

Join the discussion @ p2p.wrox.com



Wrox Programmer to Programmer™



Professional Augmented Reality Browsers for Smartphones

Programming for junaio, Layar, and Wikitude

Lester Madden

www.it-ebooks.info

PROFESSIONAL AUGMENTED REALITY BROWSERS FOR SMARTPHONES

INTRODUCTION	xxi
▶ PART I INTRODUCTION	
CHAPTER 1 Introducing Augmented Reality (AR)	3
CHAPTER 2 Natural-Feature Tracking and Visual Search	13
CHAPTER 3 Introduction to AR Browsers	21
CHAPTER 4 Latitude, Longitude, and Where to get POIs	53
▶ PART II WIKITUDE	
CHAPTER 5 Building Worlds with KML	69
CHAPTER 6 Building Worlds with ARML	85
▶ PART III LAYAR	
CHAPTER 7 Building Layar Layers	103
CHAPTER 8 Creating Filters and 2D Objects	145
CHAPTER 9 Using Layar Tools	177
▶ PART IV JUNAIO	
CHAPTER 10 Creating junaio Channels	189
CHAPTER 11 Natural-Feature Tracking and Visual Search with junaio	235
▶ PART V THE NEXT STEPS	
CHAPTER 12 Adding Advanced Functionality	267
CHAPTER 13 Taking Your Application to Market	279
CHAPTER 14 The Future of AR	291

Continues

APPENDIX A	Wikitude Support and ARML Parameters	301
APPENDIX B	Layar Support and Parameters	305
APPENDIX C	Junaio Support and Parameters	311
INDEX	319

PROFESSIONAL

Augmented Reality Browsers for Smartphones

PROFESSIONAL

Augmented Reality Browsers for Smartphones

PROGRAMMING FOR JUNAIO, LAYAR, AND WIKITUDE

Lester Madden



WILEY

Wiley Publishing, Inc.

Professional Augmented Reality Browsers for Smartphones: Programming for junaio, Layar, and Wikitude

This edition first published 2011

©2011 John Wiley & Sons,

Ltd *Registered office*

John Wiley & Sons Ltd,
The Atrium, Southern Gate,
Chichester, West Sussex,
PO19 8SQ, United Kingdom

For details of our global editorial offices, for customer services and for information about how to apply for permission to reuse the copyright material in this book please see our web site at www.wiley.com.

The right of the author to be identified as the author of this work has been asserted in accordance with the Copyright, Designs and Patents Act 1988.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, except as permitted by the UK Copyright, Designs and Patents Act 1988, without the prior permission of the publisher.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The publisher is not associated with any product or vendor mentioned in this book. This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Wikitude content reproduced with permission of Wikitude

metaio content reproduced with permission of metaio

junaio content reproduced with permission of junaio

Layar content reproduced with permission of Layar

978-1-119-99281-3

978-1-119-99286-8 (ebk)

978-1-119-99287-5 (ebk)

978-1-119-99479-4 (ebk)

A catalogue record for this book is available from the British Library.

For my wife Angelica. Without her I have nothing.

ABOUT THE AUTHOR

LESTER MADDEN (Essex, UK) has been involved with mobile technologies for 15 years. He was a product evangelist for Microsoft, travelling Europe and talking about smartphone development at conferences. He also has held marketing or developer focused roles at Skype, Nokia and Symbian. In April 2009 he started blogging on augmented reality and has been involved with the community ever since. Madden is well known in the mobile augmented reality community working with many of the application providers for content. He also runs augmented reality events in the UK and spoke at Augmented Reality Conference 2010 (Santa Clara June 2010) on trends in mobile augmented reality.

ABOUT THE TECHNICAL EDITOR

NITIN SAMANI has been working with mobile technology his entire career. He began his career as a software engineer at Symbian and then moved into technical marketing, focusing on developer programs and events. Here he gained experience working with leading handset manufacturers, operators and ISVs. He also ran a successful university outreach program (Symbian Academy), which was rolled out to more than 80 universities worldwide. Most recently, Nitin has been working in B2B business development at Samsung Electronics. To this role, he brings his breadth and depth of mobile experience to deploy solutions into key vertical markets; from Retail to Healthcare. Nitin has been involved in AR for a number of years now, writing papers, speaking at events, and conducting interviews on the subject.

CREDITS

**VP CONSUMER AND TECHNOLOGY
PUBLISHING DIRECTOR**
Michelle Leete

**ASSOCIATE DIRECTOR—BOOK CONTENT
MANAGEMENT**
Martin Tribe

ASSOCIATE PUBLISHER
Chris Webb

PUBLISHING ASSISTANT
Ellie Scott

SENIOR MARKETING MANAGER
Louise Breinholt

MARKETING EXECUTIVE
Kate Parrett

EDITORIAL MANAGER
Jodi Jensen

SENIOR PROJECT EDITOR
Sara Shlaer

PROJECT EDITOR
Box Twelve Communications, Inc.

TECHNICAL EDITOR
Nitin Samani

PRODUCTION EDITOR
Kathleen Wisor

PROOFREADER
James Saturnio, Word One

INDEXER
Robert Swanson

COVER DESIGNER
Michael Trent

COVER IMAGE
© iLexx/istockphoto.com

ACKNOWLEDGEMENTS

FOR THE PAST TWO YEARS, I have lived and breathed augmented reality (AR) via my blog (www.augmentedplanet.com), reporting the latest AR innovation from across the web and mobile devices. It's been an amazing journey and I have met some truly incredible people from the AR industry who have inspired me to keep blogging. To name just a few: Christine Perey (PEREY Research & Consulting), Myles Peyton (Total Immersion), Andy Gstoll (Mobilizy), Noora Guldemond (metaio), Danika Berlin (metaio), and James Alliban (Augmatic) — the true UK AR guru.

I have also met some great bloggers who report on AR. If you want to learn more about AR, here are some blogs and bloggers you should check out: Willy Angole (www.arnewsroom.com), Rouli (www.artimes.rouli.net), Thomas Carpenter (www.thomaskcarpenter.com), Dan Romescu (www.augmentedcitizen.org) and Tobias Kammann (www.augmented.org). If I have missed any one, I deeply apologize. You'll always find links to the best AR blogs on augmentedplanet.com

There are also those I need to personally thank for helping me with this book. There is Wiley's Birgit Gruber, whose regular phone calls to check the status helped me through those dark moments when I realized just how big the task of writing was. Jeff Riley, my editor from Box Twelve Communications, translated my jumble of words into the masterpiece you hold before you. I hope I didn't test too much of Jeff's patience. Thanks also to Nitin Samani, my technical author and good friend. Thanks to people like Frank Angermann (metaio), Xuan Wang (Layar), and Nicola Radacher (Mobilizy), who wrote fantastic documentation and samples I was able to utilize — many of which you will work through in this book. HTC kindly provided me with an HTC Wildfire that I used for creating the Android chapters.

Finally, there are those whom you neglect when you spend every weekend and evenings writing: My good friends John Wyer (who never gave up trying to tempt me with beer) and Steve Emment (who never gave up trying to get me to fix his broadband). And, of course, my wife, Angelica, who spent each weekend indoors, waiting patiently for me to finish. To Angelica, I say, "Every day, all the time."

CONTENTS

INTRODUCTION

xxi

PART I: INTRODUCTION

CHAPTER 1: INTRODUCING AUGMENTED REALITY (AR)	3
My Augmented Reality Journey	3
What is AR?	4
Why AR Is Useful?	10
Summary	11
CHAPTER 2: NATURAL-FEATURE TRACKING AND VISUAL SEARCH	13
Introducing Natural-Feature Tracking	13
How Natural-Feature Tracking Works	14
Scenarios for Natural-Feature Tracking	15
Introducing Visual Search	16
Shopping	17
Translating Languages	17
Identifying Objects	18
Marketing AR-Enabled Apps	19
Summary	19
CHAPTER 3: INTRODUCTION TO AR BROWSERS	21
AR Browser Basics	22
The Growth of AR Browsers	23
Anatomy of a Browser	24
Wikitude World Browser	25
Overview	26
Development Choices	27
Wikitude Worlds	28
Creating Your First World	34
Layar Reality Browser	36
Overview	36
Development Choices	37
Functionality	38
Layers	38
Creating Your First Layer	42

junaio	43
Overview	44
Development Choices	45
Channels	45
Testing a junaio Demo	47
Browser Accuracy	47
GPS and Compass Accuracy	49
Mapping Accuracy	50
Summary	52
CHAPTER 4: LATITUDE, LONGITUDE AND WHERE TO GET POIS	53
<hr/>	
An Overview of Latitude/Longitude	53
Working with Points of Interest (POIs)	57
Working with POI Databases	63
Summary	65
<hr/>	
PART II: WIKITUDE	
<hr/>	
CHAPTER 5: BUILDING WORLDS WITH KML	69
<hr/>	
Using the Wikitude Dashboard	70
Developing with KML	71
Creating KML With Google Earth (For Non Developers)	72
Creating a World	74
Testing	78
Simulating Locations	80
Creating KML With Google Earth (For Developers)	81
Testing	84
Understanding KML's Limitations	84
Summary	84
CHAPTER 6: BUILDING WORLDS WITH ARML	85
<hr/>	
Understanding Augmented Reality Markup Language (ARML)	86
What's New With ARML?	86
Creating a World With ARML	87
Adding the POIs	92
Completing the World	94
Creating the ARML World	98
Testing on the Device	99
Summary	100

PART III: LAYAR

CHAPTER 7: BUILDING LAYAR LAYERS	103
Creating Your Layar Account	104
Creating a Layer	105
Creating the Layer on the Publishing Site	105
Testing in the Client	107
Preparing the Database	108
Creating the Table	109
Adding POIs to the Database	112
Creating a Web Service	114
Viewing the mylayer.php Code in Full	121
Testing the Layer	125
Customizing Your Layer	128
Creating a More Compelling Listing	128
Changing POI Colors	131
Creating Icon Sets	132
Adding Layar Actions	135
Changing to Version 4.0	135
Creating an Actions Table	135
Fetching the Actions Function	137
Adding Actions	140
Adding Audio and Video	142
Adding Triggers	143
Summary	144
CHAPTER 8: CREATING FILTERS AND 2D OBJECTS	145
Using Filters	147
Creating the Real Estate Database	148
Creating the Filters	153
Connecting the Filters	157
Using SQL Queries	158
Preparing the SQL	162
Testing the Real Estate Layer	163
Troubleshooting	163
Finishing the Layer	164
Experimenting with 2D Objects	165
Changing Dimensions	173
Summary	176

CHAPTER 9: USING LAYAR TOOLS	177
Launching Layers	177
Using Layar Intent	178
The Layar Shortcut Tool	179
Requirements	180
Using the Shortcut Tool	180
Hoppala	182
Using Hoppala Augmentation	182
BuildAR	184
Using BuildAR	185
Skaloop	185
Configuring Skaloop	185
Summary	186
<hr/> PART IV: JUNAIO	
CHAPTER 10: CREATING JUNAIO CHANNELS	189
Understanding the Requirements	190
Setting up the Apache Server	190
Adding Your API Key	192
Creating Your First Channel	192
Creating the Client Listing	192
Testing Your Server Configuration	196
Setting up the POI	199
Creating Multiple POIs	204
Including Optional Parameters	206
Understanding the [name:string] Error	208
Adding Images, Sound, and Video	210
Adding Images	210
Playing Sounds	211
Playing Videos	212
Creating 3D Content	214
Debugging 3D	216
Scaling 3D Content	220
Using Animation	225
Using OBJ files	229
Creating 3D Content	230
Importing MD2 and OBJ Files	231
Summary	233

CHAPTER 11: NATURAL-FEATURE TRACKING AND VISUAL SEARCH WITH JUNAIO	235
Natural-Feature Tracking for Non-Developers	236
Creating Your First GLUE Channel	237
Adding Images and Video	241
Experimenting with Natural-Feature Tracking	242
Gluing 3D Objects to an Image	242
Natural-Feature Tracking for Developers	247
Creating a Channel	248
Building a Channel from Scratch	250
Using Visual Search	253
Overlaying Videos (Movie Textures)	257
Encoding Movie Textures	259
Image Requirements for Natural-Feature Tracking	261
Summary	263
PART V: THE NEXT STEPS	
CHAPTER 12: ADDING ADVANCED FUNCTIONALITY	267
Working with Dedicated XML Files	267
Creating Advanced Interactions	270
Adding Interactions	270
Using LLA Markers	274
Configuring an LLA	274
Retrieving Data from a Database	276
Summary	277
CHAPTER 13: TAKING YOUR APPLICATION TO MARKET	279
Marketing Your Content	280
Listing Your Content	280
Generating Excitement	283
Making Money from AR	287
Summary	290
CHAPTER 14: THE FUTURE OF AR	291
Using AR in Marketing	291
Using AR for Translation Services	292
Using AR for Interactive TV	293

Using AR in Diminishing Reality	294
Using AR in Advertising	295
Using AR in Books and Print	296
Using AR in Gaming	296
Using AR in Hardware	298
Summary	299
APPENDIX A: WIKITUDE SUPPORT AND ARML PARAMETERS	301
Support	301
ARML Parameters	301
APPENDIX B: LAYAR SUPPORT AND PARAMETERS	305
Support	305
Request Parameters	305
APPENDIX C: JUNAIO SUPPORT AND PARAMETERS	311
Support Channels	311
junaio Certification Program	311
junaio Parameters	312
Troubleshooting Guide	316
Failure of Validation Test 1 - Check Callback URL	316
Failure of Validation Test 2 - Check pois/search	316
Test 3 - Check pois/search return value error	317
Failure of Validation Test 5 - Check pois/search	317
Status Codes	318
INDEX	319

INTRODUCTION

AT AUGMENTEDPLANET.COM, I have been blogging about the rise in popularity of augmented reality (AR) since April 2009. When I started Augmented Planet, the only augmented reality applications available were either high-budget demos put together by creative agencies to wow their clients or obscure demos created by developers experimenting with the technology. Since those early days, Augmented Planet has documented the rise of augmented reality across the mobile industry and has become the leading blog and news site dedicated to all things augmented reality.

It might surprise you to know that AR isn't necessarily a new technology. You probably just haven't noticed it. AR has become popular of late because of iPhone and Android applications such as Layar and Wikitude, two of the most popular mobile AR browsers currently available. They have helped create the AR browser genre, propelling AR to the forefront of everybody's minds and capturing the attention of developers.

So what exactly is AR? In its simplest form, AR is the art of super-imposing computer graphics over a live view of the real world. AR is used in graphics for televised sporting events, whereby real-time analytical information about the game in progress is displayed on your television. It's also used with digital cameras that provide real-time information about the battery life, the number of pictures taken, or the local environment lighting level. All of that information is conveniently displayed in the camera's digital display.

AR not only blends computer graphics with live video, it extends into the field of image recognition (for example, using a computer to recognize an image and then perform a visual search whereby the image is compared to images stored in a database). For example, you might use your smartphone's camera to identify a wine label in your local supermarket. The image of that label is then compared to images in a database and, once the matching image is found, that image returns purchase information to you (such as customer reviews).

You might have already experimented with an AR browser on your smartphone that displays information about your local surroundings in the phone's camera window. These so-called browsers display everything from who is tweeting nearby to the name of the building in front of you. And it's this type of application (along with image recognition and visual search) that will be explored in this book.

WHO THIS BOOK IS FOR

This book is predominantly for developers looking to understand how to develop content for the three main AR mobile platforms:

- Layar
- junaio
- Wikitude

While I have made every effort to include examples for both experienced developers and novice/non-developers alike, you will get the most from this book if you have an understanding of PHP, XML, and MySQL. Furthermore, most of the AR applications that you will build using this book will require hosting on a publicly visible web server.

For non-developers, I have included examples of how you can create AR applications which display points of interest and how to create applications which use image recognition to display interactive 3D objects when images such are recognized. And all of this can be done without writing a single line of code. However, to get the most from this book, you should be willing to roll up your sleeves and attempt the code examples contained within.

Developing for mobile devices is never a trivial exercise. However, thanks to products such as Layar, junaio, and Wikitude, much of the complexity has been removed. So regardless of your level of programming expertise, you should be able to complete all the examples and build AR applications for the iPhone, Android (and depending on the platform, perhaps even Symbian and bada).

My goal in writing this book has *not* been to describe every single function call for all the three AR platforms in detail. My goal has been to help you understand each platform's strengths and weakness and to help you understand how to create content for all three browsers. And, of course, my goal has been to ultimately help you start your journey on the way to becoming an expert AR developer. I hope that when you do create your AR applications, you'll let us at Augmented Planet know so we can blog about your achievements.

WHAT THIS BOOK COVERS

This book will focus on the three main AR browsers platforms available today. Layar and Wikitude are two of the most popular AR browsers. The junaio browser extends the AR browser genre to support natural-feature tracking (a form of image recognition), enabling you to build applications which recognize images of family and friends as well as commercial items, such as book covers. As such, this book is divided into five main parts:

- Part I: Introduction
- Part II: Wikitude
- Part III: Layar
- Part IV: junaio
- Part V: The Next Steps

Part I: Introduction

This section introduces AR and its related technologies, such as natural-feature tracking and visual search. Additionally, this section introduces each browser and examines popular content that has been created by other developers just like you. A core part of building content for AR browsers is in the understanding of latitude and longitude and how these coordinates are used to create points of

interest (POIs). These concepts are explored in this section to help you gather the skills necessary for creating content.

Part II: Wikitude

This section covers the Wikitude AR browser and explores the two types of XML style languages. Wikitude requires no programming experience to create content when using either KML (Keyhole Markup Language) or ARML (Augmented Reality Markup Language). This section can be understood by anyone with a basic understanding of XML.

Part III: Layar

This section teaches you how to develop applications for the Layar browser. You'll learn how content can be enhanced with 3D and you'll learn how content can be triggered when users are within range of a certain destinations. Layar development requires you to have an understanding of how to configure a MySQL database and how to write PHP code. I will provide the source code and links to third-party tools that enable non-developers (or those who wish to quickly prototype) to build content without writing code. To get the most out of developing for Layar, you should have MySQL and PHP skills.

Part IV: junaio

In this section, you learn how the junaio browser enables you to add interactive 3D models and natural-feature tracking content using XML. Additionally, you will learn about indoor GPS and how to provide reliable positioning even when GPS is unavailable. Developers and non-developers alike are able to utilize much of this content to build compelling scenarios.

Part V: The Next Steps

In this final section, you learn how to take your content to market and how to attract potential users. Finally, you learn how AR is used in the wider industry as well as in other environments. You also get a glimpse into what the future might hold for AR.

HOW THIS BOOK IS STRUCTURED

While each of the three AR platforms contained in this book is different, and each uses unique programming methodologies, the book is structured such that you are gradually taken through the varying levels of complexity. The introductory section provides you with the background necessary to understand the anatomy of an AR browser as well as the basics of latitude and longitude, concepts that are essential for an understanding of the rest of the book.

Since Wikitude development is entirely based around XML and is, by far, the easiest platform to develop content, it's the logical starting point for your AR development. Layar adds additional functionality, including 3D and proximity alerts. Layar development is more complex to configure (though once your MySQL database is configured, you're all set for future development), so it's the next logical step for you to develop your skills. Junaio, the new kid on the block, is the last browser covered because it's the most advanced technology. Its features go well beyond the simple display of POIs.

WHAT YOU NEED TO USE THIS BOOK

You don't need much in order to use this book effectively. To help you decide if this book is for you, I have organized the requirements according to those that are recommended and those that are essential.

You will learn how to harvest latitude and longitude coordinates and then you'll be asked to harvest 10 locations near you. Typically, these coordinates will be within a mile or a kilometer of your home (or office). AR browsers have a maximum range, so you will use your local coordinates to populate your AR locations

Recommended

Unless you really like writing PHP or XML in NotePad, you should arm yourself with a good editor. I recommend that you use free tools, such as XML Notepad from Microsoft (<http://tinyurl.com/xmlnotepad>) for creating and editing XML, and PSPad (www.pspad.com) for creating and editing PHP. Of course, you're welcome to use your own favorite tools.

To obtain location data for the content you will build, you can use a tool such as Google Earth to harvest location-based content. Google Earth is a recommended install for the KML chapter because it exports data directly into a format that can be used by Wikitude.

This book will also point to other recommended AR applications. These are optional, of course, but they will help you get a wider view on the types of AR applications available.

Essential

To get the most from the examples in this book, you must have access to a web server you can use to create a MySQL database and host PHP files. The web server must be publicly visible and cannot be on a private network (such as IIS or Apache running on a home PC).

To test your AR content, you will need access to a smartphone. Currently the AR platforms used in this book work on the iPhone, the iPad 2, and most Android devices. Support is being added to platforms such as Symbian and bada, but availability is specific to the application you're using. In developing this book, the source code was tested on the iPhone and Android platforms.

CONVENTIONS

To help you get the most from the text and keep track of what's happening, we've used a number of conventions throughout the book.



Boxes with a warning icon like this one hold important, not-to-be forgotten information that is directly relevant to the surrounding text.



The pencil icon indicates notes, tips, hints, tricks, or and asides to the current discussion.

As for styles in the text:

- We *highlight* new terms and important words when we introduce them.
- We show file names, URLs, and code within the text like so: `persistence.properties`.
- We present code in two different ways:

We use a monofont type with no highlighting for most code examples.

We use this style to emphasize code that's particularly important

SOURCE CODE

As you work through the examples in this book, you might choose either to type in all the code manually or to use the source code files that accompany the book. All of the source code used in this book is available for download at www.wrox.com. You will find the code snippets from the source code are accompanied by a download icon and note indicating the name of the program so you know it's available for download and can easily locate it in the download file. Once at the site, simply locate the book's title (either by using the Search box or by using one of the title lists) and click the Download Code link on the book's detail page to obtain all the source code for the book.



Because many books have similar titles, you might find it easiest to search by ISBN; this book's ISBN is 978-1-1199-9281-3.

Once you download the code, just decompress it with your favorite compression tool. Alternately, you can go to the main Wrox code download page at www.wrox.com/dynamic/books/download.aspx to see the code available for this book and all other Wrox books.

ERRATA

We make every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error in one of our books, like a spelling mistake or faulty piece of code, we would be very grateful for your feedback. By sending in errata you might save other readers hours of frustration and at the same time help us provide even higher quality information.

To find the errata page for this book, go to www.wrox.com and locate the title using the Search box or one of the title lists. Then, on the book details page, click the Book Errata link. On this page you can view all errata that has been submitted for this book and posted by Wrox editors. A complete book list including links to each book's errata is also available at www.wrox.com/misc-pages/booklist.shtml.

If you don't spot "your" error on the Book Errata page, go to www.wrox.com/contact/techsupport.shtml and complete the form there to send us the error you have found. We'll check the information and, if appropriate, post a message to the book's errata page and fix the problem in subsequent editions of the book.

P2P.WROX.COM

For author and peer discussion, join the P2P forums at p2p.wrox.com. The forums are a Web-based system for you to post messages relating to Wrox books and related technologies and interact with other readers and technology users. The forums offer a subscription feature to e-mail you topics of interest of your choosing when new posts are made to the forums. Wrox authors, editors, other industry experts, and your fellow readers are present on these forums.

At p2p.wrox.com you will find a number of different forums that will help you not only as you read this book, but also as you develop your own applications. To join the forums, just follow these steps:

1. Go to p2p.wrox.com and click the Register link.
2. Read the terms of use and click Agree.
3. Complete the required information to join as well as any optional information you wish to provide and click Submit.
4. You will receive an e-mail with information describing how to verify your account and complete the joining process.



You can read messages in the forums without joining P2P but in order to post your own messages, you must join.

Once you join, you can post new messages and respond to messages other users post. You can read messages at any time on the Web. If you would like to have new messages from a particular forum e-mailed to you, click the Subscribe to this Forum icon by the forum name in the forum listing.

For more information about how to use the Wrox P2P, be sure to read the P2P FAQs for answers to questions about how the forum software works as well as many common questions specific to P2P and Wrox books. To read the FAQs, click the FAQ link on any P2P page.

PART I

Introduction

- ▶ **CHAPTER 1:** Introducing Augmented Reality (AR)
- ▶ **CHAPTER 2:** Natural-Feature Tracking and Visual Search
- ▶ **CHAPTER 3:** Introduction to AR Browsers
- ▶ **CHAPTER 4:** Latitude, Longitude, and Where to get POIs

1

Introducing Augmented Reality (AR)

WHAT'S IN THIS CHAPTER?

- What is augmented reality?
- Types of augmented reality
- Why augmented reality is useful

This chapter introduces you to the various augmented reality technologies and provides details of some applications that you may want to try.

MY AUGMENTED REALITY JOURNEY

I first stumbled upon augmented reality (AR) in February 2009 when I was working for Symbian, the company that developed the mobile OS for S60, UIQ, and MOAP smartphones. It was a Friday afternoon and I was in need of some distractions after a particularly long meeting about the annual Smartphone Show we were planning.

Like many companies, we had a forum where employees could post interesting things they have found on the web; one of those interesting things happened to be an AR video. The video had just been posted to YouTube by Microsoft Research and showed a researcher walking through the campus with his laptop screen open and a webcam filming the hallway in front of him. Displayed on the laptop screen, superimposed on top of the video from the webcam was a trail of bubbles that were emanating from an object hidden somewhere in the building. As the researcher got closer to the object, the bubbles became thicker and thicker until the object was eventually located.

I remember being amazed at how computer graphics and the live video feed had been combined in this way. Beyond simple multimedia, it was simply amazing and I knew it was going to be the next big thing for smartphones. After much frantic searching on the web, I eventually discovered a handful of AR applications (mostly available for Nokia smartphones) and I decided to create AugmentedPlanet.com as the place to document the rise of AR across mobile, web, and the desktop.

What is AR?

I like to think of AR as being the opposite of virtual reality. Virtual reality immerses the user in a computer-generated world whereas AR combines the real world with computer graphics. In effect, AR brings the computer world to us. Unlike virtual reality, which requires specialist equipment to be experienced, AR requires only a way to capture the world around you and the means to experience the computer world (typically by overlaying computer graphics in the camera window). Because the requirements are minimal, many of today's smartphones are ideal AR devices.

Before we get started on AR and its technologies, it is worth pointing out that this book takes the popular view on what AR *is* rather than presenting the view of the purist. Over the past year or so, many new solutions have been released to the various mobile application stores calling themselves AR applications. In addition, companies like Google and metaio are combining technologies like visual search (discussed in the next chapter) with their AR browsers, blurring the line defining AR. Ultimately, as developers, I think the “How do I build that?” question is far more entertaining than a discussion on the pros and cons of what is and isn't AR. With that in mind, I am going to take the all-encompassing view of AR and define AR as a technology that:

- Combines the real world with computer graphics
- Provides interaction with objects in real-time
- Tracks objects in real-time
- Provides recognition of images or objects
- Provides real-time context or data

This broader definition allows us to include technologies that are often included under the AR umbrella but are not strictly AR in the purist sense. AR, as you will learn, means different things to different people. Through blogging about AR solutions, I have learned that perceptions of what AR is and what AR isn't vary widely. Wikipedia (http://en.wikipedia.org/wiki/Augmented_reality) defines AR as a term for a live direct or indirect view of a physical real-world environment whose elements are augmented by virtual computer-generated sensory input such as sound or graphics.

Simply put, AR is the combining of computer graphics with live video feed. With such a simple description, however, you can see why it's difficult to agree on what is and what isn't AR. For example, if you are watching a live TV sporting event and the players stats are shown during the game, is that AR? If you have a digital camera, there is good chance that it has a small screen that provides you some additional information. That information might be the battery power, the number of photos you have taken, or perhaps even information about the lighting environment. Are these examples of AR? Well, they both augment your reality by providing you with additional contextual information. To many, they are valid example of usages of AR, but others will argue that to be considered true AR, they must perform tracking to keep track of objects in real-time.

As I suggested, let's not get too focused on what AR is and what AR isn't. Instead, let's look at what is generally accepted as being AR technology.

Gravimetric AR

Gravimetric AR is the latest trend in AR applications for mobile devices. These applications are typically called *browsers* and will be the focus on the applications that we build in this book. Browsers use the phone's gravimeter to determine the position of the user and the orientation. The browser genre was invented and made popular by two applications that originally appeared for the Android. These applications overlay computer data about objects that appear in the camera window. If the device is pointed at a tourist attraction (such as the Statue of Liberty shown in Figure 1-1), relevant information about the object is overlaid for the user.

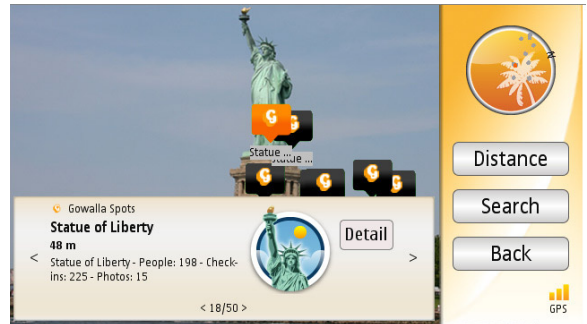


FIGURE 1-1: AR Browser example

AR browsers take advantage of a smartphone's hardware, so when the user pans the device around, new information is displayed to provide even more contextual information. Since no tracking of objects or image recognition is used, the application can be used indoors (even where a wall obstructs the view of the target object). In fact, the camera lens of a smartphone can be covered completely because the application neither knows nor cares about what the camera sees. This has led some to argue that AR browsers are not true examples of AR because the use of the camera is largely superficial, with no real-time tracking taking place.

You will learn more about AR browsers in Chapter 3, which covers terminology in more detail and provides insight into the main browsers for which you will be building content.

Fiduciary Markers

Fiduciary markers are the truest form of AR because they are used to track objects in the real-world. As shown in Figure 1-2, black and white squares are used as a point of reference or to provide scale and orientation to the application.

When the marker is recognized by the software, an action takes place. Typically that action is the overlaying of a 3D object. The software is able to track the orientation of the marker and as the orientation changes, either because the user is moving his device or moving the marker, the view of the 3D object can be changed accordingly. Ultimately, this enables the user to move around the object and view it from 360 degrees. Similarly

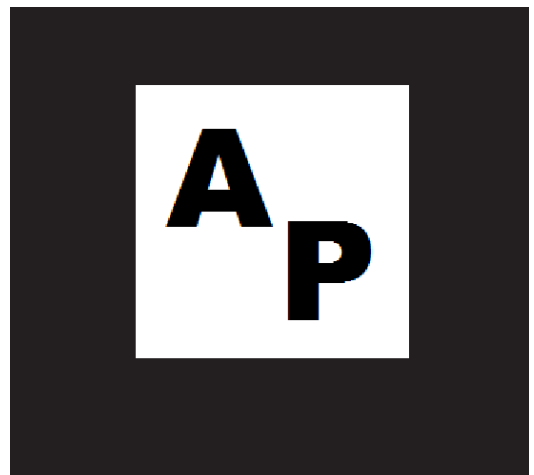


FIGURE 1-2: A fiduciary marker

the application may know about scale, and as the marker is moved nearer or further from the camera, the 3D object size can be adjusted accordingly. In Figure 1-3, the ARGirl AR application created for the iPhone by APetrus shows a 3D image drawn on a marker.

This fun application enables the user to either print or draw a marker to be recognized and tracked. When the iPhone application recognizes the marker, a 3D image of a dancer is displayed. As a nice touch, the 3D model reacts to music. Despite marker-based applications being popular with Flash developers, they haven't yet set the world on fire for mobile devices. In the case of the iPhone, Apple only added the API necessary for developers to gain access to the raw camera feed and analyze what the camera was viewing in iOS 4.0. The ability to detect markers is still fairly new for iPhone developers; Android and Symbian developers have had the functionality for some time. The number of marker-based applications, however, is relatively few.

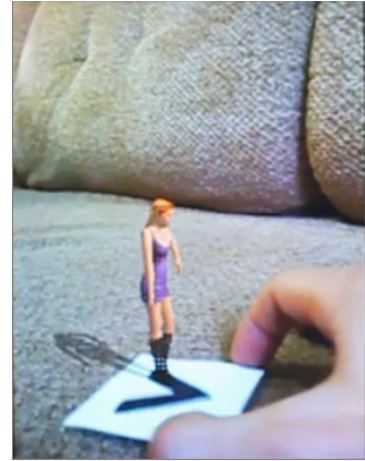


FIGURE 1-3: The ARGirl iPhone app

Barcodes

Browsers and markers represent the high end of AR technology, with markers being the earlier of the two. Some people (myself included) say that markers are a natural evolution of barcodes. If the process of recognizing a marker is considered AR, then the process of recognizing other objects—be it facial recognition or object recognition—must also be considered AR. In my opinion, the humble *barcode* (see Figure 1-4) is a form of AR.

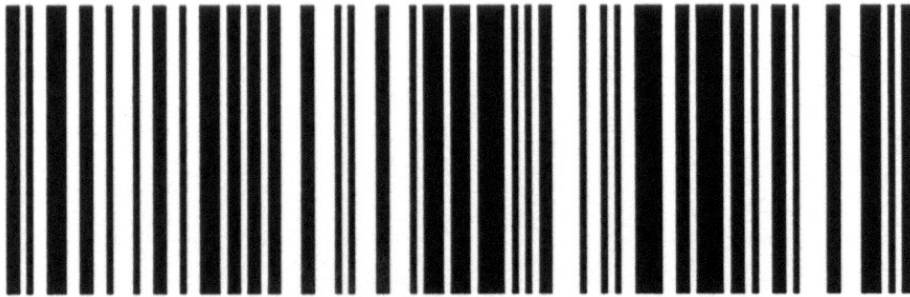


FIGURE 1-4: A standard barcode

While the barcode is not a very sophisticated image, a computer is used to recognize the code and retrieve related information when scanned. In this instance, no 3D rendering takes place; only the recognition and action occurs. Developers have taken advantage of this technology to build barcode shopping applications that enable users to scan hundreds of thousands of product barcodes to find the cheapest price online. Typically these barcode applications are not categorized as AR applications in the stores. Whether or not you consider them AR is essentially up to you.

There are many applications that enable you to use a phone's camera to scan a barcode and obtain a price online. To experiment, try Red Laser, a free app for both Android and the iPhone. Once you have installed the application, try using it on various products around the house to see what information you can retrieve. Red Laser will also retrieve nutritional information for food products, so be sure to scan a variety of barcodes to see what you can discover.

Quick Response (QR) Codes

Quick response (QR) codes are two-dimensional codes that consist of lots of squares arranged in a square pattern. Typically these are black and white, but you might discover two-color QR codes from time to time. QR codes were invented in Japan in the early 1990s and used to track the various parts in vehicle manufacturing. Today they are used as quick links to websites, quick dial for a phone number, or even to quickly send a SMS message.

While the pattern for QR code looks random, it actually contains an embedded message which can be read by a QR code reader. Once read by the camera, the action is automatically carried out. Figure 1-5 shows a QR code for AugmentedPlanet.com. If you install a free QR code reader for your device and point your camera at the code, you will automatically be taken to the Augmented Planet home page without the need to type the URL.



FIGURE 1-5: QR code for Augmented Planet

QR codes are huge in Asia, where they are frequently used in advertising and magazines. Outside Asia, however, they haven't gained mass adoption. Google is pushing the format and in 2009 distributed over 190,000 QR codes to businesses throughout the US. These codes were window stickers that enabled anyone with a QR code reader to scan the code and call up details about the business (for example, the phone number, reviews, saving the address to favorite places, and so on). In addition, the businesses were able to offer discount vouchers to the users. If you didn't see that particular campaign, you might have seen numerous web sites—primarily those that have Android applications—using QR codes to link directly to the Android Market download page. If a user scans the QR code with her Android device, she can be taken directly to the download location for the relevant application in seconds.

QR codes are not like markers, which are recognized only by applications which are programmed to specifically look for that marker pattern. Where each marker is specific to an application, QR codes contain the data encoded in the pattern. That's to say that if you created a QR code that links to a web site, the pattern that is generated follows an ISO standard with the URL encoded that can be read by different QR code applications. Since the information is embedded in the pattern, it cannot be changed and you must generate a new code to handle any changes (for example, pointing to a different URL).



There are many free web sites that enable you to create your own QR codes, so take 10 minutes to experiment with creating the various types of content. Try <http://qrcode.kaywa.com/>.

Microsoft Tags

Like QR codes, *Microsoft tags* are a quick way to enable users to access content. Unlike QR codes, however, which have all the information embedded in the image, tags enable you to dynamically update the source as often as you wish. With a tag you can update the destination or action to reflect a change in the URL or phone number just by changing the destination on the tag website. This is a clear benefit because you don't need to update any printed material. You can also password-protect the resource so it can be used only by approved users.

You might choose to add a tag to your business card and have that tag represent your home phone number. In that situation, you could give your business card to anyone, but only those who know the password would be able to access your home number. Figure 1-6 shows a tag that provides a shortcut to the Augmented Planet web site.

Tags can be either color or black and white and the default image also includes details of where users can obtain the tag reader software. Of course, this information is entirely optional, but it's useful to promote exactly what the tag is and how it can be read.

Tags also provide rich tracking functionality so you can see how often users use your tag. In addition, you can specify the duration that your tag is alive, setting both its start and expiration dates. Microsoft tags are still relatively new, but Microsoft is pushing the format to many advertisers, promoting the use of tags in their campaigns as a quick way to take a user from a printed advertisement to a website or even to place a call. Unlike QR codes, tags are—to some extent—customizable and can be combined with images.

In Figure 1-7, the image on the left is the raw tag; the image on the right is a tag that has been combined with an image of my cat. Both perform the same action. A skillful artist would be able to create a tag that utilizes the colors as part of the design to create a truly blended tag.



FIGURE 1-6: Microsoft tag for Augmented Planet

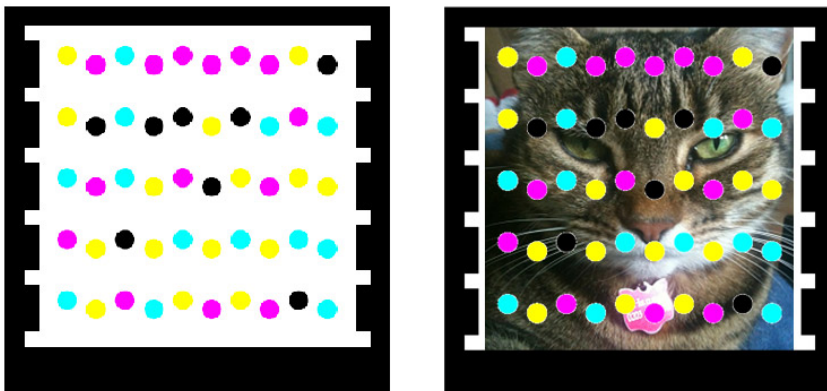


FIGURE 1-7: Customizable tags

Branding your tags with an image of your company logo is a good way to promote your software or services while also allowing users quick access to the information you're promoting.

You should download a copy of the Microsoft tag reader from your phone's application store. Once you install it, you can create tags at <http://tag.microsoft.com/> and then test them.



Barcodes, QR codes, and tags are the simplest examples of AR. They are often referred to as hyperlinking rather than AR.

Markerless AR

As shown in Figure 1-2, markers are the black and white squares which enable an application to track and detect the orientation and adjust the position of the 3D object accordingly. *Markerless AR* is using AR without tracking or tracking without a special marker. Gravimetric AR browsers are an example of markerless AR since points of interest (POIs) and other data are displayed in the camera window but the data is not attached or tracked for any particular visual object. Many of the mobile games that claim to be AR are games that simply turn on the camera as the backdrop. Other than the camera providing a local setting for the user, the use of the camera is superficial. So, again, some will claim that this is not real AR.

Figure 1-8 shows Soulbit7's entertaining AR Invaders iPhone game; this game adds an AR twist to the classic Space Invaders game by putting users in their surroundings (courtesy of the smartphone camera's ability to provide game's backdrop).

Markerless tracking is where AR is used to track objects in the real world without using special markers. Face recognition is an excellent example. You might be surprised to see just how advanced facial recognition systems are for mobile devices. Polar Rose, a Swedish company, showcased a facial recognition prototype for Android devices in 2009. The prototype, named Augmented ID, allowed a user to point his Android phone at a person's face and the application would compare the face to faces in a database in an effort to find a match. If a match was found, the application overlaid the subject's social media profile (Twitter, Facebook, LinkedIn, LastFM, and so on). The technology has since been purchased by Apple, so it gives an indication of the types of markerless AR applications we'll see in the future.

Since the acquisition, the free face recognition SDK is no longer available to download. However, at the Mobile World Congress trade show in Barcelona in February 2011, a new company named Viewdle (www.viewdle.com) showcased their free face recognition SDK for Android developers. As Figure 1-9 shows, face recognition presents interesting use-cases for mobile applications.

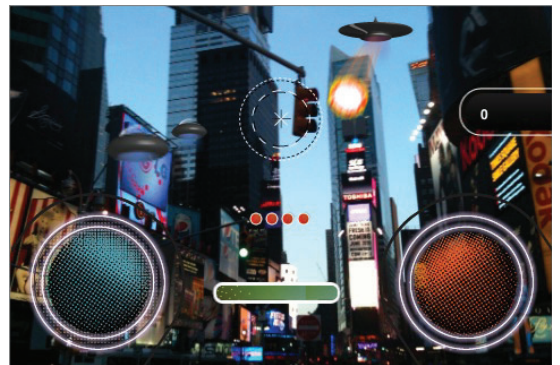


FIGURE 1-8: Soulbit7's AR Invaders iPhone game

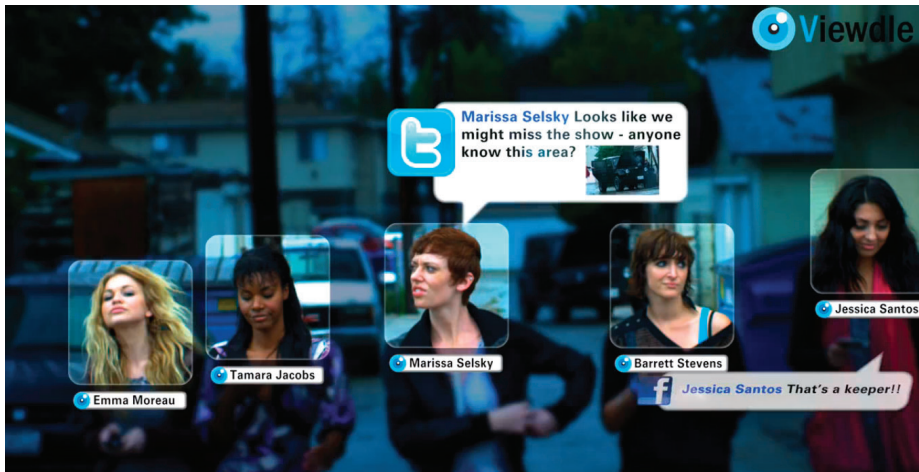


FIGURE 1-9: Viewdle's face recognition SDK

You will learn more about markerless tracking in Chapter, 2, “Natural Feature Tracking and Visual Search.”



Watch this facial recognition demo to see how advanced markerless AR is:
<http://tinyurl.com/3ynlfop>.

WHY AR IS USEFUL?

It's easy to see how useful hyper-linking applications are. Imagine reading an article on the bus to work and seeing a link to a web site to obtain more information or download an application. You could try to remember the URL for when you get to the office or you could type the URL into your phone's browser. With AR, however, you can just snap a picture of the QR code/tag and have the information instantly. Considering the usefulness of QR codes, it is surprising that they are not more widely used.

The real strength of AR browsers is their discoverability. Today, browsers have most of the attention and it's amazing how many people have yet to experience a browser for themselves. Browsers are incredibly useful ways to discover information about places and objects around you. Browsers have helped me discover information about my neighborhood that I never would have discovered otherwise.

One application for the iPhone (Get London Reading) shows all the books that were set around my immediate location; surprisingly, there are a lot of books based on my neighborhood. Another application (Museum of London) superimposes old pictures from the early 1900s so you can see how your neighborhood looked in the past. Of course, these applications I describe are specific to

London; in later chapters, you will get to experiment with AR browsers and discover content for your own neighborhood.

SUMMARY

AR is defined as a means for overlaying computer graphics over live video captured by the camera. An AR application can be as simple as the adding of a graphic to a video feed. As AR becomes more popular, so, too, does the debate of what AR is and what AR isn't. If you have experienced AR by holding a marker to your webcam and interacting with 3D, you're not likely to consider anything less as a true form of AR.

AR is also used in barcode recognition, which is considered to be among the earliest examples of AR applications. These types of applications are considered to be hyper-linking applications because they do not render any display.

The most popular type of AR application for mobile devices available today is the browser that overlays contextual data about objects or locations for your surroundings. These browsers are what you will focus on in later chapters.

2

Natural-Feature Tracking and Visual Search

WHAT'S IN THIS CHAPTER?

- An introduction to natural-feature tracking
- An introduction to visual search

In this chapter, you will learn about natural-feature tracking, which expands on the concept of the marker and enables 3D objects to be overlaid on top of real images (such as book covers, CD covers, or even your own holiday snapshots). You will also learn about visual search, which is similar to natural-feature tracking except a different action takes place when an image has been recognized. Before you begin, you'll need an Android or an iPhone.

INTRODUCING NATURAL-FEATURE TRACKING

Unlike AR solutions that use markers as their basis for recognition, *natural-feature tracking* solutions can be applied to almost any image as long as the image is complex enough. An example of a natural-feature tracking application is a mobile application that can recognize a movie poster. With natural-feature tracking, the application can analyze the poster and identify it by comparing the poster image to similar images. In contrast, a marker-based solution requires a special identifier to be included on the poster; it would be the marker that provides the identification rather than the poster image.

Markerless solutions make a better solution for a couple of reasons. Markers and QR codes are not user-friendly and you would have a difficult task persuading the marketing people behind the latest Hollywood blockbuster to include such content with their marketing materials. Secondly, any such markers would need to be included on material at the time of printing. Marketing campaigns that use marker technology cannot be created retroactively.

In the movie poster example, it's impossible to add markers for Jurassic Park because all packaging and marketing materials have already been produced. With an application capable of natural-feature tracking, there is no longer a requirement for a marker to be used. Anything in the real world (including, a face, an object, or an image) can be tracked and identified.

You'll notice that in the description of natural-feature tracking, I have gone to great lengths to avoid simply stating that the technology is image recognition. This is largely due to the action that takes place when the image is recognized. Typically, when 3D is overlaid on top of the image, the image needs to be tracked so the 3D object can respond to changes in orientation and scale. Because no marker is present, the application uses the natural features of the image to provide alignment and orientation — hence the term *natural-feature tracking*. Image recognition is still a valid term, but it's more akin to visual search solutions that simply identify an image and carry out a simple action. You will learn more about visual search later in this chapter.

How Natural-Feature Tracking Works

To recognize an object, a reference image is required. In keeping with our movie theme, let's imagine we want to recognize the Jurassic Park movie poster and draw a 3D dinosaur on the poster that can be viewed with a smartphone. Since we only want to draw the dinosaur on the correct movie poster, we obviously need to have the Jurassic Park movie poster image so it can be compared for matches. Let's call this original Jurassic Park movie poster our reference image. When users view the movie poster with their smartphones, the poster they are viewing in their smartphone camera windows must be compared to the reference image to see if the posters match. If they match, the 3D dinosaur is displayed; if not, no action takes place. That, of course, is a huge simplification of the process. In reality, it is a lot more complex.

The ability for humans to spot patterns is an amazing skill. Just by looking at a movie poster, we can identify it against a likely match. Unlike the human eye, which can instantly tell the difference between the Jurassic Park movie poster and a chocolate wrapper, a computer needs more assistance in understanding what it is seeing. Therefore the reference image is not simply a JPG or other image file residing on a server. Instead the reference image of the movie poster has to be converted to an image map of the light and dark areas that make up the poster. The reference image that is created is no longer something that can be viewed by humans. Instead it is a file that contains the encoded information that represents the light and dark areas of the poster. Don't worry too much about this file; the format is proprietary. Just accept that it's an encoded file that you host on your web server for comparison.

The application that will recognize the Jurassic Park poster will need to convert whatever is in the smartphone camera window into a similar map of light and dark areas. Let's call this the comparison image. The comparison image can be created in one of two ways:

- The user takes a picture of the movie poster and the application converts the picture into the comparison image. In this case, the comparison image is sent to the server where the image map is located and then analyzed to determine if they are the same.
- Or, as you'll see with modern smartphones that have greater processing power, the application analyses the video feed directly from the camera window (rather than requiring a phone) and constantly generates comparison images. As soon as a match is found, the desired action takes place.

Reference images that contain lots of sharp edges and high contrasts work the best. A polar bear sitting in a snow bank would not have enough contrast to produce a suitable reference image. Color is generally not an issue because, in most cases, the colors contained in the image are lost when the image is converted into the reference image.

Scenarios for Natural-Feature Tracking

Solutions that use natural-feature tracking typically overlay a 3D object over the recognized image. Like its marker-based counterpart, the image in the camera window is tracked and, as it is rotated, the 3D object is updated in real-time, enabling the user to view the 3D object from multiple angles.

As processing power for mobile phones increase, we are beginning to see more of these types of applications. They represent the future for AR and the days of the marker are numbered. Whereas hyperlinking solutions enable the user to quickly go to a web page, natural-feature tracking enables the user to interact with the content. Imagine seeing an advertisement for the latest trendy sports car in your newspaper. Hyper-linking would enable you to quickly visit the home page for the advertisement while natural-feature tracking will enable you to point your smartphone at the advertisement and see the car appear in your camera window. By moving your smartphone around the advertisement, you can view the car from any angle. Perhaps by interacting with the keys on your phone, you will even be able to customize the car to build your perfect ride.

As you will see in later chapters, these types of solutions are possible today. Figure 2-1 shows a user interacting with 3D provided by a natural-feature tracking solution. In the picture you can see the user holding their smartphone up to an image (hidden) with 3D graphics being overlaid.

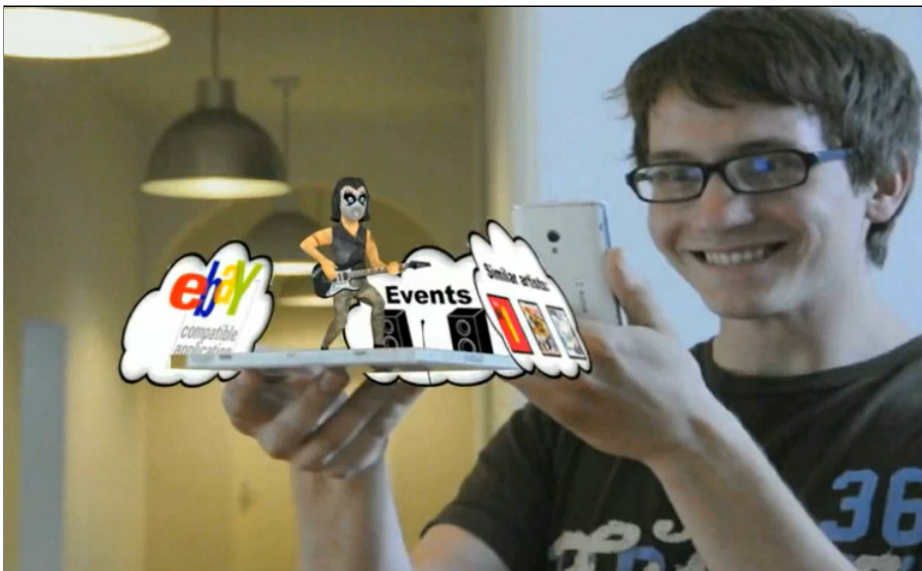


FIGURE 2-1: An example of junaio GLUE's natural-feature tracking

Gaming

Games that use natural-feature tracking are also becoming popular. One of the first popular natural-feature tracking games for the Android platform was a game called Space InvadAR. In this game, users play a clone of the popular Space Invaders game by using a picture of planet Earth. A player simply points his camera at the picture; once the image has been recognized, alien ships are launched, and the player is tasked with defending the planet. All this takes place over 3D images of Earth, providing a realistic background for the game.



If you are interested in seeing Zenitum's Space InvadAR game, you can find the YouTube video at <http://tinyurl.com/39nyxjy>.

Not all feature tracking has to rely on recognizing a predetermined image. Some applications merely detect that something is shown in the camera window and changed. The change could be that something new has been added to the video feed or something has been removed. As an example that illustrates this concept, Upsies! is an iPhone game based on the popular game of trying to keep a soccer ball in the air for as long as possible by using only your feet. The game draws a soccer ball, which is visible in the phone's display. Point your camera at your feet and then use your feet to keep the ball in the air and prevent it from touching the ground.

The game works by comparing the images taken in each video frame from the iPhone's video feed. Think of the app as creating image maps on the fly and comparing them for matches. As it compares images, it is able to detect that a foot (or any other object) has been introduced and it detects when the foot is near the 3D ball. If they connect, it counts as a valid move.



You can see an example of Upsies!, developed by Blind Mice Studios, at <http://tinyurl.com/4hpwa71>.

INTRODUCING VISUAL SEARCH

Visual search is hotly debated as to whether it actually is AR. As previously mentioned, we'll take the popular view and include it here as an example of AR — but feel free to decide for yourself. Personally, I have no problem considering visual search an AR technology. If QR codes are considered part of AR's history, then it's difficult to deny visual search the same courtesy.

Visual search applications work in a way that is similar to a natural-feature tracking application. In fact, each works the same as the other until the point at which the object/image is recognized. Natural-feature tracking applications are likely to overlay 3D; visual search applications perform a search based on what has been recognized.

Typical examples of how visual search is used include:

- Shopping
- Translating languages
- Identifying objects

Shopping

There are numerous mobile applications that enable the user to shop for a product (such as a book, a DVD, or a game) just by taking a picture of the product. This is useful if you happen to be in a store and see a product you want to buy. Rather than making a note of the name of the product, you could take a picture of the cover to obtain a price comparison from Amazon or Google advertisers. Once the photo has been taken and processed, it is sent to the server for comparison to a reference image. If the photo matches the reference image, the product is identified and the price comparison results returned to the user.

SnapTell is a free application available for both the Android and iPhone and is worth installing to get a taste of how natural-feature tracking can be used for shopping. Try installing the application and taking pictures of books you have around the house.

Translating Languages

I have long thought that AR and translation go hand in hand. Imagine traveling to a foreign country and being able to get live translations on any text you point your smartphone at. Everything from street names, menus, and instructions all at the point of a lens. Numerous applications have attempted this, but the process of recognizing the text has been a painful one. The user first needed to take a picture, then the text in the picture needed to be highlighted, and, finally, it was then submitted to a server for processing.

Word Lens from Quest Visual is the first real natural-feature tracking AR application I have seen. The application translates the text and even overwrites the original language. Figure 2-2 shows the original English text on the left; when the language is converted to Spanish (shown on the right), it is displayed in place of the original language.



© COPYRIGHT 2010, QUEST VISUAL, INC.

FIGURE 2-2: An example of a Word Lens AR translation

Applications like Word Lens rely on an optical character recognition (OCR) engine to convert the text found in the camera so it can be submitted to a translation engine and processed. Applications that use OCRs are not considered to be AR applications, but what sets this apart is the object containing the text is tracked and the results overlaid.

Identifying Objects

One of my favorite apps to demo at an event is Google Goggles, which is available for both Android and the iPhone. Like the shopping example that searches the image for a product match, Goggles identifies wine labels, works of art, and even architecture.

Imagine being on vacation and taking a photo of a famous building. Goggles will identify it for you! Images are processed in a manner similar to other examples in this chapter, only with Goggles, the images are compared to the millions of images that Google holds.

Goggles is must-try application for anyone interested in AR. Install it from your phone's app store by searching for Google Goggles on Android or Google Mobile App on the iPhone.

Once you have installed the application, you can point the camera at different classes of objects to perform a visual search. The application even distinguishes between objects and images. Take a photo of the image shown in Figure 2-3 and use Google Goggles to identify the object for you.



REPRODUCED FROM GOOGLE™

FIGURE 2-3: Using Google Goggles to identify a famous bridge in London

MARKETING AR-ENABLED APPS

In Japan and the Far East, users are already accustomed to QR codes and recognize that they have special meaning. In Europe and the Americas, QR codes are generally known only by those who are technically savvy. Markerless applications present their own challenges. Imagine if you designed an interactive CD cover that displayed a video of one of the songs from the album when viewed through a smartphone. Or imagine an interactive movie poster that played a preview of the movie. How will you tell users that the image has special AR functionality?

Total Immersion, one of the giants in desktop and web AR, is trying to educate the world about AR-enabled apps by providing a unique AR+ icon (see Figure 2-4) that AR-enabled images/products can use to denote that they are AR enabled. Although Total Immersion's efforts are still in their infancy, Total Immersion has been working with third parties (including clients, partners, and journalists) to promote the adoption of the AR+ icon.

Content providers and developers can customize the icon to indicate what special content is included. For more information about the program, visit <http://arplus.t-immersion.com/>

This move is intended to accomplish several objectives simultaneously: educate consumers, ease application development for AR providers and their partners, allow for seamless integration with existing solutions, control quality levels across the board, and promote industry growth.

Total Immersion is sensing a real demand in the market for this kind of clarity backed by an organized, collective effort. AR is moving well beyond digital marketing. Coming up fast are AR solutions in e-commerce/retail, experiential education and entertainment, medicine and science, embedded AR in durable consumer products, and public safety and transportation, among others.



© 2008-2011 TOTAL IMMERSION
— ALL RIGHTS RESERVED

FIGURE 2-4: The AR+ icon

SUMMARY

In this chapter, you have learned about the various types of natural-feature tracking and visual search applications that are available for the Android and iPhone. You have learned that AR can be used to simplify users' lives, whether it's by providing helpful product information or translating foreign languages.

Hopefully you've installed and experimented with some of the standalone applications that are available for your mobile device. If not, you should. As you learn to build AR applications for browsers later in this book, it's helpful to know what applications are available and what other developers are building.

3

Introduction to AR Browsers

WHAT'S IN THIS CHAPTER?

- An overview of the clients you will be developing content for
- Examples of AR content created by other developers
- A first look at the functionality provide by AR browsers
- Why browsers are not accurate

For the course of this book we will focus on building content for the three leading AR browser platforms. Taking this approach presents us with some benefits and, of course, the inevitable drawbacks. Before you begin, you'll need an iPhone or an Android phone and the latest versions of junaio, Layar, and Wikitude.

Benefits of using the three leading AR browser platforms include:

- It's easier to build content with little or no programming knowledge required. In many cases, you can access a database or provide XML to reference your content.
- Your content appears to an audience already interested in AR and actively seeking your content.
- It's in the platform providers' interests to actively promote their applications to users, so that means less marketing on your part.
- You can take advantage of richer features as the platform provider makes the functionality available.
- You don't have to worry about camera input, screen layout, or GPS APIs.
- The same content will work on Android, iPhone, and Symbian/Windows Phone/bada OS/MeeGo when the clients are released.
- It's becoming increasingly common for AR browsers to be preinstalled on mobile devices. This provides even more discoverability opportunities for your content.

Drawbacks include:

- Users must launch another application before finding your content.
- Your content can lay undiscovered in the platform provider's client.
- You don't have full control over the user interface.

Mobile AR is still very much in its infancy, so unless you have a specialist need and want to provide functionality beyond some of the benefits highlighted previously, your best option — for now, at least — is to build content for one of the leading platforms. These companies are investing millions of dollars into making their platforms a success; they have a marketing budget we can only dream of. It's better to ride their wave than try to fight the tide ourselves.

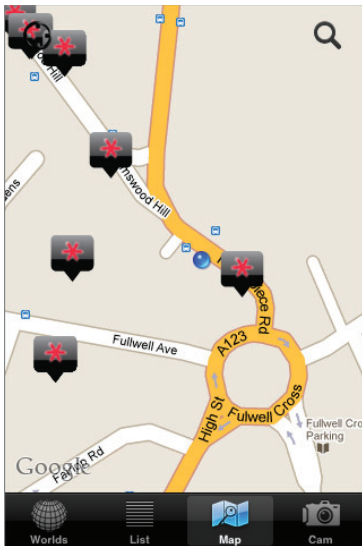
Another reason not to go it alone is the fact that on the Android and the iPhone, there are almost 100 AR browsers all fighting for user attention. Many are simple; they offer nothing more than the ability to view POIs in the camera window. A handful offer developers the ability to build their own content, but only three are known all over the world and used by millions of users. With that in mind, we will focus on building content for Wikitude, Layar, and junaio.

Each browser offers unique functionality. So before we get down and dirty, we'll take a look at each browser in detail to learn more about each platform and what kind of content developers are building.

AR BROWSER BASICS

While I said in Chapter 1 that you need a camera and a screen to experience AR, it may surprise you to know that the minimum requirement for an AR browser is simply a smartphone that has a GPS. Why no camera? Well, it is less common, but some developers have taken the approach of building applications that provide an AR experience through augmenting your hearing rather than using a visual method. Toozla is an application from a Russian company that detects the users' location via GPS. When users come into range of a POI, an audio file begins to play, giving users an overview of their current surroundings. If users remain at the POI to listen, the application provides more detailed commentary. Like most AR applications, Toozla uses the Wikipedia API to provide some of its content, but Toozla also uses professionally recorded material to give users a richer tour guide experience. We are not going to cover building content for Toozla, but they do have an API if you are interested in producing audio content. You will find them at www.toozla.com

AR browsers work by using GPS to detect the user's current location while the compass detects the direction the device is facing. With the user's current location and their direction known, the nearby POIs can be displayed in the camera window. As the user moves the mobile device around, the accelerometer detects the elevation while the compass continues to detect the direction the user is currently facing. This combination enables the application to use the map data to build the AR view. In Figure 3-1, you can see POIs from the Qype world for the Wikitude browser shown on a map, with the map indicating their actual location. In Figure 3-2 the same POIs are drawn in the AR browser window, providing a visual indication on their locations.



©2011 GOOGLE, MAP DATA
©2011 TELE ATLAS

FIGURE 3-1: POIs shown on a map

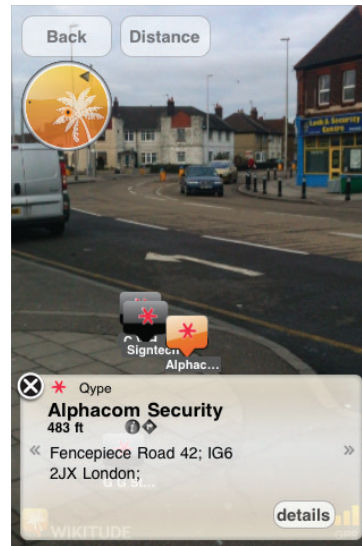


FIGURE 3-2: POIs shown in the AR browser



Many of today's mobile devices are equipped with accelerometers and magnetometers, both of which are put to good use with providing an AR experience. The accelerometer is used to sense the orientation of a device (for example, when the screen view on the device is moved from landscape to portrait) as well as detect the acceleration or how quickly a user moves the device. The accelerometer is also used to detect the angle of the device, enabling an application to know if the camera is being pointed at the sky or at the ground. In addition, the magnetometer — or digital compass, if you prefer — can track changes in the Earth's magnetic field and determine with considerable accuracy which direction the user is facing and thereby determine North, South, East or West. Hidden Sky for the iPhone is an AR application that enables users to track the location of satellites, stars, and planets in the sky. The user simply points his iPhone at the sky; the application uses the accelerometer to determine the height the device is being pointed and the magnetometer determines the direction. In this example, GPS determines the user's latitude and longitude position and then presents him with the relevant POIs.

The Growth of AR Browsers

Since the first AR browser first appeared in 2008, the genre has grown to around 300 applications available for the iPhone alone. At last count, an average of 20 new iPhone applications that exhibited AR browser behavior were being added to the iPhone App Store each month. Many of these applications combine geo-location and the camera with freely available APIs from Wikipedia, Twitter, Flickr, Google Search, or an untold number of other sources to enable users to visualize the

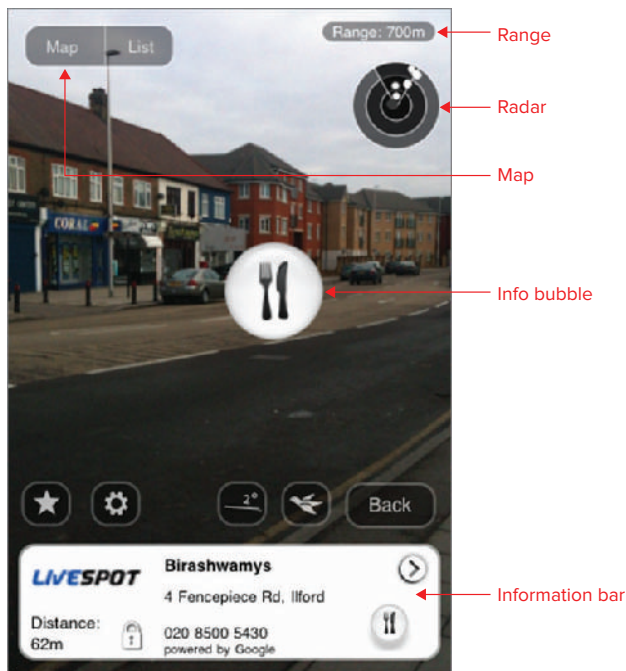
locations of POIs around them. There are applications that, for example, help you see who is tweeting around you, help you find your nearest train station, or help you find a place to eat.

Since this book is about building for the popular platforms rather than building applications from scratch, you can relax a little because a lot of the plumbing is taken care of. You won't need to worry about APIs necessary to detect the user's location, what design is best for building a user interface, or even how to figure the movement of the user's devices. In most cases, you need only to supply the XML that contains the coordinates of the POI along with relevant icons and text. The difficult choice is determining which platform to build for.

Anatomy of a Browser

Before you delve into the various platforms, however, you should learn what all the platforms have in common so you have a better understanding of what is available to you. All platforms offer similar functionality but they all name their elements differently. So for the time being, we will refer to the components of the browser as follows (see Figure 3-3):

- Radar
- Info bubble
- Information bar
- Range
- Map
- App Store



© 2009 WEB MEASURE PTY LTD

FIGURE 3-3: Anatomy of a browser

Radar

The *radar* provides a visual location of the direction where the POIs are located and essentially tells the user which way the device should be pointed.

Info Bubbles

POIs are displayed as *info bubbles*, which is an icon that visually indicates what it represents (for example, a knife and fork for food or a house that indicates a structure).

Information Bar

Once a user selects the info bubble, a short description about that POI is displayed in the *information bar*, offering the user more information about the POI active in the info bubble. If a user selects the text in the information bar, they are taken to a longer description about the POI. Depending on the browser, this can be a web page, a file, or text.

Range

The ability for the user to set the search range enables the application to limit the distance for the POIs that are returned. If you are in London looking for the nearest hotel, you probably won't want to see hotels that are 50 miles away. The range feature enables you to control the maximum distance you want content returned, making the content more relevant to your location and giving you more control over what content is displayed in the camera view.

Map

While AR is great when it comes to helping you visualize the location of the POI, it is not always useful in helping you navigate. Fortunately, most browsers include a map so you can see the POIs in a traditional map view. Included with the map is often the ability to get turn-by-turn instructions to the destination.

App Store

As you learn more about the individual browsers, you'll notice that developers have already been building an amazing amount of AR content. Naturally, to find content, you need an application store as part of the browser. This is where your content will be listed and found by users.

Now that you have a basic understanding of the features of each browser, it is time to meet the key players in more detail.

WIKITUDE WORLD BROWSER

Application name: Wikitude World Browser

Platforms: Android, iPhone, Symbian. Bada

Developer: Mobilizy

Home page: www.wikitude.org/

Originally appearing for Android devices in 2008, Wikitude has the honor of being one of the first AR browsers released. Since those early days, Wikitude has gone on to appear on the iPhone, Symbian and more recently, Samsung Bada devices. Wikitude has also won many industry awards, including being voted the best Augmented Reality Browser in the Augmented Planet Readers Choice Award in 2009 and 2010 Wikitude is also the easiest of the browsers to build content for, enabling anyone regardless of their development experience to create content through a Google Maps interface without having to write a single line of code.

Overview

The Wikitude client has undergone a number of changes since it was first created. The current version (see Figure 3-4) shows a list of Worlds that can be sorted by Featured (staff picks), alphabetically (A-Z), or by Distance from your current location.

Worlds can be loaded individually or you can select multiple Worlds to load. This can be useful if you are looking for tourist attractions and are not sure which World the content lives inside or if you want the widest possible range of results. In addition, users can search for content with the search icon and perform a search across all the Worlds to return relevant content.

Once a World has been selected, users can view the content in either the AR view, in a map, or in a list.

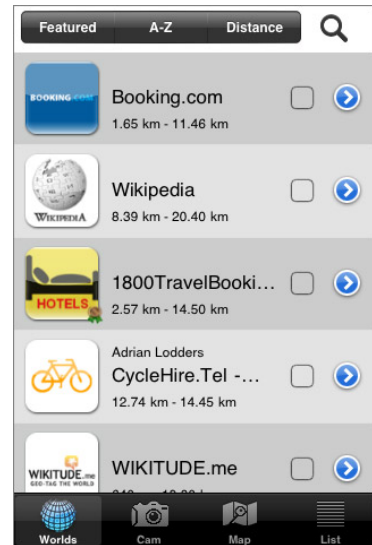


FIGURE 3-4: The Wikitude client

Wikitude Drive

Mobilizy is also pushing the boundaries with AR and recently released the world's first AR navigation system for Android devices. Aptly named Wikitude Drive (see Figure 3-5), the application displays driving instructions over the live camera feed. Although it's beyond the scope of this book, it does show the incredible investment that platform providers are placing in their products and the future direction that AR will take.



FIGURE 3-5: Wikitude Drive (Android)

Development Choices

In Wikitude speak, developers build Worlds. Once these worlds are submitted and approved, they appear directly inside the Wikitude client. Building Worlds for Wikitude is probably the simplest of the platforms we'll cover in this book. Also, it offers the greatest variety in terms of development methods. Developers can create Worlds by using:

- The Google Maps interface
- Keyhole Markup Language (KML)
- Augmented Reality Markup Language (ARML)
- WebServices
- APIs

Google Maps Interface

Developers and non-developers alike can add simple POIs through a Google Maps interface. Content created this way appears in the Wikitude.me World. This is incredibly useful if you want to geo-tag just a few locations, such as your favorite shop, bar, or restaurant.

Keyhole Markup Language (KML)

Keyhole Markup Language (KML) is an XML-based schema that describes geographical information. It was originally created by Keyhole, Inc. (since acquired by Google) and is the format used by Google Earth to describe locations, coordinates, and so on. As a common standard, it provides Wikitude developers with access to a wide range of content without them needing to format content into a new structure.

Augmented Reality Markup Language (ARML)

Augmented Reality Markup Language (ARML) is a standard XML schema proposed by Mobilizy (the creators of Wikitude) as a standard format for building cross-platform or cross-device AR browsers. In theory, if the XML is in the same format, then the same file can be used by developers in other browsers without having to tweak it. ARML offers you richer functionality over KML it enables you to include features such as phone numbers, addresses, and other helpful information.

WebServices

Both the KML and ARML files which describe the POIs are hosted on the Wikitude server. From a developer's perspective, you need only to create the XML file (either in KML or ARML format) and enter a simple form to submit it to the Wikitude client. This is why developing for Wikitude is so incredibly simple and attractive to many developers. If, however, you maintain a database of geo-coded data, you can chose to continue to host that data and provide a web services interface to enable Wikitude to access the data.

APIs

If you don't like the thought of not having control over the UI, or if you want to have your own launch icon for your content, the Wikitude APIs can be embedded in your own applications, thereby giving you more control over your products. This is especially useful when you want to create an experience that is tightly integrated and offer richer functionality.

API Example

In general, I am not a huge fan of developing standalone AR applications, particularly if the application offers nothing more than the ability to view a limited number of POIs in an AR view.

For example, developing an application that does nothing more than show users the location of the nearest airports is unlikely to give you very much reward on your investment. Building a World would be a much better option. If, however, you wanted to extend an existing application that provided users with flight arrival and departure information as well as locations of the airport shops, hotels, and car-rental agencies, then extending the application to support AR is a much better solution than trying to integrate that functionality into someone else's product.

An excellent example of this is the IBM Seer application shown in Figure 3-6, which created a compelling standalone experience for tennis fans at the Wimbledon Tennis Open. Users could view the scores of all the matches taking place across the various courts, monitor the wait time for the taxis, find amenities, and even use AR to get live information about the length of the wait time at the food counter.

The IBM Seer application was built using the Wikitude API and offered a much richer experience than an application displaying only POIs. Thus, it made more sense than trying to create a World.



IBM®

FIGURE 3-6: The IBM Seer application

Wikitude Worlds

For a product that has Wiki in its name, you might expect that it would include Wiki-based content — and you'd be correct. Initially, Wikitude was entirely based upon Wikipedia data and was primarily used as a city guide that enabled users to identify tourist attractions. Since Wikitude became a fully fledged platform with an open API, more than 500 Worlds have been developed by developers all over the world, enabling users to find AR content for everything from shopping venues to any YouTube videos that have been shot in the area.



If you haven't done so already, install Wikitude from either the Android Market or the Apple App Store and try out some of the numerous Worlds that are available.

The list of Worlds available to you might differ from those available to me because Worlds are relative to your location. Wikitude does an excellent job of only displaying content that is within

a reasonable range of your phone so searching for pizza doesn't produce results for pizza joints 3,000 miles away. As you explore the client, be sure to:

- Test the search icon and search for nearby businesses.
- Click on POIs to explore the data in the info bubbles and the descriptions.
- Test the routing and calling functionality.

This section covers some Worlds that you may want to try.

Wikipedia

The Wikipedia World is pretty much the catch-all World providing users with locations of train stations, rivers, hotels, tourist attractions, and more. Because the World takes its data from Wikipedia, it can produce a great deal of content for users to navigate. Nevertheless, it's an excellent World for discovering your surroundings. Figure 3-7 shows Wikipedia data in an AR view; Figure 3-8 shows the same data in a list view. (Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.)

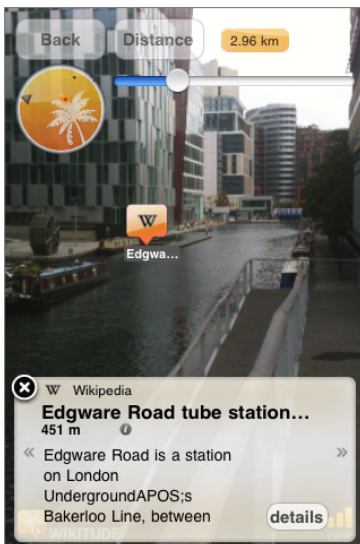


FIGURE 3-7: Wikipedia data in the AR view

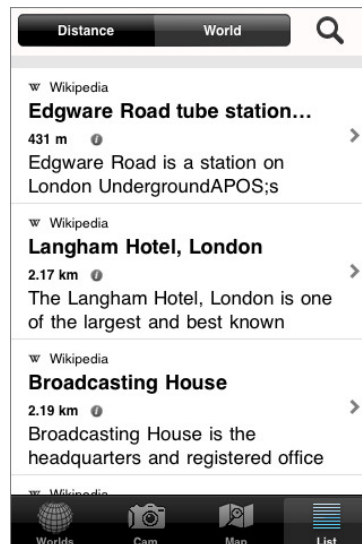


FIGURE 3-8: Wikipedia data in a list view

YouTube

YouTube features millions of videos uploaded by users from around the world. Have you ever wondered if any of that content is geo-tagged for your local area? I don't live in a particularly exciting place, but I am always amazed at how much YouTube content has been filmed and uploaded from around my neighborhood. Without AR, I never would have discovered this content. Figure 3-9 shows YouTube videos in an AR view; Figure 3-10 shows YouTube videos that have been shot around my neighborhood.

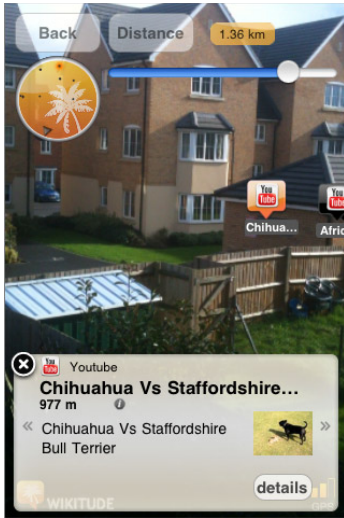
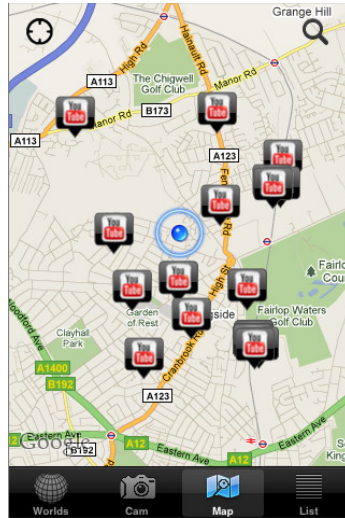


FIGURE 3-9: YouTube videos in the AR view



© 2011 GOOGLE, MAP DATA
© 2011 TELE ATLAS

FIGURE 3-10: YouTube videos filmed around my neighborhood

Webcams.travel

The Webcams.travel World contains publicly accessible webcams from around the world and is another excellent example of how AR leads to discoverability of interesting content. Figure 3-11 shows nearby webcams and Figure 3-12 shows links I can access to watch live feeds. In 2010, Augmented Planet extended the Readers Choice Awards to cover third-party applications developed for Layar, junaio and Wikitude. Webcams.travel won the award for best third-party content.

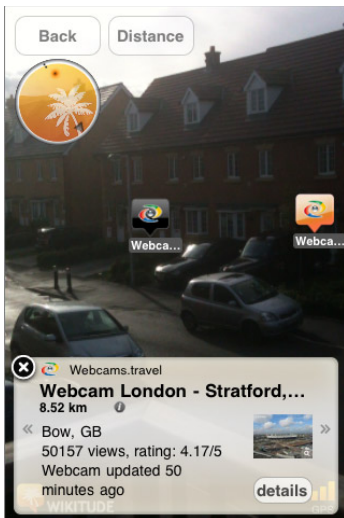


FIGURE 3-11: Webcams located near me

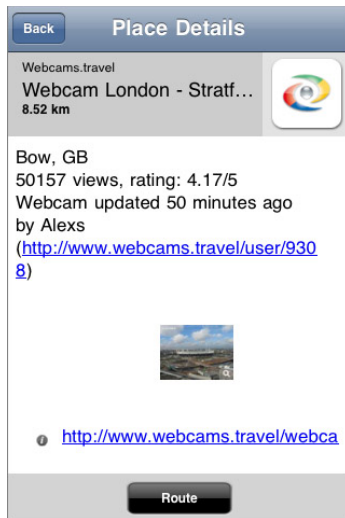


FIGURE 3-12: Link to watch the live feed

Qype

If you are looking for a reliable business or a good place to eat, the Qype World is a social networking site that contains reviews for thousands of local business around the world. Using this World, you can read real opinions of services near your current location. Figure 3-13 shows Qype reviews in an AR view. Figure 3-14 shows reviews of nearby businesses.



FIGURE 3-13: Qype reviews in the AR view

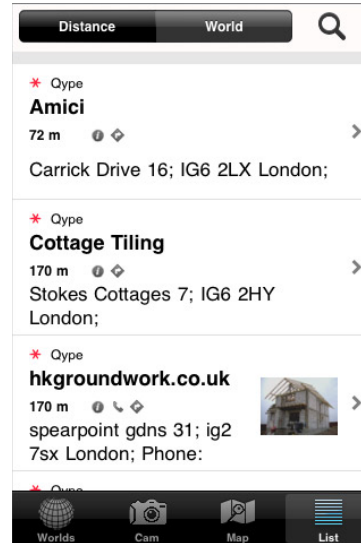


FIGURE 3-14: Nearby business reviews

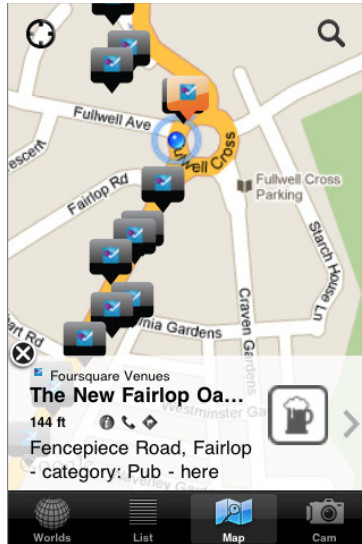
Foursquare Venues

Foursquare Venues (see Figure 3-15) is a social city guide with around eight million users worldwide. Users can find local venues and see where their friends are. Active users can earn discounts from their favorite places. Figure 3-15 shows several Foursquare venues near my home, including my local bar. Figure 3-16 shows nearby Foursquare locations on a map.



© 2011 FOURSQUARE

FIGURE 3-15: Foursquare check-in locations



© 2011 FOURSQUARE

FIGURE 3-16: Nearby Foursquare locations on a map

Google Local

Google Local (see Figure 3-17) can be used to search for local businesses and see the results in the AR view. This World is excellent for finding just about anything since it uses the Google search engine to provide results.

Flickr

Like YouTube, Flickr contains thousands of geo-coded images from around the world. Using the Flickr World opens your eyes to the amazing pictures that people are taking around your neighborhood. Figure 3-18 shows nearby Flickr content in a list view. Figure 3-19 shows Flickr content and its proximity to my neighborhood. (© 2011 Yahoo! Inc. FLICKR and the FLICKR logo are registered trademarks of Yahoo! Inc.)



REPRODUCED FROM GOOGLE™

FIGURE 3-17: Google Local search shown in the AR view

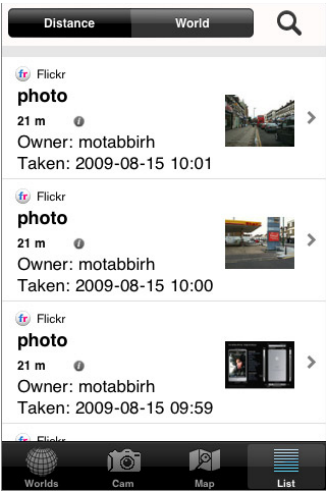
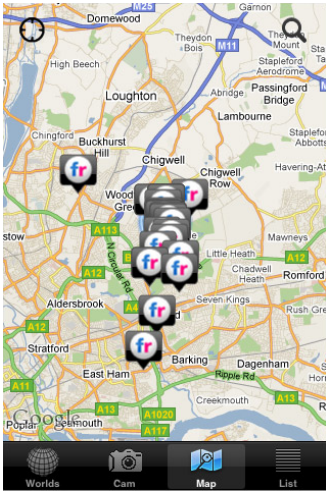


FIGURE 3-18: Nearby Flickr content shown in a list view



© 2011 GOOGLE, MAP DATA
© 2011 TELE ATLAS

FIGURE 3-19: Nearby Flickr from my neighborhood

Booking.com

Looking for a hotel? Booking.com probably has a hotel near you. Not only that, but it includes a review system so you'll be able to find a reasonable priced hotel within your budget, as well as hotels that other Booking.com users have recommend (or perhaps *not* recommended). Figure 3-20 shows nearby Booking.com content in the camera window. Figure 3-21 shows the large quantity of hotel reviews in the Central London area. Booking.com is an interesting example of how brands are adopting AR to promote their services. (© 1996–2011 Booking.com™. All rights reserved.)

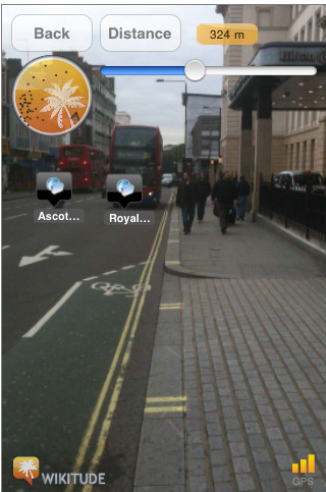
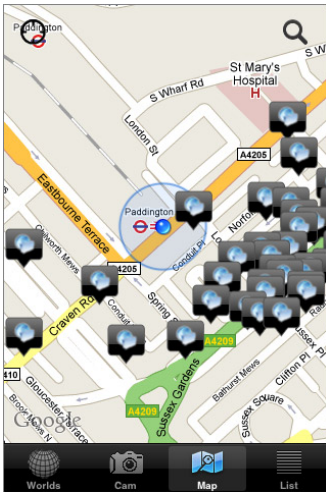


FIGURE 3-20: Hotels from Booking.com in Central London. AR View



© 2011 GOOGLE, MAP DATA
© 2011 TELE ATLAS

FIGURE 3-21: Booking.com hotels shown in a map view

Creating Your First World

The easiest way to create a World for Wikitude is by using the Wikitude.me web site. Since it requires no programming, go ahead and create your first World and get a taste of things to come:

1. Visit the Wikitude.me web site at www.wikitude.me and click the Goto Google Maps Interface button (see Figure 3-22).

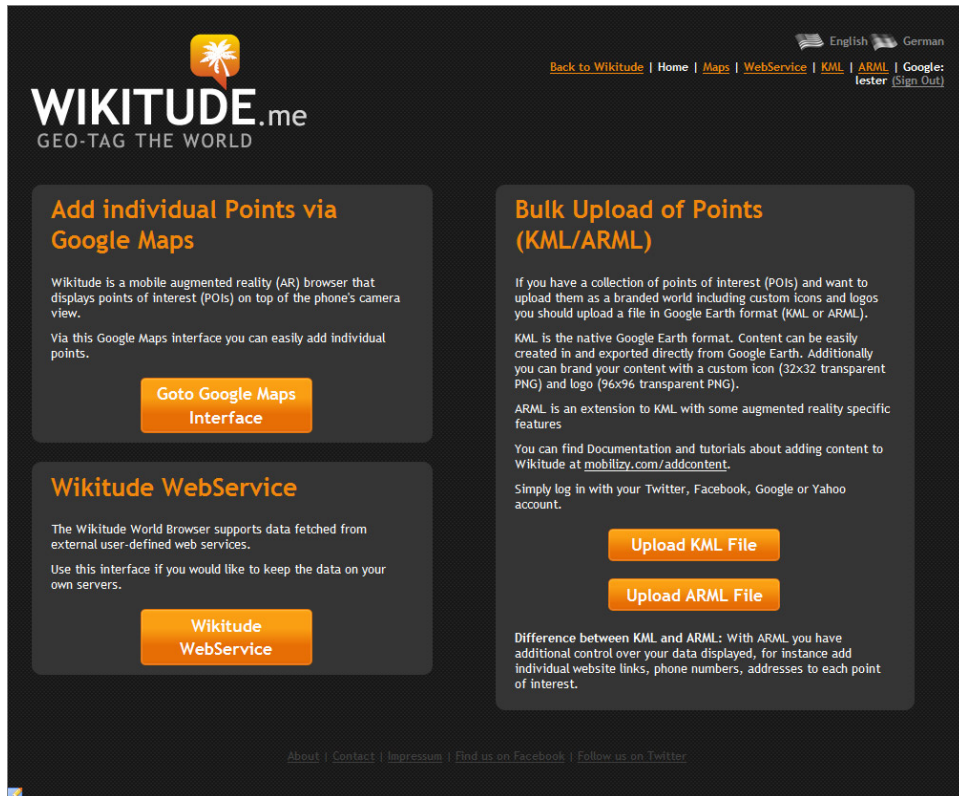


FIGURE 3-22: The Wikitude.me website

2. Sign in using either a Facebook, Twitter, Google, or Yahoo account.
3. In the Go to search box, type your home town and click the Go to button (see Figure 3-23).

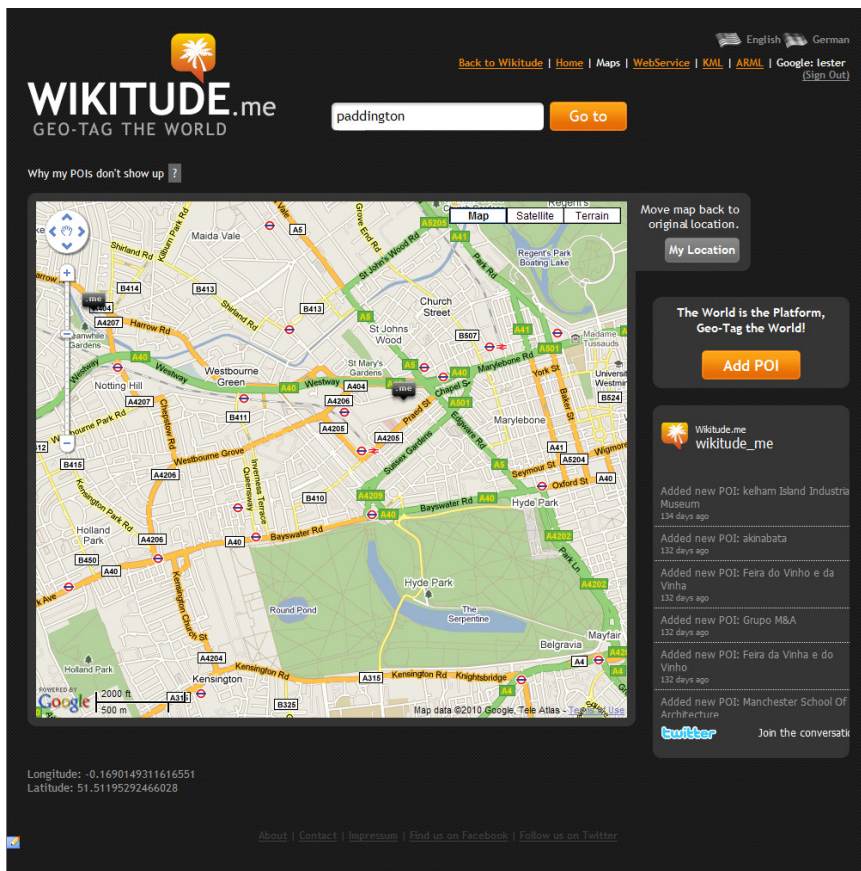


FIGURE 3-23: Searching via the Wikitude.me maps interface

4. On the Google Map that appears, click something nearby that interests you (for example, your local supermarket or bar). A Properties box will display on the right side.
5. Type your POI Title and a Description. You can also type a URL if you'd like. See Figure 3-24.
6. Click the Save button.

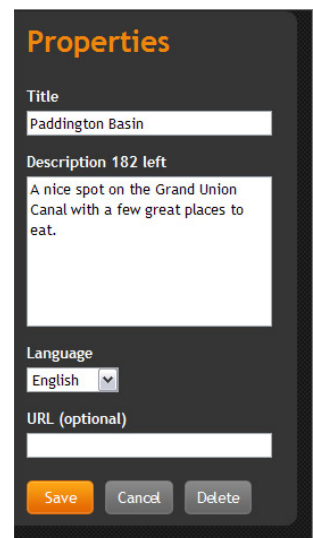


FIGURE 3-24: Creating the POI

7. Go to your Wikitude mobile application and load the Wikitude.me World. See Figure 3-25.

Congratulations, you have just created your first AR content. Of course, in later chapters, we'll look at how we can create our own Worlds with custom icons and content.

LAYAR REALITY BROWSER

Application name: Layar Reality Browser

Platforms: Android, iPhone

Developer: Layar

Home page: www.layar.com/

Layar was the first AR browser released, piping Wikitude to the Android platform in 2008. I always think of Layar and Wikitude as the Google and Apple of the AR world. Wikitude is akin to Apple, quietly doing good things. Layar, however, is the aggressor, looking to take over the world and be the dominant, platform-hungry company. And the approach appears to be successful. In only two years, Layar has produced some amazing stats: more than 1 million users, nearly 2,000 third-party layers created, and a thriving community building open-source tools to make developers' lives easier.

Overview

With the growth in content, Layar has undergone changes to help make it easier for users to find all the layers. If you haven't guessed already, layers are what Layar calls the content created by developers. Think of layers as being akin to Wikitude Worlds. Layers are sorted on a Nearby basis as well as Featured, Popular (what other users are using), and Categories. If that's not enough, you can save your favorite layers for easy access in the future. Figure 3-26 shows the Layar client (version 3.6) running on an iPhone.

A common problem with AR applications is they draw all the information bubbles on the same horizontal plane. If you have several POIs that are in the same general area, they can become difficult to select in the camera window, particularly if the



FIGURE 3-25: Viewing my first World

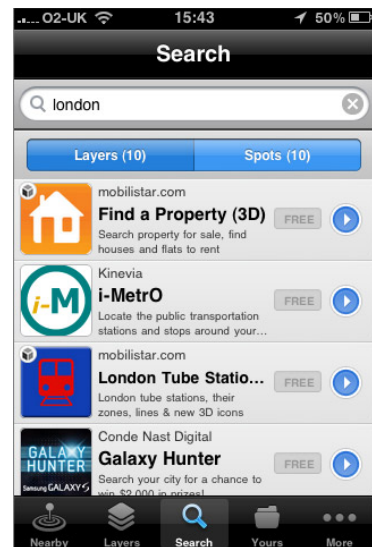
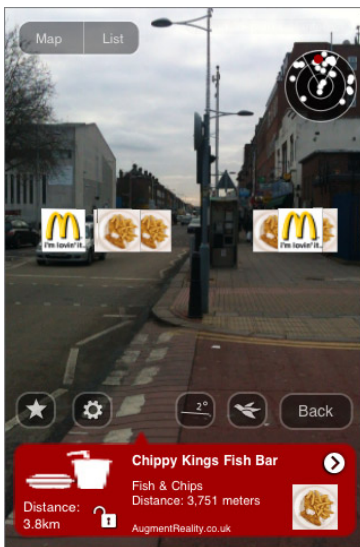


FIGURE 3-26: The Layar client

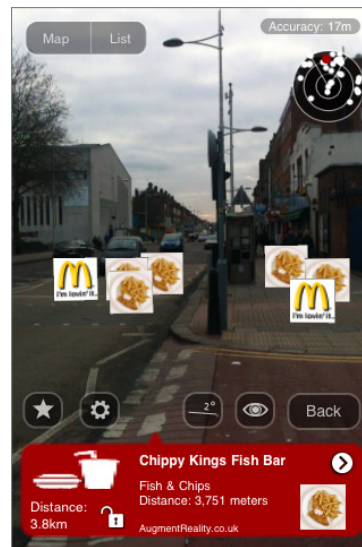
POI you are interested in is partially obscured. As shown in Figure 3-27, several POIs bubbles can become clustered together.

Layar has a really useful feature that provides a bird's-eye view of the horizontal plane so you can see a distance scale. As a result, those partly hidden POIs become visible. It's a rather nifty feature to help de-clutter the window and simplify selections. The same POIs from Figure 3-27 are shown in Figure 3-28, this time with the bird's-eye view enabled. Notice how the obscured POIs are now visible and can be selected.



AUGMENTEDREALITY.CO.UK

FIGURE 3-27: Layers on the same horizontal plane



AUGMENTEDREALITY.CO.UK

FIGURE 3-28: Layar's bird's-eye view

Development Choices

Developers building content for Layar build layers, aptly named to describe adding a layer over reality. Layar — in my view, at least — is more complicated to build content for than Wikitude because you are responsible for the hosting of the content. Layar will call your server with a request for the POIs so you will need to host a database which stores the relevant information. In turn, you will need to handle the incoming requests and return the right POI content back to the Layar client. It sounds more difficult than it is. We will go through the process step-by-step in the Layer development chapters. In this book, we will focus on building Layer content with MySQL and PHP.

Functionality

Layar pushes the boundaries with AR browsers by adding additional features to make content even more interesting. Some of the additional functionality includes:

- **3D:** Instead of simple icons, you can add 3D objects.
- **Proximity triggers:** Triggers define an action that will occur when the user comes in proximity to a location.
- **Audio:** POIs can have associated audio that can be combined with proximity triggers to play sounds when the user comes in range of a location.
- **Authentication:** Authentication enables you to authenticate the user with a username and password and have her login to interact with your layer.

With these features, developers are building a range of content, including AR games that require users to travel to nearby locations and interact with their surroundings using the camera.

Layers

Layar offers the usual mix of POIs that display content within proximity of your current location, but it pushes the boundaries with features such as proximity alerts. *Proximity alerts* enable developers to build content that triggers when the user comes in range of the POI. By combining proximity alerts with the ability to play audio, you can start to build sophisticated tour-guide solutions that play an audio tour of a tourist attraction when the user comes into range of the attraction. Layar also enables developers to charge for their content. Layar handles all the payments from users and works on a revenue-share basis, giving developers 60 percent of the profits from their work.

If you haven't done already, install Layar from either the Android Market or the Apple App Store and try out some of the numerous Layers that are available.

As you test Layar, be sure to click all the icons and:

- Test the search icon and search for nearby businesses.
- Click on POIs to explore the data in the info bubbles and the descriptions.
- Test the routing and calling functionality.

Here are some Layers that you may want to try:

- Woomba Mania
- Compass & Navigation AR
- i-MetrO
- The AR Beatles Tour
- Conquer the Game

- Maze – Labyrinthe
- Art is all around you

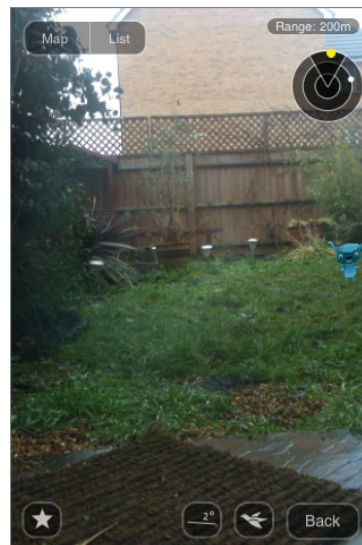
Woomba Mania

Woomba Mania is a proximity-based game where the player travels around to collect the escaped Woombas. Woombas come in different colors and have different point values. Figure 3-29 shows the points system for each type of Woomba. Expect a few surprises along the way, including rare Woombas as well as Chookies and Ghosts that need to be avoided. Once you have found a Woomba you'll need to travel to its location to collect it before it disappears. Figure 3-30 shows a blue Woomba lurking in my garden.



WWW.HOPPALA-AGENCY.COM

FIGURE 3-29: The Woomba splash screen

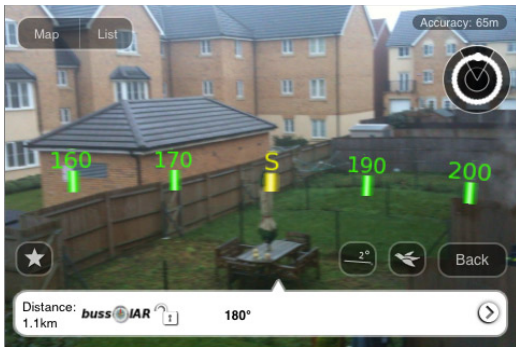


WWW.HOPPALA-AGENCY.COM

FIGURE 3-30: Finding the Woomba

Compass & Navigation AR

The Compass & Navigation AR layer displays an AR compass on the screen that can be used to indicate direction or display an orienteering arrow. While developed by an Italian developer, the application is fully localized in English and is great for anyone who needs an AR compass. This layer is a good example of providing AR without overlaying POIs. Figure 3-31 shows the compass in action. Figure 3-32 shows the integrated travel functionality.



© 2011 G-MAPS.IT

FIGURE 3-31: AR compass



© 2011 G-MAPS.IT

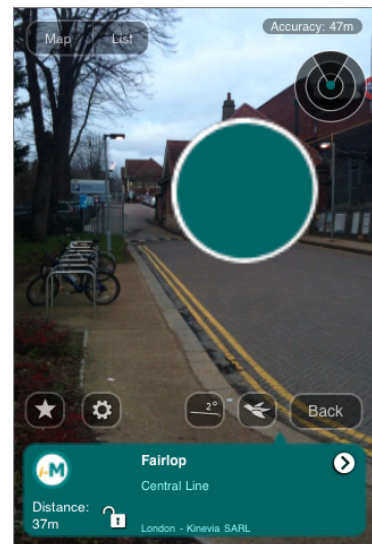
FIGURE 3-32: Navigation using AR

i-MetrO

There are many commercially available, standalone AR applications that will help you locate the nearest metro stations in your city. i-MetrO, however, can be used in Europe and the US to help travelers locate their nearest station (see Figure 3-33).

The AR Beatles Tour

If you're a fan of the Beatles, this Layer enables you to take a tour of famous Beatle locations in London. Highlights include the ability to have your photo taken with the Beatles on the famous Abbey Road crossing. While the layer is specific to the UK, it is a good layer that demonstrates how 3D can be blended with AR to provide engaging content. Figure 3-34 shows The Beatles on the famous Abbey Road crossing



© KINEVIA SARL 2008-2010 - ALL RIGHTS RESERVED

FIGURE 3-33: My nearest train station

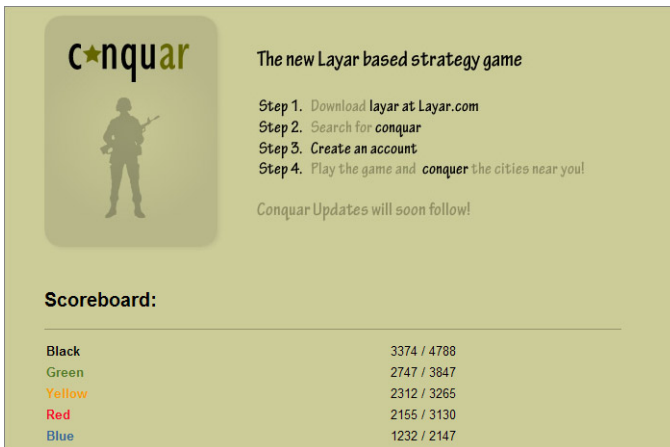


MOBILISTAR.COM

FIGURE 3-34: The Beatles in London

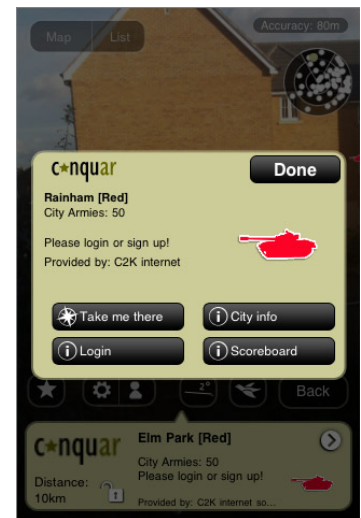
Conquer the Game

Think of capture the flag, only in AR terms. Sign up to the choice of armies and travel to the enemies' strong points to take over their territories. Don't forget to take the time to defend your team's territory by allocating some of your army to prevent it from being taken over. As shown in Figure 3-35, choose the team you want to join from the five teams available. Then you find the nearby territories available to invade, as shown in Figure 3-36.



COPYRIGHT C2K © 2011
ALL RIGHTS RESERVED

FIGURE 3-35: Choose your army



COPYRIGHT C2K © 2011
ALL RIGHTS RESERVED

FIGURE 3-36: Join the fight

Maze — Labyrinthe

Developed by a French developer but localized into English, this layer draws an entire maze in the camera window. Your task is simple: escape. You'll be doing a lot of walking, so this game is best played outdoors in an open area. This is an amazing example of what AR is all about. Figure 3-37 shows the immersive environment created by the Maze. The game feels almost like Doom; it's lacking only the monsters and power-ups hidden in the maze.



HPSC.FR

Art is all around you

Art is all around you is an interesting layer that displays famous works of art that you can interact with. The layer draws a famous piece of art with faces cut out. Users align the artwork with faces of their friends to include those faces in the photo. If you have ever wanted to put your own face on the Mona Lisa and redefine that famous enigmatic smile, then the Art is all around you layer gives you the opportunity to be creative with works of art. Figure 3-38 shows Grant Wood's American Gothic; all that's missing is two volunteers.

FIGURE 3-37: Can you escape the maze?



WWW.MUZAR.ORG

FIGURE 3-38: Create your own American Gothic Portrait

Creating Your First Layer

There are a number of ways to create Layer content, but for your first Layer, I'll show you how to use Message Central Layer developed by DnL Productions Inc.

1. Load the Layar client and locate the Message Central Layer (either by accessing the Feature items or by using Search).
2. Load the Message Central Layer.

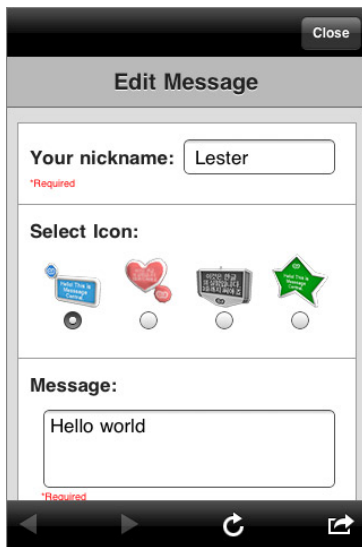
Depending on the range you have set, you'll probably see hundreds of messages from Layar users.

3. Restrict the range to just a few meters to avoid the clutter.
4. Return to the AR View and click the Create New Message info bubble. The Edit Message dialog box displays (see Figure 3-39).

If you have any difficulty locating the Create New Message bubble, just point your phone at the ground.

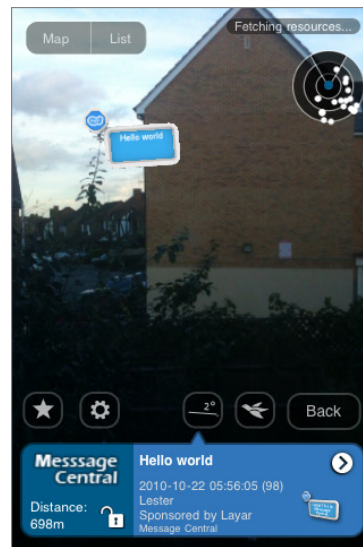
5. Type your nickname, select an icon, and type your message.
6. Press Create. (Remember, your message will be public, so be careful with what you say.)

Congratulations! You have just joined the Layar community. Figure 3-40 shows the hello world message.



WWW.DNLPRO.COM

FIGURE 3-39: Creating your first Layar POI



WWW.DNLPRO.COM

FIGURE 3-40: Viewing your message

As you've witnessed in this section, Layar is an incredibly rich environment. Without Layar, creating your own standalone application would require a lot of work on your part to approach anywhere near the functionality provided by Layar. Like Wikitude, Layar provides a nice level of integration that, for example, allows users to get turn-by-turn directions to the POI as well as access to relevant telephone numbers.

JUNAIO

Application name: junaio

Platforms: Android, iPhone

Developer: Metaio

Home page: www.junaio.com/

The junaio browser is the new kid on the block from Metaio, an AR heavyweight. If Layar and Wikitude are considered the Google and Apple of the AR world, Metaio is considered the Microsoft of the AR world. And the AR world should be on alert because the sleeping giant has awoken. junaio was released at the end of 2009, but unlike Layar and Wikitude, junaio was not released as an AR browser. It was released as the world's first AR social networking browser.

The original version of junaio focused on enabling users to post 3D objects to the virtual world and, in turn, other users could use these 3D scenes to create their own scenes to share with their friends. Since the release of junaio 2.0 in early 2010, junaio has become a fully fledged AR browser with around 150 Channels and another 650 in development.

While Layar and Mobilizy (Wikitude) are AR startups, Metaio is considered a pioneer in the AR world. Since 2003, it has created image recognition solutions and SDKs for developers. junaio has been built on its commercially available AR technology, which includes its natural feature tracking SDKs. junaio is the first AR browser that enables developers to build natural feature-tracking solutions.

Overview

With the switch in focus from a social browser to an AR browser, junaio's features have changed quite a bit. Despite these changes, it still keeps its roots with sharing content with friends, so it retains its social feel. Content in the junaio client can be found either by browsing nearby content, content your friends are viewing, or content that has been featured (staff picks). Content is restricted to what is nearby the user's location. Depending on your country or even city, you will see a list of content that is different than what appears in my client. Figure 3-41 shows the latest version of the client for the iPhone.

An exciting aspect of the client is the ability to browse the new nearby channels as they are published. This is a great way to find out what developers are doing without having to search. You can see new channels as they are published and you can be among the first to try them out (see Figure 3-42).

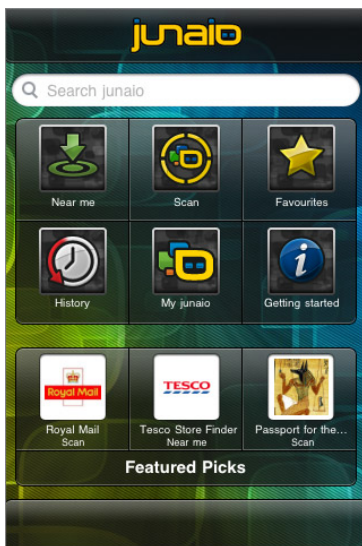


FIGURE 3-41: The junaio client

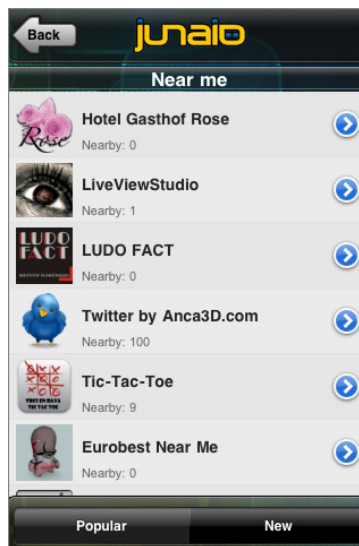


FIGURE 3-42: The new channels

Development Choices

As you might have already guessed, in the junaio universe, developers build channels. junaio presents developers with a rich set of features they can use when building content. Functionality includes:

- **Simple POIs:** As with Layar and Wikitude, junaio developers can create simple POIs that will appear in the camera window.
- **Sound and video:** Simple POIs can be enhanced with the ability to play video or audio.
- **3D and animation:** Objects can also be used in place of an icon. 3D objects can have animation actions.
- **Proximity detection:** Like with Layar, developers are able to determine when the user comes in range of a POI and take appropriate action.
- **Natural feature tracking:** junaio is the only AR browser currently available to offer developers the functionality to recognize images and either display an overlaid 3D object or play video.
- **Latitude, Longitude, Altitude (LLA):** Since GPS is not always accurate (particularly in indoor situations), LLA consists of special markers which, when scanned with the camera, position the user at a known location.

Channels

As you experiment with junaio you'll notice just how well an integrated experience junaio provides for its users. There are very few instances where the users find themselves outside the junaio client. In both Layar and Wikitude, there are occasions where control is passed outside the AR browser to another application (such as a web page). A user might find himself in a state of limbo; he might have finished an action and wants to return to the browser, but there is no easy return route. junaio provides a very tightly integrated environment whereby everything is managed by junaio.

As you test out the junaio channels, feel free to add me (LesterMadden) to your friends channel by clicking Profile > Friends > Add Friends.

If you haven't done already, install junaio from either the Android Market or the Apple App Store and try out some of the numerous available channels. You should also create an account; this account will be your developer account later.

As you test junaio, be sure to click all the icons and:

- Test the search icon and search for nearby businesses.
- Click on POIs to explore the data in the info bubbles and the descriptions.
- Test any video and audio functionality that you find.

Here are some channels that you might want to try. Bear in mind, however, that the many of the channels might not have content for your city. For this reason, I have listed only a few channels that should be visible to readers worldwide. As you explore the junaio client, be sure to explore local channels that you may have access to.

archINFORM

archINFORM overlays a map in the camera window to show you the road layout near your location. As you move your phone, the map will update to reflect the direction you are facing. This is incredibly useful for navigating because you can see the map in relation to your location. The archINFORM channel also contains a catalog of 3D buildings which will be drawn on the map if they located near your location. As shown in Figure 3-43, the combination of a map and an AR view should make it almost impossible to be lost.

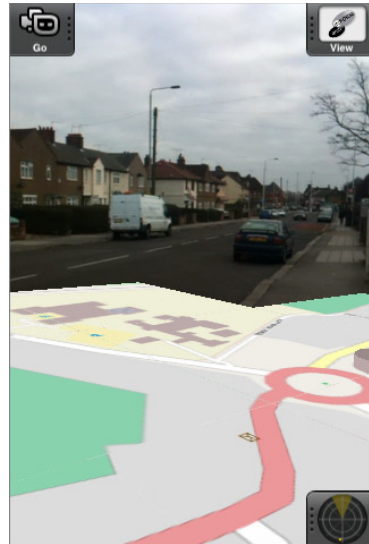


FIGURE 3-43: AR map combined with live camera feed

Wikipedia

No AR browser platform is complete without accessing content from Wikipedia. This channel, developed by Frank Angermann from Metaio, displays interesting content from the Wikitude database. In Figure 3-44, I have used junaio to provide information about my local train station. Notice how rich the information is.

WorkSnug

WorkSnug (see Figure 3-45) is aimed at helping users find nearby places they can use to work. Each POI is assessed for amenities (for example, Wi-Fi access) as well as how quiet it is. WorkSnug is also available as a standalone AR application for the iPhone. If you're looking for a place to complete your work, WorkSnug can help.



FIGURE 3-44: junaio Wikipedia channel

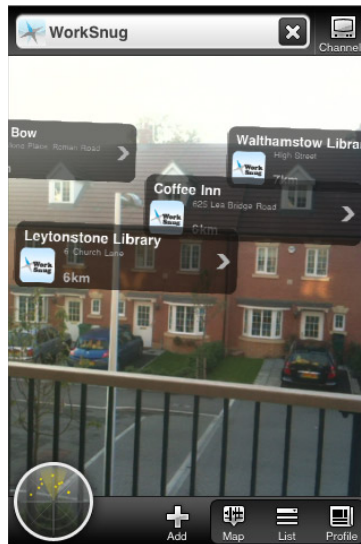


FIGURE 3-45: Places to work nearby

Testing a junaio Demo

As you tested junaio, you likely found some interesting content and you're probably eager to try natural-feature tracking. Follow these steps to test the junaio GLUE demo, which we will build in a later chapter.

1. Load junaio and use the search bar to search for the key-word GLUE.
2. In the search results, select the junaio GLUE demo.

The client gives you an indication that it is downloading resources (these resources will be the 3D objects being downloaded to your mobile device).

3. When the download has completed point your phone at the picture shown in Figure 3-46 to see a 3D character.
4. Interact with the 3D junaio Man character by clicking on the image that appears on your mobile device screen. This triggers an animation.



FIGURE 3-46: Interact with junaio Man

As you've witnessed in this section, junaio offers a great integrated experience for the user. There are very few instances where you were taken outside the junaio client to complete an action. I'm sure you'll also agree that the natural feature-tracking demo for junaio Man opens up a world of possibilities. The junaio Man example is quite simple, but you can build more compelling solutions, including those that bring printed advertisements to life. For example, if you were building a marketing campaign for a car manufacturer, you could use GLUE to recognize a printed image of a car and present your users with a 3D experience.

The only downside with GLUE right now is its tendency to display channels that don't have any relevant content for your locale. However, with another 650 channels in development, we'll soon have more relevant content at our fingertips. (Hopefully, that includes your content!)

BROWSER ACCURACY

Most of our experience with GPS is via the use of navigation software, which we assume is 100 percent accurate and therefore we know exactly where we are. If we are driving down the highway and there is a problem with the signal, the software takes into account our last known speed, the direction we were traveling, and the road we were traveling on. Assuming we don't make any turns and deviate from the programmed route, when the GPS device loses the signal or the signal becomes unreliable, the software can estimate where we're likely to be until a reliable location fix can be obtained. This all happens seamlessly and provides us with tremendous confidence that we're located exactly where the GPS says we're located. With AR browsers, users also expect the GPS to be 100 percent accurate and to have the information bubble displayed directly over the target object. Figure 3-47 shows what users expect with AR POI placement.

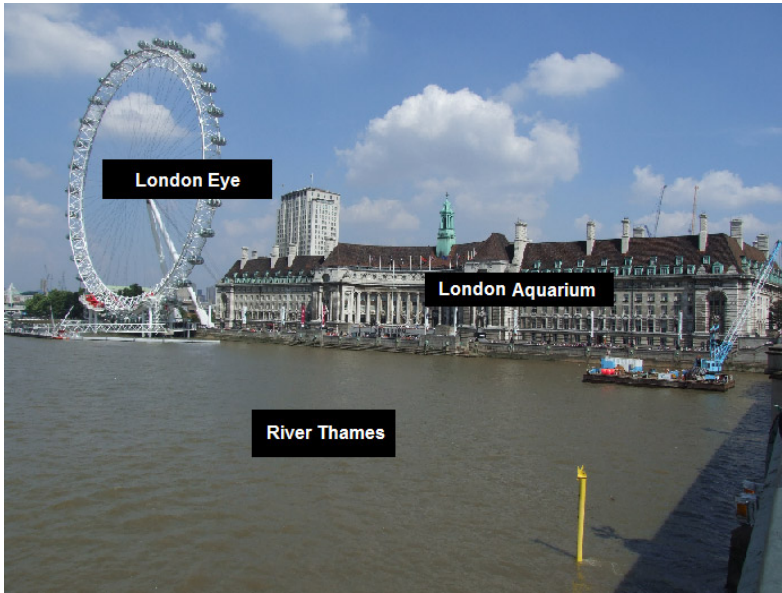


FIGURE 3-47: Users expect 100 percent GPS accuracy with AR browsers

What they often get, however, is something that resembles the user experience shown in Figure 3-48.



FIGURE 3-48: AR browsers often fail to meet those user expectations

GPS and Compass Accuracy

When you try to explain that GPS accuracy varies from a few feet to over a mile, users are often surprised. However, GPS accuracy is affected by a number of factors, including interference from nearby sources, atmospheric conditions, and even buildings. Therefore, your location and even the weather conditions will affect where the GPS thinks you are located. Of course, if the GPS position of your current location is incorrect, it will affect how the POIs are drawn in relation to your position. As the saying goes, “garbage in, garbage out.”

It might come as a surprise to learn that the accuracy of the compass on mobile devices can differ between models and can be further affected by objects that are placed in close proximity. In a simple experiment, I put my iPhone 3GS and my wife’s iPhone 4G in the same location, both aligned in exactly the same direction, yet they produced different results (see Figure 3-49).



FIGURE 3-49: An iPhone compass experiment

In this test, there was no interference — at least, no deliberate interference. The difference is purely a result of the differences in hardware. I repeated the experiment by placing the devices back in the same location, only this time touching each other. Figure 3-50 shows that when the devices are in contact with each other, the compass direction changes.



FIGURE 3-50: iPhone compass experiment with interference

Of course, any Boy Scout will advise you to never use a compass around a metallic source, so when I placed the two iPhones in close proximity to one another, the metal in each phone created interference that produced conflicting data.

Mapping Accuracy

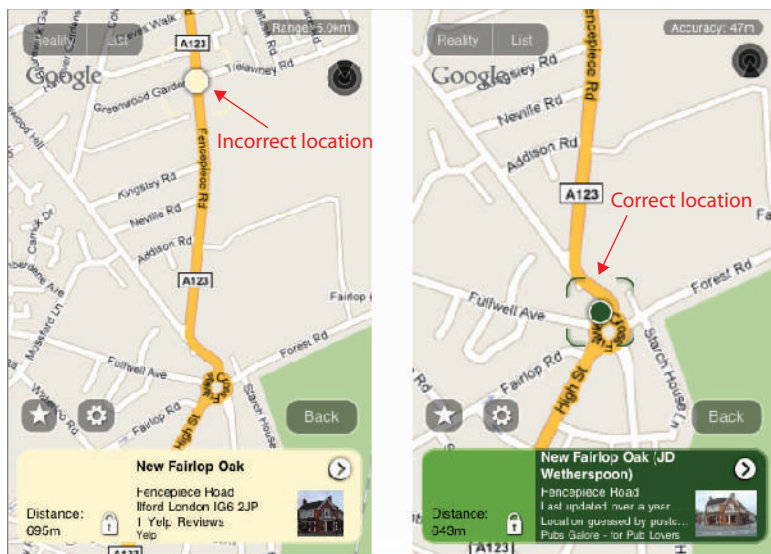
As you have learned in this chapter, GPS and compass locations are not always accurate, so we can't rely on the locations of our POIs being 100 percent accurate.

They also might be inaccurate as a result of a third-party database that hasn't been updated. For example, a business might have moved or closed and the database has not been updated to reflect that. In some cases, information might be inaccurate because the database uses a company's registered address rather than company's trading address. I live in a quiet part of London, within a 15-minute walk to the shops. But when I look at some data sources, it mistakenly shows an Indian restaurant just a few yards from my door.

Another problem is that some databases simply have the location of the POI incorrectly recorded. As shown in Figure 3-51, my local pub is shown in two different locations. In the left image, my local pub is shown using the Yelp layer for Layer on my iPhone. In this case, the problem is not caused by hardware; I would obtain the same result using an Android device. Notice how the POI for the New Fairlop Oak is positioned at the top of the map near Greenwood Gardens. In the image on the right,

the same pub is positioned correctly by the Pubs Galore layer for Layar and located by Fullwell Avenue.

In this example, it is easy to see that the data sources used by both layers are different. Again, “garbage in, garbage out.”



© 2011 GOOGLE, MAP DATA

© 2011 TELE ATLAS

FIGURE 3-51: Example of inaccurate POIs

So why does all this matter? So you’ll better understand the results you will see and also understand why the information bubbles will never be perfectly aligned with the target. Fortunately for you, however, by building for a platform like junaio, Layar, or Wikitude, the platform providers must implement the clever routines to better take advantage of the phones hardware. Your part is simply ensuring that the underlying mapping data is correct, and if you plan to license third-party data, test it first to see how accurate it really is.



Compare Qype and Wikipedia across the three browsers. Load the clients and make a note of the inconsistencies in the content that each shows you on the map around your location.

SUMMARY

In this chapter, you learned a little about the three AR browsers that you will build content for later in this book. Not only did you see points of interest in the camera window, you had your first taste of natural-feature tracking. Each platform offers vastly different functionality, and by exploring what some other developers have built, you are likely beginning to consider which platform you prefer.

The chapter also enlightened you to browser accuracy. You learned how hardware plays a large part in accuracy errors. Results are often dependent upon the weather, the location, or even the device. You also learned that not all data sources are equal; an accurate database with the accurate location of POIs is fundamental to providing a good user experience.

4

Latitude, Longitude, and Where to get POIs

WHAT'S IN THIS CHAPTER?

- How to read latitude and longitude coordinates
- How to convert degrees, minutes, and seconds to decimal
- Where to get latitude and longitude coordinates
- Resources to consider for POI harvesting

When you build AR browsers, you must provide geo-coded coordinates to the browser in the format of latitude and longitude (optionally, you can also include altitude). If you're like me, you were looking out the window during that particular lesson in geography. In this chapter, we'll have a quick review of how the latitude/longitude coordinate system works. In addition, you'll also learn of some resources where you can obtain some interesting POIs which you may use later when you build your real browsers. Before we begin, you'll need a calculator as well as a laptop running a web browser and Google Earth (optional).



In later chapters, you will use the geo-coded coordinates that you generate here.

An Overview of Latitude/Longitude

You probably have seen latitude/longitude written in either in degrees (using the degrees symbol) or written in decimal. The following examples point to the same location, which happens to be New York City:

- Degrees, hours, minutes: 40° 43' 0" N, 74° 0' 0" W
- Decimal: 40.716667, -74.0

When you build your AR browsers, you must put locations in the decimal format. Since many maps use the degrees, hours, minutes notation, you need to understand how the latitude/longitude system works and how it's converted to decimal.

Longitude

Longitude, also known as *meridians*, represents the vertical lines that are often shown on maps of the globe. These meridian lines extend from pole to pole, with the zero degree (0°) line — known as the *prime meridian*, extending through Greenwich in England. To the west of the prime meridian, the longitude extends to 180° W; to the east of the prime meridian, the line extends to 180° E. This produces a total of 360 meridian lines and 360 degrees.

The Earth's circumference measured at the equator is approximately 24,901 miles, so you can divide 24,901 by 360 to get approximately 69 miles, which is roughly how far apart the longitude lines are at the equator (see Figure 4-1).

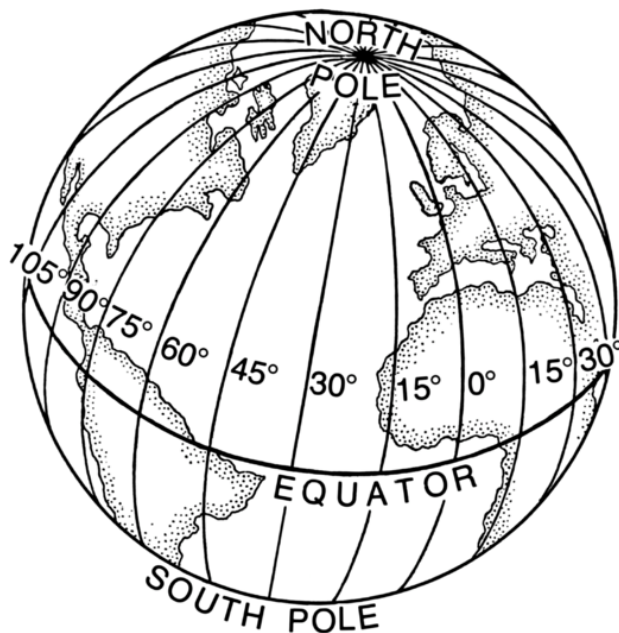


FIGURE 4-1: Longitude lines extend from pole to pole

Latitude

Latitude, also known as *parallels*, represents the lines that extend horizontally around the globe; they are equally spaced at around 69 miles between each parallel (see Figure 4-2). Everything North of the equator (0°) is represented as degrees north, with the North Pole located at 90° N and the South Pole located at 90° S. Taking into account 69 miles for every degree traveled, $180^\circ \times 69$ puts the distance from pole to pole at around 12,420 miles. These are approximations because the Earth is not perfectly round; it's an oblate spheroid, which means it bulges in the center.

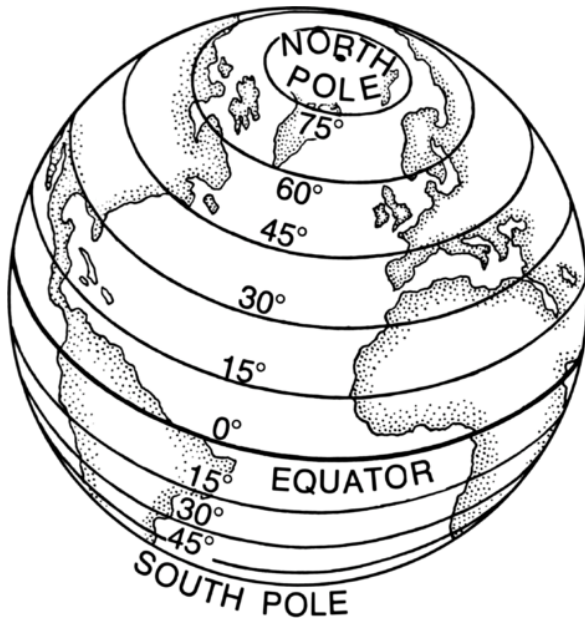


FIGURE 4-2: Latitude lines circle the globe

Okay, you now know what the longitude/latitude lines on a map represent. If you have a location of 74° W, you know that it will be west of the meridian line. If it's 40° N, it's North of the equator. If you look for 40° N, 74° W on the map shown in Figure 4-3 you should find the coordinates give us New York.

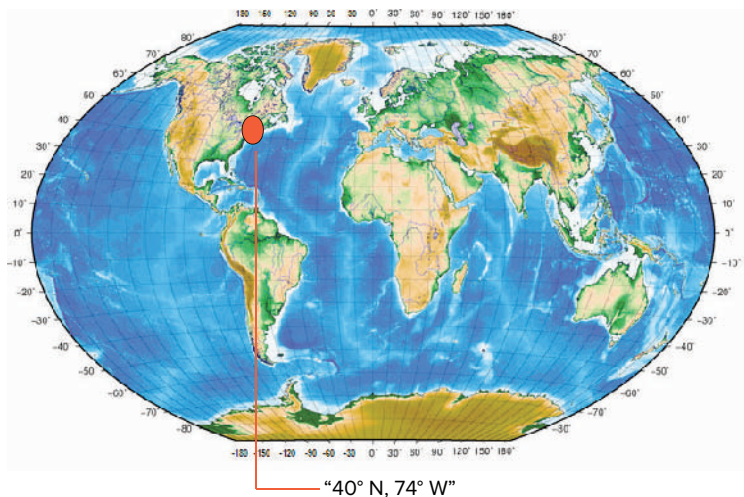


FIGURE 4-3: Typical map showing degrees

Minutes and Seconds

Sixty-nine miles is too big of an area to be of much help when navigating. Imagine arriving at JFK Airport in New York and the only details you have about your hotel is it is within a 69 mile radius of the airport! To precisely locate an object, you need to subdivide both latitude and longitude into smaller units.

Each degree therefore is made up of 60 minutes, with each minute divided into tens or hundreds of seconds, depending on how precise you need to be. If your hotel is located at Times Square in New York City, the latitude and longitude coordinates are $40^{\circ} 45' 26.16''$ N, $73^{\circ} 59' 9.02''$ W

- 40 degrees, 45 minutes, and 26.16 seconds north of the equator
- 73 degrees, 59 minutes, and 9.02 seconds west of Greenwich in England.

Decimal Notation

We know that computers prefer numbers, so working with the coordinates $40^{\circ} 45' 26.16''$ N, $73^{\circ} 59' 9.02''$ W causes untold misery. So the coordinates are converted into their decimal form:

$40.757267, -73.985839$

With the decimal notation, positive latitude values represent north of the equator (zero degrees latitude) and positive longitude values represent east of the Greenwich meridian (zero degrees longitude).

Negative values represent the southern hemisphere (or west of the Greenwich meridian).

Converting to Decimal

Converting the location of our Times Square hotel from degrees, minutes and seconds ($40^{\circ} 45' 26.16''$ N, $73^{\circ} 59' 9.02''$ W) to decimal is easy: Just convert the seconds into minutes and then add the value to the existing minutes. Then convert the total minutes into decimal.

This is a lot easier to explain with a breakdown of the calculation:

1. The first degree digit value always stays the same, so 40° will always be 40 whether expressed in degrees or decimal format. Notice that our latitude has the N notation, indicating that it is north of the equator, so it is a positive number:

40

2. The remaining coordinates are minutes and seconds ($45' 26.16''$). Before we can add the two values they need to be of the same type (for example, both written as minutes). To find the number of minutes, divide the seconds (26.16) by 60 (the number of seconds in a minute):

$$26.16 / 60 = 0.436$$

3. Now that you have converted the seconds into minutes, you can add the newly calculated minute value (0.436) to the minutes (45) in the coordinates to get the total minutes:

$$45 + 0.436 = 45.436$$

4. Divide the total minutes you just calculated (45.436) by 60 again to get the total minutes expressed in decimal notation:

$$45.436 / 60 = 0.7572$$

5. Combine the first degree (40) with the results of the calculation (0.7572) to get the latitude:

$$40 + 0.7572 = 40.7572$$

The longitude is calculated in the same way.

1. The first degree remains the same, but because it is west of the prime meridian, it's a negative number:

$$-73$$

2. Divide the seconds (9.02) by 60 to convert the seconds into minutes:

$$9.02 / 60 = 0.153$$

3. Add the result (0.153) to the minutes (59) to get the total minutes:

$$59 + 0.153 = 59.153$$

4. Divide the total minutes (59.153) by 60 to get the decimal value of minutes:

$$59.153 / 60 = 0.9858$$

5. Combine the first degree (-73) with the result of the calculation (0.9858) to get the longitude:

$$-73 + 0.9858 = -73.9858$$

In decimal form, you would then record this location as:

$$40.7572, -73.9858$$



In most cases, when you discover geo-coded POIs you can expect the coordinates to be written as latitude followed by longitude. However, there is no convention that says they have to be written this way. If your POIs start appearing several thousand miles away from where you expected, you might have reversed the coordinates.

Working with Points of Interest (POIs)

If you were starting to think that building AR browsers required you to huddle over maps like Christopher Columbus attempting to navigate the globe, don't fret. Fortunately, there are a number of ways to generate coordinates without this type of effort. For the remainder of this chapter, you

will learn some of the tools that will give you latitude/longitude locations in decimal format at the click of a button. The sites you will look at are:

- Google Maps
- Google Earth
- Wikipedia

Google Maps

Google Maps is one of my favorite places to harvest latitude and longitude coordinates. Google provides excellent search facilities and one-click access to the coordinates of your chosen location. Even better, Google Maps is web-based, so you don't need to install it and it's quick to use.

To obtain the latitude and longitude from Google Maps, follow these simple steps.

1. Visit <http://maps.google.com> and type your street and city into the search box.
2. Right-click the target and select What's here? from the menu.
3. You'll notice that the search box has been replaced with the latitude and longitude of the destination (see Figure 4-4).

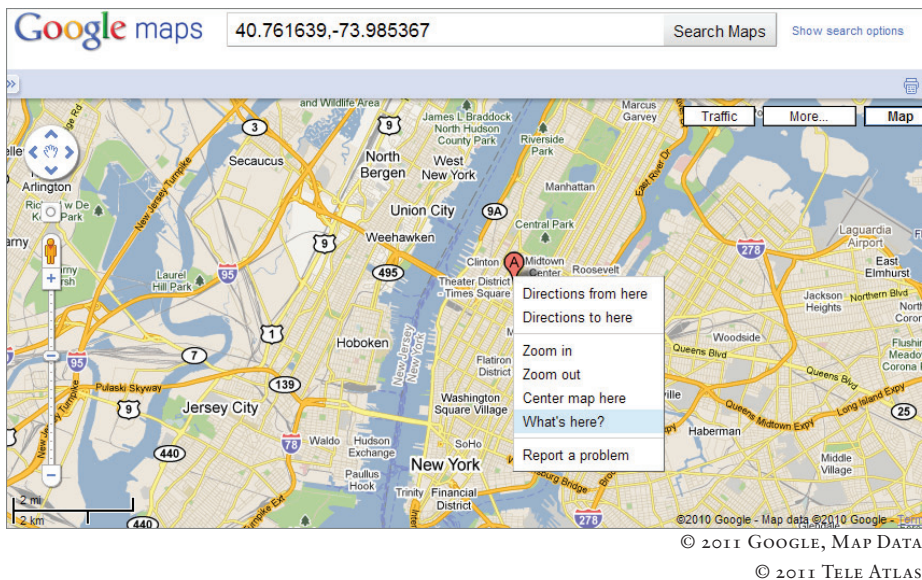


FIGURE 4-4: The latitude and longitude coordinates for an address shown in Google Maps



You will need some data points for the chapters when we get started with building content, so it's an excellent time to begin harvesting a few locations near your home (or where ever you'll be when you start writing the code). Find useful targets, such as a school, a restaurant, or a bank. Grab ten locations at various distances and add them to the form below for safe-keeping until you start writing code later in the book).

My POIs

NO	LATITUDE	LONGITUDE	DESCRIPTION
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

Google Earth

Google Earth is another great resource for finding coordinates, though you'll need to install the client from www.google.com/earth/index.html. Google Earth contains a huge amount of user-generated content. If you are looking for the nearby gas stations or hardware stores, chances are that data will be available in the Google Earth client. A lot of locations in Google Earth also contain rich descriptions (for example, a school might have a description tag that describes when the school was built and what subjects are taught). If you are going to use any data other than the latitude and longitude, you should read the terms and conditions page to ensure you are complying with the allowed usage.

To use Google Earth to harvest latitude and longitude coordinates, follow these steps:

1. Load the Google Earth client from www.google.com/earth/index.html
2. Search for a location.
3. Right-click the location and choose Copy.
4. Load Notepad and paste the results.

You should see an XML file that looks like Listing 4-1.



LISTING 4-1: Sample Google Earth XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
      xmlns:gx="http://www.google.com/kml/ext/2.2"
      xmlns:kml="http://www.opengis.net/kml/2.2"
```

Available for
download on
Wrox.com

continues

LISTING 4-1 (continued)

```

        xmlns:atom="http://www.w3.org/2005/Atom">
<Document>
  <name>KmlFile</name>
  <Style id="transit57h">
    <IconStyle>
      <scale>1.1</scale>
      <Icon>
        <href>http://kh.google.com:80/flatfile?
lf-0-icons/uk-london-metro_tiny.png</href>
        <gx:w>32</gx:w>
        <gx:h>32</gx:h>
      </Icon>
    </IconStyle>
    <LabelStyle>
  </LabelStyle>
    <BalloonStyle>
      <text>${description}</text>
    </BalloonStyle>
    <LineStyle>
      <color>00000000</color>
    </LineStyle>
    <PolyStyle>
      <color>00000000</color>
    </PolyStyle>
  </Style>
  <StyleMap id="1048858">
    <Pair>
      <key>normal</key>
      <styleUrl>#transit57n</styleUrl>
    </Pair>
    <Pair>
      <key>highlight</key>
      <styleUrl>#transit57h</styleUrl>
    </Pair>
  </StyleMap>
  <Style id="transit57n">
    <IconStyle>
      <Icon>
        <href>
          http://kh.google.com:80/flatfile?lf-0-icons/
          uk-london-metro_tiny.png</href>
        <gx:w>32</gx:w>
        <gx:h>32</gx:h>
      </Icon>
    </IconStyle>
    <LabelStyle>
      <scale>0</scale>
    </LabelStyle>
    <BalloonStyle>
      <text>${description}</text>
    </BalloonStyle>
    <LineStyle>

```



```

        <color>00000000</color>
    </LineStyle>
    <PolyStyle>
        <color>00000000</color>
    </PolyStyle>
</Style>
<Placemark>
    <name>Mansion House</name>
    <description><![CDATA[<table
cellspacing="6" cellpadding="0"><tr><td><b><font size="+2"><a id="title"
href="http://maps.google.com/maps/place?
ftid=0x487604aa7ecafcb:0xdc334cff2f
3bf3bb&q=type%3Atransit_station%3A%22Mansion+House%22" target="_blank"
style="color:#0000cc" name="Mansion House">Mansion
House</a></font></b></td></tr></table><table cellspacing="6"
cellpadding="4"><tr><td align="left" bgcolor="#ffce00" title="Circle"><font
size="+1" color="#000099"><b>Circle</b></font><td align="left"
bgcolor="#007229" title="District"><font size="+1"
color="#ffffff"><b>District</b></font></td></tr></table><table
cellspacing="6" cellpadding="0" width="100%"><tr><td id="last"><a
href="http://www.transportdirect.info/" target="_blank"><font
color="#008000">transportdirect.info</font></a><div style="background:none
repeat scroll 0 0 #E8ECF9;height:2.1em;margin:10px 0 10px;padding:4px 10px
0;color:#0000cc;min-width:200px"><a id="more_info"
href="http://maps.google.com/maps/place?ftid=0x487604aa7ecafcb:0xdc334cff2f
3bf3bb&q=type%3Atransit_station%3A%22Mansion+House%22" target="_blank">View
more information &raquo;</a></div></td></tr></table><script
type="text/javascript" src="http://mw1.google.com/mw-earth-
vectordb/scripts/placepage/transit_l110n.js"></script>]]></description>

    <styleUrl>#1048858</styleUrl>
    <Point>
        <coordinates>-0.0941871,51.5120201,0</coordinates>
    </Point>
</Placemark>
</Document>
</kml>

```



Your XML may vary greatly depending on the location you selected when you performed the copy action.

This is a form of XML used by Google Earth called *Keyhole Markup Language (KML)*. In Chapter 5, “Building with KML,” you will learn how to use the KML directly in your Worlds. For now, try this:

1. Locate the `<coordinates>` tag in the KML you just created.
2. Copy the coordinates and paste them into the Google Earth search box.

If you don't have Google Earth installed or didn't follow the previous steps, type 0.0941871,51.5120201 into Google Maps.

Did Google Earth or Google Maps take you to the location that you were expecting? I'm guessing that you ended up miles from where you expected. If you used my coordinates, you were probably looking at a stretch of ocean off the coast of Somalia and not the Tower Bridge in London, which is where I generated the POI. So why did that happen?

I always thought coordinates were written in the order of latitude first, then longitude. But it turns out there is no universal convention for how the coordinates are presented. With KML, the coordinates are expressed as longitude first, then latitude. If you are using Google Earth to generate your coordinates, keep in mind that you will need to reverse the order of the parameters if you want to use them with your applications.

Wikipedia

Wikipedia is undoubtedly one of the greatest resources on the internet, covering everything from famous people, technology, as well as cities and famous structures. Wikipedia not only contains rich descriptions of locations but the physical latitude/longitude coordinates as well.

To use Wikipedia for the source of your POIs, follow these simple steps:

1. Visit www.wikipedia.org.
2. Search for Tower Bridge London.
3. Select the first option from the list.

On the right side of the screen, you will see the geo-code location expressed as degrees (see Figure 4-5). Don't panic, no calculator is needed!

4. Click the degrees link.

The screenshot shows the Wikipedia article for "Tower Bridge". At the top right, there is a search bar and navigation links. Below the article title, the coordinates "Coordinates: [51°30'27N 0°10'32W](#)" are displayed and circled in red. A red arrow labeled "Geo-code info" points to this circled area. The article text describes the bridge as a combined bascule and suspension bridge in London, England, over the River Thames. It also includes a table with details such as "Garrage", "Crosses", "Locale", and "Maintained by".

FIGURE 4-5: Wikipedia displaying geo-code data

5. On the page that appears, you will see the decimal notation of the latitude/longitude (see Figure 4-6).



FIGURE 4-6: Wikipedia geo-coded options

Although Wikipedia is a great source for harvesting POIs, your chosen POI will have to be something Wikipedia considers worthy of being entered into the Wikipedia database. So if you're looking for the coordinates of a Starbucks on Liverpool Street, you won't likely find it in Wikipedia.

Working with POI Databases

Say you're building a collection of POIs for your AR browser that displays the locations of all the train stations in London. You could spend the day gathering the data from the various sources noted in the previous section or you could explore the many websites that offer access to the POIs of train stations, restaurants, businesses, and much more.

There are many sources of content that you may want to use and below is a list of just a few that you might find useful. Your challenge will be to wrap the data in the various databases in to a format that matches the XML structure required by your browsers.

For the purpose of this book we will focus on geo-coding a few locations around your immediate area and you will not be required to sign up, download, or even visit any of the links listed below. Think of these links, while not exhaustive, as a useful starting point for gathering content in the future as you seek to build real AR browser applications.



You should always read the relevant terms and conditions before using any of these services.

Accessing the Wikipedia Database

Wikipedia has over three million English-language articles; wouldn't it be amazing if you could programmatically access all that data? The good news is that you can. If you prefer, you can also download the entire database and search for the content you are specifically looking for, but bear in mind that three million-plus records is a lot of data.

Read the usage and how-to instructions at http://en.wikipedia.org/wiki/Wikipedia:Database_download. You will find details on how to:

- Download the database in English and other languages
- Import the database into MySQL
- Understand the SQL and XML schemas used for the data dump

If downloading more than three million articles of data is more information than you really need, there is a community effort called DBpedia that enables developers to extract information from the Wikipedia database.

DBpedia (<http://dbpedia.org>) is a community effort to extract structured information from Wikipedia and make this information available on the Web. DBpedia allows developers to perform sophisticated queries against Wikipedia and link other data sets on the Web to Wikipedia data.

Accessing the GeoNames Database

The GeoNames (www.geonames.org) geographical database covers all countries and contains over eight million place names that are available for free download.

Accessing the CityGrid Database

CityGrid Media (<http://developer.citygridmedia.com>) is a local search provider that connects users with the local businesses around them. Local searches enable you to build solutions where users can free-type what they are looking for and you return the best generated results. For example, a user could type pizza and you could return the pizza restaurants nearest to the user's location. CityGrid's local search covers more than 18 million businesses from around the world. Users can search an extensive database of listings, ratings, and reviews, as well as share their opinions about places and services in their neighborhoods

Accessing the Yelp Database

Yelp (www.yelp.com/developers) is a popular review web site that covers the USA and the UK. With Yelp, users can review restaurants, bars, hotels, local services and just about everything in between. Yelp's API provides you with access to the data and the ability to:

- Retrieve business reviews and rating information for a particular geographic region or location.
- Display review information for a particular business.
- Determine accurate neighborhood name information for a particular location.
- Track recent reviews for a particular business.
- Display pictures of highly rated local businesses and of the top reviewers for that business.
- Determine a particular business' review and rating information based on the phone number for that business.

Accessing the Zvents Database

AR is a great tool for finding out what is happening around you. *Zvents* (http://corporate.zvents.com/products/mobile_api.html) provides programmatic access the world of entertainment and what is taking place at nearby venues. If you are looking to build AR content that will inform users about cinema, theater, or concerts, this is a good starting point.

Accessing the Foursquare Database

Foursquare (www.foursquare.com/apps) is a new way to explore your city. With more than eight million users worldwide, Foursquare is a friend-finder, a social city guide, and a game that challenges users to experience new things. *Foursquare* lets users “check in” to a place when they’re there, tell friends where they are, and track the history of where they’ve been and who they’ve been there with. The Foursquare API provides you with the ability to build applications that interact with the Foursquare platform. You can use the API to create new ways to check-in to Foursquare or visualize the data generated by the foursquare community.

Accessing the Hoovers Database

While *Hoovers* (<http://developer.hoovers.com>) specializes in providing company information for CRM systems, they do have a database of 65 million US company records which are accessible via their API. Their data may useful for niche applications.

Accessing the Yahoo Database

The latest version of *Yahoo’s Local Search Web Service* (<http://developer.yahoo.com/search/local/V3/localSearch.html>) allows you to search the Internet for businesses near a specified location and now returns both the latitude and longitude and Yahoo! user ratings of the establishment as well as search-by-business categories.

Accessing the Trulia Database

AR can make searching for real estate fun. If you are looking to provide information about houses and apartments for sale, or perhaps an AR solution that provides trends and local information for house buyers, *Trulia* (<http://developer.trulia.com/>) could be a good location for you.

SUMMARY

In this chapter, you learned how to read the latitude and longitude lines on a map — if degrees, hours and minutes were a mystery to you before, you’re now an expert at map reading. You learned how to convert the degrees, minutes, and seconds format to a decimal notation that can be used with your AR browsers. You also learned how to use Google Maps to generate geo-coded coordinates that you will use when you build content for junaio, Layar, and Wikitude. Finally, you learned about several resources that might be of use to you when you’re looking for richer sources of geo-coded data.

PART II

Wikitude

- ▶ **CHAPTER 5:** Building Worlds with KML
- ▶ **CHAPTER 6:** Building Worlds with ARML

5

Building Worlds with KML

WHAT'S IN THIS CHAPTER?

- How to use the Wikitude Dashboard
- How to create your first Wikitude World using KML
- How to test your World on the iPhone and Android
- How to obtain your Wikitude beta key
- How to simulate your GPS location

In this chapter, you will kick-start your AR development with the Wikitude World browser from Mobilizy. As you go through the chapters in this book, you may wonder about the order in which they appear. You first learn about Wikitude, then Layar, and then junaio. I have chosen Wikitude as your first venture into AR because it is undoubtedly the easiest of the three platforms to create content for. The Wikitude browser may lack the 3D capabilities of Layar and the natural-feature tracking functionality of junaio, but it is consistently a user favorite. It has been voted the Augmented Planet Readers Choice best Augmented Reality Browser for 2009 and 2010.

Not only is Wikitude easy for you to develop content for, you don't need a server to upload your POIs and you don't need a lot of developer experience in order to create content. Worlds are simply a structured XML document that contains the name, description, and location of the POI.

Before you begin, you'll need the following:

- Latitude and longitude coordinates of nearby locations (generated from the previous chapter).
- Some familiarity with XML.
- The Wikitude client installed on your mobile device.
- An XML editor, such as Microsoft's XML Notepad (<http://tinyurl.com/msxmlnotepad>).

USING THE WIKITUDE DASHBOARD

To begin creating Wikitude Worlds, visit www.wikitude.me. This Wikitude.me web site is where you add content for the browser.

On the Wikitude.me page, click the Upload KML File button and log-in using your Twitter, Facebook, Google, or Yahoo account information. (Be sure you use the account where you want to host all your Wikitude Worlds.) Note the account details are used only to authenticate you; you are not granting Wikitude the ability to post Facebook or Twitter messages on your behalf.

After you log-in, the Wikitude Dashboard displays (see Figure 5-1). This is where you can configure your Worlds.

The screenshot shows the Wikitude Dashboard interface. At the top left is the Wikitude.me logo with the tagline "GEO-TAG THE WORLD". In the top right corner, there are language selection options for English and German, and a navigation menu with links: Back to Wikitude, Home, Maps, WebService, KML, ARML, Google, and a Logout button.

The main content area is divided into two sections. The left section, titled "Add", contains a form for creating a new World. The form fields are:

- Title (required) with a help icon (?)
- Description (required) with a help icon (?)
- Tags with a help icon (?)
- Provider URI with a help icon (?)
- Author with a help icon (?)
- E-Mail (required)
- Language with a dropdown menu
- World Status section containing a Status dropdown menu set to "Public"
- Icon (required) with a "Browse..." button and a "Preview (NOTE: changes will only be visible after save)" label
- Logo with a "Browse..." button and a "Preview (NOTE: changes will only be visible after save)" label
- KML/KMZ file with a "Browse..." button and an "Enter KML URL" field

 Below the form is a section for "Wikitude Content Interface Terms of Use" with a checkbox indicating that the user has read and accepted the terms. A "Save" button is located at the bottom of the form.

The right section, titled "My KML/KMZ", shows a list of the user's worlds, currently containing one entry: "My First Wikitude World" with a "(Delete)" link next to it.

FIGURE 5-1: The Wikitude Dashboard

The Dashboard is restricted to just one page with no complex submenus or arrays of options to configure; it's just one extremely simple page to master. Before you can complete the page, however, you'll need to have a KML file with your POI content.

DEVELOPING WITH KML

Wikitude developers have the option to develop Worlds using a standard form of XML known as Keyhole Markup Language (KML), which is widely used by Earth browsers (such as Google Earth, Google Maps, and many other mapping products). If you have Google Earth installed, you can create Wikitude Worlds without writing a single line of XML or without any knowledge of how XML works. Content can be created directly via the Google Earth UI and then uploaded to the Wikitude server.

For developers, Listing 5-1 shows KML generated by Google Earth for my local train station.



Available for
download on
Wrox.com

LISTING 5-1 KML example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
xmlns:gx="http://www.google.com/kml/ext/2.2"
xmlns:kml="http://www.opengis.net/kml/2.2"
xmlns:atom="http://www.w3.org/2005/Atom">
<Placemark>
  <name>Barkingside tube station</name>
  <address>Barkingside tube station, Ilford, IG6, UK</address>
  <Snippet maxLines="2"><![CDATA[<br/>]]></Snippet>
  <LookAt>
    <longitude>0.0886</longitude>
    <latitude>51.5848</latitude>
    <altitude>0</altitude>
    <heading>0</heading>
    <tilt>0</tilt>
    <range>1000</range>
  </LookAt>
  <StyleMap>
    <Pair>
      <key>normal</key>
      <Style>
        <IconStyle>
          <Icon
            <href>http://maps.gstatic.com/intl/en_uk/mapfiles
              /kml/pal3/icon60.png</href>
          </Icon>
        </IconStyle>
      </Style>
    </Pair>
    <Pair>
      <key>highlight</key>
      <Style>
```

continues

LISTING 5-1 (continued)

```

        <IconStyle>
            <scale>1.3</scale>
        <Icon>
            <href>http://maps.gstatic.com/intl/en_uk/mapfiles
                /kml/pal3/icon52.png</href>
        </Icon>
    </IconStyle>
    </Style>
</Pair>
</StyleMap>
<Point>
    <coordinates>0.0886,51.5848,0</coordinates>
</Point>
</Placemark>
</kml>

```

If you don't want to install Google Earth, you can use the KML in Listing 5-1. Either download it or type it in to your XML tool of choice. Be sure that you change the following tags to reflect a longitude and latitude location near you. You can also download the icon and logo images used in this example.

```

<longitude>0.0886</longitude>
<latitude>51.5848</latitude>
<coordinates>0.0886,51.5848,0</coordinates>

```

As we discussed in Chapter 4, there is no standard that states whether latitude should appear before longitude or vice versa. If you consider the following coordinates and use them in a product such as Google Maps, the POI will be positioned somewhere in the Indian Ocean between Somalia and the Seychelles — rather than in East London, which is the correct location.

```

<coordinates>0.0886,51.5848,0</coordinates>

```

When using KML, you express the coordinates as longitude / latitude.

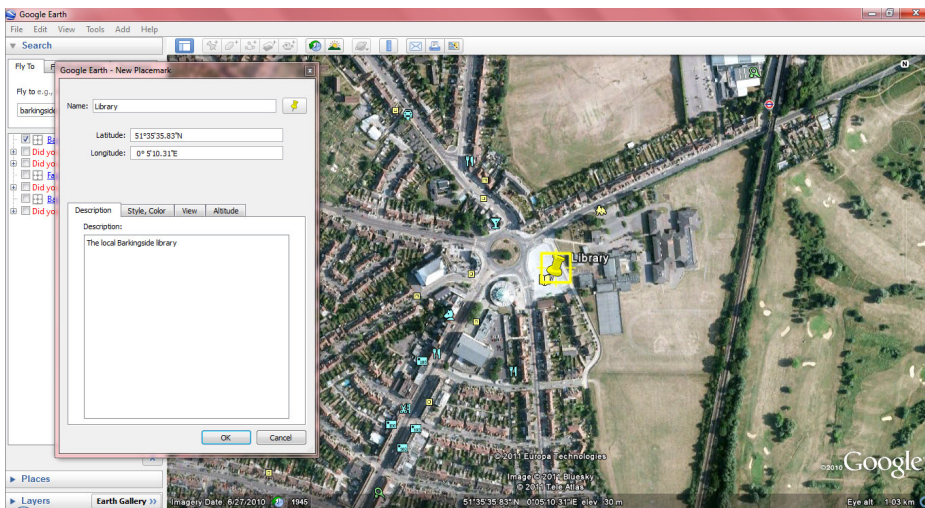
Creating KML With Google Earth (For Non Developers)

The easiest way to create KML — particularly if you have multiple POIs that you want to include with your World — is by using the Google Earth interface. If you have Google Earth installed, search for your home address. Once you have found it on the toolbar, you will notice a yellow pin (see Figure 5-2). This pin enables you to mark a location for your KML and, more importantly, enter a relevant description that will appear in the Wikitude client. This ultimately saves you the trouble of manually editing the XML in an editor at a later date.



FIGURE 5-2: Use the yellow pin (located on the far left) to mark a location for your KML

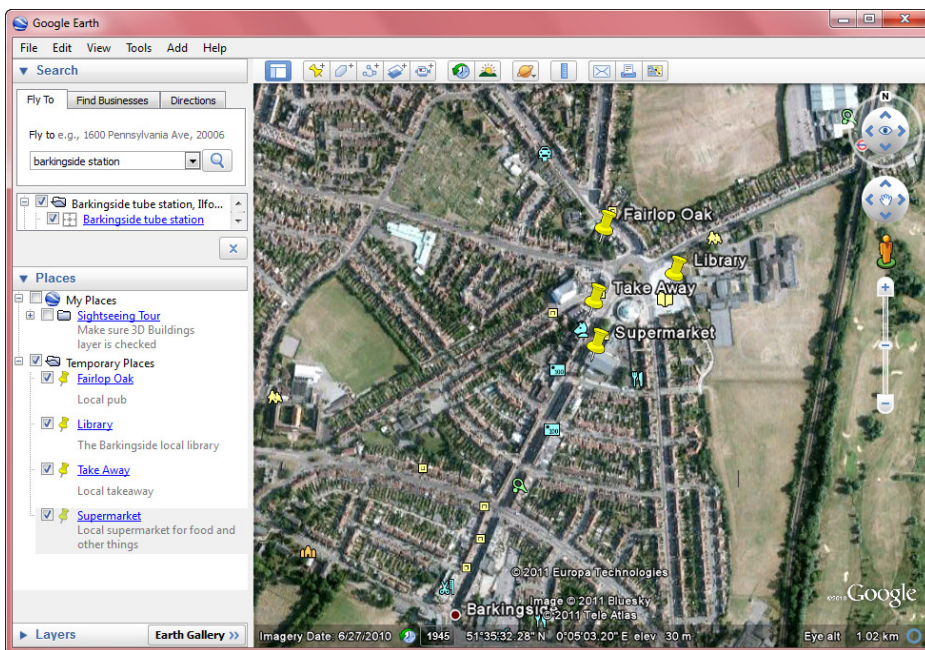
Clicking the yellow pin displays the New Placemark dialog box (Figure 5-3) that enables you to enter a name and description. On the Altitude tab, leave the setting at Absolute, which is the only setting currently supported by Wikitude. Selecting an unsupported option may result in unexpected behavior.



©2011 GOOGLE, MAP DATA ©2011 EUROPA TECHNOLOGIES IMAGE ©2011BLUESKY ©2011 TELE ATLAS

FIGURE 5-3: The New Placemark dialog box

In the sample KML shown in Figure 5-4, I have used the Google Earth application to add several POIs for my home city. You should take this opportunity to do the same using the Google Earth application.



©2011 GOOGLE, MAP DATA ©2011 EUROPA TECHNOLOGIES IMAGE ©2011BLUESKY ©2011 TELE ATLAS

FIGURE 5-4: Google Earth POIs

Once you have completed a few POIs, save the results directly to the KML format. In the left pane, the Places area displays a list of temporary places (see Figure 5-5) that you have created. Ensure that all your temporary places are selected, right-click the Temporary Places heading, select Save Places As. Ensure you select the type KML and not the default KMZ format.

Now that you now have a valid KML file, you are ready to create your first World.

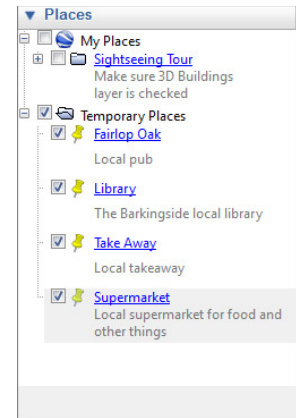


FIGURE 5-5: Temporary places

Creating a World

Return to the Wikitude dashboard (refer to Figure 5-1) to add a new World. If you have closed your web browser, you will need to revisit www.wikitude.me, log-in, and select the Upload KML option. To create the World, you will need to input a description along with some other details about the World. This data is added via the short form displayed in the Wikitude Dashboard. The form is simple, but let's step through the options anyway.

Title

The Title is the name of your World as it will appear in the Wikitude client. You should try to keep to less than 15 characters; anything more than that will be truncated.

Description

The Description also appears in the client. This is your opportunity to provide a rich description that tells potential users why they should try your World.

Tags

Tags are comma-separated lists of search words that will be used when users search the Wikitude client for content. If, for example, your World provides the location of pizza restaurants, you want to use tags such as pizza, restaurant, fast food, take away, delivery, Italian, snacks, as well as any others you would like to use. The more relevant your tags, the more likely you'll be discovered.

Provider URL

The documentation states that the Provider URL appears at the bottom-left of the screen while the application is in the AR view. In the current version, however, a Wikitude logo is shown and this Provider URL has no effect. However, you should include your Provider URL for future use. If activated, the Provider URL is the developer's home page or the home page of the World.

Author

This is where you type the name of the author who created the content. This appears in the client, so it should reflect your company name or the name you want to attribute the World to.

E-Mail

The E-Mail entry does not appear in the client but is required should Mobilizy need to contact you regarding your World.

Language

The Language option is where you specify the language for your World.

World Status

The status of Worlds can be Testing or Public. Public indicates that the World is live and can be viewed by any user. Worlds that are in testing can be viewed only by users who have entered the developer key (also known as a Beta-World Key) on their device. Unless the key is entered, it is not shown in the mobile client.

Under the World Status heading you will see the developer key related to the current world. Note that the developer key is generated and appears only after you have completed the entire form and pressed the Submit button.

Icon

The icon should be a 32×32 PNG file that represents the POI. This icon will be shown in the AR camera window when your POIs are visible.

Logo

Typically the logo will be your company logo or the logo associated with your World. The required size is 96×96 and in the format of a PNG file.

KML/KMZ File

The KML file will be the file you exported from Google Earth and saved in the previous step. Equally, the KML file could have been created from scratch, or downloaded. You can upload the KML file from your hard disk or if your KML file is located on your web server, you can specify a URL. Notice, however, that regardless of where the file is stored, the KML file you specify is copied to the Wikitude server where it will be hosted. If you upload a KML file that was located on your own web server and then make changes to the file, you must re-upload the file to the Wikitude server for the new version to come into effect. To do this, log-in to the Wikitude Dashboard and select your World from the list and make the relevant change and save the World. Any changes you have made will then be reflected in the client.

In the KML/KMZ section, you also have the option to view the source KML or test the World on Google Maps to visually see where the POIs are located. This is a useful step that enables you to quickly test your World in a web UI without testing on a live device. It can help you troubleshoot reversed coordinates problems or help you build and test Worlds for locations where you are not present. For example, you can build a KML World that contains the POIs of all the subway stations in Tokyo and test the World by using the Google Maps option directly from the Wikitude Dashboard.

You should complete the form with your details and upload the KML for your World. Figure 5-6 shows that I have completed the Wikitude Dashboard form to create my World.

Edit: My First Wikitude World

[Show data on Google Maps](#)

Title (required) ?
My First Wikitude World

Description (required) ?
This is my first Wikitude World that I will create using XML

Tags ?
testing

Provider Url ?
<http://www.augmentedplanet.com>


Author ?
Lester Madden


E-Mail (required)
lester@augmentedplanet.com

Language ?
English

World Status
Status ?
Testing

Developer Key ?
pbyccpy

Icon (required) ?
Browse...

Preview (NOTE: changes will only be visible after save)

Logo ?
Browse...
Delete

Preview (NOTE: changes will only be visible after save)

KML/KMZ file ?
Browse...
Enter KML URL

Wikitude Content Interface Terms of Use
These Wikitude Content Interface Terms of Use ("Wikitude Content Interface Terms") is an agreement between you (either an individual person or a single legal entity, who will be referred to in these Content Interface Terms as ("You"), and Mobilizy GmbH, Ginzkeyplatz 11, 5020 Salzburg/Austria, as ("Mobilizy").
 I have read the above-mentioned Terms and Conditions for uploading data to the Wikitude system and accept them.

Save Cancel Delete

FIGURE 5-6: A sample Wikitude listing

Figure 5-7 shows how the completed listing including the description and logo are displayed in the Wikitude client.

When I used Google Earth to create the KML file, I targeted a nearby train station as my source POI. Of course, when you used Google Earth to create your KML file, you used POIs from the map — so your results will differ. Figure 5-8 shows how my POI and icon are presented in the AR view.



FIGURE 5-7: Wikitude World client listing

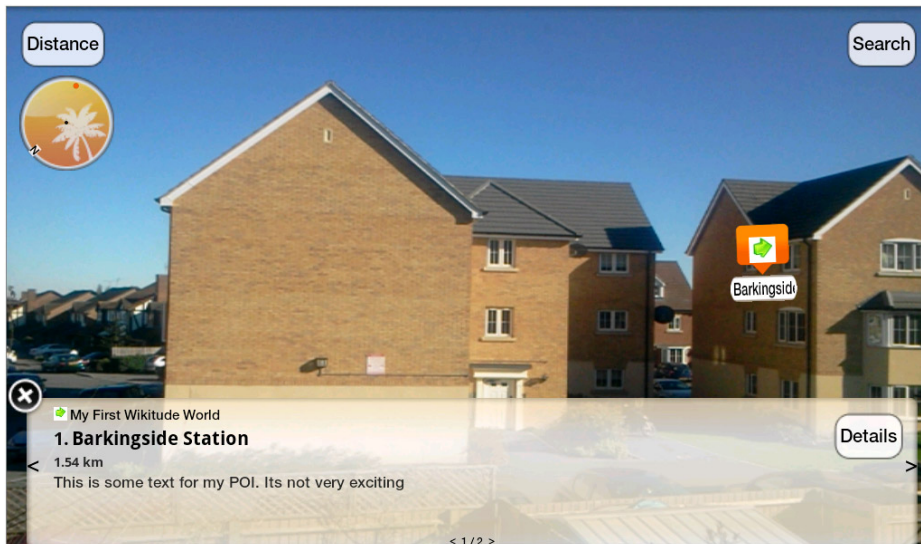


FIGURE 5-8: Wikitude World AR view

Testing

Now that you have created your World, you will need to test it on the client. Since there is no way to log-in on the Wikitude client, you need to enter the World's Beta Key directly to either your iPhone or your Android application. The process differs per device.

The Beta-World Key is shown in the Wikitude dashboard. To obtain the key, you will need to edit your World by selecting it from the list of Worlds shown on the right of the Wikitude Dashboard. Once you select the World, the developer key will be shown under the World Status section of your listing.

Android

Android owners can enter the Beta-World Key by loading the Wikitude client and pressing the Menu button. This displays a POI Settings page (see Figure 5-9).

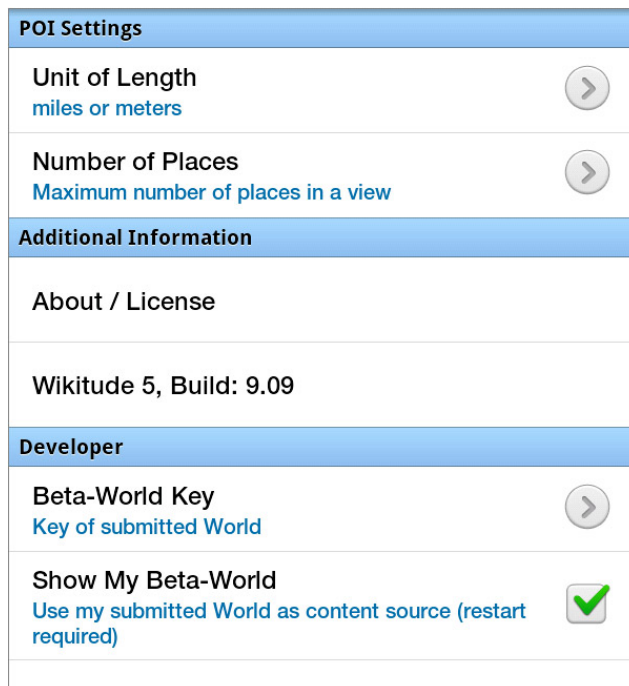


FIGURE 5-9: Android developer options

There are two steps you need to take here:

1. Make sure the Show My Beta-World option is checked.

This toggle enables you to manage the displaying of beta Worlds in the client. This is necessary because when you enable the beta Worlds option, all the other Wikitude Worlds will be hidden.

2. Enter your key by pressing the Beta-World Key menu item.

This displays the Beta-World Key dialog box (shown in Figure 5-10) used to enter your key.

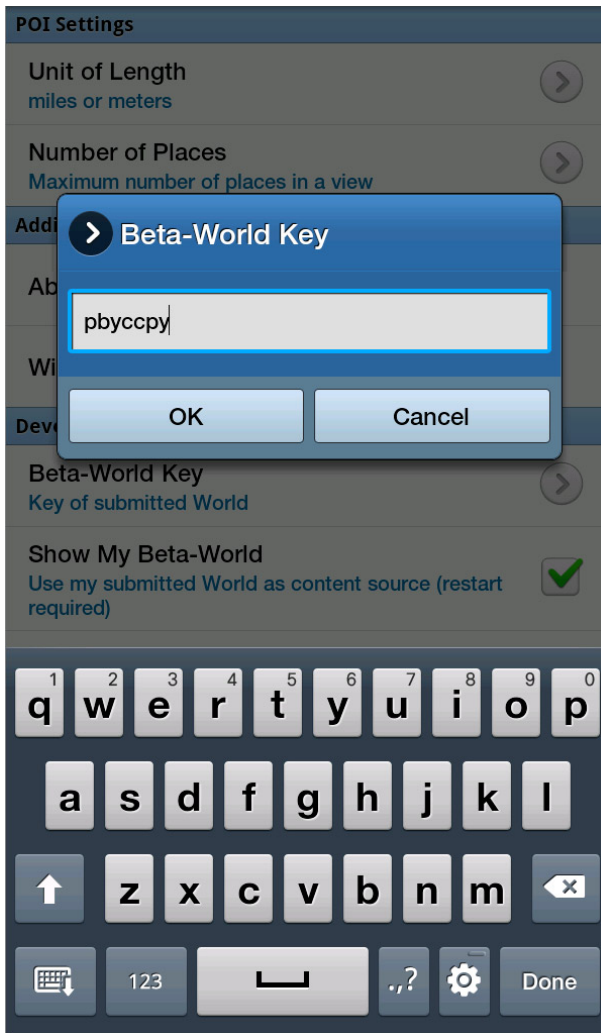


FIGURE 5-10: Android developer key

You will need to restart Wikitude. Once you reload Wikitude, you will see that most of the Worlds have been removed and your World will be displayed.

iPhone

For iPhone owners who want to test beta Worlds, follow these steps:

1. Open the iPhone's Settings application from the iPhone's home screen and select Wikitude, which displays the Wikitude Settings screen (see Figure 5-11).
2. Select the Content Developer Settings menu, which will display the Developer Settings options shown in Figure 5-12.

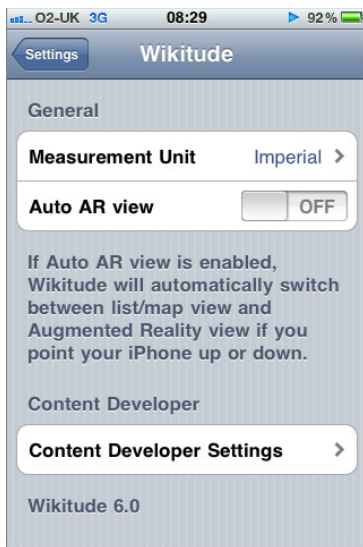


FIGURE 5-11: Wikitude settings



FIGURE 5-12: Developer Settings options

3. In the Developer Key section, type your Beta-World Key and ensure that the Show My Beta Worlds toggle is set to On.

This toggle enables you to quickly enable or disable the displaying of beta Worlds. When the option is On, all other Worlds are hidden in the client, so the toggle enables you to quickly switch between using Wikitude and testing your content.

Additionally, the iPhone enables you to simulate your location so you can test Worlds as if you were in a different location. For example, if you are based in the United States but you are building a World for the London Olympics, you can create the KML for London and then simulate your location to test the World as if you are in London.

4. Exit Wikitude and restart the Wikitude client.

When you reload the client, an incorrect key will display Unknown World in the list of available Worlds. If you receive this error, simply check that you have entered the key correctly.

If you do not see the beta World displayed, try ending the Wikitude process by double-pressing the iPhone's menu button then tapping and holding the Wikitude icon until the x appears. Then delete it.

Android owners should use a utility that enables the manual ending of processes. Following these steps, you should now be able to test your sample World in the client.

Simulating Locations

On an iPhone, you can use Wikitude to simulate your location. You can easily test this functionality by using the KML in Listing 5-1 and leaving the longitude and latitude coordinates as they are. These coordinates represent a train station in Barkingside, which is my neighborhood. To place yourself in Barkingside and test the POI, follow these steps:

1. Upload the KML from Listing 5-1 to the Wikitude Dashboard.
2. Type your developer key into the client.
3. Set the latitude to 51.585247 and the longitude to 0.088273.
4. Turn the Simulate Location setting to On (see Figure 5-13).

Once you restart the Wikitude client and select the World, you will be placed in Barkingside and you should see the POI displayed.

Remember to disable the Simulate Location setting once you have finished to ensure that the GPS correctly picks up your real location; if you don't disable it, Wikitude will continue to place you at the simulated location.

Note that the simulated location is applied only to Wikitude and all other GPS applications on the iPhone will use your real location rather than the simulated location.

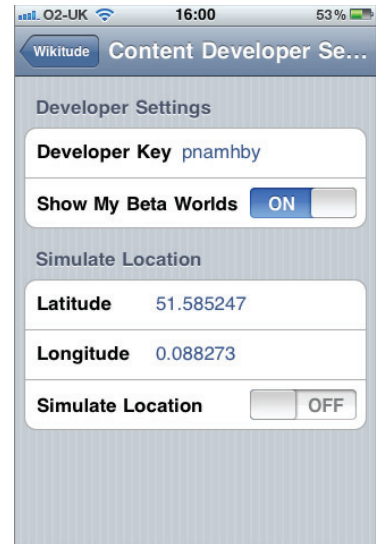


FIGURE 5-13: GPS location simulation



In addition to longitude and latitude, you will often see a third parameter present. This third parameter represents the object's altitude. In most cases, you should just leave this parameter set to 0 (zero), which lets Wikitude handle the altitude automatically. If altitude is important to your World, you can change the setting to the correct value. Use caution, however, because this setting varies according to each device's capabilities and it is my experience that all POIs are drawn on the same horizontal plane.

Creating KML With Google Earth (For Developers)

When you create KML using the Google client, additional tags are created yet ignored by the Wikitude client. If you'll be creating KML from scratch, you don't want to include unnecessary tags. Listing 5-2 shows the minimal KML that displays a POI in the Wikitude client.



LISTING 5-2: Minimal KML

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
xmlns:gx="http://www.google.com/kml/ext/2.2"
xmlns:kml="http://www.opengis.net/kml/2.2"
xmlns:atom="http://www.w3.org/2005/Atom">
<Document>
<Placemark>
```

Available for
download on
Wrox.com

continues

LISTING 5-2 (continued)

```

    <name>Barkingside Station</name>
    <description>This is some text for my POI. Its not very
        exciting</description>
    <Point>
<coordinates>0.08860000006129187,51.58480000018209,0</coordinates>
    </Point>

</Placemark>
</Document>
</kml>

```

As you can see, it is much simpler than the output from Google Earth. Each new POI that you want to include becomes a new `<Placemark>` node that is inserted before the `</Document>` tag. You can, of course, edit the KML in a text editor like Notepad, but you can easily introduce errors such as changing capitalization or forgetting to close a tag. I recommend that you try a free XML editor (such as the free XML Notepad from Microsoft).

In Chapter 4, you harvested 10 latitude and longitude coordinates of various locations near you. In Table 5-1, you will use those coordinates to build a fictitious restaurant finder World. For the longitude/latitude entries, use the coordinates you generated in the previous chapter. Note that the order is longitude/latitude/altitude for KML. If your POIs do not appear in the Wikitude client, you probably have reversed the coordinates and used latitude/longitude.

TABLE 5-1: Restaurant Finder

NAME	DESCRIPTION	LONGITUDE	LATITUDE
Bobs Pancake House	A delicious selection of savory and meaty pancakes. Great for lunches and dinners!		
The Hot Dog King	Like hot dogs? We're the kings of the dog. Try our 3-foot dog special!		
The Curry House	The best curry in the West. Try our super-hot Vindaloo. Can your tongue take it?		
The Slow Cooker	Time no object? Come and relax! We make all our meals the old-fashioned way.		
Speedy Café	You have 15 minutes to eat your food. Finish it quick or we'll take it back!		
Fruit and Stuff	Like fruit? We have a wide selection of apples, pears, bananas and other fruits.		
Healthy Options	We only sell food that is less than 300 calories.		

NAME	DESCRIPTION	LONGITUDE	LATITUDE
Deep Fried	You name it, we fry it. Try our new deep-fried chocolate bars!		
Just Cakes	If you like cakes, you'll be in heaven! We have a wide selection of creamy and chocolate cakes.		
Bread and Butter	The best sandwiches you ever tasted. We make our own bread and churn our own butter. Sandwiches don't get any better than this!		

Listing 5-3 is an extract from the POIs in Table 5-1. Notice that each restaurant has its own <Placemark> node and the number of XML tags required to display a POI is very low when compared to the KML generated by Google Earth.



LISTING 5-3: Restaurant KML

Available for
download on
Wrox.com

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
xmlns:gx="http://www.google.com/kml/ext/2.2"
xmlns:kml="http://www.opengis.net/kml/2.2"
xmlns:atom="http://www.w3.org/2005/Atom">
<Document>
  <Placemark>
    <name>Bobs Pancake House</name>
    <description> A delicious selection of savory and meaty
      pancakes. Great for lunches and dinners! </description>

    <Point>
      <coordinates>LONGITUDE/LATITUDE/ALTITUDE</coordinates>
    </Point>
  </Placemark>
  <Placemark>
    <name>Hot Dog King</name>
    <description> Like hot dogs? We're the kings of the dog.
      Try our 3-foot dog special! </description>

    <Point>
      <coordinates>LONGITUDE/LATITUDE/ALTITUDE</coordinates>
    </Point>
  </Placemark>
</Document>
</kml>
```

To add each POI, copy the <Placemark> node and add the details for <name>, <description> and <coordinates>. Notice that each tag is case-sensitive.

If you use XML Notepad (see Figure 5-14), you can right-click the <Placemark> node and select Duplicate. This will prevent errors occurring in the XML.

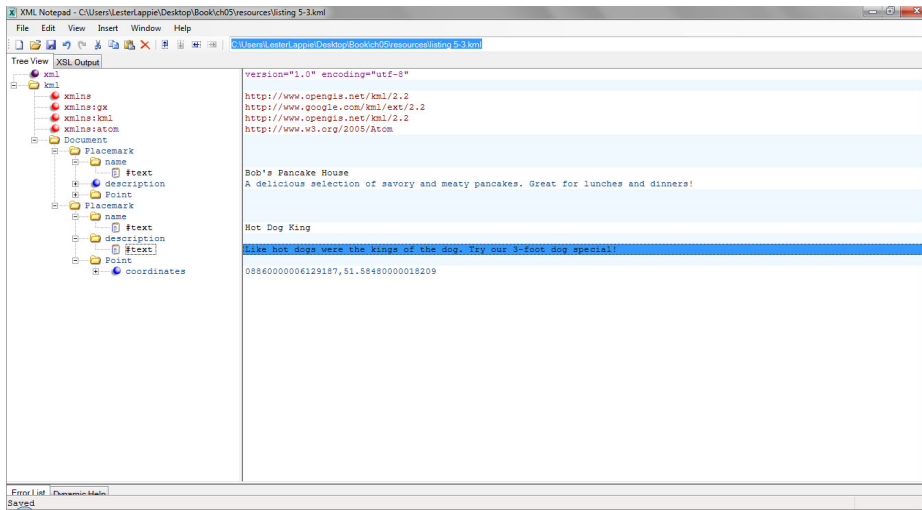


FIGURE 5-14: Using XML Notepad

Testing

Once you have built your restaurant World, you can test it by uploading the KML file via the Wikitude Dashboard and entering the developer key into the Wikitude client. You can also test Worlds by selecting the Show data on Google Maps link from the Wikitude Dashboard. This is located at the top of your World and is useful for quickly testing to ensure all POIs are in the correct locations.

Understanding KML's Limitations

As you tested your World, you probably noticed that the functionality is very basic. With a restaurant World you may have wanted to include a different image for each restaurant — perhaps a picture of the outside of the restaurant so potential customers can get a feel on the environment before they visit.

Similarly, you may have wanted to include functionality such as the ability for the user to place a phone call or email the restaurant to inquire about reservations. In KML, there are no tags to provide this functionality, so KML is really only suitable for the basic Worlds where no interactivity is required.

SUMMARY

In this chapter you created your first Wikitude World. You learned that with a product like Google Earth, virtually anyone can create Worlds that contain multiple POIs. If you're a developer and you want to create Worlds, you learned that KML is also incredibly simple to use; it requires just a few XML nodes. Content is uploaded to the Wikitude server, so there are no hosting or configuring databases to worry about. Building your own Worlds really doesn't get any easier than this.

Finally, you also learned that KML bucks the trend with latitude/longitude coordinates and is the opposite to what you will use with other browsers.

6

Building Worlds with ARML

WHAT'S IN THIS CHAPTER?

- An introduction to Augmented Reality Markup Language (ARML)
- How to create your first Wikitude World using ARML
- How to test your World

In the previous chapter, you learned how you create Wikitude Worlds by creating the KML visually using Google Earth to select your POIs. Additionally, you learned how to create Worlds by creating the KML from scratch. Now that you have created a World, you may have spotted some weaknesses with the KML format. For example, you may have noticed that each POI in the World shares the same icons, or that there is no ability to add phone numbers, email addresses, or links to web sites. These features make the interaction with POIs more compelling.

In this chapter, you will build on your KML knowledge and learn about the proposed new Augmented Reality Markup Language (ARML), which is the first step toward providing a cross-platform, cross-product standard on how AR applications will be created in the future. ARML addresses the weaknesses with KML. It includes support that allows each POI to be customized with its own image and provides the ability for the user to make phone calls, send emails, or visit related web sites directly from the Wikitude client.



The format for ARML was submitted by Mobilizy (the developers of Wikitude) to the AR Consortium (www.arconsortium.org) in September 2009. Consortimums take a long time before new formats are approved as new standards, so ARML still has a long way to go before it becomes an industry standard and no longer a proposal. You can refer to the developer specification for ARML at www.openarml.org.

Before you begin the work we'll do in this chapter, you'll need the following:

- Latitude and longitude coordinates of nearby locations (generated from the previous chapter).
- Some familiarity with XML.
- The Wikitude client installed on your mobile device.
- An XML editor, such as Microsoft's XML Notepad (<http://tinyurl.com/msxmlnotepad>).

UNDERSTANDING AUGMENTED REALITY MARKUP LANGUAGE (ARML)

A superset of the KML used in Google Earth, *Augmented Reality Markup Language (ARML)* includes AR-specific tags that enable you to build more compelling and interactive Worlds. No doubt that by the time it emerges, it will have undergone many reviews and will contain additional tags that developers will find useful in the future.

If — and ultimately *when* — ARML becomes an industry standard, it will enable you to build content that will work across any AR browser that uses the ARML specification. Think about how HTML has enabled developers to build web pages that work across any internet browser. That's why ARML is an exciting proposal for AR. As developers, you will be able to build a single ARML application to work with other AR browsers that have adopted the ARML standard. If you want to build content for all the major platforms today, you have to learn how to program ARML, KML, PHP and XML.

There is a long way to go before we see this type of standard emerge and even longer before we see this type of standard adopted by all the major players. Meanwhile, you should continue to brush up on your PHP and XML skills so you can build content for all the top platforms.

What's New With ARML?

ARML introduces AR-specific tags to enable you to build more exciting content. As you built your KML World in the previous chapter, you may have noticed that some functionality was unavailable with your POIs. In the sample application, it would have been useful to include the phone number of the restaurant so the user could instantly call and reserve a table or perhaps even provided the user with the ability to download a copy of the menu. Additionally, all the POIs in the sample World shared the same icons. Each restaurant had different food specialties, so it would have been useful to include a picture of the outside of the restaurant or perhaps use a custom icon that represents each type of restaurant visually. The Hot Dog King could have included a hot dog icon while the Fruit and Stuff restaurant could have been represented by a banana. This type of visualization presents a much more polished appearance and helps your World stand out in a crowd.

Listing 6-1 shows several of the new tags provided by ARML. Notice that the `<ar:provider>` tag controls the look and feel of the World and sets the World's metadata. The `<Placemark>` tag controls the look and feel of each POI.



Available for
download on
Wrox.com

LISTING 6-1: ARML example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
      xmlns:ar="http://www.openarml.org/arml/1.0"
      xmlns:wiktitude="http://www.openarml.org/wiktitude/1.0">

  <Document>
    <ar:provider id="mountain-tours-I-love.com">
      <ar:name>Mountain Tours I Love</ar:name>
      <ar:description>My preferred mountain tours in the alps.
        Summer and Winter.</ar:description>
      <wiktitude:providerUrl>http://www.providerhomepage.com
        </wiktitude:providerUrl>
      <wiktitude:tags>travel, hiking, skiing, mountains</wiktitude:tags>
      <wiktitude:logo>http://www.mountain-tours-I-love.com
        /wiktitude-logo.png </wiktitude:logo>
      <wiktitude:icon>http://www.mountain-tours-I-love.com
        /wiktitude-icon.png</wiktitude:icon>
    </ar:provider>
    <Placemark id="123">
      <ar:provider>mountain-tours-I-love.com</ar:provider>
      <name>Gaisberg</name>
      <description>Gaisberg is a mountain to the east of Salzburg,
        Austria</description>
      <wiktitude:info>
        <wiktitude:thumbnail>http://www.mountain-tours-I-
          love.com/gaisberg-thumb.png </wiktitude:thumbnail>
        <wiktitude:phone>555-9943</wiktitude:phone>
        <wiktitude:url>http://en.wikipedia.org/wiki/Gaisberg
          </wiktitude:url>
        <wiktitude:email>info@mountain-tours-I-
          love.com</wiktitude:email>
        <wiktitude:address>Jakob-Haringer-Str. 5a, 5020 Salzburg,
          Austria</wiktitude:address>
        <wiktitude:attachment>http://www.mountain-tours-I-
          love.com/gaisberg-map-to-print.pdf
          </wiktitude:attachment>
      </wiktitude:info>
      <Point>
        <coordinates>13.11,47.81,1158</coordinates>
      </Point>
    </Placemark>
  </Document>
</kml>
```

Creating a World With ARML

In the last chapter you created a fictitious KML World that displayed the location of 10 restaurants around your home. In this chapter, you will create the same World in ARML and take advantage of the additional functionality provided by ARML tags. Because both the structure and tags are different, you must create the World from scratch rather than extend the previous sample to support ARML.

I recommend a using the ARML starter shown in Listing 6-2 and using an XML tool such as XML Notepad, which is a free download from Microsoft. There are more tags and more complexity to the XML structure, so a tool enables you to ensure that errors don't unintentionally creep into the file.



Notice that the tags are case-sensitive!



Available for
download on
Wrox.com

LISTING 6-2: ARML starter

```
<?xml version="1.0" encoding="utf-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
xmlns:ar="http://www.openarml.org/arml/1.0"
xmlns:wiktitude="http://www.openarml.org/wiktitude/1.0">
  <Document>
    <ar:provider id="">
      <ar:name></ar:name>
      <ar:description></ar:description>
      <wiktitude:providerUrl></wiktitude:providerUrl>
      <wiktitude:tags></wiktitude:tags>
      <wiktitude:logo></wiktitude:logo>
      <wiktitude:icon></wiktitude:icon>
    </ar:provider>

    <Placemark id="">
      <ar:provider></ar:provider>
      <name></name>
      <description></description>
      <wiktitude:info>
        <wiktitude:thumbnail></wiktitude:thumbnail>
        <wiktitude:phone></wiktitude:phone>
        <wiktitude:url></wiktitude:url>
        <wiktitude:email></wiktitude:email>
        <wiktitude:address></wiktitude:address>
        <wiktitude:attachment></wiktitude:attachment>
      </wiktitude:info>
      <Point>
        <coordinates></coordinates>
      </Point>
    </Placemark>

  </Document>
</kml>
```

Once you have this basic structure, save the file with an .ARML extension. The <Placemark> node can be copied for each new POI that you want to include in the World. If you have typed or downloaded the code shown in Listing 6-1, when you open the ARML in XML Notepad, you will see the structure shown in Figure 6-1.

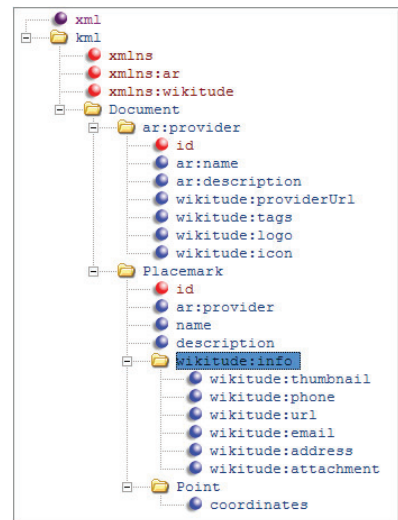


FIGURE 6-1: XML Notepad structure for an ARML World

Now that you have the structure created you can begin to populate the ARML with data. First, complete the `<ar:provider>` tags, which determine the overall look and feel of your World. Each tag is described in this section.

`<ar:provider>`

The provider is a unique name for the World. It cannot be a name that is in use by you or any other developer and must be unique across all Worlds. The best way to ensure that it is unique and reflects the nature of the World is to prefix the name with your company name or your own name. For example: `<ar:provider id="AugmentedPlanet-RestaurantFinder">` enables me to keep the name unique even if there are hundreds of other restaurant Worlds.

`<ar:name>`

This tag is the name of the World as it will be shown in the Wikitude client. You should create a catchy name that adequately explains the nature of the World to potential users. A good name can entice users to try your World. For example:

```
<ar:name>Top 10 Local Restaurants</ar:name>
```

Note, however, that there is a difference between the number of characters you can use on the iPhone and the number of characters you can use on the Android. On the iPhone, which uses the latest version of the client, you have space for 13 characters before the text is truncated. On the Android, you are allowed a lot more. It's only a matter of time before the Android client is updated to the new UI, so your challenge is to describe your World in 13 characters or less.

`<ar:description>`

The description tag is optional and may be included in the Wikitude client. In recent UI changes to Wikitude browser, this tag does not appear. Despite this fact, and despite the fact that the tag is optional, I still recommend that you include it for two reasons:

1. The documentation indicates that the description, if present, forms part of the search criteria.
2. If in the future the client is updated to show the description, you will not need to edit and republish your World.

If you do decide to include it, a good description enables you to go into more detail about why your World should be tried. For example:

```
<ar:description>Discover the top 10 places to eat in Barkingside.  
From fast food to cakes we have it covered</ar:description>
```

`<wikitude:providerUrl>`

This tag provides the means for the user to visit your web page to learn more about you or the World. In the latest versions of the client, clicking your logo no longer triggers the URL. While the tag is optional and currently has no visual effect, I still recommend that you include something because it may become enabled in the future. For example:

```
<wikitude:providerUrl>http://www.augmentedplanet.com</wikitude:providerUrl>
```

<wikitude:tags>

This tag provides a comma-separated list of keywords that relates to your World. As you create the tags, consider the keywords users might use to search for your content. Good tag matches are essential because tags rank higher than the description alone. When you create tags, consider including common misspellings or typos so you have a chance of showing up even when users type the wrong word. For example, include *restaurant* so you can still be found and ranked highly in the list. For example:

```
<wikitude:tags>restaurant,resturant,food,fastfood,fastfood,
burgers,pizza,hotdogs</wikitude:tags>
```

<wikitude:logo>

This tag provides the URL of the logo for either your company or for your World. The logo should be a 96×96-pixel PNG file and it will appear in the client. The Logo field is optional but you should consider it essential because it enables you to present a professional appearance. Without it, your World looks unfinished and unprofessional. For example:

```
<wikitude:logo>http://www.myserver.com/mylogo.png</wikitude:logo>
```

<wikitude:icon>

This tag displays the icon in the camera view to indicate the POI to the user. The format for the icon is a 32×32-pixel PNG file. This is a required field and must be included. For example:

```
<wikitude:icon>http://www.myserver.com/myicon.png</wikitude:icon>
```

Adding the Metadata

Before adding the POIs to the World, you should complete the `<ar:provider>` tags that will define the World. Use Table 6-1 to fill out the tags.

TABLE 6-1 ARML World Metadata

TAG	SETTING
ar:provider id=	<yourname>:RestaurantFinder
ar:name	Restaurants
ar:description	Restaurants around my neighborhood
wikitude:providerUrl	http://<url to your server or landing page>
wikitude:tags	Restaurants,restaurant,food,burgers,hotdogs,cakes
wikitude:logo	URL to the location of the WorldLogo.png file
wikitude:icon	URL to the location of the WorldIcon.png file

If you are using the XML Notepad tool, your screen should look similar to the screen shown in Figure 6-2.

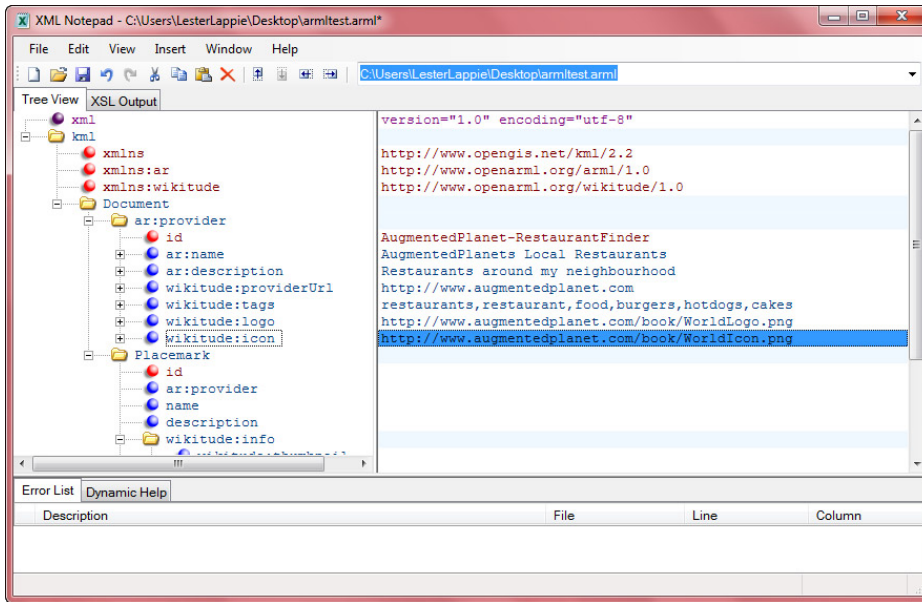


FIGURE 6-2: First steps in ARML

If you have completed the ARML in Notepad, the `<ar:provider>` tag section should resemble what's shown in Listing 6-3.

LISTING 6-3: ARML metadata

```
<ar:provider id="AugmentedPlanet-RestaurantFinder">
<ar:name>Restaurants</ar:name>
<ar:description>Restaurants around my neighbourhood</ar:description>
  <wiktitude:providerUrl>http://www.augmentedplanet.com
    </wiktitude:providerUrl>
  <wiktitude:tags>restaurants, restaurant, food, burgers, hotdogs, cakes
    </wiktitude:tags>
  <wiktitude:logo>http://www.augmentedplanet.com/book/WorldLogo.png
    </wiktitude:logo>
  <wiktitude:icon>http://www.augmentedplanet.com/book/WorldIcon.png
    </wiktitude:icon>
</ar:provider>
```

Now that you have created the World structure, you are ready to add the POIs to the remainder of the document. Note that you cannot upload a World that doesn't contain at least one POI.

Adding the POIs

Before you add the POIs to the ARML file, it's important to understand the usage and purpose of the tags in this section.

<Placemark>

Each POI requires a unique ID which is provided by the <Placemark> tag (note the uppercase P). In most cases, this should be a sequential number commencing at 1 and increasing by 1 for each new POI. Of course, you are free to create your own unique key format, but simple is often better.

```
<Placemark id="1">
```

<ar:provider>

You may choose to have a single ARML file that contains multiple Worlds or just a single World. For example, the file could contain content for Real Estate, Hotels, and Restaurants with each category its own separate World with its own description, tags, and icons. Thus, each POI must be associated with the right World definition. To match POIs to the right World, each POI is given an <ar:provider> tag that should reference the unique <ar:provider> tag in the metadata. In the example shown in Listing 6-3, I used the following:

```
<ar:provider id="AugmentedPlanet-RestaurantFinder">
```

Therefore, each POI in my World needs to reference `AugmentedPlanet-restaurantworld` to be associated with the correct World. The syntax used inside the <Placemark> node is as follows:

```
<ar:provider>AugmentedPlanet-RestaurantFinder</ar:provider>
```

If you misspell the name in the provider tag, the POI will not appear in your World and you will not receive any error messages to indicate that it is missing. This is a good reason to use a structured tool for copying the <Placemark> node.

<name>

The name tag represents the name of the POI. In the restaurant example, this is the name of the restaurant (The Hot Dog King). For example:

```
<name>The Hot Dog King</name>
```

<description>

The description tag is the associated description for the POI that you want to appear in the information bar when the POI is active. The description is optional but you should treat this tag as an essential tag because your World will look unfinished if no content is included. For example:

```
<description>Like hot dogs? We're the kings of the dog. Try our  
3-foot dog special!</description>
```


<wikitude:thumbnail>

With the thumbnail tag, each individual POI can have its own associated image that is shown in the information bar. In the restaurant example, this could be an image of the restaurant or it could be the restaurant logo. The format for the thumbnail image is a 64×64-pixel PNG file. Thumbnail images are optional. The usage is as follows:

```
<wikitude:thumbnail>http://www.yourserver.com/
thumbnail.png</wikitude:thumbnail>
```

<wikitude:phone>

Each POI in the World can have an associated phone number. Including this optional tag displays the number in the POI listing, which provides the user with an easy way to dial the business. The tag works in a very similar way to the `mailto:` parameter of a link tag in HTML. If you use this tag, I recommend that you include the full country and area code for this field and not assume that the local number will be enough.

```
<wikitude:phone>00155512344567
</wikitude:phone>
```

<wikitude:url>

Similar to the phone tag, the url tag enables you to specify a destination web site that is related to the POI. It's an optional tag. Perhaps you are building a free World?. You may consider using this url tag as the landing page where users can obtain more information. This enables you to generate money from advertising rather than simply pass traffic directly to another company's web site.

As an example, imagine a user clicks on the Hot Dog King link in the restaurant World. If that sends the user directly to the Hot Dog King's web site, you have lost the user. If however, you take them to your web site with a description of the Hot Dog King, you could include advertising on the page or an affiliate link. The user still has the option to click a link here to be taken directly to the Hot Dog King web site, but you will have generated some revenue from the advertising or affiliate links on this page.

Usage of the tag is similar to a `<a href>` tag and includes the text and the hyperlink. For example:

```
<wikitude:url name="The Hot Dog King home page">http://www.thehotdogkingpage.com
</wikitude:url>
```

<wikitude:email>

Including this optional tag with an email address displays the email address in the Wikitude client. Once clicked, the phone's email client loads with the email address pre-populated. For example:

```
<wikitude:email>hotdoginfo@thehotdogking.com</wikitude:email>
```

<wikitude:address>

The address tag is optional and not required for routing. Routing uses the <coordinates> supplied to provide turn-by-turn instructions. For example:

```
<wikitude:address>50 Fake Street, Barkingside, IG6 5ZZ, UK</wikitude:address>
```

<wikitude:attachment>

The attachment tag is optional and enables the user to download associated content (for example, a restaurant menu as an image or in a PDF format). You can include any file type, including video and audio, but you should stick to a common file type (for example, JPEG, PNG, or MPEG) that will work across any supported smartphone or the user will be unable to open the file. Wikitude will not impose any file-size limit but you should avoid large files because they may take a long time to download over a data connection.

```
<wikitude:attachment name="View the menu"
type="application/pdf">http://thehotdogking.com/menu.pdf
</wikitude:attachment>
```

The types allowed are standard Multipurpose Internet Mail Extensions (MIME) types. Common types you will encounter include:

- Audio (MPEG)
- Video (MPEG)
- Application (PDF)
- Image (JPEG, PNG, or GIF)
- Text/plain (TXT)

<coordinates>

The coordinates are entered in the format of longitude, latitude, and then altitude. Altitude is optional. If included, altitude should be given in meters.

```
<coordinates>0.080223,51.543047,30</coordinates>
```

Completing the World

In the previous chapter, you created the World based on the restaurant shown in Table 6-2. You will complete the same exercise, except this time you will include the additional tags.



Since you will be adding 10 POIs, you should decide if you are going to copy the <Placemark> node 10 times or if you will copy each node as you go. To copy, simply right-click the <Placemark> tag in XML Notepad and select Duplicate. I recommend that you first add the <ar:provider> tag data, and then perform the copy. This will save you the trouble of including the tag 10 times.

You will now need to complete the <Placemark> node for each POI with the data shown in Table 6-2. As you populate the ARML, keep in mind the following:

- The <ar:provider> tag for each POI should match what you entered for the <ar:provider id=> tag when you added the World metadata.
- To utilize the URL, email, and phone-number functionality you should include your own details. This enables you to complete a realistic test.
- The thumbnail and attachments must be hosted on your server. As you upload them, you should note their full URL so it can be added to the ARML.
- The images referenced in the code can be downloaded along with the ARML.

TABLE 6-2: Restaurant Table

AR:PROVIDER ID	NAME	DESCRIPTION	LATITUDE/ LONGITUDE	OTHER
1	Bob's Pancake House	A delicious selection of savory and meaty pancakes. Great for lunches and dinners!		thumbnail=pancakethumb.png
2	The Hot Dog King	Like hot dogs? We're the kings of the dog. Try our 3-foot dog special!		thumbnail=hotdogthumb.png Attachment=hotdogattachment.jpg
3	The Curry House	The best curry in the West. Try our super-hot Vindaloo. Can your tongue take it?		thumbnail=currythumb.png
4	The Slow Cooker	Time no object? Come and relax! We make all our meals the old-fashioned way.		thumbnail=panthumb.png
5	Speedy Café	You have 15 minutes to eat your food. Finish it quick or we'll take it back!		thumbnail=wizzthumb.png
6	Fruit and Stuff	Like fruit? We have a wide selection of apples, pears, bananas and other fruits.		thumbnail=fruitthumb.png Attachment= banana.jpg
7	Healthy Options	We only sell food that is less than 300 calories.		thumbnail= healthythumb.png

continues

TABLE 6-2 (continued)

AR:PROVIDER ID	NAME	DESCRIPTION	LATITUDE/ LONGITUDE	OTHER
8	Deep Fried	You name it, we fry it. Try our new deep-fried chocolate bars!		thumbnail=frythumb.png
9	Just Cakes	If you like cakes, you'll be in heaven! We have a wide selection of creamy and chocolate cakes.		thumbnail=cakehumb.png
10	Bread and Butter	The best sandwiches you ever tasted. We make our own bread and churn our own butter. Sandwiches don't get any better than this!		thumbnail=breadthumb.png Attachment=breadattachment.jpg

Once you have added each POI to the ARML file, save the file. If you are using XML Notepad, your screen should resemble Figure 6-3. You are now ready to upload the file to the server.

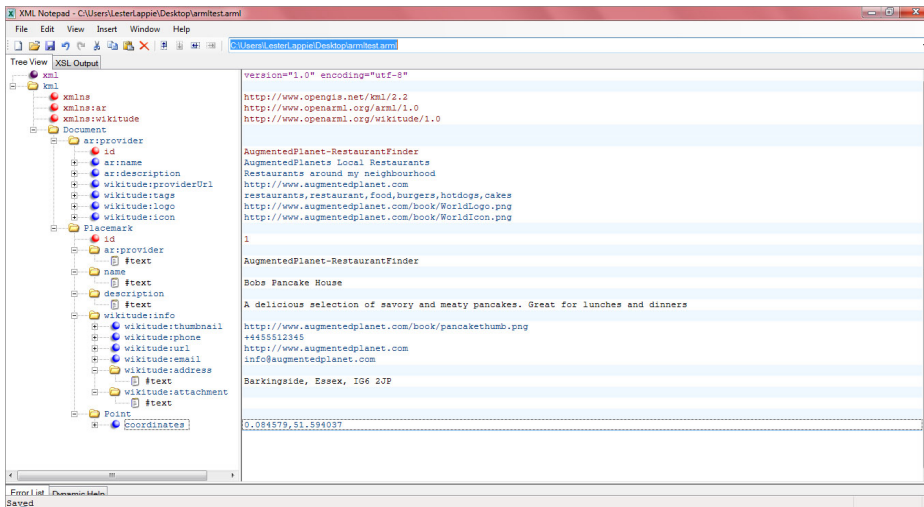


FIGURE 6-3: Completed ARML file

If you have added the ARML using Notepad or if you want to compare the output from the ARML file, it should resemble the extract from the example shown in Listing 6-4.



LISTING 6-4: ARML Restaurants

Available for
download on
Wrox.com

```
<?xml version="1.0" encoding="utf-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
xmlns:ar="http://www.openarml.org/arml/1.0"
xmlns:wiktitude="http://www.openarml.org/wiktitude/1.0">
  <Document>
    <ar:provider id="AugmentedPlanet-RestaurantFinder">
      <ar:name>Restaurants</ar:name>
      <ar:description>Restaurants around my neighbourhood</ar:description>

      <wiktitude:providerUrl>http://www.augmentedplanet.com
        </wiktitude:providerUrl>
      <wiktitude:tags>restaurants, restaurant, food, burgers, hotdogs,
        cakes</wiktitude:tags>
      <wiktitude:logo>http://www.augmentedplanet.com/book/
        WorldLogo.png</wiktitude:logo>
      <wiktitude:icon>http://www.augmentedplanet.com/book/
        WorldIcon.png</wiktitude:icon>
    </ar:provider>

    <Placemark id="1">
      <ar:provider>AugmentedPlanet-RestaurantFinder</ar:provider>
      <name>Bob's Pancake House</name>
      <description>A delicious selection of savory and meaty pancakes.
        Great for lunches and dinners!</description>
      <wiktitude:info>
        <wiktitude:thumbnail>http://www.augmentedplanet.com/book/
          pancakethumb.png</wiktitude:thumbnail>
        <wiktitude:phone>+4455512345</wiktitude:phone>
        <wiktitude:url>http://www.augmentedplanet.com</wiktitude:url>
        <wiktitude:email>info@augmentedplanet.com</wiktitude:email>
        <wiktitude:address></wiktitude:address>
        <wiktitude:attachment></wiktitude:attachment>
      </wiktitude:info>
      <Point>
        <coordinates>0.077369, 51.596249</coordinates>
      </Point>
    </Placemark>

  </Document>
</kml>
```

In your example, you will reference your own server where the images are hosted and also utilize your own longitude/latitude for the coordinate tags.

Creating the ARML World

Now that you have created the ARML file, you should ensure that the relevant thumbnail images and attachments have been uploaded to your web server to the location you specified in the ARML file. You will now create the World listing using the Wikitude Dashboard and upload your ARML file. To upload the file, revisit the Wikitude dashboard at www.wikitude.me. Where you previously selected the KML option, this time you'll click the Upload ARML File option.

As shown in Figure 6-4, the information required to create the World is similar to that required when creating the KML World. The primary difference is that unlike the KML example (where you created the World metadata including images using the Wikitude Dashboard), the ARML file now contains all the relevant images and resources which must be hosted on your server.

The screenshot shows the Wikitude Dashboard interface. At the top left is the Wikitude logo with the tagline 'GEO-TAG THE WORLD'. The main content area is titled 'Add' and contains the following form elements:

- World Status:** A dropdown menu currently set to 'Public'.
- ARML file:** A text input field with a 'Browse...' button and a placeholder '??armUrl??'.
- E-Mail (required):** A text input field.
- Terms of Use:** A checkbox labeled 'I have read the above-mentioned Terms and Conditions for uploading data to the Wikitude system and accept them.' Below this is a scrollable area containing the 'Wikitude Content Interface Terms of Use' text.
- Save:** An orange button at the bottom of the form.

On the right side of the dashboard, there is a sidebar titled 'My ARMLs' which is currently empty.

FIGURE 6-4: ARML Dashboard view

As you upload the World, ensure that you set the status to Testing so it does not inadvertently become visible to everyone. You'll also notice that there are no tags or descriptions required; remember, all the World metadata is contained in the ARML file. To complete the publishing of the World, type your email address and select the terms and conditions check box. Then click Save.

Now you should see the World developer key displayed under the combo box where you selected your World's status. If you do not see the developer key, simply select your World from the navigation on the right side of the screen and you will find it under the World status box.

Testing on the Device

The final step is to test your World on your device. As you learned in the previous chapter, you must enter your developer key into the Wikitude client. Because the client accepts only one developer key at a time, you must overwrite any existing keys.

On the iPhone, you can find the Wikitude configuration menu by selecting the iPhone's Settings application and then selecting Wikitude. The developer settings on the Android can be accessed by pressing the menu button from directly inside the Wikitude client. If you need assistance you should refer to the respective Testing sections in Chapter 5.

Once you have entered your developer key, you can test the World on your smartphone. Often, however, you must kill the Wikitude process so the new developer key is recognized. On the iPhone, you can do this by double-pressing the center button and then pressing and holding the Wikitude icon.

On the Android, use an application to terminate the process. As you test, be sure to try the attachments as well as the routing, calling, and emailing functionalities.



If you update your World and re-upload it using the Wikitude Dashboard, a new developer key is generated. You must reenter the new developer key into the client.

Figure 6-5 shows the completed World.

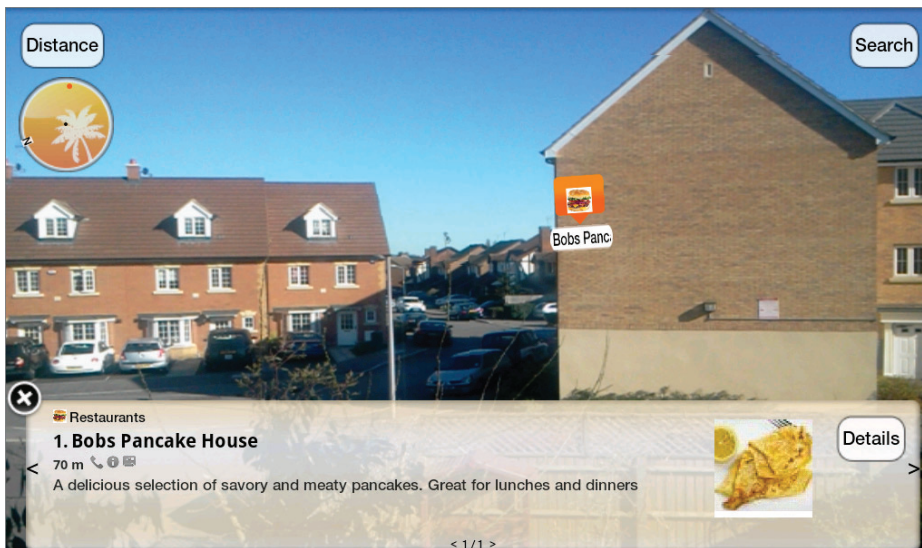


FIGURE 6-5: A POI from the completed ARML World

SUMMARY

In this chapter, you learned how to create AR content using ARML. You learned ARML gives you more control over images, allowing each POI to have its own associated image. Additionally, ARML allows you to use powerful tags that provide users with the ability to place phone calls, send emails, or obtain turn-by-turn navigation instructions from their current location to the POI.

PART III

Layar

- ▶ **CHAPTER 7:** Building Layar Layers
- ▶ **CHAPTER 8:** Creating Filters and 2D Objects
- ▶ **CHAPTER 9:** Using Layar Tools

7

Building Layar Layers

WHAT'S IN THIS CHAPTER?

- ▶ How to create a layer
- ▶ How to set up the Layar database
- ▶ How to create a web service
- ▶ How to access the layer
- ▶ How to customize the layer
- ▶ How to add actions
- ▶ How to create triggers

In Chapter 1, you learned that Layar is slightly more complicated to develop for than Wikitude. Developers always ask me why I think Layar is complicated; truth is, it's not. But as you just learned in the previous chapter, Wikitude is incredibly simple to use to build content. Layar requires a little more preparation.

Before you can begin working with Layar in this chapter, you'll need the following:

- ▶ A publicly visible web server with PHP 5.2 or newer, including JSON support.
- ▶ An FTP application to upload files to your server.
- ▶ A MySQL database (phpMyAdmin is preferred).
- ▶ A PHP editor (such as PSPad) to edit PHP files. See www.pspad.com.
- ▶ Nearby latitude/longitude coordinates.
- ▶ The Layar mobile application installed to your device.

Please note that your publicly visible web server needs to be a hosted account. You will not be able to configure your home desktop PC as a web server because the Layar client must access the PC and retrieve data from a MySQL database and access the PHP resources.

You can, of course, develop for Layar in other programming languages (such as C#, Java, and so on). You can also use other database products, such as Microsoft SQL Server. However, for the purposes of this book, I will provide examples only in PHP.

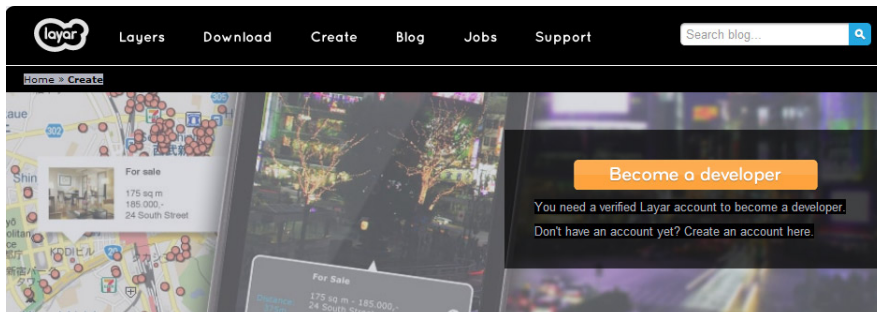
CREATING YOUR LAYAR ACCOUNT

Before you can begin building layers, you will need to have a developer account. This is normally a two-step process. The first step gives you a standard account; the second step grants you developer access.

1. To complete the first step, visit `www.layar.com` and click the Create account option at the top right of the screen.

You should receive confirmation (of your account being created) via e-mail and you are ready to proceed to step two.

2. Log in to the Layar portal. (If you are not already logged in to the Layar portal, visit `www.layar.com` and login with your account details.)
3. Once you have logged in, click the Create option from the top navigation and then click the Become a developer option (see Figure 7-1) to create a developer account.



COPYRIGHT © 2005 BY MySQL AB

FIGURE 7-1: Creating a developer account

4. Complete the short enrollment form.

You should now see a screen similar to Figure 7-2, which is your Layar Dashboard — the gateway to publishing content for the Layar platform.

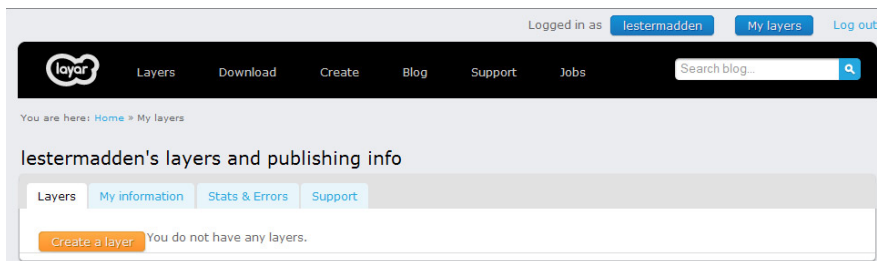


FIGURE 7-2: The developer portal

Using the Dashboard, you can:

- Create/delete layers.
- Configure your contact information.
- Configure your payment details (if you are selling your content).
- Configure your public profile (such as your company logo). This is optional; if you create a public profile, it will appear on the Layar web site. This is useful if you have created interesting layers and you are perhaps interested in developing layers for third parties.
- See stats about usage and any errors that may have occurred with your layers.
- Get support on the development of layers, suggest a feature, or send the Layar team an e-mail.

Whenever you want to amend, create, or delete layers, you will need to do it via the Layar Dashboard.

CREATING A LAYER

On your web server, you first must create a subfolder where you will host the Layar endpoint function. In this subfolder, create a file called `mylayer.php` and add the following code:

```
Message: it's working
```

Ensure that the `mylayer.php` file has been uploaded to your web server and then test that the URL is valid by typing the URL to the file. If it is correct, you should see the contents of the `mylayer.php` file displayed in your web browser. If not, double-check the URL. It should look something like:

```
www.myserver.com/mysubfolder/mylayer.php
```

Once you have the valid URL, make a note of it and proceed to creating the layer on the publishing site.

Creating the Layer on the Publishing Site

The first part of the process in creating the layer is creating the layer through the Layar Dashboard. This will configure some basic details about your layer, including:

- A simple name.
- A description.
- A URL of the endpoint function.

1. On your Layar Dashboard, click the Create option, which displays the first part of the configuration form (see Figure 7-3).

Create a Layer

Layer name:

Title:

Publisher name:

Layer type:

API endpoint URL:

Short description:

FIGURE 7-3: Creating a Layer

The options here are very simple and will define how your layer looks inside the Layar client:

- Layer name is a private name that is not published. The name you give here is not changeable and is what will be used to refer to your layer in code or when you need to communicate with Layar team.



*Note the layer name needs to be unique — it cannot be one that another developer has used. As shown in Figure 7-3, I prefixed my Layar name with **ap** (Augmented Planet). Prefixing your layer name with either your initials or company name is a good way to ensure that it is unique.*

- Title is the name as it will appear in the Layar gallery.
- Publisher name is taken from your publisher account and should generally be left the same. This field is also visible in the Layar gallery.
- Layer type specifies how the client should display your POIs (for example, as 2D objects or as 3D objects).
- API endpoint URL is the link to the mylayer.php file you created. You will need to provide the full URL here.



Note that only the default ports 80 (*http*) and 443 (*https*) are supported. If you have configured your server to support additional ports, you will need to open ports 80 and 443.

- Short description is the description that will appear in the Layar gallery. This is your opportunity to sell your layer to your potential audience.

Complete the Create a Layer form similar to how I completed mine in Figure 7-3. Be sure to use the link to the endpoint file on your own server. Once you are ready, click the Create layer button.

Testing in the Client

You now have created a very basic layer. It has no functionality, but you can still test it in the client to make sure that everything is configured OK and that it appears in the Layer client. On your iPhone or Android device, load the Layar client and select the More option from the navigation menu. You will be given the option to log in to your account. Use the same details you used when creating the developer account in the previous steps. Once you have logged in, you will see a message indicating that you have been given access to the development mode.

You will notice that the main Layer screen inside the mobile application now has a Test category (see Figure 7-4. If you haven't guessed already, this is where your development layers will appear — enabling you to test their functionality before making them public. They will be available only to you and you will need to log in using your Layar account to make them visible.

If you click the Test category, you will see the layer you created along with the short description that you added. At this point, the design is a little boring (see Figure 7-5). Later in this chapter, you'll learn how to add screenshots and richer information.

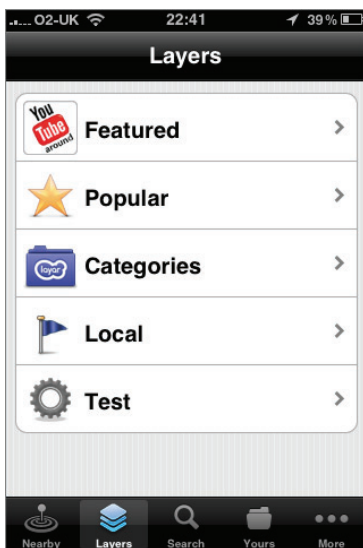


FIGURE 7-4: The Test category

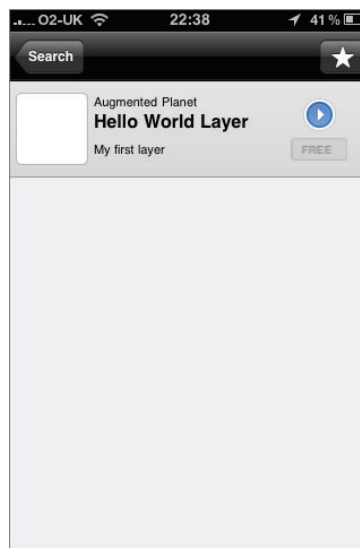


FIGURE 7-5: Your first layer

PREPARING THE DATABASE

Layar takes the POI content from a database, so your next step is to set up a MySQL database and create a simple table to hold the content. Exactly how you create the database is dependent on your hosting company and the tools they provide for you. Most hosting companies should provide you with a phpMyAdmin interface to create and modify MySQL tables.



The following instructions are specific to creating a database on the GoDaddy hosting site. If you are with another web hosting company, you will need to refer to their specific instructions on how to access the phpMyAdmin tool. Search Google for phpMyAdmin and your hosting company name.

1. Log in to your Account Manager at GoDaddy.com.
2. From the Products section, click Web Hosting.
3. Next to the hosting account you want to use, click Launch.
4. In the Databases section of the Hosting Control Center, click MySQL.
5. In the Create Database form, type a database name and configure it as shown in Figure 7-6.

The screenshot shows the 'Create Database' form in the GoDaddy Hosting Control Center. The form is titled 'Create a New MySQL Database Instructions' and includes the following fields and options:

- Description:** MyLayar Database
- MySQL Database/User Name:** LayarDBUser
- New Password:** [Strong]
- Confirm Password:** [How To Generate a Strong Password]
- Read-Only User Name (Optional):** [Not Rated]
- New Password:** [Not Rated]
- Confirm Password:** [How To Generate a Strong Password]

The MySQL Version is set to 4.1. The form also includes 'OK' and 'Cancel' buttons at the bottom right.

FIGURE 7-6: Creating a MySQL database

Be sure to make a note of the User Name and New Password. In my example, the User Name is LayarDBUser.

6. Click the OK button.

You will then need to wait a few minutes while the database is created.

7. Once the database has been created, you will see a link to manage via phpMyAdmin (see Figure 7-7).

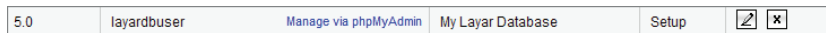


FIGURE 7-7: The Log in to the phpMyAdmin interface

8. Click the phpMyAdmin link. You will be prompted to log in to the database you created.
9. Once your database has been configured correctly and you have successfully logged in, you will be directed to screen shown in Figure 7-8.

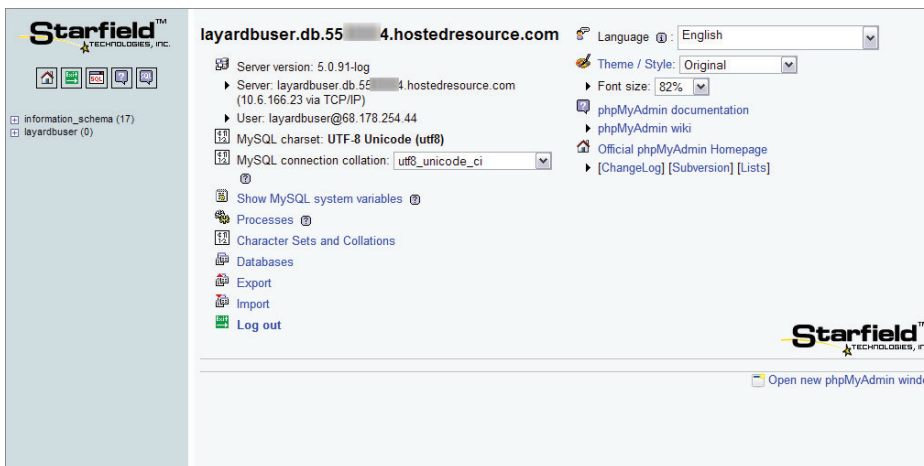


FIGURE 7-8: The phpMyAdmin interface.



If you are stuck, try a Google search on how to create a MySQL database with your particular hosting company. Or contact their technical support team for further assistance.

Creating the Table

You have created the MySQL database and successfully logged in. Your next step is to create the table that will contain the POIs.



The following steps will be valid for all MySQL users regardless of hosting company.

To create the table in the database, follow these steps.

1. On the left side of the screen, click the SQL button (see Figure 7-9) to open the SQL pane.

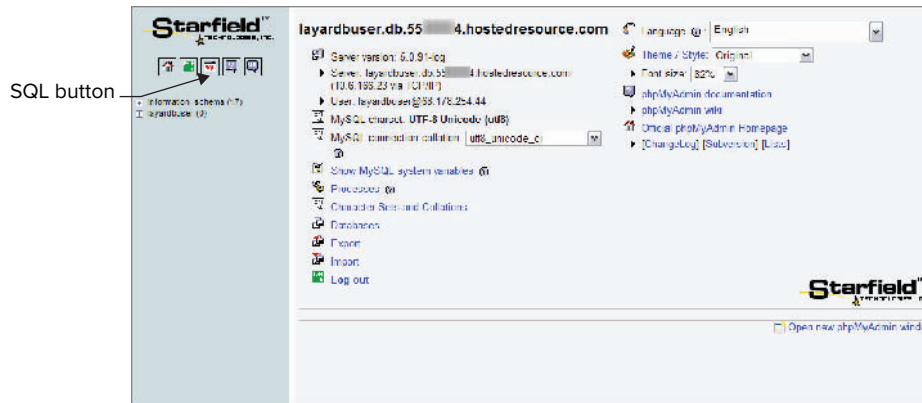


FIGURE 7-9: Using the SQL button

The SQL query window (see Figure 7-10) is where you will enter your SQL to create the table and the relevant fields.

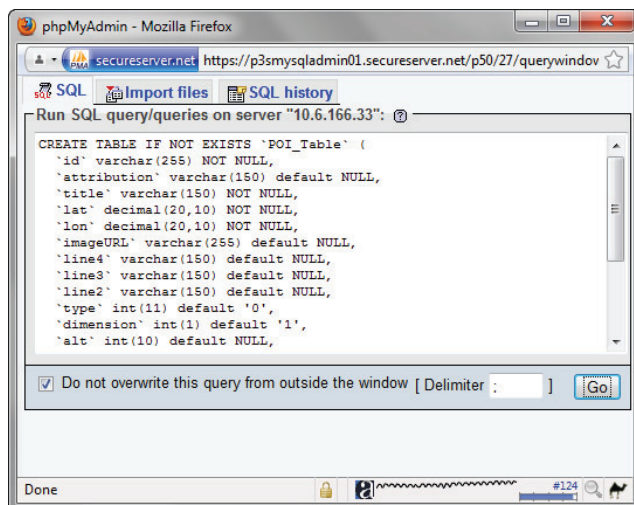


FIGURE 7-10: The SQL query window

- In the SQL query window, type the SQL shown in Listing 7-1 and click the Go button.



LISTING 7-1: Creating a POI_Table

Available for
download on
Wrox.com

```
CREATE TABLE IF NOT EXISTS `POI_Table` (
  `id` varchar(255) NOT NULL,
  `attribution` varchar(150) default NULL,
  `title` varchar(150) NOT NULL,
  `lat` decimal(20,10) NOT NULL,
  `lon` decimal(20,10) NOT NULL,
  `imageURL` varchar(255) default NULL,
  `line4` varchar(150) default NULL,
  `line3` varchar(150) default NULL,
  `line2` varchar(150) default NULL,
  `type` int(11) default `0`,
  `dimension` int(1) default `1`,
  `alt` int(10) default NULL,
  `relativeAlt` int(10) default NULL,
  `distance` decimal(20,10) NOT NULL,
  `inFocus` tinyint(1) default `0`,
  `doNotIndex` tinyint(1) default `0`,
  `showSmallBiw` tinyint(1) default `1`,
  `showBiwOnClick` tinyint(1) default `1`,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

- Close the SQL query window.
- Return to the phpMyAdmin window and see that the SQL query has completed successfully.

Your table has been created (see Figure 7-11).

Your SQL query has been executed successfully (Query took 0.0005 sec)

SQL query:

```
CREATE TABLE IF NOT EXISTS `POI_Table` (
  `id` VARCHAR(255) NOT NULL,
  `attribution` VARCHAR(150) DEFAULT NULL,
  `title` VARCHAR(150) NOT NULL,
  `lat` DECIMAL(20,10) NOT NULL,
  `lon` DECIMAL(20,10) NOT NULL,
  `imageURL` VARCHAR(255) DEFAULT NULL,
  `line4` VARCHAR(150) DEFAULT NULL,
  `line3` VARCHAR(150) DEFAULT NULL,
  `line2` VARCHAR(150) DEFAULT NULL,
  `type` INT(11) DEFAULT 0,
  `dimension` INT(1) DEFAULT 1,
  `alt` INT(10) DEFAULT NULL,
  `relativeAlt` INT(10) DEFAULT NULL,
  `distance` DECIMAL(20,10) NOT NULL,
  `inFocus` TINYINT(1) DEFAULT 0,
  `doNotIndex` TINYINT(1) DEFAULT 0,
  `showSmallBiw` TINYINT(1) DEFAULT 1,
  `showBiwOnClick` TINYINT(1) DEFAULT 1,
  PRIMARY KEY (`id`)
) ENGINE = MYISAM DEFAULT CHARSET = utf8
```

Edit Create PHP Code

COPYRIGHT © 2005 BY MySQL AB

FIGURE 7-11: The SQL window

Adding POIs to the Database

At this point you have created the MySQL database and have added a table with several fields. The next step is to insert some POIs into the table. I will assume that you didn't change the name from the `POI_Table` used in the previous steps.

There are two ways to populate your database:

- Using SQL
- Using the phpMyAdmin interface

Each produces the same results. Your preference will depend on how well you know SQL or whether you prefer a GUI. I will show you how to use both methods; use whichever method you prefer.

Using SQL

If you're a database wizard or you want to add your POIs via SQL, then to insert data into the table, follow the steps you previously used to create the table. The relevant SQL for adding a sample POI is shown in Listing 7-2.



In Chapter 4, "Latitude, Longitude, and Where to Get POIs," you harvested nearby latitude/longitude values. Insert those values into the relevant <latitude value> and <longitude value> fields in Listing 7-2 before attempting to run the query. You can also download the images referenced in Listing 7-2 and upload them to your own web server.



LISTING 7-2: Adding POIs to the database via SQL

Available for
download on
Wrox.com

```
INSERT INTO `POI_Table`
  (`id`, `attribution`, `title`,
   `lat`, `lon`, `imageURL`, `line4`, `line3`,
   `line2`, `type`, `dimension`, `alt`, `relativeAlt`,
   `distance`, `inFocus`, `doNotIndex`, `showSmallBiw`,
   `showBiwOnClick`)

VALUES
  ('1', 'My First Layer', 'Hello World',
   '<latitude value>',
   '<longitude value>', '
   '<yourserver>/helloworld.png',
   'A description', 'distance:%distance%',
   'more details about my POI', 1, 1, NULL, NULL,
   '0.0000000000', 0, 0, 1, 1);
```

Using the phpMyAdmin Interface

In this step, you will add a second POI to the database. This time, however, you will use the MySQL user interface to add the POI. Since you have added the database and the table to MySQL, the left side of the screen has been quietly updating, adding your content to the navigation (see Figure 7-12).

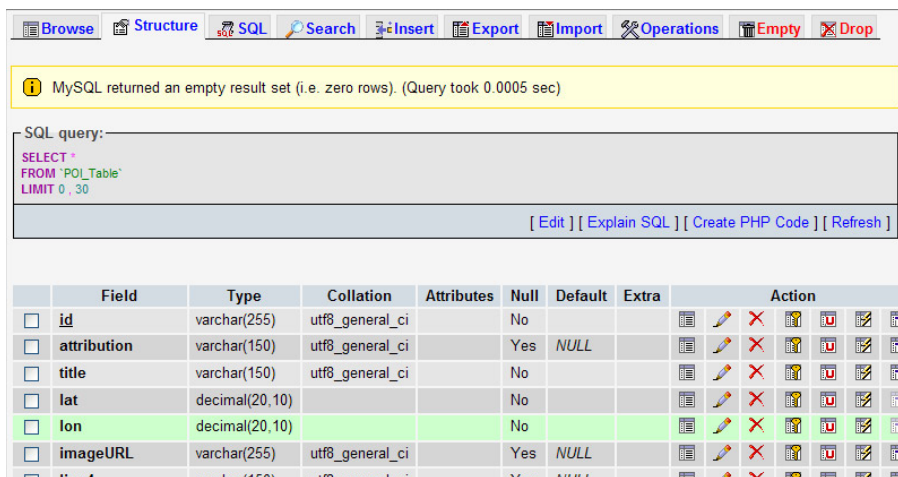


COPYRIGHT © 2005 BY MYSQL AB

FIGURE 7-12: Navigating your database

1. Expand your database and click the table name.

You should now see your table in the phpMyAdmin window (see Figure 7-13). You could make changes to your database here, but our database should be fine as it is.



COPYRIGHT © 2005 BY MYSQL AB

FIGURE 7-13: Your table

2. From the tabbed menu, click the Insert option.
3. Complete the phpMySQL form as shown in Figure 7-14.



For the sample layer you will need to enter the latitude and longitude values from just one of the POIs you harvested in Chapter 4. You also need the sample image uploaded to your web server. This can be found in Listing 7-2.

Field	Type	Function	Null	Value
id	varchar(255)			2
attribution	varchar(150)		<input type="checkbox"/>	My First Layar
title	varchar(150)			Hello World
lat	decimal(20,10)			<your latitude>
lon	decimal(20,10)			<your longitude>
imageURL	varchar(255)		<input type="checkbox"/>	nt/uploads/playground/layar/helloworld.png
line4	varchar(150)		<input type="checkbox"/>	A description
line3	varchar(150)		<input type="checkbox"/>	distance:%distance%
line2	varchar(150)		<input type="checkbox"/>	more details about my POI'
type	int(11)		<input type="checkbox"/>	1
dimension	int(1)		<input type="checkbox"/>	1
alt	int(10)		<input checked="" type="checkbox"/>	
relativeAlt	int(10)		<input checked="" type="checkbox"/>	
distance	decimal(20,10)			0.0000000000
inFocus	tinyint(1)		<input type="checkbox"/>	0
doNotIndex	tinyint(1)		<input type="checkbox"/>	0
showSmallBiw	tinyint(1)		<input type="checkbox"/>	1
showBiwOnClick	tinyint(1)		<input type="checkbox"/>	1

COPYRIGHT © 2005 BY MYSQL AB

FIGURE 7-14: Adding data via the phpMyAdmin window

Creating a Web Service

Your database now has the coordinates of several POIs that you created near your location. You also have a layer appearing in the client. The next step is to connect the two together so the client can log in to the database and access your POIs. You will do this by extending the `mylayer.php` file that you created earlier. To begin, you can remove the line of code that reads `Message: it's working` so you have an empty `mylayer.php` file.

In this file, you need to add the configuration for your MySQL database. You need to have the following details:

- localhost
- database_name
- database_username
- database_password

These details are available from the steps you followed to create the database to hold the POIs. localhost is the server used for your MySQL database; you can get the server location by logging in into your database via the MySQL panel (see Figure 7-15).



COPYRIGHT © 2005 BY MYSQL AB

FIGURE 7-15: URL for the MySQL database

Once you have these details, you can update the `mylayer.php` file as shown in Listing 7-3.



LISTING 7-3: Configuring the database

Available for
download on
Wrox.com

```
<?php
/* Configure the connection to the MySQL database */
/* add your details here */

$dbhost = "localhost"; /* your server name */
$dbdata = "database_name"; /* your database name */
$dbuser = "database_username"; /* the db username */
$dbpass = "database_password"; /* the db password */

/* connect to the MySQL server. */

$db = new PDO( "mysql:host=$dbhost; dbname=$dbdata", $dbuser,
$dbpass, array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8") );

// set the error reporting attribute to throw Exception .
$db->setAttribute( PDO::ATTR_ERRMODE , PDO::ERRMODE_EXCEPTION );
```



All of the code that follows should be added to the `mylayer.php` file unless otherwise indicated.

Inspecting the Query String

Layar will call a callback function in the `mylayer.php` file and include information about the user's physical location in the query string. You will need to examine the HTTP request to get access to the query string and determine the response.

The following is an example query string passed from Layar to the callback function:

```
http://www.augmentedplanet.com/wp-content/uploads/playground/layar/
mylayer.php?lang=en&countryCode=AF&lon=4.887339&userId=
6f85d06929d160a7c8a3cc1ab4b54b87db99f74b&developerId=0&developerHash=4
57f9999d71e53d44e57c5f7f80b6b1e8ec3a215&version=4.0&radius=1500
&timestamp=1289663298578&lat=52.377544&layerName=apmyfirstlayer&accuracy=100
```

If you look at the query string above you can see that the string contains a host of interesting data, but for our purposes right now, we are specifically interested in the following:

- `layerName` is the name of our layer.
- `lat` is the latitude of the position where the user is located.
- `lon` is the longitude of the position where the user is located.
- `radius` is the search range within which POIs should be returned.

The following code will read the query string and put the parameters into an array named `$value`:

```
// Put needed parameter names from GetPOI request in an array called $keys.
$keys = array( "layerName", "lat", "lon", "radius" );

// Initialize an empty associative array.
$value = array();

try {
// Retrieve parameter values using $_GET and put them in $value array with
// parameter name as key.
foreach( $keys as $key ) {

    if ( isset($_GET[$key]) )
        $value[$key] = $_GET[$key];
    else
        throw new Exception($key ." parameter is not passed in GetPOI request.");
} //foreach
} //try
catch(Exception $e) {
    echo 'Message: ' . $e->getMessage();
} //catch
```

Now you need to prepare your response that will display the POIs to the user:

```
// Create an empty array named response.
$response = array();

// Assign cooresponding values to mandatory JSON response keys.
```



```

$response["layer"] = $value["layerName"];

// Use Gethotspots() function to retrieve POIs with in the search range.
$response["hotspots"] = Gethotspots( $db, $value );

// if there is no POI found, return a custom error message.
if ( empty( $response["hotspots"] ) ) {
    $response["errorCode"] = 20;
    $response["errorString"] = "No POI found. Please adjust the range.";
} //if

else {
    $response["errorCode"] = 0;
    $response["errorString"] = "ok";
} //else

```

Notice the use of an `errorCode` value of 20. If you want to create your own custom error messages, use an error code between 20 and 29 and create your own `errorString` message:

```

$response["errorCode"] = 20;
$response["errorString"] = "No POI found. Please adjust the range.";

```

In the previous code, you created a custom function called `Gethotspots()`. This function is used to retrieve the POIs from the MySQL database and put them into the `$response["hotspots"]` array.

Using SQL to Fetch the POI Information

Listing 7-4 continues to build on the code you have entered in the previous steps and creates the SQL query to access the MySQL database. The query restricts the results to 50 POIs for performance reasons and uses the Haversine formula to determine the distance of the user to the POIs. The *Haversine formula* is an equation used in navigation for calculating the shortest distance between any two points on the surface of a sphere from their longitudes and latitudes coordinates.



LISTING 7-4: Fetching the POI from the database

Available for
download on
Wrox.com

```

function Gethotspots( $db, $value ) {

    /* Create the SQL query to retrieve POIs within the "radius" returned from
    GetPOI request. Returned POIs are sorted by distance and the first 50 POIs
    are selected. The distance is calculated based on the Haversine formula.
    Note: this way of calculation is not scalable for querying large database.
    */

    // ":lat1", ":lat2", ":long" and ":radius" are named parameter markers for
    // which real values will be substituted when the statement is executed.
    // $sql is returned as a PDO statement object.
    $sql = $db->prepare( "
        SELECT id,
            attribution,

```

continues

LISTING 7-4 (continued)

```

        title,
        lat,
        lon,
        imageURL,
        line4,
        line3,
        line2,
        type,
        dimension,
        ((acos(sin((:lat1 * pi() / 180)) * sin((lat * pi() / 180)) +
            cos((:lat2 * pi() / 180)) * cos((lat * pi() / 180)) *
            cos((:long - lon) * pi() / 180))
            ) * 180 / pi()) * 60 * 1.1515 * 1.609344 * 1000) as distance
    FROM POI_Table
    HAVING distance < :radius
    ORDER BY distance ASC
    LIMIT 0, 50 " );

    // PDOStatement::bindParam() binds the named parameter markers to the
// specified parameter values.
    $sql->bindParam( ':lat1', $value['lat'], PDO::PARAM_STR );
    $sql->bindParam( ':lat2', $value['lat'], PDO::PARAM_STR );
    $sql->bindParam( ':long', $value['lon'], PDO::PARAM_STR );
    $sql->bindParam( ':radius', $value['radius'], PDO::PARAM_INT );

    // Use PDO::execute() to execute the prepared statement $sql.
    $sql->execute();

    // Iterator for the response array.
    $i = 0;

    // Use fetchAll to return an array containing all of the remaining rows
// in the result set.
    // Use PDO::FETCH_ASSOC to fetch $sql query results and return each row
// as an array indexed by column name.
    $pois = $sql->fetchAll(PDO::FETCH_ASSOC);

    /* Process the $pois result */

    // if $pois array is empty, return empty array.
    if ( empty($pois) ) {

        $response["hotspots"] = array ();

    } //if
    else {

```



In Listing 7-4, the SQL query is prepared using the `PDO::prepare()` function and then executed using the `PDO::execute()` function. These functions are recommended by Layar for security reasons and will help prevent general SQL injection attacks.

Parsing the Retrieved POI Information

Now that the database query has executed, put the results into an array:

```
// Put each POI information into $response["hotspots"] array.
    foreach ( $pois as $poi ) {

// If not used, return an empty actions array.
    $poi["actions"] = array();

// Store the integer value of "lat" and "lon" using predefined function
// ChangetoIntLoc.
    $poi["lat"] = ChangetoIntLoc( $poi["lat"] );
    $poi["lon"] = ChangetoIntLoc( $poi["lon"] );

// Change to Int with function ChangetoInt.
    $poi["type"] = ChangetoInt( $poi["type"] );
    $poi["dimension"] = ChangetoInt( $poi["dimension"] );

// Change to demical value with function ChangetoFloat
    $poi["distance"] = ChangetoFloat( $poi["distance"] );

// Put the poi into the response array.
    $response["hotspots"][$i] = $poi;
    $i++;
} //foreach

} //else

return $response["hotspots"];
} //Gethotspots
```

Including Custom Support Functions

To complete the code, include the following support functions:

ChangetoIntLoc() converts a decimal GPS latitude or longitude value to an integer by multiplying by 1000000:

```
// Convert a decimal GPS latitude or longitude value to an integer by
// multiplying by 1000000.
//
// Arguments:
//   value_Dec ; The decimal latitude or longitude GPS value.
//
// Returns:
//   int ; The integer value of the latitude or longitude.
//
function ChangetoIntLoc( $value_Dec ) {
    return $value_Dec * 1000000;
} //ChangetoIntLoc
```

ChangetoInt() changes a string value to integer:

```
// Change a string value to integer.
//
// Arguments:
//   string ; A string value.
//
// Returns:
//   Int ; If the string is empty, return NULL.
//
function ChangetoInt( $string ) {

    if ( strlen( trim( $string ) ) != 0 ) {

        return (int)$string;
    }
    else
        return NULL;
} //ChangetoInt
```

ChangetoFloat() changes a string value to float:

```
// Change a value to float
//
// Arguments:
//   string ; A string value.
//
// Returns:
//   float ; If the string is empty, return NULL.
//
function ChangetoFloat( $string ) {

    if ( strlen( trim( $string ) ) != 0 ) {

        return (float)$string;
    }
    else
        return NULL;
} //ChangetoFloat
```

Write the \$JsonResponse array into JSON format and then close the connection to MySQL database:

```
// Put the JSON representation of $response into $jsonresponse.
$jsonresponse = json_encode( $response );

// Declare the correct content type in HTTP response header.
header( "Content-type: application/json; charset=utf-8" );

// Print out Json response.
echo $jsonresponse;

/* Close the MySQL connection.*/

// Set $db to NULL to close the database connection.
$db=null;
```

Finally, don't forget to close the php tag:

```
?>
```

Viewing the mylayer.php Code in Full

Congratulations, you have created your first layer and have successfully configured your database with sample content and created the web service that will access your POI and return it to the user. Listing 7-5 shows the previous code in its entirety.



Available for
download on
Wrox.com

LISTING 7-5: Complete listing to retrieve and display POIs in Laya

```
<php
/* Configure the connection to the MySQL database */
/* add your details here */

$dbhost = "localhost";
$dbdata = "database_name";
$dbuser = "database_username";
$dbpass = "database_password";

/* connect to the MySQL database. */
$db = new PDO( "mysql:host=$dbhost; dbname=$dbdata", $dbuser, $dbpass,
array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8") );

// set the error reporting attribute to throw Exception .
$db->setAttribute( PDO::ATTR_ERRMODE , PDO::ERRMODE_EXCEPTION );

// Put needed parameter names from GetPOI request in an array called $keys.
$keys = array("layerName", "lat", "lon", "radius" );

// Initialize an empty associative array.
$value = array();

try {
    // Retrieve parameter values using $_GET and put them in $value array
    // with parameter name as key.
    foreach( $keys as $key ) {

        if ( isset($_GET[$key]) )
            $value[$key] = $_GET[$key];
        else
            throw new Exception($key ." parameter is not passed in GetPOI
request.");
    }//foreach
} //try
catch(Exception $e) {
    echo 'Message: ' . $e->getMessage();
} //catch

/* Construct the response into an associative array.*/
// Create an empty array named response.
```

continues

LISTING 7-5 (continued)

```

$response = array();

// Assign cooresponding values to mandatory JSON response keys.
$response["layer"] = $value["layerName"];

// Use Gethotspots() function to retrieve POIs with in the search range.
$response["hotspots"] = Gethotspots( $db, $value );

// if there is no POI found, return a custom error message.
if ( empty( $response["hotspots"] ) ) {
    $response["errorCode"] = 20;
    $response["errorString"] = "No POI found. Please adjust the range.";
} //if
else {
    $response["errorCode"] = 0;
    $response["errorString"] = "ok";
} //else

function Gethotspots( $db, $value ) {

/* Create the SQL query to retrieve POIs within the "radius" returned from
GetPOI request.
    Returned POIs are sorted by distance and the first 50 POIs are
selected.
    The distance is caculated based on the Haversine formula.
    Note: this way of calculation is not scalable for querying large
database.
*/

// Use PDO::prepare() to prepare SQL statement.
// This statement is used due to security reasons and will
help prevent general SQL injection attacks.
// ":lat1", ":lat2", ":long" and ":radius" are named parameter markers for
// which real values
// will be substituted when the statement is executed.
// $sql is returned as a PDO statement object.
$sql = $db->prepare( "
    SELECT id,
           attribution,
           title,
           lat,
           lon,
           imageURL,
           line4,
           line3,
           line2,
           type,
           dimension,
           ((acos(sin((:lat1 * pi() / 180)) *
                sin((lat * pi() / 180)) +
                cos((:lat2 * pi() / 180)) *
                cos((lat * pi() / 180)) *
                cos((:long - lon) * pi() / 180))

```

```

        ) * 180 / pi()) * 60 * 1.1515 * 1.609344 * 1000)
        as distance
    FROM POI_Table
    HAVING distance < :radius
    ORDER BY distance ASC
    LIMIT 0, 50 " );

// PDOStatement::bindParam() binds the named parameter markers to the
// specified parameter values.
$sql->bindParam( ':lat1', $value['lat'], PDO::PARAM_STR );
$sql->bindParam( ':lat2', $value['lat'], PDO::PARAM_STR );
$sql->bindParam( ':long', $value['lon'], PDO::PARAM_STR );
$sql->bindParam( ':radius', $value['radius'], PDO::PARAM_INT );

// Use PDO::execute() to execute the prepared statement $sql.
$sql->execute();

// Iterator for the response array.
$i = 0;

// Use fetchAll to return an array containing all of the remaining rows
// in the result set.
// Use PDO::FETCH_ASSOC to fetch $sql query results and return each row
// as an array indexed by column name.
$pois = $sql->fetchAll(PDO::FETCH_ASSOC);

/* Process the $pois result */

// if $pois array is empty, return empty array.
if ( empty($pois) ) {

    $response["hotspots"] = array ();

} //if
else {

    // Put each POI information into $response["hotspots"] array.
    foreach ( $pois as $poi ) {

        // If not used, return an empty actions array.
        $poi["actions"] = array();

        // Store the integer value of "lat" and "lon" using predefined
// function ChangetoIntLoc.
        $poi["lat"] = ChangetoIntLoc( $poi["lat"] );
        $poi["lon"] = ChangetoIntLoc( $poi["lon"] );

        // Change to Int with function ChangetoInt.
        $poi["type"] = ChangetoInt( $poi["type"] );
        $poi["dimension"] = ChangetoInt( $poi["dimension"] );

        // Change to demical value with function ChangetoFloat
        $poi["distance"] = ChangetoFloat( $poi["distance"] );

        // Put the poi into the response array.

```

continues

LISTING 7-5 *(continued)*

```

        $response["hotspots"][$i] = $poi;
        $i++;
    }//foreach

}//else

return $response["hotspots"];
}//Gethotspots

// Convert a decimal GPS latitude or longitude value to an integer by
// multiplying by 1000000.
//
// Arguments:
//   value_Dec ; The decimal latitude or longitude GPS value.
//
// Returns:
//   int ; The integer value of the latitude or longitude.
//
function ChangetoIntLoc( $value_Dec ) {

    return $value_Dec * 1000000;

}//ChangetoIntLoc

// Change a string value to integer.
//
// Arguments:
//   string ; A string value.
//
// Returns:
//   Int ; If the string is empty, return NULL.
//
function ChangetoInt( $string ) {

    if ( strlen( trim( $string ) ) != 0 ) {

        return (int)$string;
    }
    else
        return NULL;
}//ChangetoInt

// Change a string value to float
//
// Arguments:
//   string ; A string value.
//
// Returns:
//   float ; If the string is empty, return NULL.
//

```



```

function ChangetoFloat( $string ) {
    if ( strlen( trim( $string ) ) != 0 ) {
        return (float)$string;
    }
    else
        return NULL;
} //ChangetoFloat

/* All data is in $response, print it into JSON format.*/

// Put the JSON representation of $response into $jsonresponse.
$jsonresponse = json_encode( $response );

// Declare the correct content type in HTTP response header.
header( "Content-type: application/json; charset=utf-8" );

// Print out Json response.
echo $jsonresponse;

/* Close the MySQL connection.*/

// Set $db to NULL to close the database connection.
$db=null;

?>

```

Testing the Layer

There are two ways to test your layer and ensure that it is working correctly. First, of course, you can test it by loading the Layar client on your mobile device, logging in, and accessing your layer via the Test category. Figures 7-16, 7-17, and 7-18 show how my layer appears.

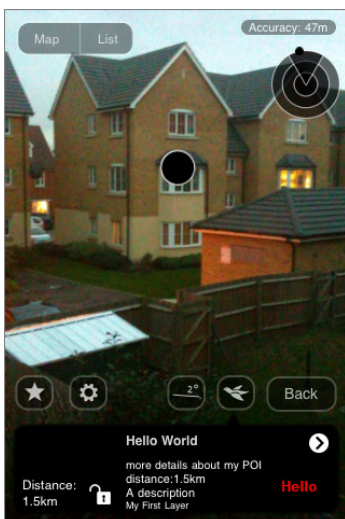


FIGURE 7-16: My POI in the Layar camera window

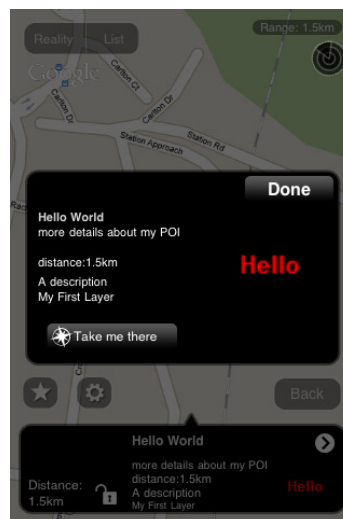


FIGURE 7-17: My POI in the map view



FIGURE 7-18: My POI in the list view

If there are any errors with the code, you won't get any useful debug information displayed; you'll get only no POI returned. Fortunately, via the Layar Dashboard, you can test your layer and access debug information. To test via the Layar Dashboard, go back the Layar web site (www.layar.com/publishing/#layers)

You should notice that there are some buttons to the right of your layer, including one that enables you to test your layer (see Figure 7-19).

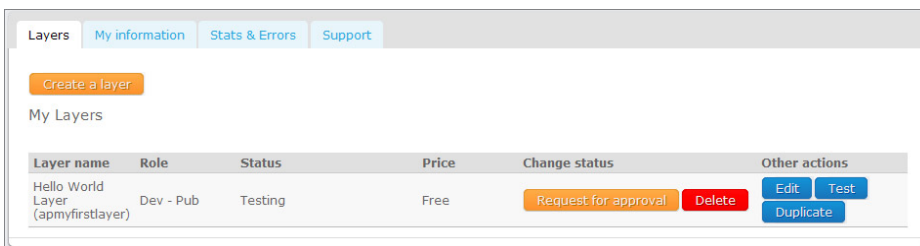


FIGURE 7-19: Testing in the Layar Dashboard

The test window defaults to Amsterdam, which is where the Layar company headquarters is based. You will probably notice that you have errors shown in the console window because there are no POIs in range. You will need to center the map on your location by typing your city or street into the geo-location search box. Once you have done this, the map will center on your location. You can make life easier by saving this location for future use and then selecting the Load layer button at the top of the screen. This should then load your layer in the test interface, as shown in Figure 7-20.

Logged in as **lestermadden** | My layers | Log out

Layar Layers Download Create Blog Support Jobs Search blog...

Test your layer

Layer name: [Load layer!](#)

Before request publication, please test your layer against layer testing instructions at <http://layar.pbworks.com/Layer-testing-instructions>

Layer: **apmyfirstlayer**

Map | Satellite | Hybrid

Filter Settings:

- Api version: 4.0
- Range slider: Search range (Current Value: 1500) 100 m 5000 m
- Accuracy: 100
- Country code (**): Afghanistan
- Language [en]: en - English

Current location: 51.59073, 0.0800816

Saved locations: [geo]: [Home \[x\]](#)

[Switch to coordinates](#)

Enter geolocation: [Go!](#)

[Save location](#) [Clear Map](#) [Get search center](#)

** Drag the person to change the center of your search.
If you can't select the person, zooming in/out once might help on some browsers.

Console:

```

Loading layer "undefined"...
found layer = apmyfirstlayer
Loading POIs
Loading POIs for page #1
oauth disabled...
FOI fwd url = http://www.augmentedplanet.com/wp-content/uploads/playground/layer/mylayer.php?lang=en&countryCode=Af&lon=4.887339&
userid=6f85d06929d160a7c8a3cc1ab4b54b87db99f74b&developerId=0&
developerHash=6ae05721079c8d2a6e5fb2f5eba7744fbcf07&version=4.0&radius=1500&
timestamp=1289667918821&lat=52.377544&layerName=apmyfirstlayer&accuracy=100
Response received from provider, validating...
response code : 200
Response validated, sending it back...
oauth disabled...FOI fwd url = http://www.augmentedplanet.com/wp-content/uploads/playground/layer/mylayer.php?lang=en&countryCode=Af&lon=4.887339&
userid=6f85d06929d160a7c8a3cc1ab4b54b87db99f74b&developerId=0&
developerHash=6ae05721079c8d2a6e5fb2f5eba7744fbcf07&version=4.0&radius=1500&
timestamp=1289667918821&lat=52.377544&layerName=apmyfirstlayers
accuracy=100,Response received from provider, validating...response code :
200,Response validated, sending it back...
Error: No POI found. Please adjust the range.
Loading layer "undefined"...
found layer = apmyfirstlayer
Loading POIs
Loading POIs for page #1
oauth disabled
    
```

POI List:

1	loc=51.585635,0.088377	Hello World
	type: 1	more details about my POI
	dimension: 1	distance:%distance
		A description
		My First Layer

Layers | Download | Create | Company

© 2010 Layar. All rights reserved. [Terms and Conditions](#) | [Privacy policy](#) | We are also on mobile: m.layar.com | Follow us on: [t](#) [f](#) [y](#) [v](#) [p](#)

FIGURE 7-20: Testing in Layar via the test UI

In the test interface, you will also notice that you have options to change the country code, accuracy, language, range, or the API version. These are parameters that are passed along with the query string and can be used to provide a custom experience to your users. Later, we'll look more at why these are useful; hopefully, you will have used the sample code and will not be faced with any errors. Your POI(s) should be shown in the test interface and you should have them working on the Layar mobile client. If that's not the case, you will need to use the error information provided in the test interface to troubleshoot the problem before you move to the next steps.

CUSTOMIZING YOUR LAYER

In the last section, you created a basic layer with a very basic listing for the Layar client. As you might have seen when testing out the other layers available, developers have gone to a lot of effort to stand out from the crowd, so you'll need to match their efforts to give a professional look and feel. Fortunately, Layar allows you to easily customize everything about your layer, so in this section you will learn how to create a more compelling listing working with the Dashboard. Additionally, you learn how to change the POI colors and how to create icon sets.

Creating a More Compelling Listing

So far, the listing you have created is lacking any excitement (refer to Figure 7-5) and doesn't really give users a compelling reason to try the layer. Of course, it's only a hello world layer, but there is no reason why it can't be the world's best hello world layer.

In the Layar dashboard, click the Edit button that appears next to your layer and then click Listing & indexing. This displays a screen (see Figure 7-21) where you can add useful information about the layer and make it more attractive to users.

As shown in Figure 7-21, I have uploaded two images files:

- An Icon 64×64 pixels in size
- A Screenshot 480×320 pixels in size

Logged in as [lestermadden](#) | [My layers](#) | [Log out](#)

[Layar](#) | [Layers](#) | [Download](#) | [Create](#) | [Blog](#) | [Support](#) | [Jobs](#) |




You are here: [Home](#) > [My layers](#)

lestermadden's layers and publishing info

[Layers](#) | [My information](#) | [Stats & Errors](#) | [Support](#)

[← Back to all layers](#)

Edit your "apmyfirstlayer" layer

General	Icon (64×64)	Upload Image Delete 	Click here for an example 
API endpoint	Screenshot (480×320)	Upload Image Delete 	
Listing & indexing	Title	<input type="text" value="Hello World Layer"/>	
Look & feel	Category	<input type="text" value="Fun"/>	
Coverage	Short description	<input type="text" value="My first layer"/>	
Filters	Detail description	<p>See the words 'hello world' displayed in augmented reality outside your own home. You wont believe your eyes.</p> <p>Learn more</p>	
Permissions	Tags	<input type="text" value="hello, world, layer, augmented, planet"/>	
Additional settings	Minimum API version	<input type="radio"/> Version 2.1 is the oldest supported version of the Layar API <input type="radio"/> Version 2.2 introduces audio and video actions on POIs <input checked="" type="radio"/> Version 3.0 introduces 3D Objects, Flexible Range, Checkbox List, Multiple Searchboxes, Multiple Custom Sliders, Auto-Triggered Actions, User authentication <input type="radio"/> Version 3.1 introduces paid layers <input type="radio"/> Version 3.5 introduces Layar Stream and texture animation on 3D objects <input type="radio"/> Version 4.0 introduces new interactive API features	
Pricing	Layar Stream options	<input checked="" type="radio"/> Allow Layar to index all POIs* for this layer (Recommended) (?) <input type="radio"/> Allow Layar to index only the title & position of the POIs of this layer. (?) <input type="radio"/> Don't allow Layar to index the POIs of this layer (?)	
	Expiration date (?)	<input type="text"/>	
	Time to live (?)	<input type="text" value="Long (1 week)"/>	
		Save Cancel	

Layers

Download

[Download](#)

[What is Layar](#)

[Supported devices](#)

[Features](#)

Create

[Create](#)

[Make layers](#)

[Examples](#)

[Platform Overview](#)

[Partners](#)

Company

[Press](#)

[Jobs](#)

[Contact](#)

[Blog](#)

© 2010 Layar. All rights reserved. [Terms and Conditions](#) | [Privacy policy](#)

We are also on mobile: [m.layar.com](#) | Follow us on: [t](#) [f](#) [g+](#) [v](#) [y](#)

FIGURE 7-21: Listing & indexing form

You'll notice in Figure 7-22 that I have been a creative with the screenshot by creating a collage of art work showing the layer in action. I did this to give the user a better feel for how the layer works.

As you complete the form, you can either create your own images using Microsoft Paint or a similar graphics package. If you use Paint, be sure to set the picture size by selecting the image, the attributes and specifying the relevant size. You can also download the images along with the source code from the Wiley site.

The rest of the options on the Listing and indexing form are as follows:

- **Title** is the name of your layer as it will appear to the user
- **Category** is the category where you want your layer to be listed on the client.
- **Short description** is a description of 60 characters or less that describes your layer.
- **Detail description** is your chance to explain what the layer does and why users should try it. This field access HTML, so you can create links and include additional images.
- **Tags** are a comma-separated list that will help users find your layer.
- **Minimum API version** of your layer is, by default, targeted at clients 3.0 and above. However, there may be users using older versions of Layar that you may want to target. Your layer will not appear in earlier versions of the client, so selecting 3.0 means any users with 2.x will not see your layer. You can also detect the client version by looking at the query string for the version parameter and programmatically creating a tailored user experience. Leaving it at 3.0 is recommended.
- **Layar Stream options** specify what indexing Layar should do on your POIs. The recommended option allows users to view all the content in your stream (for example, POIs nearby). If you were building a treasure hunt game, you could turn off the stream by selecting the Don't allow Layar to index POIs option. Users would not see any content appear. However, you could give them clues (such as *find the big clock*) and when they come into range, you could activate a proximity trigger.
- **Expiration dates** can be set for POIs and automatically removed from the stream. For example, if your layer displays the locations of beer festivals, they can automatically be removed once the event ends.
- **Time to live** is a setting that indicates how dynamic your content is.

The parameters shown in Figure 7-21 create a more complete listing in the Layar client.

In the Layar Dashboard, you can upload a screenshot of your layer by clicking the Upload Screenshot button shown next to the Screenshot (480×320) option. Additionally, the



FIGURE 7-22: The Listing & indexing client view

Detailed Description edit box accepts HTML, so you can also use an `` tag to refer to images:

```
<img src=http://www.augmentedplanet.com/wp-content/uploads/playground/layer/screenshot480x320.png>
```

You might decide to use this option if you want to use style features (such as boldfacing text in your listing).

Remember, any images you include must be downloaded to the client over 3G or slower. Use graphics sparingly and always test your layer while not connected to your Wi-Fi.

Changing POI Colors

In addition to creating a compelling listing, you can change the POI colors as they are displayed in the AR camera window. Right now, if you look at your POI, everything is black and white. The info bubbles are black with a white border; the location of the POI in the radar is difficult to see because it is black on a black background. To change the colors (perhaps to something that matches your company's brand), click the Look & feel link in the left navigation area.

As shown in Figure 7-23, I have used a variety of colors to complete the form. Of course, for your content, you might want to be a bit more restrained than I was. But you can see how these are transferred across to the client.

The screenshot shows a web interface for customizing a layer. On the left is a navigation menu with options: General, API endpoint, Listing & indexing, Look & feel (selected), Coverage, Filters, Permissions, Additional settings, and Pricing. The main content area is titled 'General' and contains several color selection fields, each with a color swatch and a hex code:

- Banner Icon (60x26): Includes 'Upload Image', 'Delete', and 'HW' buttons, and a link 'Click here for an example' with an image icon.
- Banner text color: 050500
- Banner Background Color: 34FA63
- Spot Color: FFF700
- Outer Color: AD0202
- Middle Color: 166B01
- Inner Color: F7F7F7
- BIW Background Color: 9C6C3D
- BIW Title Color: 0B11D6
- BIW Text Color: 1FFFF8

Below these is a section for 'Custom POI Indicator Widgets (CIW)' with four rows, each for a different POI type (Focus, Inner, Middle, Outer) with dimensions. Each row has 'Upload Image' and 'Delete' buttons. At the bottom left is an 'Add icon set' button. At the bottom center are 'Save' and 'Cancel' buttons, with a message 'Layer saved successfully' below them.

FIGURE 7-23: Changing the POI colors

To illustrate that the active POI changes color, I have added a couple more POIs to the database (see Figure 7-24).

In addition to the colors shown in Figure 7-24, POIs are banded together in the following groups:

- An inner group representing the POIs closest to the user's location.
- A middle group representing the POIs at mid distance from the user's location.
- An outer group representing the POIs furthest from the user.

Each band can have its own color attributes. However, using too many different colors may cause confusion with your users, be careful not to embark on a color techno-trip.

To add additional POIs to test, simply run the `INSERT INTO` query you created in an earlier step. Just take the exact same POI you added before, increase the ID value so it is unique, and add an arbitrary value such as 1000 to the latitude / longitude coordinates so that it will appear a little further way and not directly behind the other POI you created. That will create a nearby POI for you to view.

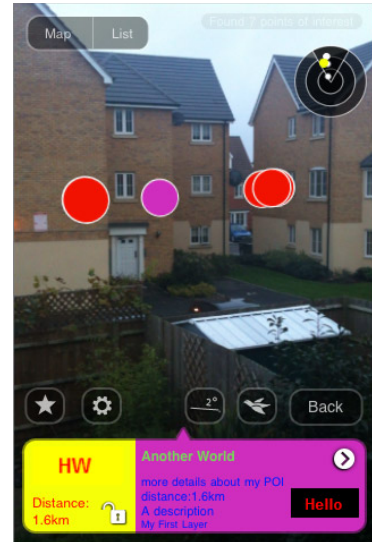


FIGURE 7-24: The new colors in the client

Creating Icon Sets

By default, POI information bubbles are displayed as circles. You can gain more control over how the POIs are displayed by creating icon sets and uploading PNG files. Using your own PNG files, you can customize how the POIs look and feel and build layers that represent the content. For example, a tree could be used to indicate a park while a house could be used to indicate property.

As shown in Figure 7-25, when using icon sets, you can specify different PNG files to represent the different groups.

- **POI Type 1 Focus (55×55)** represents the active POI.
- **POI Type 1 Inner (55×55)** represents the closest POI group.
- **POI Type 1 Middle (38×38)** represents the next group of POIs that are further from the Inner group but not as far away as the Outer group.
- **POI Type 1 Outer (28×28)** represents the POIs that are furthest away from the user.

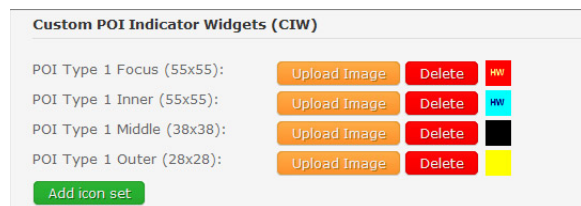


FIGURE 7-25: Creating icon sets

As shown in Figure 7-26, I used the icon sets to create an effect whereby the color changes for an active POI. Changing the color or the icon used in the icon set makes it easier to see which POI is

currently active. I know this is entirely a personal preference, but I find that square images lack the neatness of the default circles. Compare Figure 7-24 to 7-26 to see what I mean. I feel using icon sets give my layer a tacky quality. That said, you can achieve some nice results (for example, by using grayed-out icons to represent POIs that are in the distance).

What is interesting, however, is that when you use icon sets, you can create different sets of icons to represent your content. For example, let's imagine your MySQL table contains details about the location of fast-food restaurants. You could use different icons for different types of eateries (for example, you could use a burger icon for hamburger joint, a plate for table service, or chopsticks for a Chinese restaurant).

To use icons sets in this way the POI_Table you created in the database includes a Type field. Thus far, each POI record you have added to the table uses the value of 1 in the Type field. The value of 1 corresponds to the first POI icon set created via the Layar Dashboard. (It's shown with the name of POI Type 1 in the Dashboard.) Adding a new icon set creates a group called POI Type 2, POI Type 3, and so on. To apply these icon sets to the POIs, simply set the Type field to the corresponding value from the POI Type.

As shown in Figure 7-27, I have added three icon sets that each displays different colors. Of course, the images could represent anything.

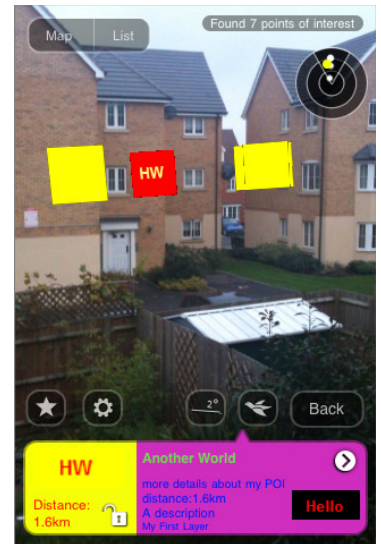


FIGURE 7-26: Icon sets in the camera window

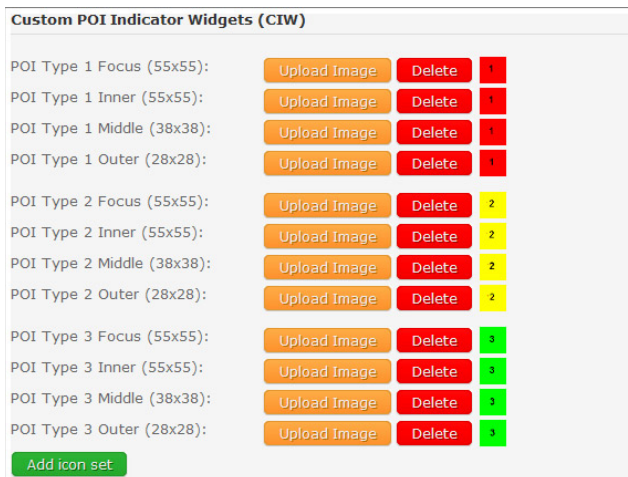


FIGURE 7-27: Creating multiple icon sets

In my database, I added a few new POIs and used the values 1, 2 and 3 accordingly in the Type field. These represent the POI Type 1, POI Type 2, and POI Type 3 icon sets. As shown in Figure 7-28, each POI now has its own color based on the icon set and type value.

Since you probably added a few new POIs in the previous steps, you'll find it relatively simple to add some icon sets yourself. If you are already a database whiz, just add a few new POIs and change the Type field value of each new record. Then create the new icon sets.

In a previous set of steps, you added the POI to the database using the MySQL query that follows. In the following steps, you will add a few more POIs:

1. Open your MySQL browser again and get to the point where you are looking at the query window shown in Figure 7-10.

Note in the SQL syntax below you will need to:

2. Increase the value of the `id` field so it is a unique value.
3. Change the longitude/latitude values either by using Google Maps to find locations near you or by adding 1000 to the coordinates.
4. Add a value to the Type field that corresponds to the icon set value.

The MySQL looks as follows:

```
INSERT INTO `POI_Table`
(`id`, `attribution`, `title`, `lat`, `lon`, `imageUrl`, `line4`, `line3`
, `line2`, `type`, `dimension`, `alt`, `relativeAlt`, `distance`, `inFocus`,
`doNotIndex`, `showSmallBiw`, `showBiwOnClick`)

VALUES

('<ID>', 'My First Layer', 'Hello World', '<latitude value>',
'<longitude value>', '<server>helloworld.png', 'A description',
'distance:%distance%', 'more details about my POI', <this is the type field>,
1, NULL, NULL, '0.0000000000', 0, 0, 1, 1);
```

Run the query a few times, each time changing the field values. You should see different icons shown in the client for your POIs.

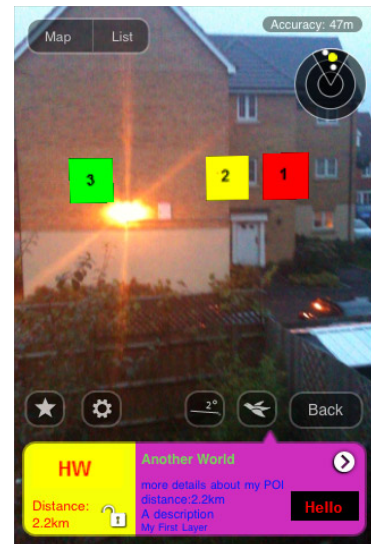


FIGURE 7-28: POI colors determined by icon set and type value

ADDING LAYAR ACTIONS

Layar supports a number of user actions that can be performed by the user. For example, you might have already seen the Take Me There button that calls up a Google Maps interface showing your location in relation to the POI. In addition to this Google Maps action, you can create actions that include:

- Open a web page
- Play an audio file
- Play a video file
- Make a telephone call
- Send an e-mail



For a full list of the supported actions available, please refer to the latest documentation from Layar at <http://layar.pbworks.com/w/page/30763878/Activity-types-for-POI-actions>.

In this section, you will learn how to extend your existing layer to support these actions. To do this:

1. Change from Version 3.0 of the API to Version 4.0.
2. Create a new actions table in your MySQL database.
3. Create a new function in your PHP code.

Changing to Version 4.0

You might remember when you created your layer and added the listing and indexing that one of the options on the form was to specify which version of the API your layer targeted. From the Layar dashboard, edit your layer and navigate your way to the Listing & indexing screen (see Figure 7-29). New actions (such as making calls or sending e-mails) were added in Version 3.0 of the API, but Version 4.0 is the recommended version to use because it includes additional changes to the action objects. Make the necessary change to the configuration setting and then save your changes.

Minimum API version

- Version 2.1 is the oldest supported version of the Layar API
- Version 2.2 introduces audio and video actions on POIs
- Version 3.0 introduces 3D Objects, Flexible Range, Checkbox List, Multiple Searchboxes, Multiple Custom Sliders, Auto-Triggered Actions, User authentication
- Version 3.1 introduces paid layers
- Version 3.5 introduces Layar Stream and texture animation on 3D objects
- Version 4.0 introduces new interactive API features
- Version 5.0 introduces simple object animations

FIGURE 7-29: API Version 4.0 support

Creating an Actions Table

You will need to create a new table in the MySQL database. This *actions table* will hold a list of the actions available for each POI.

Hopefully, you should be getting used to MySQL by now and should have no problem adding the SQL shown in Listing 7-6 to the database. If you're stuck, review the adding POIs topic at the "Adding POIs to the Database" section earlier in this chapter.



Available for
download on
Wrox.com

LISTING 7-6: Creating the ACTION_Table

```
CREATE TABLE IF NOT EXISTS `ACTION_Table` (
  `poiID` varchar(255) NOT NULL,
  `label` varchar(30) NOT NULL,
  `uri` varchar(255) NOT NULL,
  `autoTriggerRange` int(10) default NULL,
  `autoTriggerOnly` tinyint(1) default NULL,
  `ID` int(10) NOT NULL,
  `contentType` varchar(255) default 'application/vnd.layar.internal',
  `method` enum('GET','POST') default 'GET',
  `activityType` int(2) default NULL,
  `params` varchar(255) default NULL,
  `closeBiw` tinyint(1) default '0',
  `showActivity` tinyint(1) default '1',
  `activityMessage` varchar(255) default NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Once you press OK, you will see the table listed on the left navigation area.

The fields for the table you created are explained as follows:

- `poiID` is the id of the POI from the `POI_Table`.
- `label` is the label to display in the client. For example, the e-mail action might have the label of Contact Us.
- `uri` is the action that will be carried out (for example, `mailto://contactus@xyzcorp.com`).
- `autoTriggerRange` is the action automatically triggered when we come in range of the POI (the range is measured in meters).
- `autoTriggerOnly` is whether or not the action can be triggered when not in range and perhaps triggered manually.
- `ID` is the required unique identifier.
- `contentType` is the mime-type of content (for example, `audio.mpg`).
- `method` is the action type `GET` (the default) or `POST`.
- `activityType` is the numerical identifier to display the relevant icon for the action.
- `params` is a comma-separated list of parameters to be added to the request. For example, `params` will be added in the URL “`lat=45.462&lon=2.42346&alt=560`”.
- `closeBiw` determines where the user returns (POI view, AR, map, list, and so on) after the action has opened.
- `showActivity` is typically used with `autoTrigger` actions and informs the client when there are pending background activities relating to fetching an action result. For example, when `showActivity` is used with the `autoTrigger` parameter, you can choose to show or hide the resulting action.



In situations where multiple POIs have autoTrigger actions, only the closest POI will trigger. In addition, if the autoTriggerOnly parameter is set, the action will not appear when the user selects the POI outside the range.

Fetching the Actions Function

Now that you have created the table, you need to create a function that retrieves the actions for each POI. You can extend the mylayer.php file that you created previously to access the MySQL database and retrieve the POIs. Add the code shown in Listing 7-7 before the `?>` closing statement in the file.



Available for
download on
Wrox.com

LISTING 7-7: Fetching actions from the database

```
// Put fetched actions for each POI into an associative array.
// The returned values are assigned to $poi[actions].
//
// Arguments:
//   poi ; The POI handler.
//   $db ; The database connection handler.
//
// Returns:
//   array ; An array of received actions for this POI.
//
function Getactions( $poi, $db ) {

    // A new table called "ACTION_Table" is created to store actions,
    // each action has a field called
    // "poiID" which shows the POI id that this action belongs to.
    // The SQL statement returns actions which have the same poiID as the
    // id of $poi ($poi['id']).

    $sql_actions = $db->prepare( " SELECT label,
        uri,
        autoTriggerRange,
        autoTriggerOnly,
        contentType,
        method,
        activityType,
        params,
        closeBiw,
        showActivity,
        activityMessage
    FROM ACTION_Table
    WHERE poiID = :id " );

    // Binds the named parameter markers ":id" to the specified
    // parameter values "$poi['id']".
    $sql_actions->bindParam( ':id', $poi['id'], PDO::PARAM_INT );

    // Use PDO::execute() to execute the prepared statement $sql_actions.
```

continues

LISTING 7-7 *(continued)*

```

$sql_actions->execute();

// Iterator for the $poi["actions"] array.
$count = 0;

// Fetch all the poi actions.
$actions = $sql_actions->fetchAll( PDO::FETCH_ASSOC );

/* Process the $actions result */

// if $actions array is empty, return empty array.
if ( empty( $actions ) ) {

    $poi["actions"] = array();

} //if
else {

    // Put each action information into $poi["actions"] array.
    foreach ( $actions as $action ) {

        // Assign each action to $poi["actions"] array.
        $poi["actions"][$count] = $action;

        // put 'params' into an array of strings
        $paramsArray = array();

        if ( substr_count( $action['params'],' ' ) ) {
            $paramsArray = explode( " ", $action['params'] );
        } //if
        else if( strlen( $action['params'] ) ) {
            $paramsArray[0] = $action['params'];
        }

        $poi["actions"][$count]['params'] = $paramsArray;

        // Change 'activityType' to Integer.
        $poi["actions"][$count]['activityType'] =
            ChangetoInt( $poi["actions"][$count]['activityType'] );

        // Change the values of "closeBiw" into boolean value.
        $poi["actions"][$count]['closeBiw'] =
            ChangetoBool( $poi["actions"][$count]['closeBiw'] );

        // Change the values of "showActivity" into boolean value.
        $poi["actions"][$count]['showActivity'] =
            ChangetoBool( $poi["actions"][$count]['showActivity'] );

        // Change 'autoTriggerRange' to Integer, if the value is NULL,
        // return NULL.

```

```

$poi["actions"][$count]['autoTriggerRange'] =
    ChangetoInt( $poi["actions"][$count]['autoTriggerRange'] );

// Change the values of "autoTriggerOnly" into boolean value,
// if the value is NULL, return NULL.

$poi["actions"][$count]['autoTriggerOnly'] =
    ChangetoBool( $poi["actions"][$count]['autoTriggerOnly'] );

$count++;

} // foreach

} // else

return $poi["actions"];

} // Getactions

```

You also need to add the following support function to convert values to a boolean:

```

// Convert a TinyInt value to a boolean value TRUE or FALSE
// Arguments:
//   value_Tinyint ; The Tinyint value (0 or 1) of a key in the database.
// Returns:
//   value_Bool ; The boolean value, return 'TRUE' when Tinyint is 1.
//   Return 'FALSE' when Tinyint is 0.
//
function ChangetoBool( $value_Tinyint ) {

    if ( $value_Tinyint == 1 )
        $value_Bool = TRUE;
    else
        $value_Bool = FALSE;
    return $value_Bool;

} // ChangetoBool

```

Finally, you need to track down the following line of code in the `foreach` loop of the `Gethotspot` function:

```

// If not used, return an empty actions array.
$poi["actions"] = array();

```

Once you track it down, change it to the following:

```

// Use function Getactions() to return an array of actions associated with
// the current POI.
$poi["actions"] = Getactions ( $poi, $db );

```

Once you have finished editing the `.php` file, upload it to your server.

Adding Actions

In the previous two steps, you created the actions table and added the code necessary to inspect the actions table and retrieve any associated actions for your POIs. However, the actions table is currently empty, so in the following steps, you will add data to the table to make your POIs more interesting.

Since you already have a POI with the ID of 1, go ahead and apply some actions to that POI. You can either add the action by running the following SQL in the same way you have done in the past to create the POIs or, if you prefer, you can enter the action directly into the table.

1. To add directly to the table, open your MySQL database in your web browser.
2. From the left navigation, click ACTION_Table.
3. At the top navigation bar, click the Insert tab.

Figure 7-30 shows the MySQL options to directly add records to the ACTION_Table.

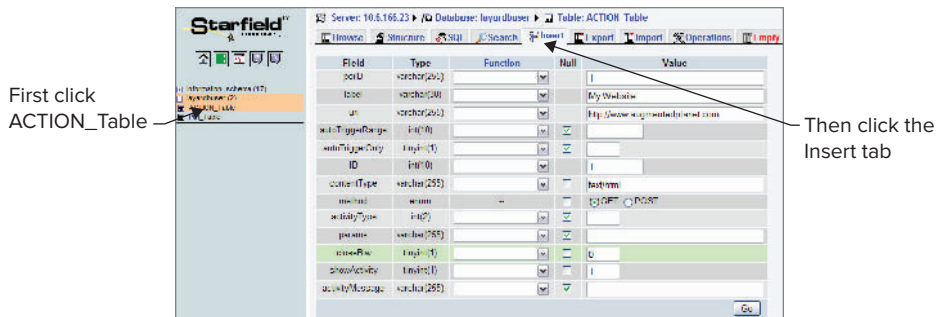


FIGURE 7-30: Adding directly to the table via the MySQL interface

Whether you choose to add the action via running a query or by inputting directly in the table via the MySQL UI, each method does the exact same thing as the other. Once you click the Go button to add the record, you can even see the MySQL that has been generated. The code shown in Listing 7-8 is useful if you are not sure what SQL syntax to use.



LISTING 7-8: Adding actions to the action table

```
INSERT INTO `layardbuser`.`ACTION_Table` (`poiID`, `label`, `uri`,
`autoTriggerRange`, `autoTriggerOnly`, `ID`, `contentType`, `method`,
`activityType`, `params`, `closeBiw`, `showActivity`, `activityMessage` )
```



```
VALUES ('1', 'My Website', 'http://www.augmentedplanet.com', NULL , NULL ,
'1', 'text/html', 'GET', NULL , NULL , '0', '1', NULL );
```

Listing 7-7 adds an action of type 1 (open a web page) to the POI in the POI_Table with the ID of 1. The label field is set to My Website, which will be the name of the button that will appear in the client. The contentType is set to text/html.

Other than those parameters, the default values are used. As shown in Figure 7-31, the POI now displays the My Website button.

Adding support for other actions (such as playing audio) is as simple as adding a new record to ACTION_Table specifying the contentType and which POI the action should be applied to. Table 8-1 shows the valid actions list.



FIGURE 7-31: POI with actions

TABLE 8-1: Actions Table

ACTION PERFORMED	ACTIVITYTYPE	CONTENTTYPE	URL
Opens web page	1	text/html	http:// or https://
Plays audio	2	audio/mpg	http:// or https://
Plays video	3	video/mp4	http:// or https://
Places a call	4	application/vnd.layar.internal	tel:
Sends e-mail	5	application/vnd.layar.internal	mailto:

With each activityType there are associated icons (see Figure 7-32).

As shown in Figure 7-33, I have added support for placing a call and sending an e-mail. Notice that in the current version of the Layar client, the iPhone doesn't pick up the activityType icon correctly, so there is slight difference between what is shown on the iPhone and what is shown on the Android. If your icons are not working as you expect, it is not necessarily a problem with your code.

Activity types / POI actions

	id	Example
	1	Info
	2	Audio
	3	Video
	4	Phone / Call
	5	Email / Message

FIGURE 7-32: activity Type icons

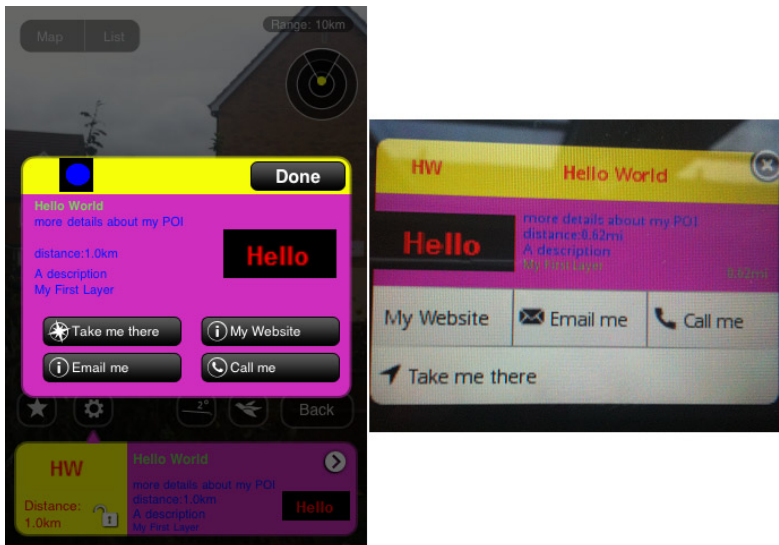


FIGURE 7-33: Actions in the iPhone/Android client

I'm sure now you're on your way to becoming a MySQL guru, so Listing 7-9 shows the SQL used to add multiple records to ACTION_Table.



LISTING 7-9: Adding multiple actions to the action table

Available for
download on
Wrox.com

```
INSERT INTO `layardbuser`.`ACTION_Table` (`poiID`, `label`, `uri`,
`autoTriggerRange`, `autoTriggerOnly`, `ID`, `contentType`, `method`,
`activityType`, `params`, `closeBiw`, `showActivity`, `activityMessage`)

VALUES

('1', 'Call me', 'tel:+44123456789', NULL, NULL, '2', 'application/vnd.layar
.internal', 'GET', 4, NULL, '0', '1', NULL),

('1', 'Email me', 'mailto:info@yourserver.com', NULL, NULL, '3', 'application/vnd
.layar.internal', 'GET', 5, NULL, '0', '1', NULL);
```

Adding Audio and Video

Both audio and video are supported and can make your POIs interesting. You'll hear me say this a lot: When you use either audio or video, test your layer off your Wi-Fi network. Downloading a 2 MB file is not a great user experience over a slow 3G connection. When using videos, your video must support progressive download so it can be streamed to the device. It should use the MP4 format because that format is supported on both the Android and the iPhone. You can use the .MOV format, but that format works only on an iPhone. For testing purposes, either search the web for a MP4 file that is a reasonable size or use your phone to capture 10 seconds of video and upload it to your server. Listing 7-10 shows the code for adding video/audio to ACTION_Table.

Available for
download on
Wrox.com**LISTING 7-10: Adding video/audio to the action table**

```

INSERT INTO `layardbuser`.`ACTION_Table` (`poiID` , `label` , `uri`
, `autoTriggerRange` , `autoTriggerOnly` , `ID` , `contentType` , `method`
, `activityType` , `params` , `closeBiw` , `showActivity` , `activityMessage` )

VALUES

('1', 'Video', '<server>video.mp4', NULL , NULL ,
'4', 'video/MP4', 'GET', 3, NULL , '0', '1', NULL ),

('1', 'Audio', '<server>trainsound.mp3', NULL , NULL , '5',
'audio/mpeg', 'GET', 2, NULL , '0', '1', NULL );

```



I use a .MOV file in Listing 7-10. Android developers will need to find an MP4 file to substitute.

Adding Triggers

Triggers enable you to specify when an action appears in the client and enables you to automatically trigger the action when a user comes in range. You will also hear these referred to as *Spots*. Spots can be useful for creating games or for providing users with functionality that is specific to their locations. For example, let's imagine you have restaurant POI. By default, it always gives the user the ability to browse the web site, e-mail the restaurant, or place a call to the restaurant, regardless of how far the user is away from the restaurant's location. However, when the user is within 500 meters, perhaps you want to automatically play a sound file that is related to the type of food served by the restaurant. To create this functionality, you must create triggers.

To test triggers for yourself, I suggest that you use Google Maps to find the latitude/longitude for a location that is about 50 meters away from your current location. This will save you some shoe leather; you won't have to walk far to trigger the actions. Once you have your latitude and longitude, update the POI_Table. Updating the existing record that you have added the actions to will save you the trouble of having to generate more actions. Load the Layar client and find the POI; once you have found it in the camera view and clicked it, Layar tells you how far away it is. For the sake of argument, let's assume Layar reports it as 50 meters away. You can now update your ACTION_Table and the autoTriggerRange field to 45 (meters). This means that when the user comes within 45 meters of this POI, the autoTriggerRange will be operational. Hopefully, it means need only to walk a few meters. Note, however, that GPS accuracy varies, so a location 50 meters away five minutes ago could be nearer or further away the next time you load the client.

If you still feel lazy, however, you can just keep manipulating the autoTriggerRange with a high and low value to simulate being in and out of range.

autoTriggerRange

If `autoTriggerRange` is included, then it indicates that the action is auto-triggered when in range of the POI. You can include only one `autotrigger`; if you include more than one, the first entry in the array is triggered and the others are ignored. It also appears that the action to automatically make a telephone call is not enabled (thankfully).

autoTriggerOnly

If the `autoTriggerRange` field has been set (is not NULL), this field should be a boolean value of true or false.

False indicates that the relevant button will be shown even outside the range and the user can trigger the action manually. The action is automatically triggered when the user is in range.

If `autoTriggerOnly` is set to true then the button is only visible when the user is within the `autoTriggerRange`, the action is triggered automatically when the user is in range.

Depending on the type of mobile device he's using, the user may see a message indicating that the object has an automatic action which he can either accept or ignore.

Your `ACTION_Table` will look something like the table shown in Figure 7-34.

	poiID	label	uri	autoTriggerRange	autoTriggerOnly	ID	contentType	method	activityType	params	closeBtn	showActivity
	1	My Website	http://www.augmentedplanet.com	NULL	NULL	1	text/html	GET	1	NULL	0	1
	1	Email me	mailto:info@yourserver.com	NULL	NULL	3	application/vnd.layar.internal	GET	5	NULL	0	1
	1	Call me	tel:+44123456789	NULL	NULL	2	application/vnd.layar.internal	GET	4	NULL	0	1
	1	Video	http://www.augmentedplanet.com/wp-content/uploads/...	NULL	NULL	4	video/MP4	GET	3	NULL	0	1
	1	Audio	http://www.augmentedplanet.com/wp-content/uploads/...	NULL	NULL	5	audio/mpeg	GET	2	NULL	0	1

Check All / Uncheck All With selected:

FIGURE 7-34: Your `ACTION_Table`

Update one of your actions by setting the `autoTriggerRange` to a numerical value (for example, 10). In addition, set the `autoTriggerOnly` to true. When you come within 10 meters of the POI you updated, you should see the button appear and the action automatically trigger when you are in range.

SUMMARY

In this chapter, you configured your first layer for the Layar browser. Much of what you did here sets the groundwork for future chapters. After completing this chapter, you have successfully created the Layar database and web service that is used to retrieve the data from the MySQL database. You also learned how to customize your layer by creating the listings and by uploading graphics. In addition, you further customized your layer by creating icon sets and changing the colors of the info bubbles that appear in the camera window. To make your POIs more engaging for users, you learned how to add actions to your layer. These actions enabled you to include video or audio, web site links, or even provide the user with the ability to contact you. Finally, you got your first taste of triggers, which enable you to determine when actions should be triggered.

8

Creating Filters and 2D Objects

WHAT'S IN THIS CHAPTER?

- ▶ How to add filtering options
- ▶ How to extend database tables
- ▶ How to add 2D objects to the layer

In the previous chapter, you created your first layer. You now have the database configured and you should be familiar enough with MySQL to be able to add new POIs and add actions to your POIs. In this chapter, you will build upon your knowledge of Laya by adding advanced functionality (such as radio buttons or checkboxes to filter content). Before you begin, make sure you've read Chapter 7.



Since you may want to keep the layer you created in the previous chapter, make a copy of your work by taking a copy of the `mylayer.php` file. For the sample code used in this chapter, I have created a new layer with the name `aprealstate` by copying the `mylayer.php` file created in the previous chapter to create a new file called `realestate.php`. As you upload the new `realestate.php` file, remember the URL because this will be the endpoint you enter when you create the layer.

Also remember that the name you give to the layer needs to be unique. Since I have already taken `aprealstate`, you will need to create a layer with a slightly different name.

In the Layar Dashboard you should create a new layer using the settings shown in Table 8-1.

TABLE 8-1: Configuration for the real estate layer

PARAMETER / SETTING	VALUE
Layar name	Unique lowercase no spaces
Title	Real Estate Finder
Publisher name	Leave to the default
Layar type	Generic 2D
API endpoint URL	URL to the realestate.php file on your server
Short description	My real estate layer showing how to use filters

By default, new layers are set to use Version 3.0 of the API. You should use the latest version of the API so you can take advantage of the latest features and fixes, so be sure to change the API version to Version 4.0 in the Look & Feel settings of your layer's Dashboard once it has been created. Figure 8-1 shows the Minimum API version section on the Create a Layer page.

Create a Layer

Layer name:

Title:

Publisher name:

Layer type:

API endpoint URL:

Short description:

Minimum API version

- Version 2.1 is the oldest supported version of the Layar API
- Version 2.2 introduces audio and video actions on POIs
- Version 3.0 introduces 3D Objects, Flexible Range, Checkbox List, Multiple Searchboxes, Multiple Custom Sliders, Auto-Triggered Actions, User authentication
- Version 3.1 introduces paid layers
- Version 3.5 introduces Layar Stream and texture animation on 3D objects
- Version 4.0 introduces new interactive API features

FIGURE 8-1 Creating a new layer and choosing an API version

USING FILTERS

If you have a layer that has lots of POIs, you may want to provide the user with a way to limit the information returned in the camera window. For example, if your layer returns apartments for rent in the user's immediate vicinity, there could be hundreds displayed. But perhaps the user is interested only in apartments with two bedrooms and up to a maximum price of \$1,200 per month. By using filters, you can provide a way for users to select their criteria and limit the results displayed.

You will use the Laya Dashboard to create the filters and UI elements and then add the functionality to the PHP file that will apply the filter criteria.

Let's start with a quick overview on how filters work. In the Laya Dashboard, edit your layer and navigate to the Filters option on the left of the screen. By default, you already have a slider control (see Figure 8-2) with the Label of Search range. You may have already seen this slider control in your layer; this particular control is configured to control the search distance of your POIs. As shown in this figure, the slider is configured to display POIs within a maximum range of 5,000 (meters) and is set to a default search range of 1,500 (meters).

Range Slider	
Label:	Search range
Min Value:	100
Max Value:	5000
Default Value:	1500

▲
▼
Remove Filter

FIGURE 8-2: The default filter

Although you can't see it in this figure, this filter is named *radius* and the radius value is passed as a parameter to the endpoint function along with the query string, as shown in the following code:

```
http://www.yoursever.com/hellofilters.php?lang=en&countryCode=AF&lon=
4.887339&userId=6f85d06929d160a7c8a3cc1ab4b54b87db99f74b&developerId=
0&developerHash=f7ecc1fexxxxxxxxxxxxxxxxxxca4398baf5629a8&version=4.0&
radius=1500&timestamp=1290876467822&lat=52.377544&layerName=
aphellofilter&accuracy=100
```

In the code you created in the previous chapter, you created an array and added the radius value from the query string into the array, as shown here:

```
// Put needed parameter names from GetPOI request in an array called $keys.
$keys = array("layerName", "lat", "lon", "radius" );
```

This was then used in the SQL to limit the results so the layer displayed only results within the user's search range. The process for any new filters you create is:

1. Add the relevant filter control via the Laya Dashboard.
2. Take the new parameter from the query string and add in to the array.
3. Modify the SQL to limit the search results.

The steps used to create the real estate database to hold the POIs will also be used to create the release estate sample layer.

Creating the Real Estate Database

The structure for the real estate database is identical to the database you created in the previous chapter but with the addition of a few extra fields that are relevant to what you might find in a production real estate layer.

The new field names you will use are:

- **Radiolist:** Indicates if the property is for sale or rent
- **Checkbox:** Indicates the type of property
- **Custom slider:** Indicates property price

These field names aren't particularly user-friendly, but these are the parameter names that will be passed to our function via the query string.

To complete this chapter, you will need the latitude and longitude coordinates for several locations near you. You will use some fictitious street names but you will need to use the latitude and longitude coordinates you harvested for the locations near you.

Creating the Database

You created several tables in the previous chapter, so I'm sure you are well onto becoming a MySQL database expert now. Load your SQL tab in the MySQL window and execute the query shown in Listing 8-1.



LISTING 8-1: POI_RealEstate_Table

Available for
download on
Wrox.com

```
CREATE TABLE IF NOT EXISTS `POI_RealEstate_Table` (
  `id` varchar(255) NOT NULL,
  `attribution` varchar(150) default NULL,
  `title` varchar(150) NOT NULL,
  `lat` decimal(20,10) NOT NULL,
  `lon` decimal(20,10) NOT NULL,
  `imageUrl` varchar(255) default NULL,
  `line4` varchar(150) default NULL,
  `line3` varchar(150) default NULL,
  `line2` varchar(150) default NULL,
  `type` int(11) default '0',
  `dimension` int(1) default '1',
  `alt` int(10) default NULL,
  `relativeAlt` int(10) default NULL,
  `distance` decimal(20,10) NOT NULL,
  `inFocus` tinyint(1) default '0',
  `doNotIndex` tinyint(1) default '0',
  `showSmallBiw` tinyint(1) default '1',
```



```

`showBiwOnClick` tinyint(1) default '1',
`Radiolist` enum('sale','rent') default 'sale',
`Checkbox` enum('1','2') default NULL,
`Custom_Slider` int(7) default NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

```

Once the SQL has executed, you will see the table named `POI_RealEstate_Table` added to the database navigation on the left.

Adding the Real Estate Data

Now that you have the table successfully created, you need to add sample data to the table. There are two ways to do this:

- Open the table in the MySQL editor (see Figure 8-3) and add each record to the table individually.
- Or copy the MySQL below and execute it in the MySQL client:

Field	Type	Function	Null	Value
id	varchar(255)			1
attribution	varchar(150)		<input type="checkbox"/>	aprealestate
title	varchar(150)			Water Street
lat	decimal(20,10)			51.583609
lon	decimal(20,10)			0.061769
imageUrl	varchar(255)		<input type="checkbox"/>	m/book/layer/houses/apartment_water.jpg
line4	varchar(150)		<input type="checkbox"/>	Large apartment by the river
line3	varchar(150)		<input type="checkbox"/>	Distance:%distance%
line2	varchar(150)		<input type="checkbox"/>	2 bedrooms
type	int(11)		<input type="checkbox"/>	1
dimension	int(1)		<input type="checkbox"/>	1
alt	int(10)		<input checked="" type="checkbox"/>	
relativeAlt	int(10)		<input checked="" type="checkbox"/>	
distance	decimal(20,10)			
inFocus	tinyint(1)		<input type="checkbox"/>	0
doNotIndex	tinyint(1)		<input type="checkbox"/>	0
showSmallBiw	tinyint(1)		<input type="checkbox"/>	1
showBiwOnClick	tinyint(1)		<input type="checkbox"/>	1
Radiolist	enum	--	<input type="checkbox"/>	<input checked="" type="radio"/> sale <input type="radio"/> rent
Checkbox	enum	--	<input type="checkbox"/>	<input type="radio"/> 1 <input checked="" type="radio"/> 2
Custom_Slider	int(7)		<input checked="" type="checkbox"/>	550000

FIGURE 8-3: Editing the MySQL database

If you are adding the records manually to the database, use the information provided in Table 8-2.

TABLE 8-2: POI_RealEstate_Table

ID	ATTRIBUTION	TITLE	LAT	LON	IMAGE URL
1	<your layer name>	Water Street	<latitude>	<longitude>	<http://www.yourserver.com/apartment_water.jpg
2	<your layer name>	Water Street	<latitude>	<longitude>	<http://www.yourserver.com/apartment_water.jpg
3	<your layer name>	Poor Street	<latitude>	<longitude>	<http://www.yourserver.com/apartment.jpg
4	<your layer name>	Rich Road	<latitude>	<longitude>	<http://www.yourserver.com/apartment3.jpg
5	<your layer name>	Poor Hill	<latitude>	<longitude>	<http://www.yourserver.com/house.jpg
6	<your layer name>	Poor Street	<latitude>	<longitude>	<http://www.yourserver.com/house2.jpg
7	<your layer name>	Super Rich Street	<latitude>	<longitude>	<http://www.yourserver.com/house3.jpg
8	<your layer name>	Main Street	<latitude>	<longitude>	<http://www.yourserver.com/house5.jpg
9	<your layer name>	Central Lane	<latitude>	<longitude>	<http://www.yourserver.com/house6.jpg
10	<your layer name>	Hobbit Hill	<latitude>	<longitude>	<http://www.yourserver.com/house_green.jpg

LINE4	LINE3	LINE2	TYPE	RADIOLIST	CHECKBOX	CUSTOM_SLIDER
Large apartment by the river	Distance:% distance%	2 bedrooms	1	Sale	2	550000
Nice water view apartment	Distance:% distance%	3 bedrooms	1	Rent	2	3000
Small apartment	Distance:% distance%	2 bedrooms	1	Sale	2	120000
Nice apartment in a good area	Distance:% distance%	4 bedrooms	1	Sale	2	420000
House on the hill	Distance:% distance%	3 bedrooms	1	Rent	1	3100
Needs a little work	Distance:% distance%	4 bedrooms	1	Sale	1	70000
Very nice house in a rich area	Distance:% distance%	4 bedrooms	1	Sale	1	800000
House on Main Street	Distance:% distance%	7 bedrooms	1	Sale	1	950000
House with a large garage	Distance:% distance%	2 bedrooms	1	Rent	1	4500
Unique underground house	Distance:% distance%	2 bedrooms	1	Rent	1	5200

If you don't want to add the records individually, use the MySQL in Listing 8-2 to add the records. You will need to change the following fields before you execute the query:

- <your layer name> is the unique name of your layer.
- <lat> is the latitude location of your POI.
- <lon> is the longitude location of your POI.



Available for
download on
Wrox.com

LISTING 8-2: MySQL bulk update query that populates the POI_RealEstate_Table

```
INSERT INTO `layardbuser`.`POI_RealEstate_Table` (
  `id` ,
  `attribution` ,
  `title` ,
  `lat` ,
  `lon` ,
  `imageUrl` ,
  `line4` ,
  `line3` ,
  `line2` ,
  `type` ,
  `dimension` ,
  `alt` ,
  `relativeAlt` ,
  `distance` ,
  `inFocus` ,
  `doNotIndex` ,
  `showSmallBiw` ,
  `showBiwOnClick` ,
  `Radiolist` ,
  `Checkbox` ,
  `Custom_Slider`
)
VALUES (
  '1', '<your layer name>', 'Water Street', '<lat>', '<lon>',
  'http://www.yourserver.com/apartment3_water.jpg', 'Large apartment by the
  river', 'Distance:%distance%', '2 bedrooms', '1', '1', NULL , NULL ,
  '', '0', '0', '1', '1', 'sale', '2', '550000'),

  ('2', '<your layer name>', 'Water Street', '<lat>', '<lon>',
  'http://www.yourserver.com/apartment3_water.jpg', 'Large apartment by the
  river', 'Distance:%distance%', '3 bedrooms', '1', '1', NULL , NULL ,
  '', '0', '0', '1', '1', 'rent', '2', '3000'),

  ('3', '<your layer name>', 'Poor Street', '<lat>', '<lon>',
  'http://www.yourserver.com/apartment3.jpg', 'Small apartment',
  'Distance:%distance%', '2 bedrooms', '1', '1', NULL , NULL , '', '0', '0',
  '1', '1', 'sale', '2', '120000'),

  ('4', '<your layer name>', 'Rich Road', '<lat>', '<lon>',
  'http://www.yourserver.com/appartment4.jpg', 'Nice apartment in a
  rich area', 'Distance:%distance%', '4 bedrooms', '1', '1', NULL ,
  NULL , '', '0', '0', '1', '1', 'sale', '2', '420000'),

  ('5', '<your layer name>', 'Poor Hill', '<lat>', '<lon>',
```

```

'http://www.yourserver.com/house.jpg', 'House on the hill',
'Distance:%distance%', '4 bedrooms', '1', '1', NULL , NULL ,
'', '0', '0', '1', '1', 'rent', '1', '3100'),

('6', '<your layer name>', 'Poor Street', '<lat>', '<lon>',
'http://www.yoursever.com/house2.jpg', 'Needs a little work',
'Distance:%distance%', '4 bedrooms', '1', '1', NULL , NULL ,
'', '0', '0', '1', '1', 'sale', '1', '70000'),

('7', '<your layer name>', 'Super Rich Street', '<lat>', '<lon>',
'http://www.yourserver.com/house3.jpg', 'Very nice house in a
rich area', 'Distance:%distance%', '4 bedrooms', '1', '1', NULL ,
NULL , '', '0', '0', '1','1', 'sale', '1', '800000'),

('8', '<your layer name>', 'Main Street', '<lat>', '<lon>',
'http://www.yourserver.com/house5.jpg',
'House on main street',
'Distance:%distance%', '7 bedrooms', '1', '1', NULL , NULL ,
'', '0', '0', '1', '1', 'sale', '1', '950000'),

('9', '<your layer name>', 'Central Lane', '<lat>', '<lon>',
'http://www.yourserver.com/house6.jpg',
'House with a large garage',
'Distance:%distance%', '2 bedrooms', '1', '1', NULL , NULL ,
'', '0', '0', '1', '1', 'rent', '1', '4500'),

('10', '<your layer name>', 'Hobbit Hill', '<lat>', '<lon>',
'http://www.yourserver.com/house_green.jpg',
'Unique underground house',
'Distance:%distance%', '2 bedrooms', '1', '1', NULL , NULL,
'', '0', '0', '1', '1', 'rent', '1', '5200');

```



In the table update shown in Listing 8-2, `type = 1` sets the type of icon that is used.

So far, you have created the database table and created 10 POIs from locations near your current position. Of course, Hobbit Hill and Super Rich Street won't be in your immediate vicinity, but the sample data gives you enough of an idea to test out filters. There are two additional steps that you'll need to take:

1. Create the filters in the Laya Dashboard.
2. Add the code to read the query string passed from Laya and filter the results.

Creating the Filters

In this step, you will create the filters in the Laya Dashboard that enable users to filter the results based on their requirements. The filters will allow users to search property based on whether it is for sale or for rent, what the price is, and what type of property it is. You have already created the fields in the table that hold the relevant values.

```

`Radiolist` enum('sale','rent') default 'sale',
`Checkbox` enum('1','2') default NULL,
`Custom_Slider` int(7) default NULL,

```

In addition to searching on these values, you will also add a filter to enable users to search for properties by entering a street name.

To add the filters, go into the Layar Dashboard for your real estate layer and click the Filters section (see Figure 8-4).

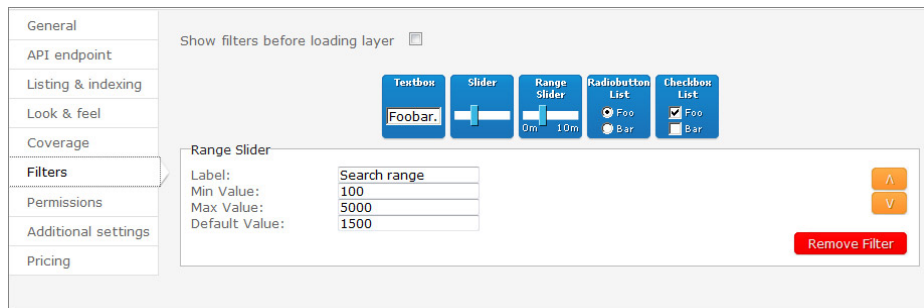


FIGURE 8-4: The default search range filter

You have probably seen this filter already in your layer, particularly if you have changed the search distance for your POIs. This filter is provided by default since it is always a good idea to enable users to search their radius. This particular filter uses the Range Slider control, is named Search range, has a minimum range of 100 (meters), a maximum range of 5,000 (meters), and the default range the slider control is set to when the layer is opened is 1,500 (meters). Users can use the slider control to select any value that falls within these values. Near the top of the screen, five types of filters are available, all of which you will use in the sample real estate layer:

- Textbox
- Slider
- Range Slider
- Radiobutton List
- Checkbox List

Textbox

The Textbox filter enables users to type in any freeform text. This is useful for enabling searches where it is not possible to limit the choices by radio buttons. In this sample, the textbox filter enables users to enter the names streets where they are looking for properties.

Slider

The Slider filter is very much like the Range Slider that is used for controlling the distance except that it includes an additional field where you can enter the exact unit of measurement. For the Range Slider, the measurement is meters (but there is nothing in the UI that tells us this). For your real estate layer, use the Slider filter to indicate the cost of the property.

Radiobutton List

If you have done any programming at all, you know that radio buttons are mutually exclusive options for selecting just one option from a list. In the real estate layer, use the Radiobutton List to filter on property that is for sale or property that is for rent.

Checkbox List

While radiobuttons are exclusive (only one option can be selected), the Checkbox List enables users to select as many options from the list as required. Use this to enable users to search for apartments, houses, or both.

The rest of the form is not exciting. You can click the up/down arrows to move your filters into a more logical order. The Show filters before loading layer option, which is disabled by default, can be selected if you want to present the filters to users when the layer is loaded and before the augmented reality view is displayed. For a real-estate layer that could potentially present hundreds of POIs to users, enabling this feature is a good way to limit how much content is initially downloaded to the device so users can begin searching more quickly.

For our example here, select the Show filters before loading the layer. Additionally, create the filters and values shown in Table 8-3. Leave the existing Range Slider as it is.

TABLE 8-3: Filter configuration

FILTER	VALUE
Text Box	
Label:	Street or road name
Default Value:	
Radio Buttons	
Label:	Looking to
Value 1:	Buy
Value 2:	Rent
Checkbox List	
Value 1:	Houses (select default)
Value 2:	Apartments (select default)
Slider	
Label:	Price Range
Label Unit:	\$
Min Value:	1000
Max Value:	1000000
Default Value:	300000

Once you have created each type of filter control, your screen should look similar to Figure 8-5.

Logged in as **lestermadden** | **My layers** | [Log out](#)

layar | [Layers](#) | [Download](#) | [Create](#) | [Blog](#) | [Support](#) | [Jobs](#) |

You are here: [Home](#) > [My layers](#)

lestermadden's layers and publishing info

[Layers](#) | [My information](#) | [Stats & Errors](#) | [Support](#)

[← Back to all layers](#)

Edit your "aphellofilter" layer

General
 Show filters before loading layer:

Filters

Range Slider

Label:

Min Value:

Max Value:

Default Value:

Text Box

Label:

Default Value:

Radio Buttons

Label:

Options:

Value	Display Text	Default
<input type="text" value="1"/>	<input type="text" value="Buy"/>	<input checked="" type="radio"/>
<input type="text" value="2"/>	<input type="text" value="Rent"/>	<input type="radio"/>

Checkbox List

Label:

Options:

Value	Display Text	Default
<input type="text" value="1"/>	<input type="text" value="Houses"/>	<input checked="" type="checkbox"/>
<input type="text" value="2"/>	<input type="text" value="Apartments"/>	<input checked="" type="checkbox"/>

Slider

Label:

Label Unit:

Min Value:

Max Value:

Default Value:

Layers | **Download** | **Create** | **Company**

Download: [Download](#), [What is Layar](#), [Supported devices](#), [Features](#)
 Create: [Create](#), [Make layers](#), [Examples](#), [Platform Overview](#), [Partners](#)
 Company: [Press](#), [Jobs](#), [Contact](#), [Blog](#)

© 2010 Layar. All rights reserved. [Terms and Conditions](#) | [Privacy policy](#) | We are also on mobile: [m.layar.com](#) | Follow us on: [ht](#) [fb](#) [yt](#) [in](#) [tw](#)

FIGURE 8-5: Real estate filter settings

Load your layer on your phone. As soon as your layer loads, you should be presented with the filter screen shown in Figure 8-6, enabling you to adjust the search options.

Of course, the filters don't do anything yet because you haven't connected them to the database and added the relevant code. You'll add this code in the next section.

Connecting the Filters

Now that the database is configured with sample data and the filters provide a means for the user to filter the data, you need to tie the two pieces together.

In the Laya Dashboard, return to the list of layers (see Figure 8-7) and select the Test option for the Real Estate Finder.

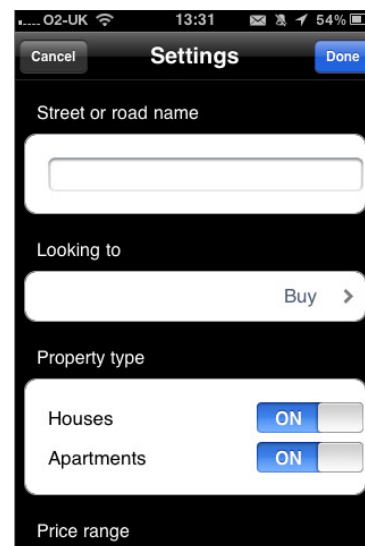


FIGURE 8-6: Real estate filters

Layer name	Role	Status	Price	Change status	Other actions
Hello World Layer (apmyfirstlayer)	Dev - Pub	Testing	Free	Request for approval Delete	Edit Test Duplicate
Real Estate Finder (aphellofilter)	Dev - Pub	Testing	Free	Request for approval Delete	Edit Test Duplicate

FIGURE 8-7: Testing the Real Estate Finder

Earlier in this chapter, I briefly described how the parameters are passed to the endpoint function via the query string and that in your code, you need to add the parameters to the array. To see the parameters that will be passed to the endpoint function, use the Laya test page:

```
http://www.yourserver.com/realestate.php?lang=en&countryCode=AF&lon=4.887339
&userId=6f85d06xxxxxxxxxxxxxxxxxxxxxxxxdb99f74b&developerId=0&developerHash=
2f00f338abf35fe827ee2114e6431f628f98fff&RADIOLIST=1&CHECKBOXLIST=2&version=
4.0&radius=1500&CUSTOM_SLIDER=30000&timestamp=1291470516281&lat=52.377544&
layerName=aprealestate&SEARCHBOX=&accuracy=100
```

The highlighted fields (except for the free text search box) have the corresponding fields in the database:

```
`Radiolist` enum('sale','rent') default 'sale'
`Checkbox` enum('1','2') default NULL)
`Custom_Slider` int(7) default NULL)
```



When you click the Test option to test a layer, the Layar test page displays. At the bottom of the page, you will see a Console section. This is where you will find the list of parameters that are being passed to your endpoint function.

As revealed in the query string, you must extend the code in `realestate.php` to add the parameters to the array. You need only to add the parameters that you are actually interested in; you don't need to add all the parameters in the query string.

Open the `realestate.php` file and locate the following lines of code (which should be near the top of the file):

```
// Put needed parameter names from GetPOI request in an array called $keys.
$keys = array("layerName", "lat", "lon", "radius" );
```

Amend it to include the filters you created:

```
// Put needed parameter names from GetPOI request in an array called $keys.
$keys = array( "layerName", "lat", "lon", "radius",
              "RADIOLIST", "CHECKBOXLIST",
              "CUSTOM_SLIDER", "SEARCHBOX" );
```

The code is now configured to search the query string for the filters.

Using SQL Queries

Of course, the filters by themselves are not very useful. You need to take the values that have been passed and add them into the query. Then action the query against the MySQL database to filter based on the users' input. However, before the parameters can be submitted to the SQL, they need to be cleaned up a little.

The following functions should be added to your `realestate.php` file. The functions can be added to the bottom of the file before the closing `?>` statement.

Searchbox Parameter

The `searchbox` parameter (see Listing 8-3) is the Textbox filter you created in the Layar Dashboard. Use this control to enable users to optionally enter street names to search.



Available for
download on
Wrox.com

LISTING 8-3: Realestate.php (searchbox function)

```
// Prepare the search value which will be used in SQL statement.
// Arguments:
//   searchbox ; the value of SEARCHBOX parameter in the GetPOI request.
//
// Returns:
//   searchbox_value ; If searchbox parameter has an empty string,
//   return a string which is a combination of numbers, letters
//   and white spaces. Otherwise, return the value of searchbox parameter.
```

```

function GetSearchValue ( $searchbox ) {

    // if $searchbox exists, prepare search value.
    if ( isset($searchbox) ) {

// initiate searchbox value and set it to any string that consists of
// numbers, letters and spaces using regular expression.
        $searchbox_value = '[0-9a-zA-Z\s]*';

        // if $searchbox is not an empty string, return the $searchbox value.
        if ( !empty($searchbox) )
            $searchbox_value = $searchbox;

        return $searchbox_value;
    } //if
    else {

// If $searchbox does not exist, throw an exception.
// throw new Exception("searchbox parameter is not passed in GetPOI
// request.");

    } //else

} // GetSearchValue

```

radiolist Parameter

The `radiolist` parameter (see Listing 8-4) enables users to select properties that are available for sale or rent. In the `POI_RealEstate_Table`, you created a field called `Radiolist` that contains the value of sale or rent. However, the corresponding `radiolist` parameter passes a numerical value to indicate the options. This function converts the `radiolist` parameter value to the text equivalent:

- `radiolist value 1` converts to sale
- `radiolist value 2` converts to rent



LISTING 8-4: radiolist function

Available for
download on
Wrox.com

```

// Prepare radiolist value which will be used in SQL statement.
// In this function, we convert the returned value into the ones that are
// stored in the database.
//
// Arguments:
//   radiolist ; the integer value of radiolist parameter in the
//   GetPOI request.
//
// Returns:
//   radio_value ; the value that can be used to construct the right SQL
//   statement.

function GetRadioValue ($radiolist) {
    // if $radiolist exists, prepare radio_value.
    if( isset( $radiolist ) ) {

```

continues

LISTING 8-4 (continued)

```

$radio_value;
// if $radiolist == 1, return $radio_value ="sale";
// if $radiolist == 2, return $radio_value ="rent";
switch ($radiolist) {
  case '1':
    $radio_value = "sale" ;
    break;
  case '2':
    $radio_value = "rent" ;
    break;
  default:
    throw new Exception( "invalid radiolist value:".$radiolist );
} //switch

return $radio_value;
} //if
else {
  throw new Exception("radiolist parameter is not passed in GetPOI
    request.");
} //else

```

checkboxlist Parameter

The `checkboxlist` parameter (see Listing 8-5) enables the user to specify what type of property they are searching for. Users can indicate houses, apartments, or even both. This function will check to see if the user has made a selection and prepare the query.



Available for
download on
Wrox.com

LISTING 8-5: checkboxlist function

```

// Prepare checkbox value which will be used in SQL statement.
// In this function, we add all the numbers in $checkboxlist parameter.
// If $checkboxlist is empty, then we return 0.
//
// Arguments:
// checkboxlist ; the value of CHECKBOXLIST parameter in the GetPOI request.
//
// Returns:
// checkbox_value ; the value that can be used to construct the right
// SQL statement.
function GetCheckboxValue ( $checkboxlist ) {

// if $checkboxlist exists, prepare checkbox_value.
  if( isset( $checkboxlist ) ) {

// Initialize returned value to be 0 if $checkboxlist is empty.
    $checkbox_value = 0;

// If $checkboxlist is not empty, return the added value of all
// the numbers splited by ','.

```

```

if (!empty($checkboxlist)) {
    if ( strstr($checkboxlist,',') ) {
        $checkbox_array = explode(',', $checkboxlist);
        for( $i=0; $i<count($checkbox_array); $i++ )
            $checkbox_value+=$checkbox_array[$i];
    }//if
    else
        $checkbox_value = $checkboxlist;
} //if

return $checkbox_value;
} //if
else {
    throw new Exception("checkboxlist parameter is not passed in GetPOI
        request.");
} //else
} //GetCheckboxValueCustom Slider Filter

```

custom_slider Parameter

The `custom_slider` parameter (see Listing 8-6) enables users to set the maximum property price. This code is short because you need only to retrieve the value from the `custom_slider` parameter.



LISTING 8-6: custom_slider function

Available for
download on
Wrox.com

```

// Prepare custom_slider value which will be used in SQL statement.
// In this function, we simply return the value of $customslider defined
// in the GetPOI request.
//
// Arguments:
// customslider ; the value of CUSTOM_SLIDER parameter in the GetPOI request.
//
// Returns:
// customslider ; the value that can be used to construct the right SQL
// statement.

function GetSliderValue ( $customslider ) {

// if $customslider exists, return its value.
    if( isset( $customslider ) )
        return $customslider;
    else
        throw new Exception("custom slider parameter is not passed in GetPOI request.");
} //GetSliderValue

```

Preparing the SQL

The SQL query statement needs to be modified to take into account the filter parameters. The code already has a `Gethotspots` function that prepares and executes the query to retrieve the POIs from the database. Currently, however, it is still reading from the `POI_Table` created in the previous chapter.

Locate the `Gethotspots` function in the `realestate.php` file (it should be near the top of the file). You will see the following SQL statement being prepared:

```
$sql = $db->prepare( "
    SELECT id,
    attribution,
    title,
    lat,
    lon,
    imageURL,
    line4,
    line3,
    line2,
    type,
    dimension,
    (((acos(sin((:lat1 * pi() / 180)) * sin((lat * pi() / 180)) +
        cos((:lat2 * pi() / 180)) * cos((lat * pi() / 180)) *
        cos((:long - lon) * pi() / 180))
    ) * 180 / pi()) * 60 * 1.1515 * 1.609344 * 1000) as distance
    FROM POI_Table
    HAVING distance < :radius
    ORDER BY distance ASC
    LIMIT 0, 50 " );
```

To add the filters, just modify the `FROM` statement to replace the `POI_Table` table name to `POI_RealEstate_Table` and add an additional `WHERE` statement to filter based on the users' filtering options.

```
FROM POI_RealEstate_Table
WHERE title REGEXP :search
AND Radiolist = :radiolist
AND ( Checkbox & :checkbox )!=0
AND Custom_Slider <= :slider
HAVING distance < :radius
ORDER BY distance ASC
LIMIT 0, 50 " );
```

The parameters need to be binded in the `PDOStatement`. At the bottom of the `Gethotspots` function, where you previously added the binding, add the binding for the filter parameters:

```
// PDOStatement::bindParam() binds the named parameter markers
// to the specified parameter values.
$sql->bindParam( ':lat1', $value['lat'], PDO::PARAM_STR );
$sql->bindParam( ':lat2', $value['lat'], PDO::PARAM_STR );
$sql->bindParam( ':long', $value['lon'], PDO::PARAM_STR );
$sql->bindParam( ':radius', $value['radius'], PDO::PARAM_INT );

// Custom filter settings parameters.
// The four Get functions can be customized.
```

```

$sql->bindParam( ':search', GetSearchValue ( $value['SEARCHBOX'] ),
                PDO::PARAM_STR );
$sql->bindParam( ':radiolist', GetRadioValue ( $value['RADIOLIST'] ),
                PDO::PARAM_STR );
$sql->bindParam( ':checkbox', GetCheckboxValue ( $value['CHECKBOXLIST'] ),
                PDO::PARAM_INT );
$sql->bindParam( ':slider', GetSliderValue ( $value['CUSTOM_SLIDER'] ),
                PDO::PARAM_INT );

```

Once you have added all the code, save your file and upload it to your server.

Testing the Real Estate Layer

Congratulations! You now have a fully functional layer complete with filters to limit the data. To test the layer, load the client on your device. You should see the Filters dialog box appear as soon as the layer is loaded. Experiment with the various filters and how they limit the amount of data displayed. See Figure 8-8 and Figure 8-9.

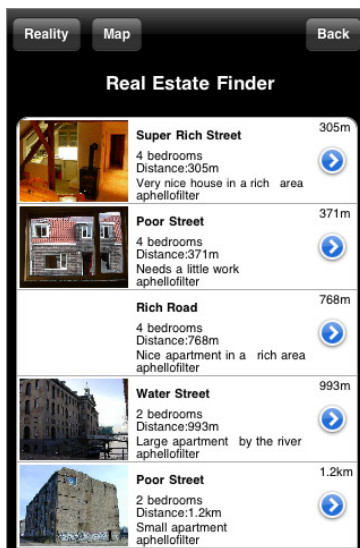


FIGURE 8-8: The Real Estate Finder list view

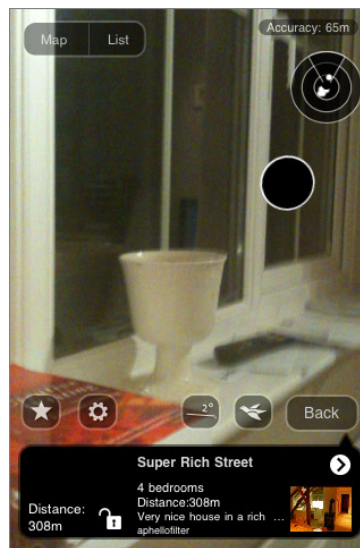


FIGURE 8-9: A property shown in the AR view

Troubleshooting

If your layer isn't working as expected, there are two ways you can troubleshoot the problem:

1. Type the URL to the realestate.php file into your browser window. This helps you track down PHP-related errors:

```

Fatal error: Uncaught exception 'PDOException' with message 'SQLSTATE[28000]
[1045] Access denied for user 'layardbuser'@'68.178.254.124' (using password:

```

```
YES)' in /playground/layer/realestate.php:10 Stack trace: #0
/playground/layer/realestate.php(10): PDO->__construct('mysql:host=laya...',
'layardbuser', 'password', Array) #1 {main} thrown in
/playground/layer/realestate.php on line 10
```

If the problem is not a PHP error, then try the next troubleshooting option.

2. Use the test functionality contained in the Layer Dashboard, which helps you understand what Layer is passing to your application.

Using the dashboard, you can perform the same actions you can on the client, making it easier to figure out the problem.

Finishing the Layer

As you learned in the previous chapter, you can create colors for the various augmented reality elements to give the layer a professional feel. Unlike the last chapter, where I encouraged you to try every color combination, the real estate layer requires you to create a more professional image.

Enter the values into the Look & feel section (see Table 8-4) and the Listing & indexing section (see Table 8-5) to complete your layer.

TABLE 8-4: Look & feel values

Banner Text Color	Realestate_icon.png
Banner Background Color	FFFFFF
Spot Color	F53416
Outer Color	F53416
Middle Color	F53416
Inner Color	F53416
BIW Background Color	147AFF
BIW Title Color	F53416
BIW Text Color	FFFFFF

TABLE 8-5: Listing & indexing values

Icon	Houseicon.jpg
Category	Accommodation
Detailed description	This layer shows real estate around me that is for sale. It also shows how to use filters in Layer.

Figures 8-10, 8-11 and 8-12 show how these parameters are used in the Layar client. I'm sure you will agree that minimal use of the colors presents a much more attractive finish.

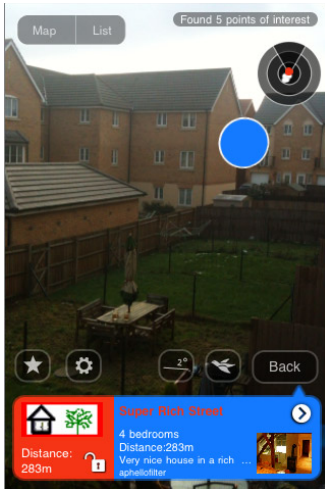


FIGURE 8-10: The real estate layer shown in the AR view

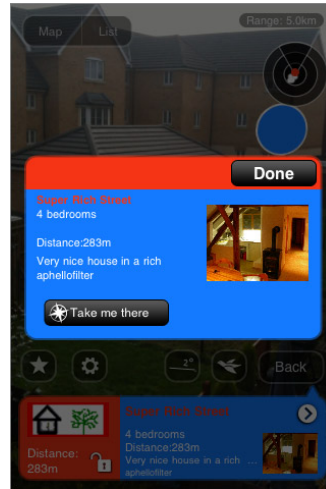


FIGURE 8-11: The real estate layer when clicked



FIGURE 8-12: The real estate layer list view



To create the full real-estate listing, you've had to type a lot of code. If you prefer, you can download the completed listing (Listing 8-7) at the companion web site.

EXPERIMENTING WITH 2D OBJECTS

Until now, you have used only generic, one-dimensional icons and icons with custom image sets. In this section, you will experiment with advanced 2D manipulation to obtain more control over the 2D object and control aspects such as size, rotation, and angle. To work with 2D objects, you first need to set the option in the Layar Dashboard and then create two additional tables in the MySQL database to hold the objects and the settings.

In the Layar Dashboard, edit your real-estate layer and select the General option. Then change the type from generic 2D to 3D and 2D objects in 3D space (see Figure 8-13).

General	Status	Testing
API endpoint	Developer email	lester@lestermadden.com
Listing & indexing	Publisher email	lester@lestermadden.com
Look & feel	Layer type	3D and 2D objects in 3D space
Coverage		
Filters		

FIGURE 8-13: Configuring Layar for 3D

Now that you have changed the layer type, examine your test account on the device. Notice the 3D cube that has been added to your layers logo? Figure 8-14 shows the icon that Layar automatically adds to indicate 3D content.



FIGURE 8-14: The 3D layer icon

Creating the Tables

The Object table is the first table you need to create. As shown in Listing 8-8, this table holds the URL of the objects. Notice that it holds the URL of various sizes of the object.



Available for
download on
Wrox.com

LISTING 8-8: Create OBJECT_Table

```
CREATE TABLE IF NOT EXISTS `OBJECT_Table` (
  `ID` int(10) NOT NULL auto_increment,
  `poiID` varchar(255) NOT NULL,
  `baseURL` varchar(255) NOT NULL,
  `full` varchar(255) NOT NULL,
  `reduced` varchar(255) default NULL,
  `icon` varchar(255) default NULL,
  `size` float(15,5) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=4 ;
```

The fields for this table are as follows:

- ID is a unique identifier for the OBJECT_Table.
- poiID is the id of the POI from the POI_RealEstate_Table where the object will be applied.
- baseURL is the domain and directory where the objects will be located. (for example, www.yourserver.com/files/objects/)
- full is the file name of the full size object (for example, house_full.jpg). This image/object is shown to users when the POI is closer than 50 meters.
- reduced is the file name of the reduced size object (for example, house_reduced.jpg). This field is optional. This image/object is shown when the POI is 50-100 meters from the user's current location.

- `icon` is the file name of the icon that represents the object (for example, `house_icon.jpg`). This field is optional. This should be a 32x32 PNG file and it is displayed when the user is more than 100 meters from the POI.
- `size` represents the scale. The default is 1.00.



Once you have created this table in your MySQL database, add the records shown in Table 8-6. The images in the table can be downloaded from the companion web site.

TABLE 8-6: OBJECT_Table records

ID	POIID	BASEURL	FULL	REDUCED	ICON	SIZE
1	1	URL to the folder on your server holding the images	house_full.jpg	house_reduced.jpg	house_icon.jpg	1
2	2	URL to the folder on your server holding the images	house_full.jpg	house_reduced.jpg	house_icon.jpg	1
3	3	URL to the folder on your server holding the images	house_full.jpg	house_reduced.jpg	house_icon.jpg	1
4	4	URL to the folder on your server holding the images	house_full.jpg	house_reduced.jpg	house_icon.jpg	1
5	5	URL to the folder on your server holding the images	house_full.jpg	house_reduced.jpg	house_icon.jpg	1

With only a few records, it's probably easiest to add them using the MySQL UI. But if you want to practice your MySQL coding skills, the bulk update query code is shown in Listing 8-9.

**LISTING 8-9: OBJECT_Table update**Available for
download on
Wrox.com

```

INSERT INTO `layardbuser`.`OBJECT_Table` (
  `ID` ,
  `poiID` ,
  `baseURL` ,
  `full` ,
  `reduced` ,
  `icon` ,
  `size`
)
VALUES ('1','1', '<Full URL to resource folder>', 'house_full.jpg',
'house_reduced.jpg', 'house_icon.jpg', '1.00000'
),

('2', '2', '<Full URL to resource folder>',
'house_full.jpg', 'house_reduced.jpg', 'house_icon.jpg', '1.00000'
),

('3', '3', '<Full URL to resource folder>', 'house_full.jpg',
'house_reduced.jpg', 'house_icon.jpg', '1.00000'
),

('4', '4', '<Full URL to resource folder>', 'house_full.jpg',
'house_reduced.jpg', 'house_icon.jpg', '1.00000'
),

('5', '5', '<Full URL to resource folder>', 'house_full.jpg',
'house_reduced.jpg', 'house_icon.jpg', '1.00000'
);

```



When adding to the `baseURL` field in the database, you must include a postfix / (for example, `http://www.yourserver.com/myfolder/`).

However, you choose to add the records, you will have added support for 2D images for the first five POIs in your `POI_RealEstate_Table`. The remaining POIs will continue to use the one-dimensional icons.

The Transformation table is the second table you need to create. This table (see Listing 8-10) holds the angle and scale you want to apply to the object.

**LISTING 8-10: Create TRANSFORM_Table**Available for
download on
Wrox.com

```

CREATE TABLE IF NOT EXISTS `TRANSFORM_Table` (
  `ID` int(10) NOT NULL auto_increment,
  `poiID` varchar(255) NOT NULL,
  `rel` tinyint(1) default '0',
  `angle` decimal(5,2) default '0.00',

```

```

`scale` decimal(12,2) NOT NULL default '1.00',
PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=4 ;

```

The fields for this table are as follows:

- `ID` is the unique identifier for the `OBJECT_Table`.
- `poiID` is the id of the POI from the `POI_RealEstate_Table`.
- `rel` is the relative Boolean value used to determine if the rotation is calculated relative to the user's position. If true, the user is always facing the object regardless of his position. If that's the case, the user will can never walk around an object.
- `angle` is the decimal value of the angle you want to apply to the object. This parameter rotates the object. Positive values turn the object clockwise; negative values turn the object counter-clockwise.
- `scale` is the decimal value of scale you want to apply to the image or object.

Once you have created this table in your MySQL database, add the records shown in Table 8-7. With so few records to add, it's often easier to type them via the MySQL UI. However, the bulk update is included in Listing 8-11.

TABLE 8-7: TRANSFORM_Table

ID	POIID	REL	ANGLE	SCALE
1	1	1	0	1.00
2	2	1	0	1.00
3	3	1	0	1.00
4	4	1	0	1.00
5	5	1	0	1.00



Available for
download on
Wrox.com

LISTING 8-11: TRANSFROM_Table Update

```

INSERT INTO `layardbuser`.`TRANSFORM_Table` (
  `ID` ,
  `poiID` ,
  `rel` ,
  `angle` ,
  `scale`
)
VALUES ('1', '1', '1', '0', '1.00'),
       ('2', '2', '1', '0', '1.00'),
       ('3', '3', '1', '0', '1.00'),
       ('4', '4', '1', '0', '1.00'),
       ('5', '5', '1', '0', '1.00');

```



If you enter a record into the OBJECT_Table, it must have a corresponding entry in the TRANSFORM_Table. Otherwise, the layer generates errors.

Extending the Code

Now you will extend the real estate sample that you just created to support the new functionality for the 2D objects. To do this, add the code shown in Listing 8-12 to your realestate.php file.

The getObject function shown in Listing 8-12 retrieves the object for the current POI from the OBJECT_Table and adds the object to the \$poi array.



LISTING 8-12: getObject function

Available for
download on
Wrox.com

```
// Put fetched object related parameters for each POI into an associative
// array. The returned values are assigned to $poi[object].
//
// Arguments:
//   poi ; The POI handler.
//   $db ; The database connection handler.
//
// Returns:
//   array ; An array of received object related parameters for this POI.
//
function Getobject( $poi, $db ) {

    // A new table called "OBJECT_Table" is created to store object related
    // parameters, namely "baseURL", "full", "reduced", "icon", and "size".

    // "poiID" which shows the POI id that this object belongs to.
    // The SQL statement returns object which has the same poiID as
    // the id of $poi ($poi['id']).

    $sql_object = $db->prepare( " SELECT baseURL, full, reduced, icon, size

                                FROM OBJECT_Table
                                WHERE poiID = :id
                                LIMIT 0,1 " );

    // Binds the named parameter markers ":id" to the specified parameter
    // values "$poi['id']".

    $sql_object->bindParam( ':id', $poi['id'], PDO::PARAM_INT );

    // Use PDO::execute() to execute the prepared statement $sql_object.
    $sql_object->execute();

    // Fetch the poi object.
```

```

$object = $sql_object->fetchAll( PDO::FETCH_ASSOC );

/* Process the $object result */

// if $object array is empty, return NULL.
if ( empty( $object ) ) {
    $poi["object"] = null;

} //if
else {

// Since each POI only has one object. Logically, only one object should
// be returned. Assign the first object in the array to $poi["object"]
    $poi["object"] = $object[0];

// Change "size" type to float.
    $poi["object"]["size"] = ChangetoFloat( $poi["object"]["size"] );
    } //else

    return $poi["object"];

} //Getobject

```

The `Gettransforms` function shown in Listing 8-13 checks the `TRANSFORM_Table` for transformations relating to the current POI. If a transform is located, it is added to the `$poi` array.



Available for
download on
Wrox.com

LISTING 8-13: `Gettransforms` function

```

// Put fetched transform related parameters for each POI into an associative
// array. The returned values are assigned to $poi[transform].
//
// Arguments:
//   poi ; The POI handler.
//   $db ; The database connection handler.

// Returns:
//   array ; An array of received transform related parameters for this POI.
//
function Gettransform( $poi, $db ) {

// A new table called "TRANSFORM_Table" is created to store transform related
// parameters, namely "rel", "angle" and "scale
// "poiID" which shows the POI id that this transform belongs to.
// The SQL statement returns transform which has the same poiID as the
// id of $poi ($poi['id']).

    $sql_transform = $db->prepare( " SELECT rel, angle, scale
                                   FROM TRANSFORM_Table

```

continues

LISTING 8-13 (continued)

```

        WHERE poiID = :id
        LIMIT 0,1 " );

// Binds the named parameter markers ":id" to the specified parameter
// values "$poi['id']".
    $sql_transform->bindParam( ':id', $poi['id'], PDO::PARAM_INT );

// Use PDO::execute() to execute the prepared statement $sql_transform.
    $sql_transform->execute();

// Fetch the poi transform.
    $transform = $sql_transform->fetchAll( PDO::FETCH_ASSOC );

/* Process the $transform result */

// if $transform array is empty, return NULL.
    if ( empty( $transform ) ) {
        $poi["transform"] = null;
    } //if
    else {

// Since each POI only has one transform. Logically, only one transform should
// be returned. Assign the first transform in the array to $poi["transform"]

        $poi["transform"] = $transform[0];

// Change the value of "rel" into boolean value,if the value is NULL
// return NULL.

        $poi["transform"]["rel"] = ChangetoBool( $poi["transform"]["rel"] );

// Change the values of "angle" and "scale" to demical.

        $poi["transform"]["angle"] = ChangetoFloat( $poi["transform"]["angle"] );
        $poi["transform"]["scale"] = ChangetoFloat( $poi["transform"]["scale"] );
    } //else

        return $poi["transform"];
    } //Gettransform

```

In the `realestate.php` file, locate the `Gethotspots` function. Inside the function, locate the `foreach` loop that adds the POI information into the `hotspots` array. The relevant section of code you are searching for is shown in Listing 8-14.

**LISTING 8-14: Gethotspots function**Available for
download on
Wrox.com

```
// Put each POI information into $response["hotspots"] array.
    foreach ( $pois as $poi ) {

// Use function Getactions() to return an array of actions associated with
// the current POI.

$poi["actions"] = Getactions ( $poi, $db );
```

Now add the code shown in Listing 8-15 to the top of the `foreach` loop. This adds the results from the `Getobject` and `Gettransform` calls to the array.

**LISTING 8-15: Extending the Getobject function**Available for
download on
Wrox.com

```
// Put each POI information into $response["hotspots"] array.
    foreach ( $pois as $poi ) {

// If POI "dimension" =2 or 3, use function Getobject() to return an object
// associated with the current POI.

        if ( $poi["dimension"] == '2' || $poi["dimension"] == '3' )
            $poi["object"] = Getobject ( $poi, $db);

// If POI "dimension" =2 or 3, use function Gettransform() to return a
// transform dictionary associated with the current POI.
        if ( $poi["dimension"] == '2' || $poi["dimension"] == '3' )
            $poi["transform"] = Gettransform ( $poi, $db);

// Use function Getactions() to return an array of actions associated with
// the current POI.

$poi["actions"] = Getactions ( $poi, $db );
```

Once you have finished editing the file, upload it to your server.

Changing Dimensions

You can try testing your layer at this point but you may be surprised to find that it hasn't changed; it still has the same look and feel as before. That's because there is one final change you need to make. When you created the table, you created it with a dimension field value of 1. The dimension field specifies what type of content the POI should display.

- 1: 1d. This is the usual POI (icons) and is the default value.
- 2: 2d. This is an image used for the POI.
- 3: 3d. This is used when referencing the 3D object for the POI.

Since the dimension field of the POI_RealEstate_Table is currently set to 1 for all the POIs in the table, you need to change the value from 1 to 2. However, it is important that you change only the dimension field for the records that have an entry in the OBJECT_Table and TRANSFORM_Table.

If you inadvertently change the dimension value for a record that doesn't have a corresponding entry, your layer will stop working. If that's the case, in the Layar test tool, load your layer to obtain the error-tracking information to track down the cause of the problem. If you have changed the dimension value of a POI record and there is no corresponding entry in the TRANSFORM_Table, you will see the following error message in the Layar test tool:

```
error validating provider response - Traceback (most recent call last): File
"/var/www/dev.layar.com/layar/publishing/api_testpage_mainproxy.py", line 50,
in validate_response data = validator.validate(data, schema) File
"/var/www/dev.layar.com/layar/util/jsontransmogriifier.py", line 524, in
validate raise ValidationError(message=" ".join(self.errors),
errors=self.errors) ValidationError: Field 'transform' is required by field
'dimension' if the value of 'dimension' is in [2, 3].
```

```
JSON parse error( most likely)!, validate your response at http://www.jsonlint.com/
error : Field 'transform' is required by field 'dimension' if the value of
'dimension' is in [2, 3].
error : Field 'transform' is required by field 'dimension' if the value of
'dimension' is in [2, 3].
```

This error simply means that a record in the POI_RealEstate_Table has a field with a dimension value of 2 or 3 and no corresponding entry in the OBJECT_Table or TRANSFORM_Table. Figure 8-15 shows the dimension field in the POI_RealEstate_Table.

imageURL	line4	line3	line2	type	dimension	alt	relativeAlt	distance	InFocus
http://www.augmentedplanet.com/book/layar/houses/h...	Unique underground house	Distance: % distance%	2 bedrooms	1	1	NULL	NULL	0.0000000000	0
http://www.augmentedplanet.com/book/layar/houses/h...	House on main street	Distance: % distance%	7 bedrooms	1	1	NULL	NULL	0.0000000000	0
http://www.augmentedplanet.com/book/layar/houses/h...	House with a large garage	Distance: % distance%	2 bedrooms	1	1	NULL	NULL	0.0000000000	0
http://www.augmentedplanet.com/book/layar/houses/h...	Very nice house in a rich area	Distance: % distance%	4 bedrooms	1	1	NULL	NULL	0.0000000000	0

FIGURE 8-15: The dimension field

To gain access to advanced functionality for 2D images (such as automatic scaling), you need to edit the dimension field for POIs 1-5 and change the value from 1 to 2. Then Layar will load the POIs with the respective 2D objects you added to the OBJECTS_Table.

In plain English, edit the POI_RealEstate table. For fields with IDs of 1-5, change the dimension field from 1 to 2 (ee Figure 8-16).

Field	Type	Function	Null	Value
id	varchar(255)		<input type="checkbox"/>	5
attribution	varchar(150)		<input type="checkbox"/>	aphellofilter
title	varchar(150)		<input type="checkbox"/>	Poor Hill
lat	decimal(20,10)		<input type="checkbox"/>	51.5993070000
lon	decimal(20,10)		<input type="checkbox"/>	0.0793500000
imageURL	varchar(255)		<input type="checkbox"/>	http://www.augmentedplanet.com/book
line4	varchar(150)		<input type="checkbox"/>	House on the hill
line3	varchar(150)		<input type="checkbox"/>	Distance:%distance%
line2	varchar(150)		<input type="checkbox"/>	4 bedrooms
type	int(11)		<input type="checkbox"/>	1
dimension	int(1)		<input type="checkbox"/>	2

FIGURE 8-16: Changing the dimension field

You can do this by editing the table in the MySQL window and manually changing the value of the dimension field or by running the update query shown in Listing 8-16.



LISTING 8-16: Updating the dimension field

Available for
download on
Wrox.com

```
UPDATE `<dbusername>`.`POI_RealEstate_Table` SET `dimension` = '2'
WHERE CONVERT( `POI_RealEstate_Table`.`id` USING utf8 ) = '5' LIMIT 1 ;

UPDATE `<dbusername>`.`POI_RealEstate_Table` SET `dimension` = '2'
WHERE CONVERT( `POI_RealEstate_Table`.`id` USING utf8 ) = '4' LIMIT 1 ;

UPDATE `<dbusername>`.`POI_RealEstate_Table` SET `dimension` = '2'
WHERE CONVERT( `POI_RealEstate_Table`.`id` USING utf8 ) = '3' LIMIT 1 ;

UPDATE `<dbusername>`.`POI_RealEstate_Table` SET `dimension` = '2'
WHERE CONVERT( `POI_RealEstate_Table`.`id` USING utf8 ) = '2' LIMIT 1 ;

UPDATE `<dbusername>`.`POI_RealEstate_Table` SET `dimension` = '2'
WHERE CONVERT( `POI_RealEstate_Table`.`id` USING utf8 ) = '1' LIMIT 1 ;
```

Once you have done this, you can test your layer. You will see a mixture of one-dimensional and two-dimensional images. The further a POI, the smaller the icon. This scaling effect is shown in Figure 8-17.

If you don't see your 2D images, use the Layar test window to help troubleshoot. Make absolutely certain that any POIs with dimension values of 2 have corresponding OBJECT_Table and TRANSFORM_Table entries. Figure 8-18 shows how to debug layers in the Layar client.

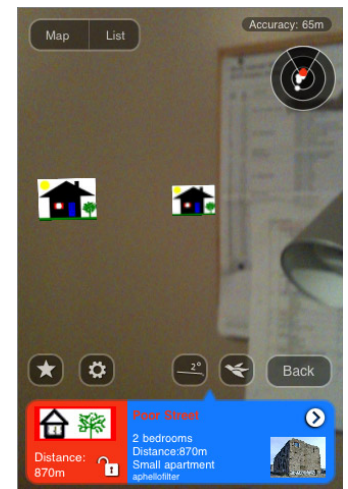


FIGURE 8-17: Scaling 2D images in the camera window



FIGURE 8-18: Debugging problems

2D Scaling

Depending on how far away your POIs are from your current location, you may notice a scaling effect for the 2D images. For POIs that are less than 50 meters from the user's current location, Layar uses the object described in the full field in the OBJECT_Table (in the example, it's the house_full.jpg file). For POIs that are 50-100 meters from the user, Layar uses the object described in the reduced field (house_reduced.jpg). For POIs more than 100 meters away, Layar uses the icon field (house_icon.jpg).

You should keep all images smaller than 100kb. If you don't include a relevant object (for example, you don't include a reduced image), Layar will load the icon. Layar will also attempt to load the icon if it is unable to load the relevant object. If the icon fails, one-dimensional icons will be displayed.

SUMMARY

Congratulations on reaching this point! In this chapter, you have written a lot of code and created several new tables for the database. You now have a layer that returns multiple POIs, has 2D images, and has functionality that enables a user to filter the results returned from your database. The layer you have created is a realistic example of a production-level layer with all the functionality that users will expect. What you have learned in this chapter will be useful when you create your layers in the future.



Using Layar Tools

WHAT'S IN THIS CHAPTER?

- ▶ How to launch layers
- ▶ How to use the Layar Shortcut Tool
- ▶ How to add the launcher icon to Android devices
- ▶ How to create and prototype layers with third-party tools

In the previous two chapters, you learned the basics of Layar development. As you have seen, Layar development is easy once you have mastered MySQL. In this chapter, you will learn about some of the tools available from Layar and third parties that simplify the developing and loading of Layar.

To test the Layar Shortcut Tool, you will need a PC installed with the Android SDK from Google and a development IDE (such as Eclipse). You will also need an Android device.

LAUNCHING LAYERS

URL: None.

Technical Knowledge Required: None

The process of asking your users to load Layar, navigate to the accommodation category, and then scroll down until they find the Real Estate layer you created is a long and tedious one. Fortunately you can make your layer much more discoverable by providing users with a shortcut to the layer. This enables users to launch your content via a web page or even with a QR code.

Using Layar Intent

Layar Intent is a way to tell Layar how it should handle incoming request from a web browser. Intent can be one of two types:

- **layar://your_unique_layer_name.** This format should be used to link to a layer from within the Layar client (for example, redirects or links within your layer). This format will also work within the iPhone and Android web browsers if the Layar client is installed. If Layar is not installed, the user receives a cryptic error message and will not be prompted to download Layar.
- **http://m_layar.com/open/your_unique_layer_name.** This syntax is the preferred method for linking to a layer from external applications (such as web browsers or email clients). The users are taken to a web page that allows them to directly open the layer. If the Layar client is not installed, users can be taken to the relevant application store to download the client.

Since you already have Layar installed on your device, why not try out the two different ways to launch your layer. You will need the unique name for your layer. I suggest using the real estate layer that you created in the previous chapter, but you're free to use any of your layers.

1. Load your mobile web browser and type the following URL:
`layar://your_unique_layer_name`



You do not need to prefix this URL with `http://`.

You should be taken directly to your layer. Of course if the user doesn't have Layar installed the browser won't know what to make of a URL prefixed with `layar://` and will throw an error message.

2. Return to your mobile browser and type
`http://m_layar.com/open/your_unique_layer_name`

This time you will be taken to an intermediate page (see Figure 9-1) that asks you what action you want to take next.

Usage Tips

Use the `http://` prefix when you want to:

- Publish a link to your layer on a web site.
- Send users links to your layer via email or SMS.
- Provide a URL by another means (for example, a QR code).

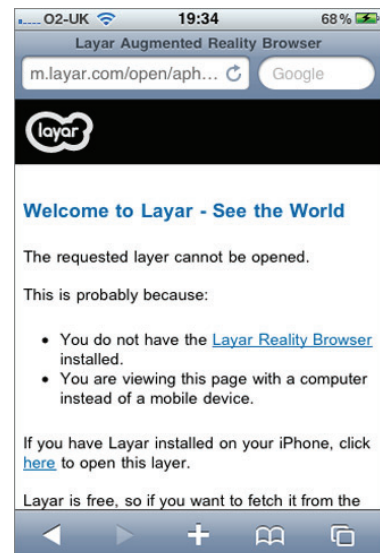


FIGURE 9-1: The confirmation page

Use the `layar://` prefix when you want to:

- Link to another layer from inside Layer (for example, your own layer or an advertised layer).
- Link to a layer directly from the web browser. For example, your Layar may direct the user to a web page to perform an action. Using the `layar://` syntax, you can return the user directly back to your layer once he has completed the action.

In addition to linking to your layer, you can include parameters to set filters. In the previous chapter, you created a real estate layer. Using the following syntax, you can populate the various filters you created to show houses for rent up to the value of 1000000.

```
layar://your_unique_layer_name/?action=refresh&CUSTOM_SLIDER=1000000
```

This syntax can be used in conjunction with a QR code. Imagine a scenario in which a real estate advertisement board has a QR code embedded. Scanning the QR code with a reader could take the user directly to your layer related to the house rather than require the user to load the layer first and then set the search criteria.

For some extra fun, try combining your layer with a QR code. Load a QR code reader and point your camera at the QR code shown in Figure 9-2 for some Woomba Mania fun.



FIGURE 9-2: A Woomba Mania QR code



For the full list of the Layar intents that you can use, please visit: <http://layar.pbworks.com/w/page/7783232/Layar-Intent>

THE LAYAR SHORTCUT TOOL

URL: <http://layar.pbworks.com/f/LayarLauncherCreator.jar>

Technical Knowledge Required: Development experience (Android only)

It's helpful to provide users with the ability to launch your layer without having to hunt around the Layar UI. It makes your layer sticky and something that is always on hand for users. Using the Layar Intent syntax enables you to create quick launches for your layer from web sites and email programs but it doesn't help you provide a quick launch icon on the device. To solve this problem, you could distribute a shortcut to the device that uses the `layar://` syntax. It's doubtful, however, that Google would agree to publish such a simple solution to the application store.

For Android developers, Layar provides a solution — albeit an unsupported one — that enables you to create a packaged binary that can be installed on Android devices. The solution enables you to configure your own launch icons which will appear on the device so it will appear to users as if they are launching your application. Once clicked, the application checks to see if the Layar client is installed.

If it's not installed, the user is prompted to install the Layar client from Google Marketplace. If Layar is installed, then your layer is loaded. Beyond the benefits of having your own launch icon, the binary can be hosted on your own web page or even on third-party application stores like GetJar (www.getjar.com). You can also try to publish it to the Google Marketplace.

Requirements

If you are already developing for the Android platform and you have all the relevant tools in place, this could be a viable option for distributing a shortcut to your layer. If, however, you are not familiar with developing for the Android platform and don't have any of the tools, then the expression “using a sledgehammer to crack a walnut” springs to mind.

To create your shortcut, you must have the following installed:

- Java development environment
- Android SDK
- USB drivers for your device
- Development environment (for example, Eclipse)



Since this is a lot of software that needs to be installed and configured, it's easier to just ask a friendly Android developer if he can create the binary for you. If you want to go it alone, then you should read <http://developer.android.com/sdk/index.html> which will help you install and configure the tools.

Using the Shortcut Tool

First, you need to download the Layar Launcher Creator Tool from the Layar web site:

<http://layar.pbworks.com/f/LayarLauncherCreator.jar>

1. After you have downloaded it, run the file, which will display a dialog box asking you to complete some information about your layer (see Figure 9-3).

The layer name should be the same as your actual unique layer name; the package name should be your layer name and your company or domain name.

2. Click the Create launcher project button.

The tool creates a sample project for you. From inside the Eclipse IDE, you must create a new project from an existing source, with the source being the project you created in the previous step.

3. A new project can be created by clicking File and choosing New Android Project, which displays the dialog box shown in Figure 9-4.

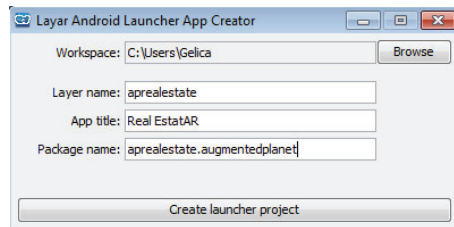


FIGURE 9-3: The Layar Launcher Creator Tool

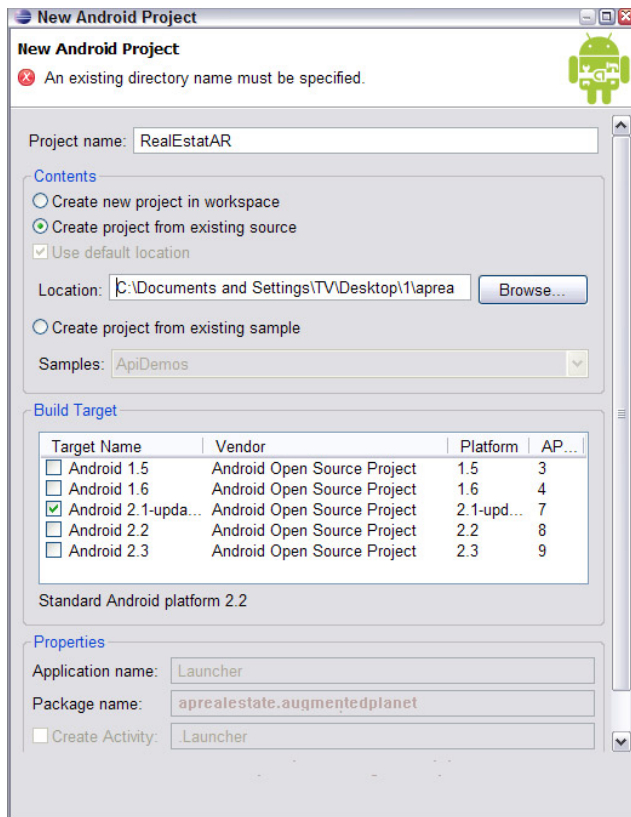


FIGURE 9-4: Creating a new Android project

4. Select the Create project from existing source radio button.

This option auto completes most of the fields in this dialog box. You need only to give the workspace a name and chose the target Android platform. Ignore any warning messages you might receive that indicate a problem with the API level not matching the Min SDK version.

You should include your own icons for the quick launch, so you need to overwrite the existing Layar icons. Overwriting the Layar icons saves you the time and effort of trying to change file names in the code, particularly if you are not familiar with the Eclipse IDE. The sizes you should provide are 48×48 and 72×72. These should be placed in the relevant \res folders (see Figure 9-5).

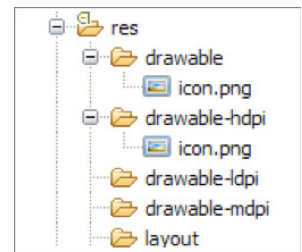


FIGURE 9-5: Icon folders

You can now build and publish your shortcut. If you are not familiar with how to build Android applications, refer to the Android publishing document:

<http://developer.android.com/guide/publishing/publishing.html>

The Layar Launcher Creator Tool is an unsupported tool from Layar. While Layar won't support its use, I'm sure they will continue to develop it and publish known issues. I strongly suggest that you take a look at the documentation for the tool at:

<http://layar.pbworks.com/w/page/28695224/Layar-shortcut-tool>

This documentation contains late-breaking information and known issues.

HOPPALA

URL: <http://augmentation.hoppala.eu>

Technical Knowledge Required: None

If you have found developing layers difficult or you don't have a web server to host a MySQL database and the other related content, then Hoppala has the answer for you.

Hoppala, the developers behind one of my favorite augmented reality games (Woomba Mania), has developed a tool that enables anyone, regardless of his developer expertise, to create augmented reality content for Layar. The tool, Hoppala Augmentation, is a web-based tool that enables you to create POIs and upload resources (such as icons, audio, video and 3D) with just a few mouse clicks. There is no hosting, configuring of databases, or writing code — just a simple but powerful web UI to build augmented reality content. Simply put, Hoppala Augmentation is the most advanced Layar creator out there — second only to building your own content, of course.

Using Hoppala Augmentation

Hoppala Augmentation is a POI generation tool that enables you to create Layar-based content via a simple web-based user interface without requiring you to write a single line of code. Better yet, you won't even need to host the MySQL database — all the hard work is taken care of for you.

To get started, register for a free Hoppala user account at <http://augmentation.hoppala.eu>. You will also need a Layar developer account (which you have already created). Getting started is very simple; first you will need to create a new Layar service using the Hoppala Dashboard (see Figure 9-6).

Title	Name	API endpoint URL
My Hoppala Layer	myhoppalalayar	http://augmentation.hoppala.eu/poi?l=04fac0600b55ae04b8f1f5094e8f5710

FIGURE 9-6: The Hoppala Dashboard

The Hoppala Dashboard is where you give your layer a title and a unique name. Hoppala will automatically create the endpoint function on the <http://augmentation.hoppala.eu> domain and append a GUID to the URL to make the location of your Layar unique.

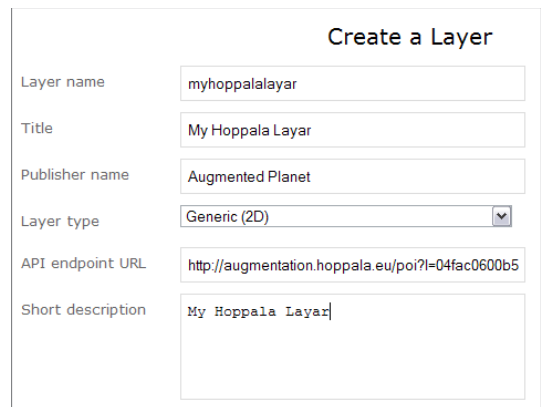
The details you added via the Hoppala Dashboard need to be provided to Layar. Already you can probably see that you need to create a new layer using the name and the end point function given

in the Hoppala Dashboard. All the details will need to be identical to ensure that the layer can be correctly found and called. The Layer Dashboard is shown in Figure 9-7.

You should copy and paste the details from Hoppala to Laya, saving you the effort of trying to troubleshoot any typos that you might inadvertently introduce. At this point you will have created a new layer with the endpoint URL residing on the Hoppala server. It will exist in both the Laya Dashboard and the Hoppala Dashboard.

The next step is to create the POIs for your layer. Fortunately, as previously mentioned, Hoppala creates all the POIs for you. You need only to click your layer in the Hoppala Dashboard to bring up the editor and use the map to search for your POI location. When you have the location, right-click and then fill out the simple form to include images, sounds, actions, and so on (see Figure 9-8).

Once you have finished building the list of your POIs, you can save the layer in Hoppala and view it in the Laya client on your mobile device (see Figure 9-9). Just about anything you have learned from the previous Laya chapters can be done without writing a single line of code.



Create a Layer	
Layer name	myhoppalalayar
Title	My Hoppala Layer
Publisher name	Augmented Planet
Layer type	Generic (2D)
API endpoint URL	http://augmentation.hoppala.eu/poi/?l=04fac0600b5
Short description	My Hoppala Layer

FIGURE 9-7: The Laya Dashboard

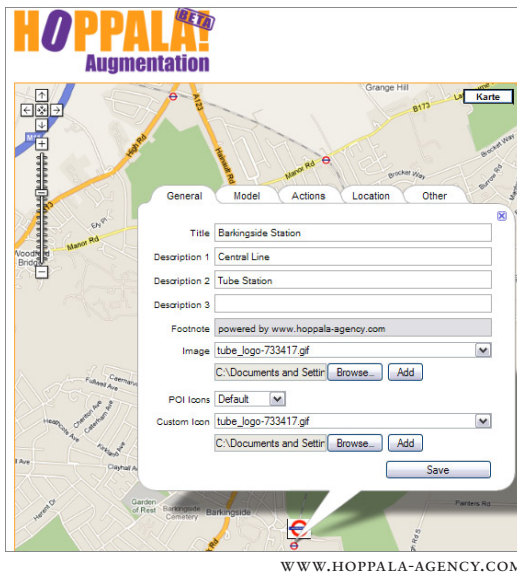


FIGURE 9-8: The Hoppala editor

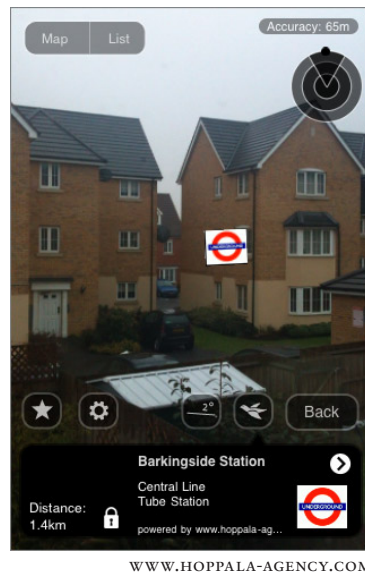


FIGURE 9-9: My Hoppala layer

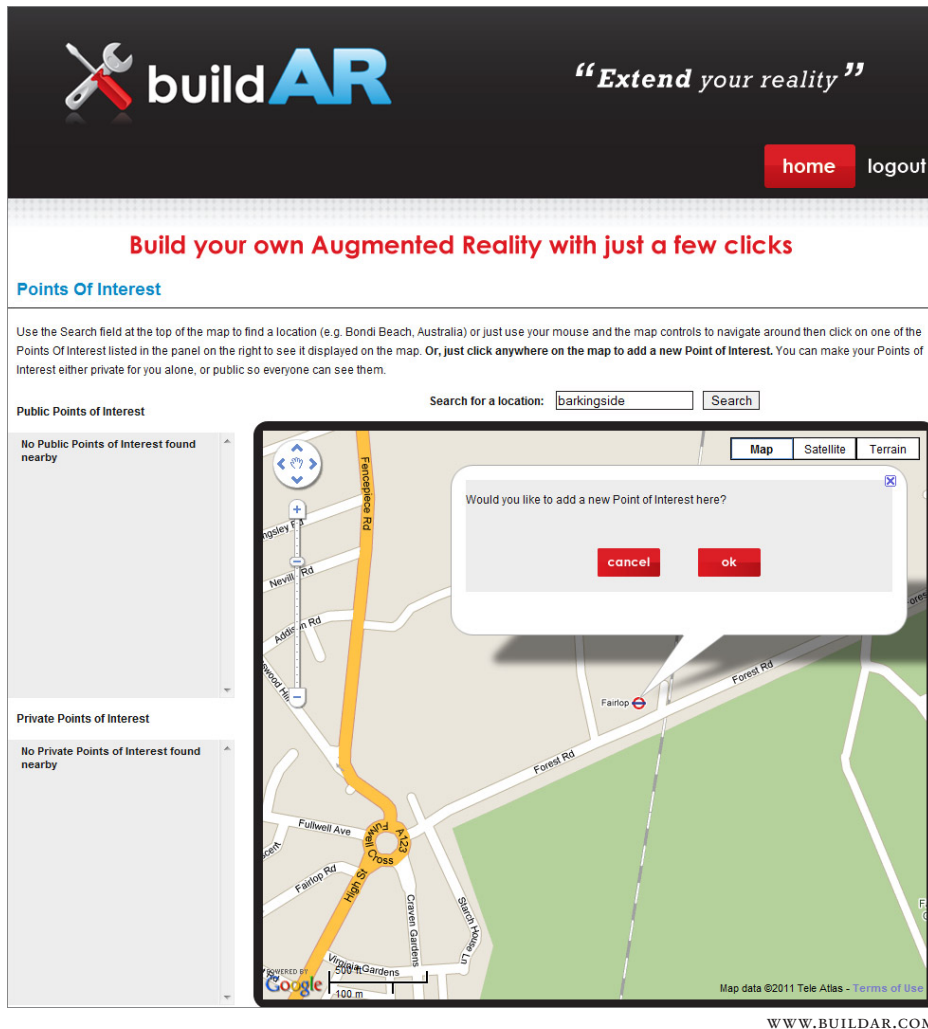
Hoppala is an amazing tool that takes away much of the pain involved in development, particularly if you are a novice developer looking to produce augmented reality content. You will notice that your layers contain a footnote referencing Hoppala. If you want to remove this, simply contact Hoppala regarding licensing terms.

BUILDAR

URL: <http://buildar.com/index.html>

Technical Knowledge Required: None

While Hoppala represents the Rolls Royce of quick and easy Layar development, there are other great tools that will enable you to build Layar content quickly and easily. BuildAR is another web interface application that uses Google Maps to enable you to quickly and easily drop POIs in any location. BuildAR doesn't provide the rich capabilities of Hoppala (for example, you cannot use video, audio, 3D, and so on), but it's still a great tool for quickly geotagging content. The BuildAR Dashboard is shown in Figure 9-10.



WWW.BUILDAR.COM

FIGURE 9-10: The BuildAR Dashboard

Using BuildAR

To use BuildAR, you first need to sign up for an account. Accounts are free only for a trial period, so it is worth purchasing if you are going to build and host your own POIs/layers or if you are looking for a third-party solution. The version I tested does not allow you to create your own layer, so any content you create will live inside the BuildAR layer. Users need to load the BuildAR layer on their mobile devices to see your content. Additionally, your content will be shown with all other developers who have published via the BuildAR web site. Future versions of the BuildAR tool will enable you to create your own Layer so your content will not be mixed with other parties' POIs. For now, however, that's a limiting factor and the flexibility you give up in return for easy layer creation.

Creating POIs with BuildAR is simple: just search the Google Maps interface, click, and then provide some simple details about the POI. BuildAR also enables you to create POIs directly from your mobile device so you can geo-tag interesting content as you find it on your phone.

BuildAR is a particularly useful tool if you want to showcase augmented reality to potential clients. You can create a simple POI that shows the power of AR and how they can list their brands while you focus on the up-sell of creating 3D, video, and all the other wonderful features of LayaR.

SKALOOP

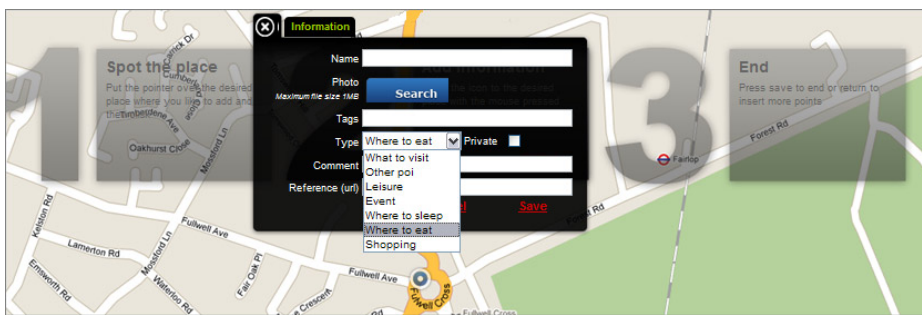
URL: <http://www.skaloop.com>

Technical Knowledge Required: None

Like BuildAR, Skaloop is another web-based tool that enables any novice developer to create LayaR content. Like BuildAR, Skaloop doesn't allow you to create your own layers; any content you create will be shared with all other Skaloop content providers and will be accessible via the Skaloop layer. However, Skaloop enables the uploading of photos and has a social aspect around the content.

Configuring Skaloop

Figure 9-11 shows the POI configuration for Skaloop.

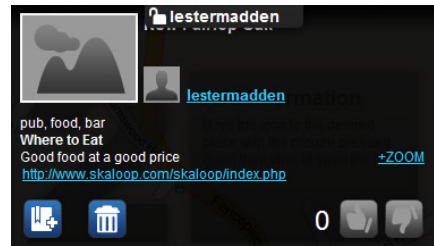


SKALOOP DEVELOPED BY SIGNO INGENIERÍA DEL TERRITORIO SL.

FIGURE 9-11: POI configuration for Skaloop

When you create a Skaloop POI, you specify a category type (for example, perhaps it's a place to eat). Other Skaloop users can view your POI and click a thumbs-up icon or a thumbs-down icon to offer their opinions of the quality of that particular establishment (see Figure 9-12).

From the developer's perspective, Skaloop is probably not going to become a tool in your development arsenal because its usefulness as a rapid prototyping tool is limited. However, it's useful when you want to upload POIs that you are looking for feedback on.



SKALOOP DEVELOPED BY SIGNO INGENIERÍA
DEL TERRITORIO SL.

FIGURE 9-12: Providing feedback on POIs

SUMMARY

In the last three chapters, you have learned a lot about developing for the Layar browser. Yet despite all you have learned, there is still so much more to Layar. As you learned in this chapter, Layar has a community of developers building content and tools for other developers to use. Tools like Hoopala and BuildAR, which enable you to quickly prototype layers, show the power of AR. In this chapter, you also learned how to use Layar Intent to combine your layers with QR codes and web sites to enable users to quickly start using your content. Now that you have mastered the basics of Layar, you're ready to learn about junaio and begin experimenting with 3D models and objects.

PART IV

junaio

- ▶ **CHAPTER 10:** Creating junaio Channels
- ▶ **CHAPTER 11:** Natural-Feature Tracking and Visual Search with junaio

10

Creating junaio Channels

WHAT'S IN THIS CHAPTER?

- How to set up the Apache web server
- How to add location-based POIs
- How to add 3D content
- How to add simple animation

In this chapter, you will learn how to develop junaio channels that are available for both the Android and the iPhone. Developed by metaio (www.metaio.com), which is widely regarded as a pioneer in AR, junaio is built upon metaio's Unifeye platform, which has been used to create solutions for millions of users. In fact, if you have tried any of the markerless demos at augmentedplanet.com, there is a good chance that you used metaio's technology in some capacity. What makes junaio so advanced is that not only does it support 3D, it empowers you to animate your objects and provide user interaction. It is also the only browser that supports natural-feature tracking, so you can build powerful image recognition content.

Before you can begin building junaio channels, you must create an account to register for your API key. Since junaio content lives on your server, the API key will be used to validate the junaio server to the Callback function on your server. This is a safeguard in case somebody requests information from your server without providing a valid junaio key. If that happens, you can deny access and protect your data.

Before you get started, you'll also need the following:

- An FTP server and an application to upload files to your server (such as FileZilla).
- A PHP editor (such as PSPad) to edit PHP files (visit www.pspad.com).
- Nearby latitude and longitude coordinates.



Please register now for your free developer account at www.junaio.com/publisher/signup/user/new.

UNDERSTANDING THE REQUIREMENTS

To create junaio channels, you will need access to a publicly visible web server that you will use to host your content. Setting up the server can be tricky; it requires several packages to be installed and configured.

First, your server must be configured with the following software:

- Apache web server
- PHP5

Or

- ASP.NET

In addition, if you want to send notifications to your channel or use encryption, you must install the Zend Framework (Apache web server only). I recommend that you install the Zend Framework minimal package to ensure that all the sample code works as expected.



Download the minimal Zend Framework package from <http://framework.zend.com/download/current>

SETTING UP THE APACHE SERVER

metaio provides you with several setup packages that simplify the creating of channels. These packages include a sample Callback function that contains the code necessary to display content to the user. To get started, download a package from the junaio developer site. You will notice that there are several packages that can be downloaded. metaio generously provides “Out of the Box” samples for each type of channel you can create. These are designed to get developers up and running quickly and they contain relevant plumbing code. For now, however, download the basic Getting Started Package at www.junaio.com/publisher/serversetup. Figure 10-1 shows the various junaio packages that can be installed.

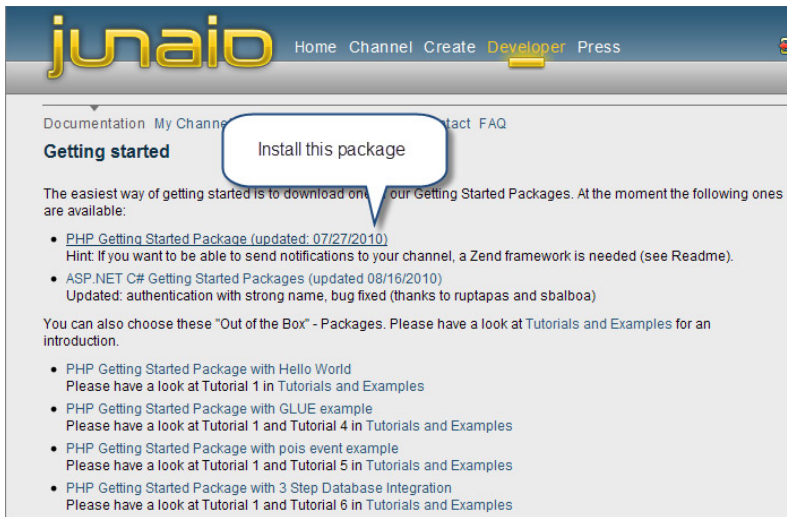


FIGURE 10-1: junaio's Getting Started Packages

In addition to the steps detailed here, I suggest you also read the readme file that accompanies the package; it contains updated information and changes relevant to the setup process. I will assume that you are already familiar with your chosen web server and have an FTP program like FTP Surfer installed to upload your files.

With that in mind, once you have download the startup package and the Zend Framework, follow these steps to setup your server:

1. Unpack all files to your computer.
2. Create a Zend subfolder under the junaio library folder on your server and copy the Zend Framework files you downloaded.
3. On the server, change the `_.htaccess` filename to `.htaccess`

Once all the files have been successfully copied to your server, find your API Key by visiting: www.junaio.com/publisher/main. You should always keep this API Key private because it is unique to you and will be used to authenticate junaio requests to your server. Figure 10-2 shows where to obtain your unique API key.



FIGURE 10-2: The junaio API key

Adding Your API Key

One of the files installed by the startup package is a file called `config.php`. Once you have located your API key, enter it into the `/config/config.php` file, as shown in Listing 10-1.



Available for
download on
Wrox.com

LISTING 10-1: config.php

```
<?php

/**
 * @copyright Copyright 2009 metaio GmbH. All rights reserved.
 * @link      http://www.metaio.com
 * @author    Bernhard Heindl
 **/

//General channel stuff
define('JUNAIO_API_HOST', 'http://api.junaio.com'); //
//URL to the junaio API
define('AUTH_DATE_TOLERANCE', 15 * 60 * 1000); //
//Set time period for valid requests: 15 minutes
define('JUNAIO_KEY', 'Your_API_KEY_HERE'); // Junaio developer key
```

Locate the following line in the `config.php` file:

```
define('JUNAIO_KEY', 'Your_API_KEY_HERE'); // Junaio developer key
```

Copy and paste your 32-digit API key between the quotes. Once complete, it should look something like the following extract:

```
define('JUNAIO_KEY', '58xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx4e');
// Junaio developer key
```

Save the file and ensure the latest version is uploaded to your web server.

CREATING YOUR FIRST CHANNEL

Before you can check your configuration, you must create a channel. The first part of the process requires you to create the channel with the details as they will appear in the junaio client. This is what will appear to your users once you have published and when they browse your channel. The second part of the process requires you to edit the PHP files and add the code to make your channel do something useful.

Creating the Client Listing

For your first channel, we will walk through the process so you can see what happens at every step. To create a new channel:

1. Log into the developer portal on the junaio web site (www.junaio.com/publisher/mychannels).
2. In the navigation bar on the right of the page, click the New Channel link.

You should see a screen similar to that shown in Figure 10-3.

FIGURE 10-3: Creating a new junaio channel

Most of the form is pretty straightforward. However, it is worth taking a look in more detail and covering some of the finer points of creating channels:

- **State of the channel.** This field specifies the current state of the channel. Initially the channel should be placed in the New (non-public) state so it is visible only to you. When you have fully developed and tested the channel and you're happy with how it works, submit it to metaio for approval by selecting the Submit state. Once metaio has approved it, you can take it offline and hide it from the general public by selecting the Archive state. This effectively disables your channel for everyone but you.
- **Channel Name.** This required field is the name of the channel as it will appear to users in the client. You should make this name as descriptive as possible.
- **Channel description.** This field enables you to enter a description of up to 512 characters to describe your channel. This text will appear in the client and you should aim to make this as descriptive as possible to encourage users to try your channel. A good description can make the difference between a user trying your channel or ignoring it.
- **Channel Short Name.** This field is a short name allocated by you to describe the channel. This is hidden from view and serves mainly as a reminder to the developer.

- **Thumbnail.** This thumbnail graphic represents the icon that will appear in the junaio client and will be associated with your channel. It is not the thumbnail used with POIs. Think of it as your launch icon or your company logo.
- **Homepage URL.** If specified, the information in this field enables users to navigate to your home page from your channel's information page. If you leave the home page URL blank, junaio omits the Open Web button.
- **Callback URL.** This required field is the URL to the location on your server where you will host the Callback function. The Callback function handles the authentication and loads the channel and actions requested by users.



If you use the Getting Started Package, the Callback function is named `index.php` and it can be located in the `/html/` folder. Depending on the location where you installed the junaio package, the URL will be something like `www.myserver.com/html/index.php`.

- **Channel Categories.** In this required field, select a channel that best describes your channel so potential users can easily find your channel. You can select several channels from the list so your channel will appear in several categories. This is particularly useful if your channel spans several genres; however, if you select all the categories, your channel will probably be rejected when you attempt to publish.
- **Channel Visibility.** Once you publish your channel, it's visible to all junaio users. If you want to restrict the visibility to just your friends, you can mark it as private. This can be useful if you are publishing private content (for example, Uncle Bob's house, David's holiday home, and so on).
- **Enable in-channel search.** If your channel contains a lot of content or you want users to search your channel, you can enable it here. Once enabled, an additional search box will be displayed to users of your channel and junaio will pass the search string to your server.
- **Supported Features.** This field specifies the features of your channel.
- **Channel refresh time.** Your channel is automatically refreshed when users open your channel or if they change their locations. In this field, you can specify in the number of seconds that elapse before the channel is automatically refreshed.

Now it's time to create your first channel. Be sure to:

- Leave the State of the channel set to New. (Let's not publish just yet.)
- In the Callback URL field, type the full URL to the `index.php` file located in the HTML folder.
- If you don't have a thumbnail image, leave this blank.
- Leave Supported Features set to None.

Figure 10-4 shows how I created my channel.

Documentation My Channels Tutorials and Examples Contact FAQ

Please be aware, that after editing a channel, it will have to be resubmitted!
You are editing this channel:

18121 **Lesters Testing Channel**
Information channel
Subscribed by: 1

State of the channel: new (non public)

Channel Name: Lesters Testing Channel

Channel description: This channel is used to test how to create junaio channels.

Channel Short Name: My Testing

Thumbnail:

Homepage URL: http://www.augmentedplanet.com

Callback URL: http://www.augmentedplanet.com/wp-conti

Channel Categories (multiple selection possible): Banking, Culture, Food, Games, News

Channel Visibility: private

Enable in-channel search:

Supported Features: None Junaio GLUE Visual Search LLA Marker

Channel refresh time (in seconds): 0

© metaio inc. 2010 Developer | Press | FAQs | Contact | Terms and Privacy | Imprint

FIGURE 10-4: My junaio channel

Figure 10-5 shows how the information is used in the junaio client.

Notice that I have several testing channels. If you don't specify a thumbnail image, your channel will be assigned the default junaio image. Clicking the channel enables the user to get more information about the channel. As shown in Figure 10-6, junaio clearly indicates that the channel has not been approved. This is where you need a compelling description to tell your potential users exactly why they should use your channel.

Depending on the mobile device you are using, you may also have the ability to manage your channel directly from your phone. The iPhone, for example, enables me to edit the name of the channel, the description, the listed category, or even hide/enable the channel. Figure 10-7 shows how to manage your junaio channel from the iPhone.

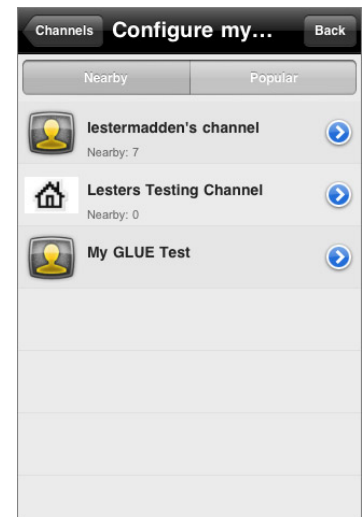


FIGURE 10-5: My channels in the junaio client

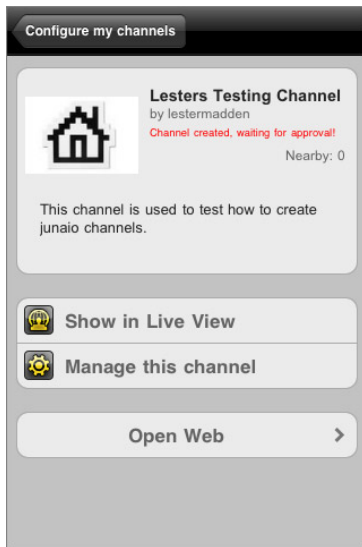


FIGURE 10-6: About my channel

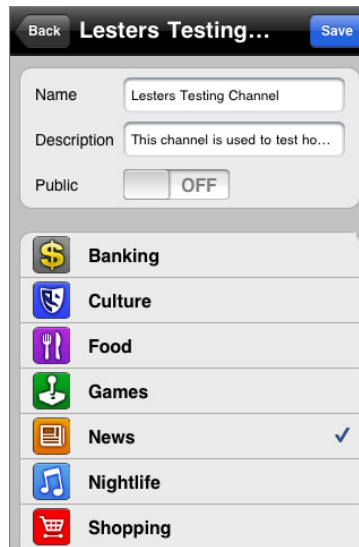


FIGURE 10-7: Managing channels on the iPhone

At this stage, load junaio and log in to your account using the same credentials you used to create your developer account. You need to use the same account to find your channels because they are currently not published. Once you load the client, click the My junaio button and type your details. Then navigate to Favorites and you should see your channel listed. Of course, at this point, you haven't configured any POIs, so the channel does nothing except look nice.

Once you have entered your login details into the junaio client, return to the junaio developer web site to ensure that your server is configured correctly.

Testing Your Server Configuration

Now that your channel has been added, you should see some additional options that can be performed.

- **Validate it.** This option performs several checks, including checking to ensure the XML is valid.
- **Edit.** This option returns you to the New Channel screen and enables you to make changes.
- **Delete.** This option removes your channel and deletes it permanently.
- **Submit.** This option sends your channel to metaio for approval and publishing.

Back on the junaio web site, you should be looking at a screen similar to the one shown in Figure 10-8. If you see something different, just click the My Channels link from the right navigation menu on the junaio web site.

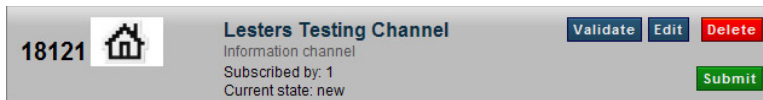


FIGURE 10-8: Channel options

You might have noticed a number displayed next to your channel. This is a unique number assigned automatically by junaio to identify channels. Should you ever need to contact junaio support, you should provide them with this number. The option you are interested in is the Validate button. You should use the Validate option often because it's the only way to find errors in the XML and configuration files. It helps with those annoying errors where the client doesn't do something as expected. Running a validation report helps to discover the problem. For now, at least, you want to check that the server is setup correctly.

Click the Validate button and several tests will take place. The tests check to ensure the server is configured correctly

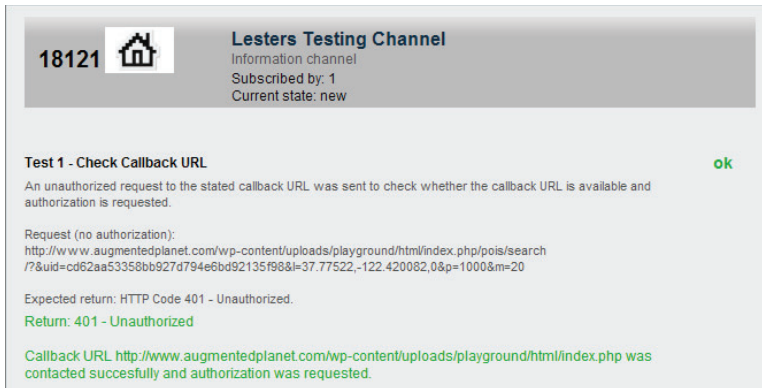
Validation Tests


The first test ensures the Callback function can be found. If you see an error similar to what's shown in Figure 10-9, you have not specified the correct URL to the index.php file located in the html folder. Remember, this should look something like: `www.myserver.com/html/index.php` but will be dependent on the directory structure on your server. Before you can continue, you need to fix the problem by editing the channel and fixing the URL. If you still have problems, refer to Appendix C, "junaio Troubleshooting and Support," for more information.



FIGURE 10-9: Results of test 1 (fail)

If your Callback function is found and is working correctly, you should see a screen similar to what's shown in Figure 10-10.



18121  **Lesters Testing Channel**
Information channel
Subscribed by: 1
Current state: new

Test 1 - Check Callback URL ok

An unauthorized request to the stated callback URL was sent to check whether the callback URL is available and authorization is requested.

Request (no authorization):
http://www.augmentedplanet.com/wp-content/uploads/playground/html/index.php/pois/search/?&uid=cd62aa53358bb927d794e6bd92135f98&l=37.77522,-122.420082,0&p=1000&m=20

Expected return: HTTP Code 401 - Unauthorized.

Return: 401 - Unauthorized

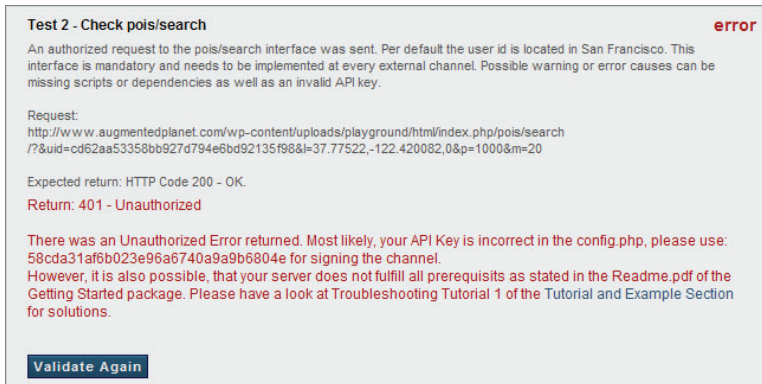
Callback URL <http://www.augmentedplanet.com/wp-content/uploads/playground/html/index.php> contacted successfully and authorization was requested.

FIGURE 10-10: Results of test 1 (pass)

If that's the case, you know that junaio can communicate with your server and you are installed correctly.

API Key Tests

The second test checks that the `/src/search.php` file can be read correctly. If this returns an error, you most likely entered the API key incorrectly or the `search.php` file could not be found. Figure 10-11 shows a failure of test 2.



Test 2 - Check pois/search error

An authorized request to the pois/search interface was sent. Per default the user id is located in San Francisco. This interface is mandatory and needs to be implemented at every external channel. Possible warning or error causes can be missing scripts or dependencies as well as an invalid API key.

Request:
http://www.augmentedplanet.com/wp-content/uploads/playground/html/index.php/pois/search/?&uid=cd62aa53358bb927d794e6bd92135f98&l=37.77522,-122.420082,0&p=1000&m=20

Expected return: HTTP Code 200 - OK.

Return: 401 - Unauthorized

There was an Unauthorized Error returned. Most likely, your API Key is incorrect in the `config.php`, please use: `58cda31af6b023e96a6740a9a9b6804e` for signing the channel.
However, it is also possible, that your server does not fulfill all prerequisites as stated in the `Readme.pdf` of the `Getting Started` package. Please have a look at `Troubleshooting Tutorial 1` of the `Tutorial and Example Section` for solutions.

[Validate Again](#)

FIGURE 10-11: Results of test 2 (fail)



Before you can continue, you will need to fix any error that relates to test 1 or test 2. (Errors beyond test 2 are not relevant at this point.) Typically, test 1 errors occur because of an incorrect Callback URL; test 2 errors are likely caused by an incorrect API key. If necessary, refer to the Appendix C, “junaio Support and Parameters.”

Hopefully, both tests have completed successfully and you are looking at a screen similar what's shown in Figure 10-12.

The screenshot shows the Junaio Developer interface for a channel named "Lesters Testing Channel". The channel is currently in a "new" state and has 1 subscriber. The interface displays three test results:

- Test 1 - Check Callback URL:** Passed (ok). An unauthorized request to the stated callback URL was sent to check whether the callback URL is available and authorization is requested. The request (no authorization) was: `http://www.augmentedplanet.com/wp-content/uploads/playground/html/index.php/pois/search/?7&uid=cd62aa53358bb927d794e6bd92135f98&=37.77522,-122.420082,0&p=1000&m=20`. The expected return was HTTP Code 401 - Unauthorized, and the return was 401 - Unauthorized. The callback URL `http://www.augmentedplanet.com/wp-content/uploads/playground/html/index.php` was contacted successfully and authorization was requested.
- Test 2 - Check pois/search:** Passed (ok). An authorized request to the pois/search interface was sent. Per default the user id is located in San Francisco. This interface is mandatory and needs to be implemented at every external channel. Possible warning or error causes can be missing scripts or dependencies as well as an invalid API key. The request was: `http://www.augmentedplanet.com/wp-content/uploads/playground/html/index.php/pois/search/?7&uid=cd62aa53358bb927d794e6bd92135f98&=37.77522,-122.420082,0&p=1000&m=20`. The expected return was HTTP Code 200 - OK, and the return was 200 - OK. A message states: "Pois search request was sent successfully." A "Show request response" button is visible.
- Test 3 - Check pois/search return value:** Failed (error). A request to the pois/search interface was sent. The return XML will be checked whether it is a valid XML file and whether it is empty or not. Possible warning or error causes can be invalid xml creation and/or return or wrongly received parameters (be aware that this is a GET method). The request was: `http://www.augmentedplanet.com/wp-content/uploads/playground/html/index.php/pois/search/?7&uid=cd62aa53358bb927d794e6bd92135f98&=37.77522,-122.420082,0&p=1000&m=20`. The expected return was HTTP Code 200 and valid XML, and the return was 200 - OK. An error message states: "An invalid XML was returned. 3 error(s) and 0 warning(s)." A box contains the following error details:


```
Error: POI #0[poi_id:string]: POI ID ([poi_id:string]) has invalid characters.
Error: POI #0[poi_id:string]: Missing or invalid interaction/feedback (-poi interaction/feedback=??). It must be set to "none" or "click"
Error: POI #0[poi_id:string]: Unknown MME Type. Please see Layout and Interaction.
```

 A "Show request response" button is visible.

The interface also includes a "Validate Again" button and a "Feedback" sidebar with sections for Publisher Information, Channels, Reference Guide, Misc, and Change Lists.

FIGURE 10-12: Tests 1 and 2 passed!

You might have noticed in Figure 10-12 that the third test returned an error. This is because you currently do not have any POI content in your search.php file. Let's rectify that problem by adding some content to the channel.

Setting up the POI

As part of the startup package you installed, metaio provided you with a sample file named search.php. This file is the function that returns your POI content that will be displayed in the camera window. As shown in Listing 10-2, the file contains the basic XML structure for a POI.

Available for
download on
Wrox.com**LISTING 10-2: The search.php file from the startup package**

```

<?php

/**
 * @copyright Copyright 2009 metaio GmbH. All rights reserved.
 * @link      http://www.metaio.com
 * @author    Bernhard Heindl
 **/

/**
 * When the channel is being viewed, a poi request will be sent
 * $_GET['l']...(optional) Position of the user when interacting with
 the POI
 * $_GET['p']...(optional) perimeter of the data requested in meters.
 * $_GET['uid']... Unique user identifier
 * $_GET['m']... (optional) limit of to be returned values
 * $_GET['page']...page number of result. e.g. m = 10: page 1: 1-10; page
 2: 11-20, e.g.
 **/

echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>
  <poi id=\"[poi_id:string]\" interactionfeedback=\"(none|click|text)\">
    <name><![CDATA[[name:string]]></name>

    <description><![CDATA[[description:string]]></description>

    <date>[placed_date:datetime]</date>

    <l>[latitude:float],[longitude:float],[altitude:float]</l>
    <o>[X:float],[Y:float],[Z:float]</o>

    <minaccuracy>[accuracy:int]</minaccuracy>
    <maxdistance>[maxdistance:int]</maxdistance>

    <mime-type>[mime-type:string]</mime-type>

    <mainresource>[model_uri:string]</mainresource>
    <thumbnail>[thumbnail_url:string]</thumbnail>
    <icon>[icon_url:string]</icon>

    <phone>[phonenummer:string]</phone>
    <mail>[email:string]</mail>
    <homepage>[homepage:url]</homepage>
  </poi>
</results>";
?>

```

Listing 10-3 can be used to replace the search.php file used in Listing 10-2. This sample adds a single POI to our channel.

**LISTING 10-3: Your first POI (search.php)**Available for
download on
Wrox.com

```

<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

    <!-- A unique number for our POI, and settings for user interaction-->
    <poi id=\"1\" interactionfeedback=\"none\">

        <!-- The name of our POI as will be seen in the camera window-->
        <name><![CDATA[My First POI]]></name>

        <!-- The description of the POI that will be shown when selected-->
        <description><![CDATA[My first junaio POI]]></description>

        <!-- The lat/lon of the POI-->
        <!-- Use a location near you for testing -->
        <l>51.5848,0.0886,0</l>

        <!-- Orintation of the POI bubble. (mainly used for 3D content)-->
        <o>0,0,0</o>

        <!-- The mime-type, for now we are using text-->
        <mime-type>text/plain</mime-type>

        <!-- The link to the image that will be shown in the camera window-->
        <!-- Should not exceed 150x150px-->
        <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>

        <!-- The link to the image that will be shown in the map view -->
        <!-- Should not exceed 40x40px-->
        <icon>http://www.yourserver.com/mappin.jpg</icon>

    </poi>
</results>";
?>

```



In Listing 10-3, I am using the latitude and longitude of a location near me. This probably will not be visible to you. In Chapter 4, you should have used a tool like Google Maps to obtain the latitudes and longitudes of some nearby locations. You should replace my <l> with your own coordinates.

```

<!-- The lat/lon of the POI-->
<!-- Use a location near you for testing -->
<l>51.5848,0.0886,0</l> <!-- your lat/lon here -->

```

Examining the XML

Since this is your first channel, it is worth walking you through the XML so you can edit it to contain the coordinates and description for a location near you.

The following syntax simply assigns the POI a unique ID and sets the user interaction to none:

```
<poi id="1\" interactionfeedback="none\">
```

The following syntax provides the name of the POI as it will show up in the augmented reality camera view:

```
<name><![CDATA[My First POI]]></name>
```

The following syntax provides the description shown to users when they click the POI:

```
<description><![CDATA[My first junaio POI]]></description>
```

The following syntax provides the latitude, longitude, and altitude of our POI:

```
<l>51.5848,0.0886,0</l>
```

Notice that there are no spaces. If you copy the latitude and longitude from another source, ensure you remove the spaces between the coordinates or you will have errors. Also, as you learned in Chapter 4, if you use Google Earth to create the POI, you must swap the coordinates around.

The following syntax sets the orientation of the POI:

```
<o>0,0,0</o>
```

This tag is mainly used when working with 3D content, but it needs to be present to pass the validation tests. If it is missing, junaio will report an error. To ensure the validation test is passed successfully, set the tag to a value of 0,0,0.

The following syntax determines that our POI is text-only:

```
<mime-type>text/plain</mime-type>
```

In the following syntax, the thumbnail is the icon that will appear in the AR view. Dimensions for the thumbnail should not exceed 150x150 pixels.

```
<thumbnail>http://www.yourserver.com/logo.jpg </thumbnail>
```

Figure 10-13 shows the image used for the logo.

The following syntax details the icon that will appear when users view the map. Dimensions should not exceed 40x40 pixels.

```
<icon>http://www.yourserver.com/mappin.jpg </icon>
```

Figure 10-14 shows the image used for the icon.



FIGURE 10-13:
Thumbnail image



FIGURE 10-14:
The image to appear on the map view

 You can download all the images and resources for this chapter from the Wrox web site.

Once you have edited the search.php file, save it and upload it to your server. It is always a good idea to return to the My Channels page on the junaio web site and run a validation test. This helps remedy errors such as missing or invalid tags.

Fix any errors and then load the junaio client on your mobile device. Locate your channel in the UI and your channel should return one POI in the camera view. In Figure 10-15, notice that the name of the POI has been taken from the <name> tag added in the XML.

When the POI has been selected additional information about the POI can be displayed to the user. This is provided by the <description> tag you added to the XML. Figure 10-16 shows the description tag in action.

If you view the POI in the map view, you see the POI drawn on the map and the location indicated by the image you used in the <icon> tag. Figure 10-17 shows the icon image displayed on the map.

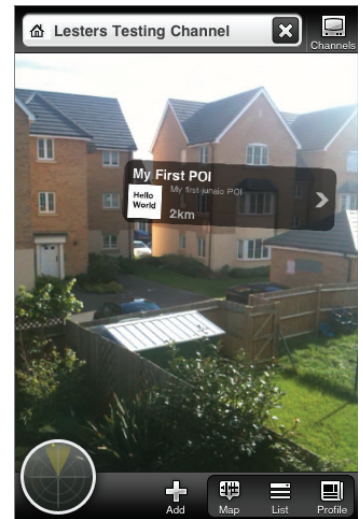


FIGURE 10-15: My first POI in junaio



FIGURE 10-16: My POI description

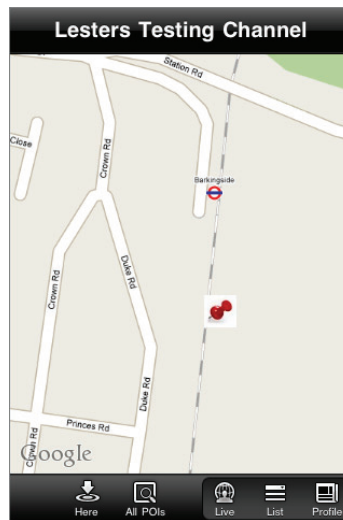


FIGURE 10-17: The POI on the map with pin

Creating Multiple POIs

Obviously, a channel becomes more exciting when there are more POIs displayed in the channel. Listing 10-4 shows how support for additional POIs can be added to the search.php file.



Available for
download on
Wrox.com

LISTING 10-4: Several POIs (search.php)

```
<?php

echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

    <!-- A unique number for our POI, and settings for user interaction-->
    <poi id=\"1\" interactionfeedback=\"none\">

        <!-- The name of our POI as will be seen in the camera window-->
        <name><![CDATA[My First POI]]></name>

        <!-- The description of the POI that will be shown when selected-->
        <description><![CDATA[My first junaio POI]]></description>

        <!-- The lat/lon of the POI-->
        <!-- Use a location near you for testing -->
        <l>51.5848,0.0886,0</l>

        <!-- Orientation of the POI bubble. (mainly used for 3D content)-->
        <o>0,0,0</o>

        <!-- The mime-type, for now we are using text-->
        <mime-type>text/plain</mime-type>

        <!-- The link to the image that will be shown in the camera window-->
        <!-- Should not exceed 150x150px-->
        <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>

        <!-- The link to the image that will be shown in the map view -->
        <!-- Should not exceed 40x40px-->
        <icon>http://www.yourserver.com/mappin.jpg</icon>

    <!-- second POI added here -->

</poi>

    <!-- Notice the unique number -->
    <poi id=\"2\" interactionfeedback=\"none\">

        <name><![CDATA[My Second POI]]></name>
        <description><![CDATA[Yet another junaio POI]]></description>
        <l>51.579527,0.088402,0</l>
```



```

        <o>0,0,0</o>
        <mime-type>text/plain</mime-type>

    <!-- Using a different logo than before-->
    <thumbnail>http://www.yourserver.com/logo2.jpg</thumbnail>

    <!-- Using a different icon than before -->
    <icon>http://www.yourserver.com/mappin2.jpg</icon>

    </poi>

</results>;
?>

```

As you can see, to add multiple POIs, simply create an additional `<poi>` node and add the relevant POI content. Each POI can have its own `<thumbnail>` and `<icons>` if needed, so you can build channels that display different classes of content.

In Listing 10-4, I have added a new `<poi>` node before the `</results>` tag. You'll notice that the new node has some differences.

First, the `<poi id>` value has been incremented. The `<poi id>` should always be a unique value. You are free to create your own numbering structure, but in most cases, a simple incremental system will be enough. You may access POIs from a database where you have no control over the value. As long as the number is unique, the value isn't important.

```
<poi id="2" interactionfeedback="none">
```

Each POI in junaio can be unique with different thumbnail and icons images. As shown in the following syntax, I used a thumbnail and icon different from the previous POI:

```

    <thumbnail>http://www.yourserver.com/logo2.jpg</thumbnail>
    <icon>http://www.yourserver.com/mappin2.jpg</icon>

```

As you would expect, each POI can have a different name and a different description tag:

```

    <name><![CDATA[My Second POI]]></name>
    <description><![CDATA[Yet another junaio POI]]></description>

```

And, finally, I used the latitude and longitude of another nearby location:

```
<l>51.579527,0.088402,0</l>
```

The previous XML produces the screen displayed in Figure 10-18 when viewed in the client.

In Figure 10-19, there are different icons shown on the map. This is useful if you are displaying different classes of POIs. For example, you might be showing where people can eat and drink. A burger icon could represent fast food while a plate could represent table dining.

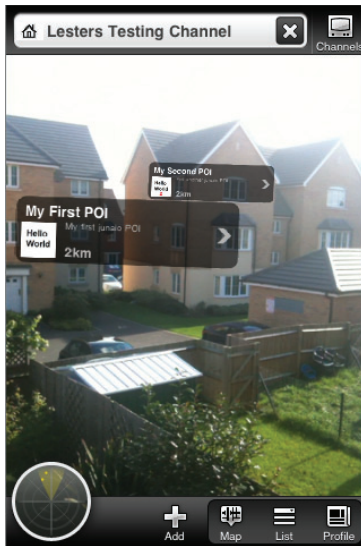


FIGURE 10-18: Several POIs in the junaio client

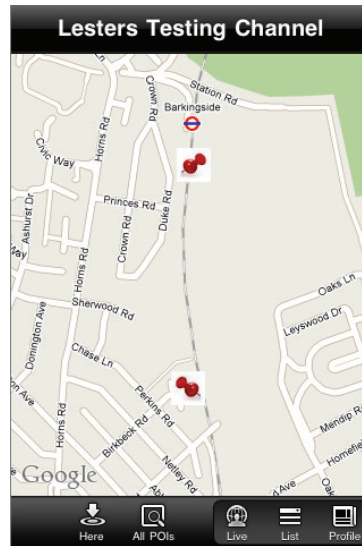


FIGURE 10-19: Several POIs in the map



junaio automatically refreshes the channel when it's opened or when the user changes position. Since you are testing, you probably haven't moved, so you may need to force a refresh. In most cases, this can be done by opening another channel and then selecting your channel again. In some cases, however, it is useful to clear the cache and force junaio to download all the graphics again. To clear the cache, iPhone users should select Settings from the iPhone menu, then select junaio, and then click the Clear on Next Launch button. Android users should kill the process if experiencing unwanted behavior.

Including Optional Parameters

As you have seen with Layar and Wikitude, you can include optional content with our POI. By including the following tags in your XML, you can add functionality such as placing a call, sending an e-mail, or visiting the relevant home page:

```
<phone>+441112222</phone>
<mail>an_email_address@email.com</mail>
<homepage>www.augmentedplanet.com</homepage>
```

It's good practice to prefix the phone number with the full country and area code so you can ensure there are no unwanted dialing problems. If you want to offer the user routing functionality to your POI, you can simply include the following tag:

```
<route>true</route>
```

Listing 10-5 adds support for placing a call, sending an e-mail, visiting a web site, and displays the route from the user's current location to the POI on a Google map.



LISTING 10-5: POIs with optional parameters (search.php)

Available for
download on
Wrox.com

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

    <!-- A unique number for our POI, and settings for user interaction-->
    <poi id=\"1\" interactionfeedback=\"none\">

        <name><![CDATA[junaio POI]]></name>
        <description><![CDATA[A junaio POI with
            functionality]]></description>

        <!-- The lat/lon of the POI-->
        <!-- Use a location near you for testing -->
        <l>51.5848,0.0886,0</l>
        <o>0,0,0</o>
        <mime-type>text/plain</mime-type>
        <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
        <icon>http://www.yourserver.com/mappin.jpg</icon>

        <!-- This POI supports the following-->
        <!-- Calling a phone number-->
        <phone>+441112222</phone>
        <!-- Sending an email-->
        <mail>an_email_address@email.com</mail>
        <!-- Visiting a website -->
        <homepage>www.augmentedplanet.com</homepage>
        <!-- Enable routing -->
        <route>>true</route>

    </poi>

</results>";
?>
```



Figure 10-20 shows how the XML is displayed in the channel once the optional parameters have been included.

FIGURE 10-20: My Channel with optional functionality



As you try out the additional tags in the XML, be sure to run the validation test to check for any errors that might have crept into the code.

Understanding the [name:string] Error

You may have spotted an error in the junaio client telling you about a [name:string] error (see Figure 10-21). Usually this occurs when you select a blank area in the camera view.

First, there is nothing wrong with your code. If you return to the junaio web site and validate your channel, you will see that even though it works correctly, there is an error at Test 5. Figure 10-22 shows an error with Test 5.

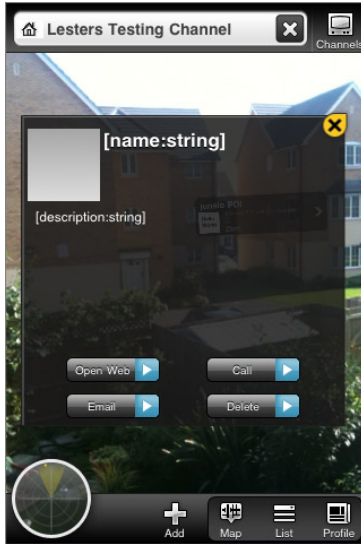


FIGURE 10-21: A [name:string] error in the client

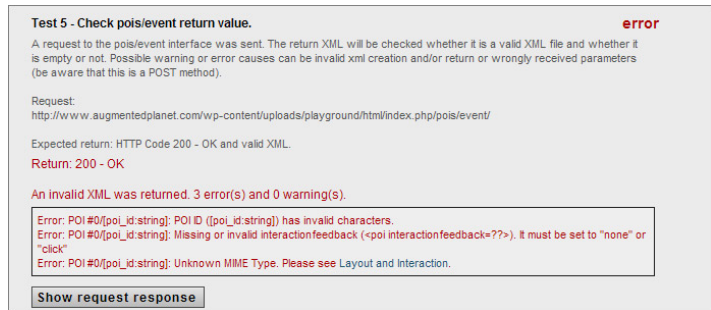


FIGURE 10-22: A Test 5 error

Because you are using a sample package that is set up for user interaction, one of the files contains sample code that you don't need right now. In chapter 12, when you add interactions to channels, you will require this functionality. But for now, let's walk before we learn to run. The easiest way to fix this is to edit your Callback function to remove a call to the `event.php` file in the `index.php` file.

Edit the `index.php` file located in the `html` folder and comment out the line that reads:

```
include '../src/event.php';
```

should become:

```
// include '../src/event.php';
```

After changing the file, run another validation and Test 5 now passes (see Figure 10-23) with a warning rather than a failure. If you now test your channel in junaio, you will see that the [name:string] error has also disappeared.

junaio
Home Channel Create **Developer** Press
Logout Download Newsletter

You are currently validating this channel...

18121

Lesters Testing Channel
Information channel
Subscribed by: 1
Current state: new

Test 1 - Check Callback URL ok

An unauthorized request to the stated callback URL was sent to check whether the callback URL is available and authorization is requested.

Request (no authorization):
http://www.augmentedplanet.com/wp-content/uploads/placeholder/index.php/pois/search?78&uid=c082aa53355b869276f94e6b92135f984a-37.77522-122.420082.0&p=1000&m=20

Expected return: HTTP Code 401 - Unauthorized.
Return: 401 - Unauthorized

Callback URL http://www.augmentedplanet.com/wp-content/uploads/placeholder/index.php was contacted successfully and authorization was requested.

Test 2 - Check pois/search ok

An authorized request to the pois/search interface was sent. Per default the user id is located in San Francisco. This interface is mandatory and needs to be implemented at every external channel. Possible warning or error causes can be missing scripts or dependencies as well as an invalid API key.

Request:
http://www.augmentedplanet.com/wp-content/uploads/placeholder/index.php/pois/search?78&uid=c082aa53355b869276f94e6b92135f984a-37.77522-122.420082.0&p=1000&m=20

Expected return: HTTP Code 200 - OK
Return: 200 - OK

Pois search request was sent successfully.

[Show request response](#)

Test 3 - Check pois/search return value. ok

A request to the pois/search interface was sent. The return XML will be checked whether it is a valid XML file and whether it is empty or not. Possible warning or error causes can be invalid xml creation and/or return or wrongly received parameters (be aware that this is a GET method).

Request:
http://www.augmentedplanet.com/wp-content/uploads/placeholder/index.php/pois/search?78&uid=c082aa53355b869276f94e6b92135f984a-37.77522-122.420082.0&p=1000&m=20

Expected return: HTTP Code 200 and valid XML.
Return: 200 - OK

A valid XML was returned. 0 errors and 0 warnings.

[Show request response](#)

Test 4 - Check pois/event. ok

A request to the pois/event interface was sent. By default the user id is located in San Francisco and clicked validation_poi_1 (type "click") was used. It is not a mandatory interface, so only warnings will be given, unless the API key is incorrect. Possible warning or error causes can be missing scripts or dependencies.

Request:
http://www.augmentedplanet.com/wp-content/uploads/placeholder/index.php/pois/event/

Expected return: HTTP Code 200 - OK
Return: 200 - OK

Pois event request was sent successfully.

Test 5 - Check pois/event return value. warning

A request to the pois/event interface was sent. The return XML will be checked whether it is a valid XML file and whether it is empty or not. Possible warning or error causes can be invalid xml creation and/or return or wrongly received parameters (be aware that this is a POST method).

Request:
http://www.augmentedplanet.com/wp-content/uploads/placeholder/index.php/pois/event/

Expected return: HTTP Code 200 - OK and valid XML.
Return: 200 - OK

An empty XML was returned.

Geo Location - lat/lonalt:

POI ID:

[Validate Again](#)

Test 6 - Check pois/visualsearch. ok

A request to the pois/visualsearch interface was sent. For validation purposes there will be no image sent. It is only a mandatory interface, if the channel supports the Visual Search feature. Possible warning or error causes can be missing scripts or dependencies. Keep in mind that this is a POST method.

Test skipped, because this channel does not support visual search.

Test 7 - Check pois/visualsearch return value. ok

A request to the pois/visualsearch interface was sent. For validation purposes there will be no image sent. It is only a mandatory interface, if the channel supports the Visual Search feature. Possible warning or error causes can be missing scripts or dependencies. Keep in mind that this is a POST method.

Test skipped, because this channel does not support visual search.

Test 8 - Check channels/subscribe. ok

A request to the channels/subscribe interface was sent. For validation purposes a user id with action "subscribe" was sent. It is not a mandatory interface, so only warnings will be given, unless the API key is incorrect. Possible warning or error causes can be missing scripts or dependencies. The request does not expect any return value besides http status code for success or errors. Keep in mind that this is a POST method.

Request:
http://www.augmentedplanet.com/wp-content/uploads/placeholder/index.php/channels/subscribe/

Expected return: HTTP Code 200 - OK
Return: 200 - OK

Channel subscribe request was sent successfully.

[Show on Map](#) [Validate Again](#)

Publisher Information

- Overview
- Getting Started
- Layout and Interaction
- Development Guidelines
- 3D Content
- LLA Markers
- Junaio GLUE
- Tutorials and Examples

Channels

- What are channels?
- New Channel
- My Channels

Reference Guide

- Parameter Overview
- channels / subscribe
- pois / search
- pois / visualsearch
- pois / event
- models / encrypt (deprecated)
- tools / modelencrypt
- tools / trackingxml
- channels / notifications

Misc

- FAQs
- Error Codes and Authentication

Change Lists

- Change List iPhone
- Change List Android
- Change List Website

© metaio inc. 2010
Developer | Press | FAQs | Contact | Terms and Privacy | Imprint

Feedback

FIGURE 10-23: All the tests now pass

www.it-ebooks.info

ADDING IMAGES, SOUND, AND VIDEO

So far, you have restricted the channel to simple text. However, junaio allows you to easily add images, sound, and video to your POI content.

Adding Images

Adding images to your POI enables you to build much more compelling content. For example, if you are building a channel to display the location of restaurants, you can include an image of the menu so potential customers could view the type of food served. If you are building a real estate channel, you can ensure users can browse for apartments and houses, download floor plans, or view additional pictures.

To add images, sound, or video, you need to change `<mime-type>` to reflect the content you are returning and you need to provide the URL of the resource you want to include. Listing 10-6 shows how to display an image.



Available for
download on
Wrox.com

LISTING 10-6: Displaying images (search.php)

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

    <!-- A unique number for our POI, and settings for user interaction-->
    <poi id=\"1\" interactionfeedback=\"none\">

        <name><![CDATA[junaio POI]]></name>
        <description><![CDATA[A junaio POI with an image]]></description>

        <!-- The lat/lon of the POI-->
        <!-- Use a location near you for testing -->
        <l>51.5848,0.0886,0</l>
        <o>0,0,0</o>

        <!-- Changing the mime type to PNG -->
        <mime-type>image/png</mime-type>
        <thumbnail>http://www.yoursever.com/logo.jpg</thumbnail>
        <icon>http://www.yourserver.com/mappin.jpg</icon>

        <!-- The image we want to load-->
        <mainresource>http://www.yourserver.com/appicture.png</mainresource>

    </poi>

</results>";
?>
```

The Listing 10-6 code places the View Image button in the client (see Figure 10-24).



FIGURE 10-24: Including an image with the POI

In the code, the `<mime-type>` image was changed to PNG:

```
<!-- Changing the mime type to PNG -->
<mime-type>image/png</mime-type>
```

The `<mainresource>` tag included the URL to the relevant PNG file that you want to include:

```
<!-- The image we want to load-->
<mainresource>http://www.yourserver.com/appicture.png</mainresource>
```

Both JPG and PNG image formats are supported. There are no restrictions in the dimensions of the file you can use. However, bear in mind that the file will be downloaded to the device over a 3G connection. This could be a slow process, so use care with large files.

At this point, edit your `search.php` file to include support for images. Feel free to use the image I provided here or use your own image.

Playing Sounds

You can include MP3 files as part of your POI. When using the `<mainresource>` tag, you need to include a link to the relevant MP3 file. In the code shown in Listing 10-7, the `<mime-type>` is changed to an audio of type MP3. The `<mainresource>` tag is used so your POI will display an additional button to play the sound.



The supported format for audio files is MP3 (MPEG-1 Audio Layer 3) up to 48 kHz with stereo audio. If you attempt to use an audio file with higher quality, the sound will not play and the user will not be given any reason for the failure.



Available for
download on
Wrox.com

LISTING 10-7: Playing sounds (search.php)

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

  <!-- A unique number for our POI, and settings for user interaction-->
  <poi id=\"1\" interactionfeedback=\"none\">

    <name><![CDATA[junaio POI]]></name>
    <description><![CDATA[A junaio POI with sound]]></description>

    <!-- The lat/lon of the POI-->
    <!-- Use a location near you for testing -->
    <l>51.5848,0.0886,0</l>
    <o>0,0,0</o>

    <!-- Changing the mime type to MP3 -->
    <mime-type>audio/mp3</mime-type>
    <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
    <icon>http://www.yourserver.com/mappin.jpg</icon>

    <!-- The mp3 we want to load-->
    <mainresource>http://www.yourserver.com/trainsound.mp3</mainresource>

  </poi>
</results>";
?>
```

The previous code enables the Play button. Sounds are a great way to build engaging content for your users. For example, you could provide an audio tour of the POI. Be aware, however, that sound files can get quite large, so make sure you test your content on a real 3G network.

Playing Videos

You can also include video files as part of our POI. When using the `<mainresource>` tag, you must include a link to the relevant file. In Listing 10-8, the `<mime-type>` is changed to a video of type MP4. When the `<mainresource>` tag is used, your POI will display an additional button to play the video.



junaio supports the playback of MP4 video files up to a maximum of 640x480 pixels and 30fps. The video should be compliant with the H.264 standard (http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC).

If you attempt to play a video that doesn't satisfy the requirements the video will fail to play. Instead, when the user clicks the Play button the screen will change to the video player and exit immediately, returning to junaio. No error message or indication why the video failed to play will be displayed. I recommend that when you use video in your channels, test the channel on an iPhone and an Android to ensure that the video works as expected. Supported mime types for video format include video/QuickTime and video/MP4.



Available for
download on
Wrox.com

LISTING 10-8: Playing videos (search.php)

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

  <!-- A unique number for our POI, and settings for user interaction-->
  <poi id=\"1\" interactionfeedback=\"none\">

    <name><![CDATA[junaio POI]]</name>
    <description><![CDATA[A junaio POI with video]]</description>

    <!-- The lat/lon of the POI-->
    <!-- Use a location near you for testing -->
    <l>51.5848,0.0886,0</l>
    <o>0,0,0</o>

    <!-- Changing the mime type to quicktime -->
    <mime-type>video/quicktime</mime-type>
    <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
    <icon>http://www.yourserver.com/mappin.jpg</icon>

    <!-- The movie we want to load-->
    <mainresource>http://www.yourserver.com/cat.mp4</mainresource>

  </poi>

</results>";
?>
```

The previous code enables the Play button in the client (see Figure 10-25).

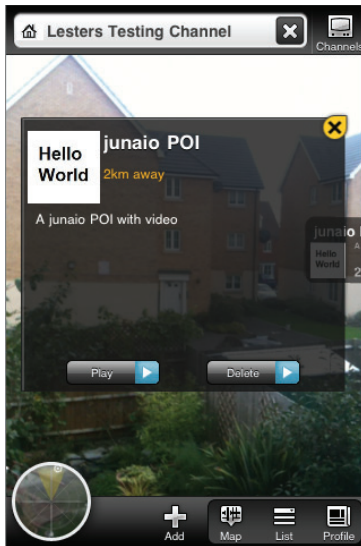


FIGURE 10-25: Playing videos in junaio



If you want to use your own videos, you can use your iPhone or Android to record a video and upload it to your server. Video is an expensive resource; you should always test your video on a 3G connection to test the user's experience. Always avoid using large files.

CREATING 3D CONTENT

You have learned how to create standard 2D POIs, and junaio also offers you the ability to present rich, interactive 3D content to your users. The latest version of the junaio client provides support for the following types:

- OBJ Models
- MD2 Models

Using 3D content is a little more involved than using a standard POI because you have to deal with the complexity of scale and orientation. Distance can also affect how your 3D objects are displayed.

Because there are more tags involved with the 3D sample shown in Listing 10-9, let's take a quick look at what's new.



LISTING 10-9: 3D sample (search.php)

Available for
download on
Wrox.com

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

  <!-- A unique number for our POI, and settings for user interaction-->
  <poi id=\"1\" interactionfeedback=\"none\">

    <name><![CDATA[3D junaio POI]]></name>
    <description><![CDATA[A 3D junaio POI]]></description>

    <!-- The lat/lon of the POI-->
    <!-- Use a location near you for testing -->
    <l>51.5848,0.0886,0</l>
    <o>0,0,0</o>

    <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
    <icon>http://www.yourserver.com/mappin.jpg</icon>

    <!-- Changing the mime type to md2 -->
    <mime-type>model/md2</mime-type>

    <!-- Forcing the 3D image to be shown rather than
         the standard POI bubble-->
    <force3d>true</force3d>

    <!-- Setting a scale -->
    <s>1</s>

    <!-- Loading a 3D resource -->
    <mainresource>http://www.junaio.com/publisherDownload/
      tutorial/metaioman.md2_enc</mainresource>
    <resources>

      <resource>http://www.junaio.com/publisherDownload/
        tutorial/metaioman.png</resource>
    </resources>

  </poi>

</results>";
?>
```

Notice the use of the `<mime-type>` of `model/md2`. Also, `<force3d>` is set to `true`. `<force3d>` causes junaio to ignore the `<thumbnail>` image while the `<s>` tag sets the scale of the 3D image. The size of the image is dependent on the distance of the POI and the size of the image in the object source file. If your POI location is far away, the 3D image will be very small. In the sample code, the 3D resource being loaded is specified in the `<mainresource>` tag and the relevant image map is applied in the `<resource>` tag.

THE RIGHT TOOLS FOR THE 3-D JOB

3D models typically consist of the 3D wireframe and a skin that is applied to the model. The `metaioman.md2` is the model file, while the `metaioman.png` file represents the clothing the image will wear. Figure 10-26 shows the `metaioman.png` file. As you can see, it would be difficult to edit without the correct tools.



FIGURE 10-26: The `metaioman.png` file

Debugging 3D

For the moment, build the sample exactly as you see in Listing 10-9 (changing the `<1>` to a latitude and longitude near you). Be sure to validate the XML when you upload it and then load the channel in the junaio client. You will see a message indicating that the image is being downloaded. You will see the POI location appear in the radar, but you probably won't see the 3D graphic appear. If you do see the junaio man 3D image, just follow along and I will explain why you may or may not see the image. Figure 10-27 shows the visual indicator that 3D content is being downloaded.

Figure 10-28 shows that the POI appears in the radar but no content is shown in the client.

Figure 10-29 indicates that the POI should be present because it's drawn correctly on the map view.

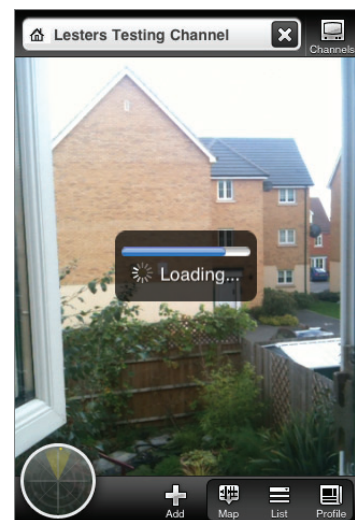


FIGURE 10-27: 3D content downloading



FIGURE 10-28: 3D content appears in the radar but isn't visible

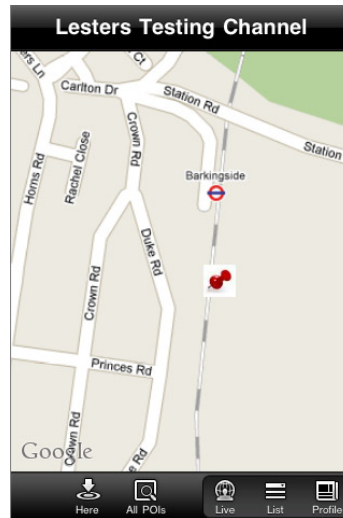


FIGURE 10-29: 3D content displayed on the map

We know it downloaded because we saw the message indicating it was being downloaded and it is shown on the map. At first you may think that there is an error with the XML or a problem with your 3D file. A useful troubleshooting tip at this point is to change the `<force3d>` tag to `false`.

```
<force3d>false</force3d>
```

This will display the default thumbnail or POI dialog box. Notice that if you click the POI, you now see a Load 3D button. Clicking this button loads the 3D model and proves that it has been referenced correctly and that it has been downloaded to the client. Figure 10-30 shows the effect of using the `<force3d>` tag.

So why didn't the 3D image appear? Well, it actually was there, but in this case, the scale needs to be adjusted before you can see it. Your own models may be different as it depends on the actual size of the model, but this example uses the junaio man model (which is around 10cm). In this example, the POI latitude and longitude is about 2km from my home. Once you take into account the scale for distance, the model can quickly become one

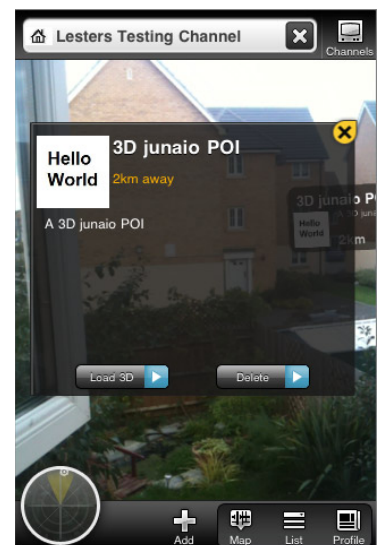


FIGURE 10-30: Changing the `<force3d>` tag to `false`

pixel in size and impossible to find. With this in mind, your 3D objects are dependent on several factors:

- ▶ The viewer's distance from the location of the latitude and longitude coordinates.
- ▶ The size of the model in the relevant OBJ or MD2 file.

To make the object visible, make the following changes:

```
<force3d>true</force3d>
<s>6000</s>
```

Of course, the actual value of `<s>` depends on how far your POI is located and the size you want it to appear in the client. When working with 3D models, it's very much a trial-and-error approach to get the perfect size. You will also notice that the junaio man graphic is horizontal rather than vertical. Figure 10-31 shows the effect of setting the scale to 6000.

The actual size of your 3D image will depend on how far the POI latitude and longitude coordinates are from your current location.

Rotating the 3D Object

You can rotate the 3D object by adjusting the orientation (`<o>`) tag. The `<o>` tag accepts a float and will adjust the x, y and z positioning of the object. (`<o>x, y, z</o>`)

To bring the junaio man upright (see Figure 10-32), use the following code:

```
<o>0, -1.57, 0</o>
```

The values given in the `<o>` tag are euler angles expressed in radians. Unlike degrees, which range from 0 to 360, radians range from 0 to 6.28. Positive values rotate right to left; negative values rotate left to right.

The size of the character is probably a little on the large side right now, but for testing purposes, leave it as it is.

By further adjusting the x parameter, you can rotate the 3D image to view from a different angle (see Figure 10-33):

```
<o>4, -1.57, 0</o>
```

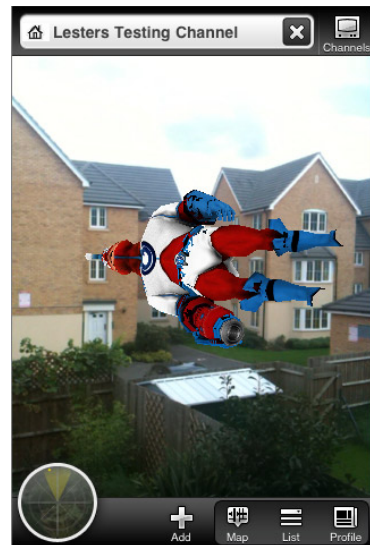


FIGURE 10-31: The junaio man in 3D



FIGURE 10-32: The junaio man upright and scaled



FIGURE 10-33: The junaio man rotated

Listing 10-10 sets the scale of the 3D model and sets the correct orientation.



Available for
download on
Wrox.com

LISTING 10-10: Scaling and orientating the 3D junaio man (search.php)

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

<!-- A unique number for our POI, and settings for user interaction-->
<poi id=\"1\" interactionfeedback=\"none\">

    <name><![CDATA[3D junaio POI]]></name>
    <description><![CDATA[A 3D junaio POI]]></description>

<!-- The lat/lon of the POI-->
<!-- Use a location near you for testing -->
    <1>51.5848,0.0886,0</1>

    <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
    <icon>http://www.yourserver.com/mappin.jpg</icon>

<!-- Changing the mime type to md2 -->
```

continues

LISTING 10-10 (continued)

```

    <mime-type>model/md2</mime-type>

    <!-- Forcing the 3D image to be shown rather than
         the standard POI bubble-->
    <force3d>true</force3d>

    <!-- Setting a scale -->
    <s>6000</s>

    <!-- Set the orientation of the 3D object-->
    <o>2,-1.57,0</o>

    <!-- Loading a 3D resource -->
    <mainresource>http://www.junaio.com/publisherDownload/tutorial
        /metaioman.md2_enc</mainresource>
    <resources>
    <resource>http://www.junaio.com/publisherDownload/tutorial
        /metaioman.png</resource>
    </resources>

    </poi>

</results>";
?>

```

Scaling 3D Content

To illustrate scale and how it affects your 3D content, Listing 10-11 adds two POIs at a distance of 4km and 8km, respectively.

**LISTING 10-11: Scaling demo (search.php)**

Available for
download on
Wrox.com

```

<?php

echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

    <!-- A unique number for our POI, and settings for user interaction-->
    <poi id=\"1\" interactionfeedback=\"none\">

        <name><![CDATA[3D junaio POI]]></name>
        <description><![CDATA[A 3D junaio POI]]></description>

    <!-- The lat/lon of the POI-->
    <!-- Use a location near you for testing -->
    <l>51.5848,0.0886,0</l>

    <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>

```



```

    <icon>http://www.yourserver.com/mappin.jpg</icon>

    <!-- Changing the mime type to md2 -->
    <mime-type>model/md2</mime-type>

    <!-- Forcing the 3D image to be shown rather than
The standard POI bubble-->
    <force3d>>true</force3d>

    <!-- Setting a scale -->
    <s>6000</s>

    <!-- Set the orintiation of the 3D object-->
    <o>0,-1.57,0</o>

    <!-- Loading a 3D resource -->
    <mainresource>http://www.junaio.com/publisherDownload/tutorial
    /metaioman.md2_enc</mainresource>
    <resources>
    <resource>http://www.junaio.com/publisherDownload/tutorial
    /metaioman.png</resource>
    </resources>

</poi>

<poi id="\2\" interactionfeedback="\none\">
    <name><![CDATA[2nd junaio man]]></name>
    <description><![CDATA[junaio man 3D demo]]></description>

    <!-- Location of the second POI-->
    <l>51.575288,0.120649,0</l>

    <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
    <icon>http://www.yourserver.com/mappin.jpg</icon>
    <mime-type>model/md2</mime-type>

    <force3d>true</force3d>
    <s>6000</s>
    <o>0,-1.57,0</o>

    <mainresource>http://www.junaio.com/publisherDownload/tutorial
    /metaioman.md2_enc</mainresource>
    <resources>
    <resource>http://www.junaio.com/publisherDownload/tutorial
    /metaioman.png</resource>
    </resources>

</poi>

<poi id="\3\" interactionfeedback="\none\">

```

continues

LISTING 10-11 (continued)

```

<name><![CDATA[3rd junaio man]]></name>
<description><![CDATA[junaio man 3D demo]]></description>

<!-- Location of the second POI-->
<l>51.566037,0.18465,0</l>

<thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
<icon>http://www.yourserver.com/mappin.jpg</icon>
<mime-type>model/md2</mime-type>

<force3d>>true</force3d>
<s>6000</s>
<o>0,-1.57,0</o>

<mainresource>http://www.junaio.com/publisherDownload/tutorial
/metaioman.md2_enc</mainresource>
  <resources>
    <resource>http://www.junaio.com/publisherDownload/tutorial
/metaioman.png</resource>
  </resources>
</poi>

</results>";
?>

```

As shown in Figure 10-34, the 3D models scale according to distance.

At some point, the 3D images become so small that the overhead of downloading them to the client becomes too great. In this case, it is more efficient simply to not download them. After all, downloading 20 3D models occupying 1 pixel each is the same as downloading 20 3D models that are visible and interactive. Fortunately, you can control the maximum distance for 3D content, so you can exclude content beyond a specified range.

To exclude content, use the `<maxdistance>` tag and provide an integer. The integer will determine the maximum distance of a POI before it will be displayed and cause the client to not load any POIs where the maximum distance is exceeded. The `<maxdistance>` tag accepts an integer which represents a meter. As you build your channel, if you are using 3D and if you have many different 3D resources that need to be loaded, you will want determine the scale and the range for content.



FIGURE 10-34: Scaling junaio man

Listing 10-12 sets the maximum distance for your 3D objects. Beyond this point, they will not be downloaded or displayed in the client.



Available for
download on
Wrox.com

LISTING 10-12: 3D content with <maxdistance> (search.php)

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

    <!-- A unique number for our POI, and settings for user interaction-->
    <poi id=\"1\" interactionfeedback=\"none\">

        <name><![CDATA[3D junaio POI]]></name>
        <description><![CDATA[A 3D junaio POI]]></description>

    <!-- The lat/lon of the POI-->
    <!-- Use a location near you for testing -->
    <l>51.5848,0.0886,0</l>

    <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
    <icon>http://www.yourserver.com/mappin.jpg</icon>

    <!-- Changing the mime type to md2 -->
    <mime-type>model/md2</mime-type>

    <!-- Forcing the 3D image to be shown rather than
         the standard POI bubble-->
    <force3d>true</force3d>

    <!-- Setting a scale -->
    <s>6000</s>

    <!-- Set the orintiation of the 3D object-->
    <o>0,-1.57,0</o>

    <!-- Set the max distance to 2000 meters. Beyond this
         dont download the POI-->
    <maxdistance>2000</maxdistance>

    <!-- Loading a 3D resource -->
    <mainresource>http://www.junaio.com/publisherDownload/tutorial/
        metaioman.md2_enc</mainresource>
    <resources>
        <resource>http://www.junaio.com/publisherDownload/tutorial/
            metaioman.png</resource>
```

continues

LISTING 10-12 *(continued)*

```

</resources>

</poi>

<poi id="2" interactionfeedback="none">
  <name><![CDATA[2nd junaio man]]></name>
  <description><![CDATA[junaio man 3D demo]]></description>

  <!-- Location of the second POI-->
  <l>51.575288,0.120649,0</l>

  <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
  <icon>http://www.yourserver.com/mappin.jpg</icon>
  <mime-type>model/md2</mime-type>

  <force3d>true</force3d>
  <s>6000</s>
  <o>0,-1.57,0</o>
  <maxdistance>2000</maxdistance>

  <mainresource>http://www.junaio.com/publisherDownload/
    tutorial/metaioman.md2_enc</mainresource>
  <resources>
    <resource>http://www.junaio.com/publisherDownload/
      tutorial/metaioman.png</resource>
  </resources>
</poi>

<poi id="3" interactionfeedback="none">
  <name><![CDATA[3rd junaio man]]></name>
  <description><![CDATA[junaio man 3D demo]]></description>

  <!-- Location of the second POI-->
  <l>51.566037,0.18465,0</l>

  <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
  <icon>http://www.yourserver.com/mappin.jpg</icon>
  <mime-type>model/md2</mime-type>

  <force3d>true</force3d>
  <s>6000</s>
  <o>0,-1.57,0</o>
  <maxdistance>2000</maxdistance>

  <mainresource>http://www.junaio.com/publisherDownload/

```

```

        tutorial/metaioman.md2_enc</mainresource>
    <resources>
        <resource>http://www.junaio.com/publisherDownload/
tutorial/metaioman.png</resource>
    </resources>
</poi>

</results>";
?>

```

In this example, only one POI will be displayed because the other two POIs are beyond the 2000-meter limit.



You can try this yourself by using Google Maps to generate the latitude and longitude coordinates of locations near you. Experiment with the `<maxdistance>` tag to find the optimum range for your 3D content.

Accuracy

Another parameter that you can use to determine what POIs are displayed is the `<minaccuracy>` tag. As you learned in Chapter 3, GPS is not entirely accurate and has an error margin based on a variety of factors. The `<minaccuracy>` tag enables you to determine the accuracy of the signal; should it fall outside your criteria, it will exclude content.

This tag enables you to set a desired accuracy before the tag is displayed. However, I'm not really sure why you would want to do this. I think it would be very confusing to users if one day the POI appeared, then the following day it is missing. Unless you have a very good reason to exclude content in this way, I would suggest that you avoid this tag with POIs. This tag is useful (and is typically used) with natural-feature tracking where it is used to set confidence thresholds on image recognition.

USING ANIMATION

MD2 files also support animation, enabling you to build content that performs an action when clicked by users. There are two states:

- The idle state (which is the default)
- The clicked state (which is an event fired when the user selects the 3D model).

Animations must be part of the MD2 file. junaio simply triggers the animation rather than containing the code to plot the animation on an action-by-action basis. The example uses the junaio man MD2, which has a 3D action. To use the action, use the following format:

```
<behaviours>
  <behaviour type="\click\">
    <length>6</length>
    <node_id>close_up</node_id>
  </behaviour>
</behaviours>
```

The tags required to invoke animation are very simple:

- If `<behaviour type>` is set to `idle`, the stated animation of the model will be started. After the model is done loading, "click" animations will be started upon clicking.
- `<length>` sets the number of frames that make up the animation. 0 indicates that the animation is to loop repeatedly.
- `<node_id>` specifies the animation you want to play. (MD2 files may contain different animations.)

In Listing 10-13, the animation length is 6 frames as specified by the `<length>` tag. The animation will be triggered when the 3D object is selected.



Available for
download on
Wrox.com

LISTING 10-13: junaio man animation 1 (search.php)

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

  <!-- A unique number for our POI, and settings for user interaction-->
  <poi id=\"1\" interactionfeedback=\"none\">

    <name><![CDATA[3D with animation POI]]></name>
    <description><![CDATA[Animation demo]]></description>

    <!-- The lat/lon of the POI-->
    <!-- Use a location near you for testing -->
    <l>51.5848,0.0886,0</l>

    <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
    <icon>http://www.yourserver.com/mappin.jpg</icon>

    <!-- Changing the mime type to md2 -->
```

```

    <mime-type>model/md2</mime-type>

<!-- Forcing the 3D image to be shown rather
      than the standard POI bubble-->
    <force3d>true</force3d>

<!-- Setting a scale -->
    <s>6000</s>

<!-- Set the orientation of the 3D object-->
    <o>0,-1.57,0</o>

<!-- Loading a 3D resource -->
    <mainresource>http://www.junaio.com/publisherDownload/tutorial
      /metaioman.md2_enc</mainresource>
    <resources>
      <resource>http://www.junaio.com/publisherDownload/tutorial
        /metaioman.png</resource>
    </resources>

    <!-- setting up the animation-->
    <behaviours>
      <!-- when the animation should start-->
      <behaviour type="\click\">
        <!-- the length of the animation -->
        <length>6</length>
        <!-- the name of the animation as defined by the md2 file-->
        <node_id>close_up</node_id>
      </behaviour>
    </behaviours>

  </poi>
</results>";
?>

```

Figure 10-35 shows the junaio man animation.

Remember when you changed the `<o>` tag to change the orientation of the character? Try adjusting the first parameter of the `<o>` a few times to see the animation from different angles:

```
<o>-11,-1.57,0</o>
```

You can automatically play the animation once it has loaded by changing the `<behaviour type>` to `idle`. As shown in Listing 10-14, you can have it repeat indefinitely by changing the length to 0.

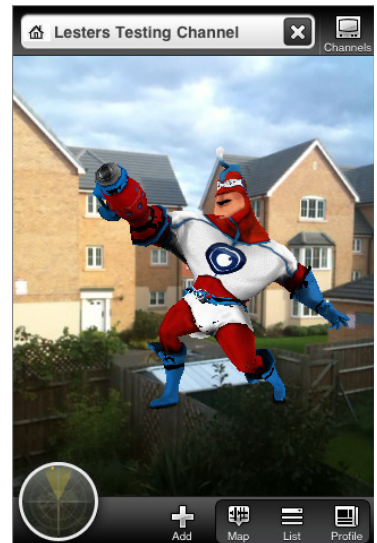


FIGURE 10-35: junaio man animation

Available for
download on
Wrox.com**LISTING 10-14: Indefinite animation on load (search.php)**

```

<?php

echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

    <!-- A unique number for our POI, and settings for user interaction-->
    <poi id=\"1\" interactionfeedback=\"none\">

        <name><![CDATA[3D with animation POI]]></name>
        <description><![CDATA[Animation demo]]></description>

    <!-- The lat/lon of the POI-->
    <!-- Use a location near you for testing -->
    <l>51.5848,0.0886,0</l>

    <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>
    <icon>http://www.yourserver.com/mappin.jpg</icon>

    <!-- Changing the mime type to md2 -->
    <mime-type>model/md2</mime-type>

    <!-- Forcing the 3D image to be shown rather than
    the standard POI bubble-->
    <force3d>>true</force3d>

    <!-- Setting a scale -->
    <s>6000</s>

    <!-- Set the orientation of the 3D object-->
    <o>0,-1.57,0</o>

    <!-- Loading a 3D resource -->
    <mainresource>http://www.junaio.com/publisherDownload/
        tutorial/metaioman.md2_enc</mainresource>
    <resources>
        <resource>http://www.junaio.com/publisherDownload/
            tutorial/metaioman.png</resource>
    </resources>

    <!-- setting up the animation-->
    <behaviours>
    <!-- when the animation should start-->
    <behaviour type=\"idle\">
    <!-- the length of the animation -->
    <length>0</length>
    <!-- the name of the animation as defined by the md2 file-->
    <node_id>close_up</node_id>

```



```

    </behaviour>
  </behaviours>

  </poi>

</results>";
?>

```

It might not be such a good idea to have your content animate once in the idle state. If you have a low number of frames for the animation, it may be over before the user finds your content in the camera window. There is no repeat tag and specifying a length of 24 with an animation of 6 frames does not cause the animation to repeat 4 times. So as you build your channel, consider if the animation should play repeatedly (so it's not annoying) or if it should play when the user clicks the 3D object. Some animation repeating could be beneficial (for example, to represent a fast-food restaurant, you may want to display a revolving burger).

USING OBJ FILES

Thus far, you have used MD2 files as the source of your 3D content. junaio also supports the OBJ format. MD2 files do not support adding multiple textures to the image. For example, you couldn't add a change of costume to a character and trigger this as your animation. OBJ files, however, support multiple textures. So they offer more flexibility but they do not support animation.

Listing 10-15 uses an OBJ file instead of an MD2 file. There is no animation with this sample.



LISTING 10-15: T-Rex demo (search.php)

Available for
download on
Wrox.com

```

<?php

echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>

  <!-- A unique number for our POI, and settings for user interaction-->
  <poi id=\"1\" interactionfeedback=\"none\">

    <name><![CDATA[T-Rex Demo]]</name>
    <description><![CDATA[A 3D T-Rex]]</description>

  <!-- The lat/lon of the POI-->
  <!-- Use a location near you for testing -->
  <1>51.5848,0.0886,0</1>

  <thumbnail>http://www.yourserver.com/logo.jpg</thumbnail>

```

continues

LISTING 10-15 (continued)

```

<icon>http://www.yourserver.com/mappin.jpg</icon>

<!-- Changing the mime type to obj -->
<mime-type>model/obj</mime-type>

<!-- Forcing the 3D image to be shown rather than
the standard POI bubble-->
<force3d>>true</force3d>

<!-- Setting a scale -->
<s>700</s>

<!-- Set the orientation of the 3D object-->
<o>0,-1.57,0</o>

<!-- Loading a 3D resource -->
<!-- Notice that OBJs are contained in a zip file!-->
<mainresource>http://www.junaio.com/publisherDownload/
junaio_model_obj.zip</mainresource>
<resources>
</resources>

</poi>

</results>";
?>

```

Figure 10-36 shows the results of the code.



FIGURE 10-36: The T-rex demo

CREATING 3D CONTENT

You will want to use your own 3D content with junaio, and that's easy enough to do. Make sure the file sizes are small for your 3D content and encrypt the files. As you have seen, junaio supports both MD2 and OBJ files and there are many tools on the market that will enable you to create content.

Blender (www.blender.org) is a particularly good tool for creating 3D content; it's also free and open source, so it's widely supported by the community. If you use Blender to create your 3D content, please refer to the tutorial from metaio that will help you export your models in the right format:

www.metaio.com/fileadmin/downloads/Unifeye_Demos/Unifeye_mobile_SDK/UnifeyeSDKMobile_ContentCreation_Tutorial.pdf

If you want to test junaio with other 3D models and, like me, you are not very artistic, there are some fantastic free models and image maps you can download. MD2 and OBJ are popular formats with game developers, so you will find plenty of resources.

To you get started, take a look at www.md2.sitters-electronics.nl, which has some excellent resources and a low polygon count.

Importing MD2 and OBJ Files

In this final section, you will learn how you can use your own MD2 or OBJ files with junaio. Before a file can be used, it must be encrypted.

In the following steps, you will download a 3D model from a third-party web site, encode it into the correct format for junaio, and then use it in your junaio Channel.

1. Visit www.md2.sitters-electronics.nl.
2. Click the Models link on the menu at the top of the page. A list of model packs available for download displays.
3. Choose Model pack 3.
4. Download the models.

The file will be compressed in .rar format, so you will need a .rar extractor. If you don't have one, try 7-zip, which can be downloaded free from www.7-zip.org/.

5. Extract the files to a location on your PC. (Notice that the file contains examples of both MD2 and OBJ models.)
6. Navigate to the car subfolder.
7. Visit www.junaio.com/publisher/placedmodels

The following steps differ, depending on whether you are encoding MD2 files or encoding OBJ files.

Encoding MD2 Files

To complete the encoding for the MD2 file, follow these steps.

1. On the junaio page you opened in Step 7, move to the bottom of the page and locate the Browse button and Encrypt button (see Figure 10-37).

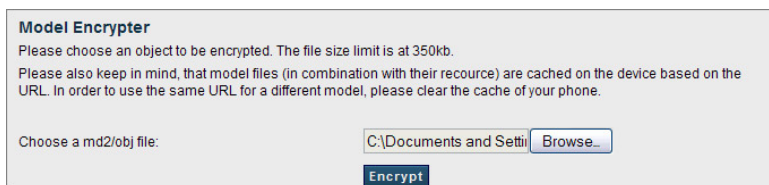


FIGURE 10-37: Encoding the file

2. Click Browse and navigate to the car.md2 file that you extracted. Once you have found it, click the Encrypt button.
3. The file will now be encrypted and you will be prompted to download the encrypted file. Do this and save it somewhere where you will be able to find it. (Notice the file will have _enc appended to the file name.)
4. You will now need to upload both the car.md2_enc file you just downloaded and the car.png file that was included in the .rar file you extracted to your web server.

In your search.php file, you can now reference the car.md2 file and the image map file using the same syntax that you used for the junaio man demo.

```
<mainresource>http://www.yourserver.com/car.md2_enc</mainresource>
<resources>
  <resource> http://www.yourserver.com/car.png</resource>
```

As discussed in the scale and orientation section, you may need to adjust the <o> and <s> tags accordingly.

Encoding OBJ Files

OBJ files need to be encoded and zipped along with the relevant image map.

1. On the junaio page you opened in Step 7, move to the bottom of the page and locate the Browse button and Encrypt button (refer to Figure 10-38).
2. Click Browse and navigate to the car.obj file that you extracted. Once you have found it, click the Encrypt button.
3. The file will now be encrypted and you will be prompted to download the encrypted file. Do this and save it somewhere where you will be able to find it. (Notice the file will have _enc appended to the file name.)
4. When using an OBJ file, it must be named **model.obj_enc**. So rename the car.obj_enc file to model.obj_enc.
5. Before you can upload the file to your server, you must create a zip called car.zip and add both the model.obj_enc file and the car.png file (which was part of the .rar package you download).
6. Upload the car.zip file to your server.

In your search.php file, you can now reference the OBJ map file using the same syntax that you used for the T-Rex demo:

```
<mainresource>http://www.yourserver.com/car.zip</mainresource>
```

Testing 3D Models

3D models can become quite large and should always be tested on a 3G network to verify usability. metaio advises that a model with around 500 vertices, 1000 polygons, and a total frame number

of 200 (with all animations) will consume about 480 kb of storage and an additional 750 kb of system memory. Don't assume your users will be on super-fast Wi-Fi connections when using your application. Test the experience under all possible scenarios.



If you have time, try out other 3D models and add them to junaio.

SUMMARY

In this chapter, you built your first Channel for junaio and you should now have a fully working junaio server that is able to serve real content to junaio users. Although you were restricted to building POIs, you made the content richer by adding support for images, videos, and 3D. As part of this chapter, you also learned that scale becomes an issue with 3D. If the object is too far from the user's location, the 3D image becomes too small to be seen. You also learned how to manipulate the 3D image with rotation and how to trigger animation.

11

Natural-Feature Tracking and Visual Search with junaio

WHAT'S IN THIS CHAPTER?

- ▶ How to use natural-feature tracking and visual search if you're a non-developer
- ▶ How to use natural-feature tracking and visual search if you're a developer
- ▶ How to overlay videos
- ▶ Image requirements for natural-feature tracking

Natural-feature tracking is a very powerful tool that enables you to recognize almost any existing product (for example, books, DVDs, games, or even photos) without having to apply any special markers to the image. In contrast, a marker or a QR code would need to be applied to the packaging when the book cover is created, so you can't apply AR after the fact.

Why is this useful? If you want to create a solution that enables users to play tracks from the latest chart-topping album, the record label isn't likely to allow you to place a special marker or QR code on the CD sleeve. The marketing would have to be prepared in advance and could add some cost to the printing and design of the cover.

With natural-feature tracking, however, users can use their phones to recognize the CD cover even if the CD is already selling in the shops. Once recognized, a music video of one of the tracks could play on the users' smartphones without requiring them to search or type anything to find it. If they like what they hear and see, they might be more willing to purchase the album.

Because the information is stored on a server, you could update the recognition action after a couple of weeks to play an exclusive interview with the band — effectively ensuring the information is fresh to the consumer. Of course, it's not all about sales and marketing. You may just want to send family members a picture of your newborn, and when the picture is recognized, your family members are taken to a Flickr page where they can see more pictures.

In this chapter, you will continue experimenting with junaio, which is currently the only AR browser that provides developers and non-developers with the opportunity to build natural-feature tracking solutions. In junaio, this functionality is called *GLUE* because you are effectively *gluing* content onto an image.

Before you get started with this chapter, make sure you have read Chapter 10 and that you have a junaio developer account. Additionally, a publicly visible web server is required for the PHP development section.

NATURAL-FEATURE TRACKING FOR NON-DEVELOPERS

Until recently, the only way to create a natural-feature tracking channel for junaio was by building the channel using PHP. metaio recently provided the functionality to enable even non-developers to take advantage of the technology and build GLUE channels without writing a single line of code. Thanks to this functionality, GLUE channels can be created using a very simple UI.

Using the non-developer UI, you can create a natural-feature tracking channel that plays video when an image is recognized or overlays a 3D object very easily. In the junaio client, search for Professional AR Apps and load the channel. Once it's loaded, point your device at each image shown in Figure 11-1. When the image is recognized, a short-video will play. As you'll see in the various videos, my cats are not that energetic — but the entire channel took less than five minutes to create.



FIGURE 11-1: Point your device at each image to watch a video

It's worth pointing out that the entire picture is being matched, not the actual cat in the picture. My channel is not using a fancy new cat-recognition engine; if the cat is shown at any different angle, a video would not play because the image would not match the image on the server.

Creating Your First GLUE Channel

In this section, you will recreate your own channel based on the demo from the previous section; you can download and use my resources to create the demo. To begin, download the following:



- The reference images to use for recognition.
- The MP4 videos which will play when the images are recognized.
- The 3D object used for the natural-feature tracking component.

Because this section is for non-developers, you don't need access to a dedicated web server; all the resources will be uploaded to the junaio server and hosted there. To create your first GLUE channel:

1. Navigate to the following <http://creator.junaio.com> and log-in. (You may need to login using the account details you already created in the previous chapter.)

Once you have logged in to the site, you are presented with a list of all your GLUE channels. You'll notice that the list shown in Figure 11-2 is empty; it doesn't include the channels you created in the previous chapter. Don't panic — nothing has been lost. This is a much simpler interface that hides much of the details required by the PHP developer site — details that aren't required here.

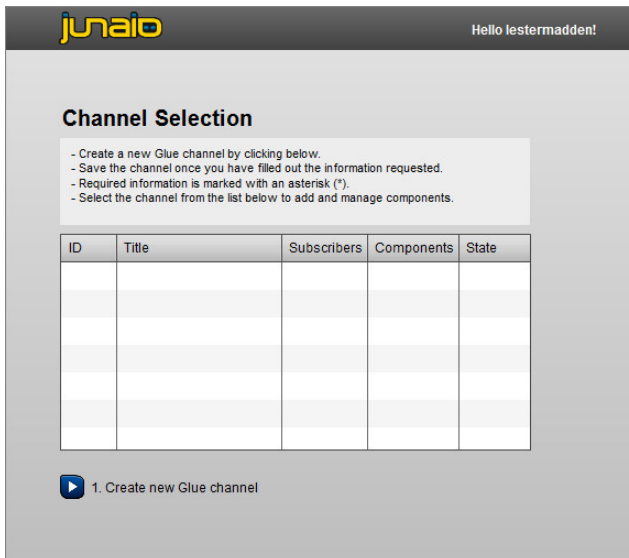


FIGURE 11-2: Creating a new GLUE channel

2. Below the channel list, click the Create new Glue channel link.

This displays a short Channel Configuration form (see Figure 11-3) for you to complete. The details entered here are used for the description of your channel as it appears in the junaio client.

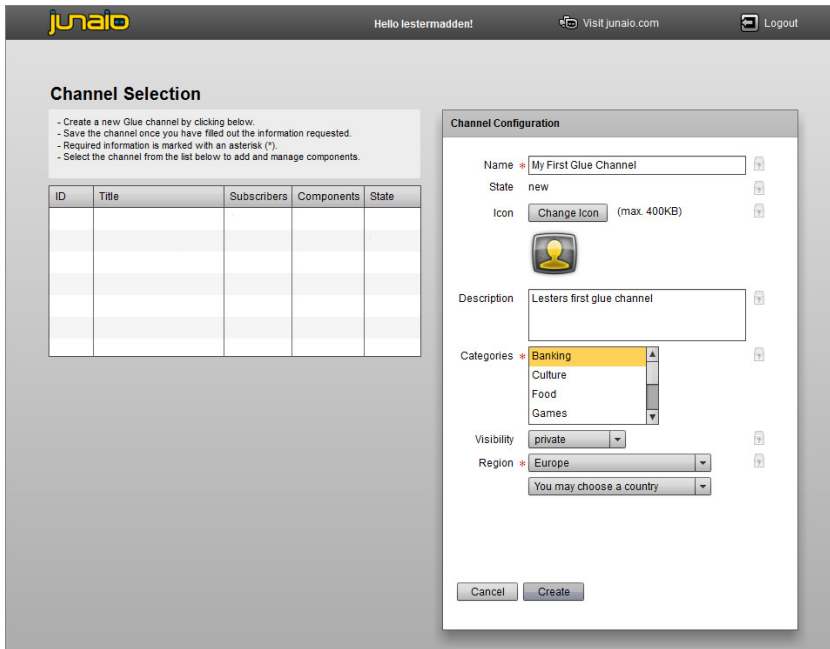


FIGURE 11-3: Configuring your channel

I recommend that you set the Visibility to private, which ensures that only you or those you invite can access your content. This is the ideal setting for testing or sharing content with friends and family. The Categories listing is largely irrelevant for this demo because there isn't a matching category. Since the channel is private and there isn't a category that suits your content, just select any category from the Categories list for now. You can always update the category later.

When you create the channel, you must specify the region where the channel will be visible. This enables you to restrict your audience to those who are only within that particular region. This might not seem beneficial, but it you might indeed have channels which are specific to a regional audience. For example, some branded products might be available only in a particular country. During your test, you should choose your home region.

3. Once you have completed the configuration form, click Create.

You're returned to the Channel Selection list shown in Figure 11-2, only the channel you just created will be shown in the list.

The next step is to add the image that you want to be recognized. This image or pattern is called the *tracking image* or *reference image*.

- Simply click your channel in the list and the screen shown in Figure 11-4 displays.

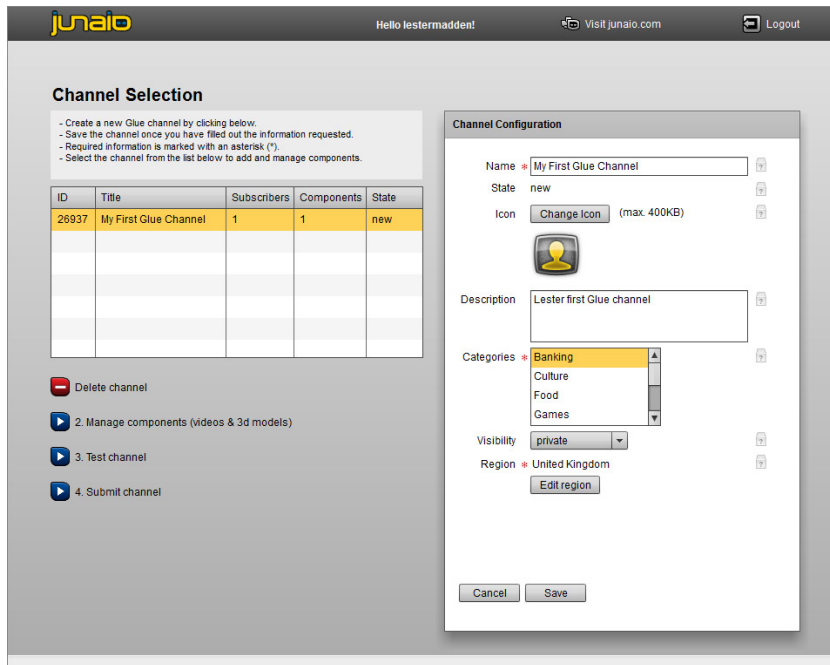


FIGURE 11-4: Managing components

This is essentially the same screen as that shown in Figure 11-3 only it includes a few other options below the Channel Selection list.

- Select the Manage components (videos & 3D models) option. The screen shown in Figure 11-5 displays.

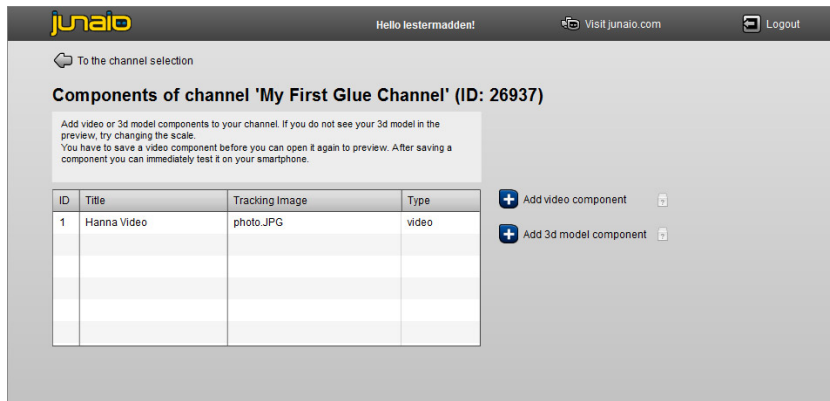


FIGURE 11-5: Adding the entry

On this screen, you will specify whether your channel will use a video object or a 3D object. In Chapter 1 and Chapter 2, I described the differences between AR and visual search. Your solution here is considered a visual search channel because you will recognize an image and then launch a video.

6. Select the Add video component option and the New video component screen displays (see Figure 11-6).

The screenshot shows the Junaio web interface. At the top, there is a navigation bar with the Junaio logo, the user name 'Hello lestermadden!', a link to 'Visit junaio.com', and a 'Logout' button. Below the navigation bar, there is a breadcrumb 'To the channel selection' and a heading 'Components of channel 'My First Glue Channel' (ID: 72972)'. A message box states: 'Add video or 3d model components to your channel. If you do not see your 3d model in the preview, try changing the scale. You have to save a video component before you can open it again to preview. After saving a component you can immediately test it on your smartphone.' Below this is a table with columns 'ID', 'Title', 'Tracking Image', and 'Type'. The table is currently empty. Below the table is a 'New video component' form with the following fields: 'Title *' (text input), 'Author' (text input), 'Video *' (with an 'Upload Video' button, a '(max. 10MB)' limit, and a note: 'Find out how to properly encode your video on junaio.com'), and 'Tracking Image *' (with an 'Upload Image' button and a '(max. 500KB)' limit). At the bottom of the form are 'Cancel' and 'Create' buttons.

FIGURE 11-6: Completing the Video component information

Title

The channel will use a collection of images. In the demo, you learned the channel recognized four different pictures and subsequently took four different actions. In the Title field, give each image its own unique descriptive name.

Author

Author is an optional field should you can use to provide attribution to the creator of the content or as a personal reminder of who created the component. It is not displayed publicly but could be used if you have several people on your team creating content.

Video

The Upload Video button is used to select the video that will play in the device's video player when the image is recognized. The video needs to be an MP4 format — which ensures that the video plays across a variety of devices — and smaller than 10MB.

Tracking Image

The Tracking Image button is used to select the image that you want junaio to recognize. Later in this chapter, you will learn how to select a good tracking image.

Adding Images and Video

Now it's time to create four video components using the image and video for each cat as shown in Table 11-1.

TABLE 11-1: Demo Resources

TRACKING IMAGE	VIDEO
hanna.jpg	hanna.mp4
bill.jpg	bill.mp4
ray.jpg	ray.mp4
bobby.jpg	bobby.mp4

Once you have created video components for all four images, the Video component screen should resemble Figure 11-7. Note that the order in which your videos are listed might differ from mine. No worries — what's important is that the right video is associated with the correct image.

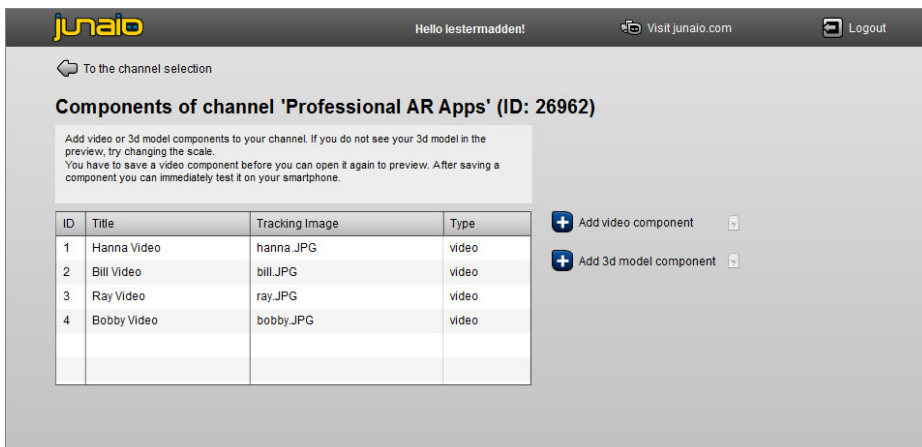


FIGURE 11-7: Completed components for the demo

Testing the Channel

Believe it or not, you have just finished your first natural-feature tracking channel! Now you need only to test the channel in the junaio client. To test it, simply load junaio and select the Favorite icon. Your new channel will display in the list. (Note that you will need to be logged in to your junaio account in order for your channel to appear.) Once the channel loads, simply point the camera at the images shown in Figure 11-1.

Experimenting with Natural-Feature Tracking

I'm sure you'll be keen to test natural-feature tracking on other images, particularly when you use your own videos. If you don't have any MP4 videos to use, use your mobile phone to capture video and use an online converter (such as www.online-convert.com) to convert the video to MP4. This site allows you to convert via a web interface, so there is no software to download.

Gluing 3D Objects to an Image

In this section you will extend your sample channel to support gluing a 3D object to an image. Revisit my Professional AR apps channel and point your camera at the image shown in Figure 11-8.

The pirate flag is very simple background and shouldn't be confused with a marker. I felt the skull and crossbones were a good match for the 3D model of a skeleton. Of course, the tracking images can be more complex. You could overlay the model over the movie poster for *Pirates of the Caribbean*, but you should always get permission before you use copyrighted images. Figure 11-9 shows the 3D object rendered on the pirate flag.



FIGURE 11-8: Pirate flag



FIGURE 11-9: A rendered skeleton

Including support for 3D objects in your channel is almost as simple as adding support for video:

1. In your Channel Selection list (refer to Figure 11-5), click your channel.
2. In the Components screen that displays, select the Add 3d model component option.

The 3D model form is a little more complex than the video form (see Figure 11-10).

The screenshot shows the Junaio dashboard interface. At the top, there's a navigation bar with the Junaio logo, a user greeting 'Hello lestermadden!', a 'Visit junaio.com' link, and a 'Logout' button. Below the navigation bar, there's a breadcrumb 'To the channel selection' and a heading 'Components of channel 'My First Glue Channel' (ID: 72972)'. A small informational box states: 'Add video or 3d model components to your channel. If you do not see your 3d model in the preview, try changing the scale. You have to save a video component before you can open it again to preview. After saving a component you can immediately test it on your smartphone.' Below this is a table with columns 'ID', 'Title', 'Tracking Image', and 'Type'. The table is currently empty. Below the table is a 'New model component' form. The form contains the following fields: 'Title *' (text input), 'Author' (text input), 'MD2 Model *' with an 'Upload Model' button (max. 350KB), 'Model Texture *' with an 'Upload Texture' button (max. 500KB), and 'Tracking Image *' with an 'Upload Image' button (max. 500KB). There is a checked checkbox 'Display extra information when the model is clicked'. Below this are four more text input fields: 'Description', 'Homepage', 'Phone', and 'E-Mail'. At the bottom of the form are 'Cancel' and 'Create' buttons.

FIGURE 11-10: Completing the 3-D model information

Title

Each component you add to your channel will be identified by its own name. The name is used in the junaio Dashboard to help you identify the component in the future.

Author

You can provide an optional author for the component.

MD2 Model

When creating GLUE channels via the web interface, you are restricted to using MD2 models — a popular file format for creating 3D objects. MD2 files can be a maximum size of 350KB.

Model Texture

MD2 models represent the skeleton of the object; the model texture is the skin that is applied to the model. The model texture should be a PNG file up to 500KB in size.

Tracking Image

The tracking image is the image that you want to be recognized by junaio. As you use the junaio application, junaio will constantly check the images from the camera window for matches to the tracking image. If there's a match, the relevant GLUE component will be triggered.

Description

The Description field is displayed when you've selected the Display extra information when the model is clicked option. Selecting this option enables you to include POI-related information with the 3D object. When users click the 3D object, a window is displayed on the device with the information you included. If you don't enable the option, the 3D object doesn't do anything when clicked — essentially, you have a cool-looking 3D object that has no functionality. The Description is a text field that will be displayed when the user clicks the 3D model. Homepage

If a URL has been entered, users are presented with a button when the 3D object has been clicked. The button provides users with the option to visit the web site. This is an optional parameter.

Phone

If a phone number has been entered, users are presented with a button when the 3D model has been clicked. The button enables users to place calls directly from the junaio client. It is good practice to include the full country and area code for any phone numbers so the number can be reached from any location. This is an optional parameter.

E-Mail

If an e-mail address has been entered, users are presented with a button when the 3D model has been clicked. The button loads the phone's default e-mail client with the e-mail address you've entered. This is an optional parameter.

Complete the form using the information shown in Table 11-2 and then click the Create button.

TABLE 11-2

PARAMETER	RESOURCE
MD2 Model	skeleton.md2
Model Texture	skeleton.png
Tracking Image	mrbones.jpg

If you'd like, enter the optional parameters (such as a Description, Homepage, Phone, or E-Mail). After you have clicked the Create button, you will be presented with additional options specific to the size and placement of the 3D object. These options are shown in Figure 11-11.

FIGURE 11-11: Scaling and positioning

Rotation/Orientation in Degrees

When you add the 3D model to the GLUE channel, you can control various aspects of its positioning. The rotation and orientation option is a series of X, Y, and Z numerical values expressed in degrees that adjust how the model is displayed. By changing these values, you can view the model from the back, front, or side. You can even turn it upside down.

Scale

The scale allows you to specify the size of the 3D model. When you first add a 3D model, it might be invisible until you increase the size.

Position on the Tracking Image

This setting enables you to position where the 3D image appears on the tracking image. In most circumstances, you will position the model in the center but you are free to align the model in the best position. For example, if you are positioning a speech bubble, you would align the image with the speakers' mouth.

Idle-Animation

If the 3D object has animations, you can enable them to play when the 3D object is idle (for example, when the user is not interacting with it). Animation needs to be part of the MD2 file, so it will be designed by the 3D model creator. If the model has actions, those actions will be shown in the drop-down list.

OnClick-Animation

OnClick animations are triggered when the user clicks the object. Animations are required to be part of the MD2 file, so they will be designed by the original model designer. If the model supports animations, the available animations will be shown in the drop-down list.

Complete the form using the parameters shown in Table 11-3.

TABLE 11-3

PARAMETER	SETTING
Rotation/Orientation in degrees	
X	-25
Y	180
Z	0
Scale	2.7
Position on the tracking image	
X	-4
Y	-54
Z	0
Idle-Animation	none
OnClick-Animation	none

When you have finished positioning and scaling the 3D model, click the Save button. You can now view the channel on your smartphone by pointing at the reference image shown in Figure 11-8. By moving your smartphone around the reference image you can view the skeleton from different angles, giving you a true 3D experience. Figure 11-12 shows the click actions from the Display extra information when the model is clicked parameters.

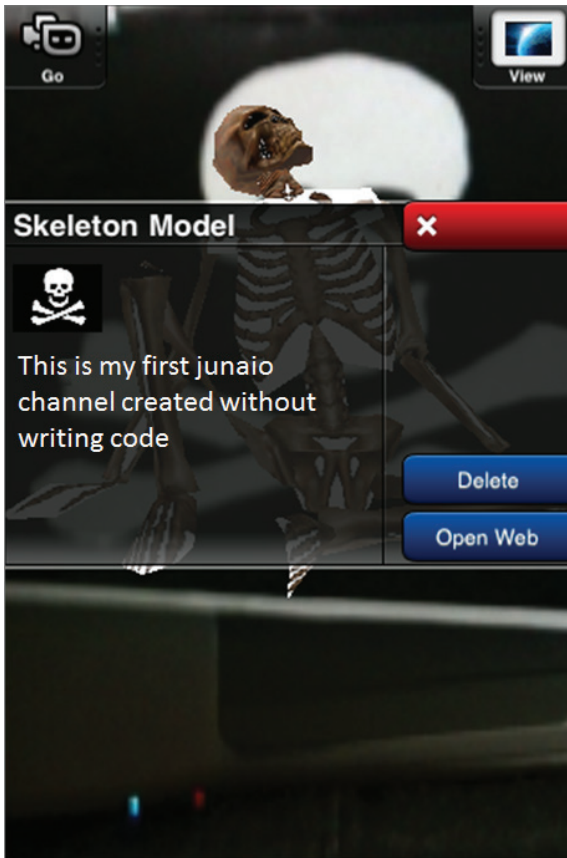


FIGURE 11-12: The 3D model with click action



If your channel doesn't work as expected, clear the cache.

NATURAL-FEATURE TRACKING FOR DEVELOPERS

As a developer, you might have been tempted to skip the previous section in favor of getting straight into the code. I suggest, however, that if you skipped the non-developer section, you should at least walk yourself through the creating of the sample. This is helpful because when it comes to building natural-feature tracking applications with code, you'll need to create the scale, position the model on the tracking image, arrange the orientation of the 3D object, and so on. Using the non-developer environment will enable you to experiment with position and see the results of the various values without a lot of trial and error with setting code values. It's the closest thing you have to a visual IDE.

Creating a Channel

In the previous chapter, you learned how to set up a junaio channel for POIs. Setting up a GLUE channel is just as easy. The first step to preserving your location-based channel is to create a new channel and a new PHP file.

1. From the junaio dashboard, select the New Channel option and create the channel as shown in Figure 11-13.

The screenshot shows the 'Create Channel' form in the junaio dashboard. The form is titled 'GLUE Test Channel' and is for an information channel. The channel ID is 29325 and it has 1 subscriber. The form fields include:

- State of the channel:** new (non-public)
- Channel Type:** Location Based Channel (selected), junaio GLUE Channel
- Channel Name:** GLUE Test Channel
- Channel description:** My GLUE test channel
- Channel Short Name:** GLUE Test Channel
- Thumbnail:** (Browse... button)
- Homepage URL:** http://www.augmentedplanet.com
- Callback URL:** http://www.augmentedplanet.com/wp-core
- Channel Categories (multiple selection possible):** Banking, Culture, Food, Games, News
- Channel Visibility:** private
- Channel refresh time (in seconds):** 0
- Support Visual Search:** (unchecked)
- Filter Options:** Search Filter Mask (unchecked), Filter URL, Show on Start (unchecked), Show on Demand (unchecked)
- Region (mandatory and no "World" for GLUE Channels only):** Europe

At the bottom of the form are 'Cancel' and 'Save' buttons. The footer of the dashboard includes '© metaio inc. 2010' and 'Developer | Press | FAQs | Contact | Terms and Privacy | Imprint'.

FIGURE 11-13: Creating a GLUE channel

Be sure to select a Channel Type of junaio GLUE Channel. In the next step, you will create a Callback file called `gluechannel.php` (as you did in the previous chapter). In the Callback URL field, specify the path to the `gluechannel.php` file. For the sake of simplicity, the URL should be the same folder as the `index.php` file created in the previous chapter. For the time being, leave the Support Visual Search option unchecked.

As you create a GLUE channel, you will need to specify the region in which it is visible. For testing purposes, you should select the part of the world where you are located. This option prevents GLUE channels showing up in the client for users outside your intended region. If you want to have global channels, you can create one channel for each region or contact junaio and ask them to make the channel available worldwide.

After you complete your settings, click Save. Now you need to create the `gluechannel.php` file and add the XML/PHP that will link to the 3D resources on the server. In Chapter 3, you tested the junaio man demo, so a good starting point will be to recreate the sample here. Create the

gluechannel.php file and add the code shown in Listing 11-1. There is no need to download the 3D resources and tracking image for this listing; they can be accessed directly from the junaio server.

Once you have added the code, upload the file to the Callback URL location you specified when creating the channel.



LISTING 11-1: gluechannel.php

Available for
download on
Wrox.com

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results trackingurl=\"http://
www.junaio.com/publisherDownload/tutorial/tracking_tutorial.xml_enc\">
  <poi id=\"1\" interactionfeedback=\"none\">
    <name><![CDATA[metaio Man]]></name>
    <author><![CDATA[metaio]]></author>
    <translation>0.0,0.0,0.0</translation>
    <o>-1.0,-3.5,5.0</o>
    <mime-type>model/md2</mime-type>

  <mainresource>
    <![CDATA[http://www.junaio.com/publisherDownload/tutorial
/metaiomman.md2_enc]]>
  </mainresource>
    <thumbnail>http://www.junaio.com/publisherDownload/tutorial
/icon.jpg</thumbnail>
  <icon>http://www.junaio.com/publisherDownload/tutorial/icon.jpg</icon>
  <route>false</route>
  <force3d>true</force3d>
  <s>1</s>
  <behaviours>
    <behaviour type=\"click\"><length>6</length>
      <node_id>close_up</node_id></behaviour>
    <behaviour type=\"idle\"><length>0</length>
      <node_id>idle</node_id></behaviour>
  </behaviours>
    <customizations/>
  <resources>
    <resource>http://www.junaio.com/publisherDownload/
tutorial/metaiomman.png</resource></resources>
  </poi>
</results>";

?>
```

Once you have uploaded the file to your server, locate the channel in the junaio client. Figure 11-14 shows the tracking image used for the demo.

You'll probably notice that the code is very similar to the 3D sample from the previous chapter but with a few new tags.



FIGURE 11-14: The tracking image

The first new tag you have added is the location of the tracking URL. This is the image map of the tracking image you want to recognize:

```
<results trackingurl=\"http://
  www.junaio.com/publisherDownload/tutorial/tracking_tutorial.xml_enc\">
  <poi id=\"1\" interactionfeedback=\"none\">
```

Unlike the non-developer UI, you need to encode your tracking image. But this is as simple as visiting a web site, uploading the image, and then downloading the image map once the junaio server works its magic.

The second new tag is the translation tag, which enables you to specify where the 3D model will be placed on the image:

```
<translation>0.0,0.0,0.0</translation>
```

By default, the image is drawn in the center of the tracking image. Using X and Y coordinates, you can adjust its position. The Z coordinate controls the rotation.

Building a Channel from Scratch

In this example, you will build — from scratch — a natural-feature tracking channel similar to the junaio man. The sample will require you encode the tracking image and encrypt the 3D image and then upload the results to your server. You will use a public domain image from www.publicdomainpictures.net (shown in Figure 11-15) and a 3D model from www.md2.sitters-electronics.nl.

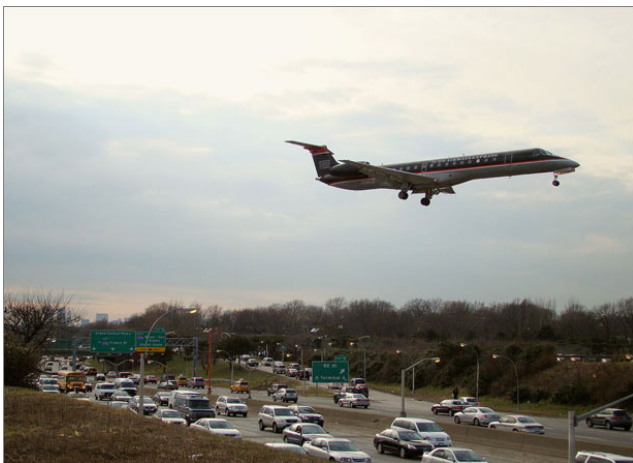


FIGURE 11-15: A public domain image

1. Encode the tracking image by uploading the picture to the online XML creator shown in Figure 11-16. This is located at the bottom of the page located at www.junaio.com/publisher/junaio glue.

6. Online tracking XML creator

You can choose any image file (jpg, png, gif) to be used as a pattern. Simply choose your image file and click "Create Tracking XML". Store the returned file and upload it to your server.

The order of files is important and must be remembered. This is the order the cosID must be assigned to your POIs. Please see a basic xml structure below after choosing the amount of images.

Please also keep in mind, that tracking xmls are cached on the device based on the URL. In order to use the same URL with changed information, please clear the cache of your phone.

For which channel do you want to create the pattern? 29325 - GLUE Test Channel

How many patterns do you want to generate? 1

Please choose image 1: \models\flightimage.jpg

In order to assign the right POI to the right pattern, please include the attribute `cosid` to your search request return:

```
<?xml version="1.0" encoding="UTF-8"?>
<results trackingurl="ABSOLUTE_PATH_TO_TRACKING_XML">
  <poi id="image1" cosid="1" interactionfeedback="none">
    ...
  </poi>
</results>
```

FIGURE 11-16: An online XML creator tool

The tool is fairly self-explanatory.

2. Select the channel you want to apply the tracking image to, select how many images you want upload, and then upload the tracking image(s) you want to encode.
3. Click the Create Tracking XML button and download the file.

The file you have downloaded represents the tracking information of the picture you just uploaded. It is unique to this image and contains a map of all the light and dark areas of the image.

4. Upload the file to your server.
5. You will need to encrypt the 3D model, which is done by uploading the 3D model to the junaio web site and having it perform the encryption for you. Visit www.junaio.com/publisher/placedmodels and at the bottom of the page, upload the file.

Notice that there is a size limit of 350kb. The sample model you will use is well within that limit.

6. Clicking the Encrypt button prompts you to download the encrypted file. Download the file and note where it is located.
7. As you have learned, 3D models have a separate image file that contains the model texture, upload both the encrypted model you downloaded in Step 6 and model texture to your sever.

Your server should now have the following files:

- tracking.xml_enc
- figher.md2_enc
- fighter.png

8. Replace the XML in the gluechannel.php with the Listing 11-2 code to finish the sample.



LISTING 11-2: gluechannel.php

Available for
download on
Wrox.com

```
<?php
    echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>

    <results trackingurl=\" LOCATION TO THE TRACKING.XML_ENC FILE
    ON YOUR SERVER \">
        <poi id=\"1\" cosid=\"1\" interactionfeedback=\"none\">
            <name><![CDATA[fighter]]></name>
            <translation>0.0,0.0,0.0</translation>
            <o>0.0,0.0,0.0</o>
            <mime-type>model/md2</mime-type>
            <mainresource>
                <![CDATA[LOCATION TO THE FIGHTER.MD2_ENC FILE
                ON YOUR SERVER]]></mainresource>

            <route>>false</route>
            <force3d>>true</force3d>
            <s>1</s>
            <behaviours>

        </behaviours>
        <customizations/>
        <resources>
            <resource>LOCATION TO THE FIGHTER.PNG FILE
            ON YOUR SERVER </resource>
        </resources>
    </poi>
</results>";

?>
```

As part of the `<poi>` tag, you added the `cosid=\"1\"` parameter, which is used identify the tracking image if you have uploaded several images to your channel.

9. Save the file and upload it to your server.
10. Load the channel on your mobile device and view the image shown in Figure 11-17. You should see the 3D fighter appear as shown in figure.

You can adjust the `<s>` tag to change the size and the `<o>` parameters to change the plane's orientation. Of course, channels can contain more than one tracking image. If you want to include more, just create a new `<results trackingurl>` node after the `</results>` tag. Remember, the `<poi id>` value has to be unique.

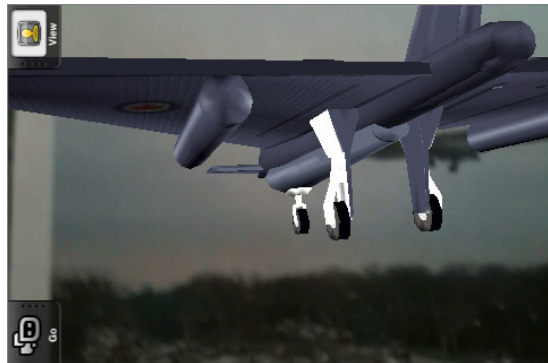


FIGURE 11-17: Fighter model on the tracking image

You also need to include a `cosid` that represents the number of the tracking image you want to use. The `cosid` was assigned when you uploaded the image in the steps shown in Figure 11-16. If you uploaded more than one image, the first image has the value 1, the second has the value 2, and so on. Only one `tracking.xml_enc` file is generated, so the `cosid` represents the index of the image in the file:

```
<results trackingurl=\".../tracking.xml_enc\">
  <poi id=\"1\" cosid=\"1\" interactionfeedback=\"none\">
    ...
  </poi>

  <results trackingurl=\".../2ndtracking.xml_enc\">
    <poi id=\"2\" cosid=\"2\" interactionfeedback=\"none\">
      ...
    </poi>
  </results>
```

USING VISUAL SEARCH

When an image is recognized, you can choose to link to a web page for more information instead of loading 3D or playing video. In chapter 2, you might have tested this concept with the Snpattell application that recognized book covers and linked to the relevant Amazon web page where consumers could purchase the item. As you might already know, Amazon has an affiliate program whereby it pays a 10 percent commission on sales by customers who are redirected by your URL or application.

Other web sites, such as Commission Junction (<http://cj.com>), have thousands of potential companies which will pay a percentage of any sales you generate. Combining visual search with an affiliate program creates an opportunity for you to make money from AR. All you need is the product image and an affiliate account to where you will send the traffic. These types of solutions are becoming increasingly popular with many supermarkets now utilizing the technology to enable consumers to shop for products like wine just by taking a picture of the label.

To prepare your channel for visual search, you need to encode two images: one you will use to play video when detected, the other you will use to link to a web site.

You will use an image of a cat shown in Figure 11-1. This image will demonstrate calling the associated video for that particular cat. The other image is of the Augmented Planet logo shown in Figure 11-18. This image will demonstrate how to load a web site when the image is recognized.



FIGURE 11-18: The Augmented Planet logo

1. As you did in the previous section, navigate to www.junaiio.com/publisher/junaioglue and from the online tracking XML creator, select your channel from the list.
2. This time, however, select the option to create two tracking patterns (because your channel will support multiple images).

Pattern one should be the image of the cat named Bill (Bill.jpg). Pattern two should be `aplogo.jpg`. Make sure you add them in this order because this is how they will be stored in the `tracking.xml_enc` file and how they will be called in the sample code.

3. Once you have created and downloaded the file, upload it to your server along with the `Bill.mp4` file that will be played.

Listing 11-3 carries out the visual search actions.



LISTING 11-3: gluechannel.php

Available for
download on
Wrox.com

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results trackingurl=\"<URL TO THE TRACKING IMAGE>\">
  <poi id=\"1\" cosid=\"1\" interactionfeedback=\"none\">

    <o>0.0,0.0,0.0</o>
    <l>0,0,0</l>
    <translation>0.0,0.0,0.0</translation>

    <mime-type>video/mp4</mime-type>
    <route>>false</route>
    <customizations>
      <customization>
        <type>video</type>
        <node_id>onTargetDetect</node_id>
        <value>
          <![CDATA[<URL TO THE MP4 FILE]>]
        </value>
      </customization>
    </customizations>
  </poi>

  <poi id=\"2\" cosid=\"2\" interactionfeedback=\"none\">

    <o>0.0,0.0,0.0</o>
    <l>0,0,0</l>
    <translation>0.0,0.0,0.0</translation>

    <mime-type>text/plain</mime-type>
    <route>>false</route>
    <customizations>
      <customization>
        <type>url</type>
        <node_id>onTargetDetect</node_id>
        <value>
```

```

        <![CDATA[www.augmentedplanet.com]]>
        </value>
    </customization>
</customizations>
</poi>
</results>";

?>

```

You might have been surprised that Listing 11-3 included a requirement for orientation and latitude/longitude. If you don't include them, the XML will not parse and will generate errors. Remember, you can always check to ensure your XML is valid by selecting the validate option for your channel. This will perform a series of checks, including testing the XML for completeness.

If you're job hunting and you want to stand out from the crowd, why not include a visual search component for your CV? Most recruiters have no idea just how easy natural-feature tracking is, so the technology is a good way to make your CV stand out from the crowd. You could include a very simple link to your LinkedIn profile when the recruiter views your CV. Or if you have the money (or expertise), you could play a 3D version of you talking about your skills and experience.



David Wood, a former colleague of mine from Symbian, demonstrates this to good effect in his AR CV at <http://tinyurl.com/davidwoodcv>.

Of course, you could use a QR Code or a Microsoft tag, but I'm sure you will agree that natural-feature tracking has more of a wow factor. In the next sample, I have taken a rather unflattering photo of myself (see Figure 11-19) and used this as my tracking image. In Figure 11-20, I have added the photo to an abstract of an old CV along with the junaio logo, which is designed to promote the fact that the image is AR-enabled. When recruiters view the CV image with the junaio channel, they will be taken to my LinkedIn profile. You could, of course, link to a video that shows you discussing your skills and sharing a portfolio of your work.

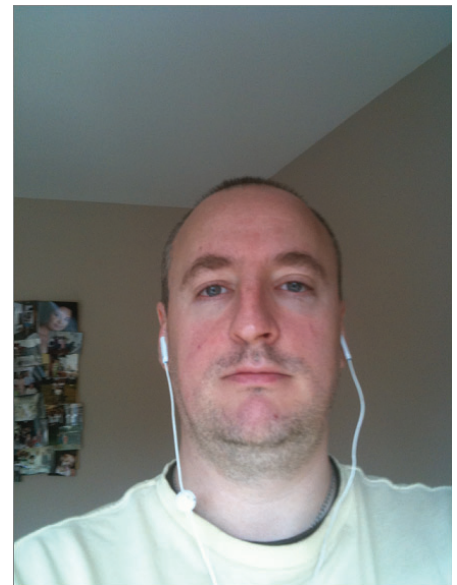
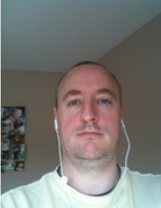



FIGURE 11-19: An image of Lester Madden

Lester Madden

Career Overview



With 15+ years' experience at the cutting edge of technology, Lester has held executive roles in Marketing and Project Management and has successfully developed and executed Go to Market strategies supporting product roadmaps, corporate goals and objectives across the Mobile, VoIP and IT industries. Formerly Developer Marketing Manager at **Symbian** Foundation/Symbian Software Ltd, developing key relationships, building worldwide marketing plans and identifying potential partners - with developers, e-retailers, mobile operators, manufacturers, digital licensees and other players. Previously Partner and Developer Relations Manager at **Skype**, launching and integrating the Skype application store, setting up the developer community and increasing the portfolio of third party applications that use the Skype API. Also held the role of **Project Manager, EMEA Developer Platform Evangelism Group at Microsoft Ltd & Microsoft EMEA**. Experience in leading and implementing Training to large groups and nominated for e-learning awards in several categories, alongside Channel 4 and The BBC.



View the Lester Madden channel in junaio

Highlights of Expertise

- Creating marketing Building Go to Market programs bringing products to market.
- Unique combination of technical, marketing, projects on time and on budget.
- Accomplished in online, digital, and B2B marketing.
- Strong problem
- Able to communicate effectively at the technical and executive level.
- Confident and accomplished public speaker.

FIGURE 11-20: An image of Lester Madden empowered by natural-feature tracking

Use the Listing 11-4 code to recreate the sample. The `lm_tracking.xml_enc` is also available for download. Only the image has to be encoded; you don't have to encode the entire CV. So I would be free to use my picture in different locations, all of which would link to my LinkedIn profile. Try re-creating the sample using your own image and links to your Facebook or LinkedIn profiles.



LISTING 11-4: Visual search on a face

Available for download on Wrox.com

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
  <results trackingurl=\"http://www.augmentedplanet.com/
wp-content/uploads/playground/junaio/lm_tracking.xml_enc\">
  <poi id=\"1\" cosid=\"1\" interactionfeedback=\"none\">
    <name><![CDATA[A Name]]></name>
    <author><![CDATA[metaio]]></author>
    <l>0.0,0.0,0.0</l>
    <o>0.0,0.0,0.0</o>
    <translation>0.0,0.0,0.0</translation>

    <mime-type>text/plain</mime-type>
    <mainresource><![CDATA[]]></mainresource>
    <route></route>
    <customizations>
```

```

    <customization>
      <name>Name</name>
      <type>url</type>
      <node_id>onTargetDetect</node_id>
      <value><![CDATA[http://www.linkedin.com/in/lestermadden]]></value>
    </customization>
  </customizations>
</poi>
</results>";

?>

```

Using a picture is just one example of how to provide interaction; a company logo is another example. Of course, you'll need to provide an indication that the image features an AR aspect or it will never be used.

OVERLAYING VIDEOS (MOVIE TEXTURES)

You've learned how visual search can be used by both non-developers and developers alike. You might be wondering why you would ever write code when you can do everything without the brainpower and effort necessary to building channels in PHP. But writing PHP gives you much more control over how and where video is displayed.

For example, you can play video directly over the tracking image instead of playing the video full-screen. Playing video this way is useful because the original tracking image is still visible, so the overall experience feels more connected than when you're simply transferred to a full-screen video. It's particularly useful if you are building an AR experience around a magazine advertisement. Figure 11-21 shows an example of a similar campaign created by metaio. To test the functionality, locate the InsideAR channel in your junaio client by searching for the keyword *insidear*. Then point your camera at the image shown in Figure 11-21.



FIGURE 11-21: metaio's InsideAR advertisement

As shown in Figure 11-22, once the recognition has taken place, the video begins to play. Rather than consume the entire screen, the video is formatted to play in a smaller area. As you can imagine, this is beyond just using a lower-resolution video, which would just be up-scaled to fill the screen. An important difference to note with playing videos in situ is that the videos have to be downloaded first and they are not streamed. Thus, it's not advisable to use large videos or create channels that have an abundance of video content because each video will need to be downloaded to the client before the user can interact.

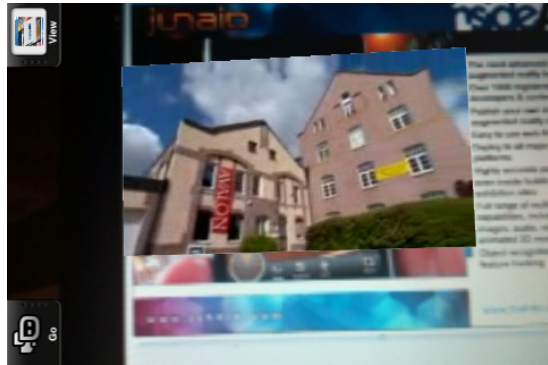


FIGURE 11-22: In-place video

The code in Listing 11-5 recreates the aforementioned demo. All the resources are located on the junaio server. Note if you have any problems with the demo (for example, the resources download but the video does not play), re-encode the `insidear.jpg` and host this on your own server. By doing this, you'll be certain of the tracking image.



Available for
download on
Wrox.com

LISTING 11-5: gluechannel.php

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results trackingurl=\"http://www.junaio.com/publisherDownload/
    tutorial/tracking_tutorial.xml_enc\">
<poi id=\"1\" cosid=\"1\" interactionfeedback=\"click\">
<o>0.0,0.0,0.0</o>
<l>0.0,0.0,0.0</l>
<translation>0,0,0.0</translation>
<mime-type>model/md2</mime-type>
    <mainresource>http://www.junaio.com/publisherDownload/
        tutorial/movieplane.md2_enc</mainresource>
<s>.3</s>
<force3d>>true</force3d>
<resources>
    <resource>http://www.junaio.com/publisherDownload/
        tutorial/insideAR.3gp</resource>
</resources>
</poi>
</results>";
?>
```

Encoding Movie Textures

You probably noticed a few new resources in Listing 11-5. In fact, the code looked more like the code you used to for 3D graphics rather than the code you used for playing video. When creating movie textures (for example, videos that will play in situ), there are a couple of hoops and hurdles that you need to be aware of. Unfortunately, it is not as straightforward as setting a scale parameter to determine the size. First, you'll notice Listing 11-5 doesn't mention an MP4 video file. Second, you might be wondering why we are using an MD2 file — which, if you recall, is the extension of a 3D model. Let's answer the MP4 question first. In Listing 11-5, the video file used is of type 3GP.

3GP and 3G2 files are multimedia files based on the MPEG-4 standard, but the format has been specially designed for transmitting to mobile devices and they are much more efficient at transmitting over bandwidth constraints. The video has to be downloaded rather than streamed, so you want this to happen as quickly as possible — and 3GP files are the best solution. The good news is there are lots of tools to convert your MP4 files to this format. The tool I use is called SUPER from eRightSoft (www.erightsoft.com). It's as simple as setting a few options and downloading the resulting converted file.

For junaio to play your file, you need to encode your MP4 files with the following settings.

- VideoCompression: MPEG4 codec / 3G2 container.
- VideoResolution: 176x144px @ 20fps
- AudioCompression: AAC LC
- AudioResolution: 22050kHz Stereo
- AspectRatio: Will be determined by the size of the MD2 plane

Figure 11-23 shows how these settings are made in the SUPER client.

During the encoding process, you will be asked if you want a file in 3GP or 3G2 format. Select 3GP because it's a superset of the functionality. Install SUPER if you'd like. If you do, encode the hanna.mp4 file and use the settings shown in the previous bulleted list. Once you have created the file, it should automatically be saved in the same location as the original. If you don't want to install the SUPER software, just use the hanna.3gp file I have provided. Either way, upload the 3GP file to your server and replace the following line in your gluechannel.php file:

```
<resource>http://www.junaio.com/publisherDownload/tutorial/insideAR.3gp</resource>
```

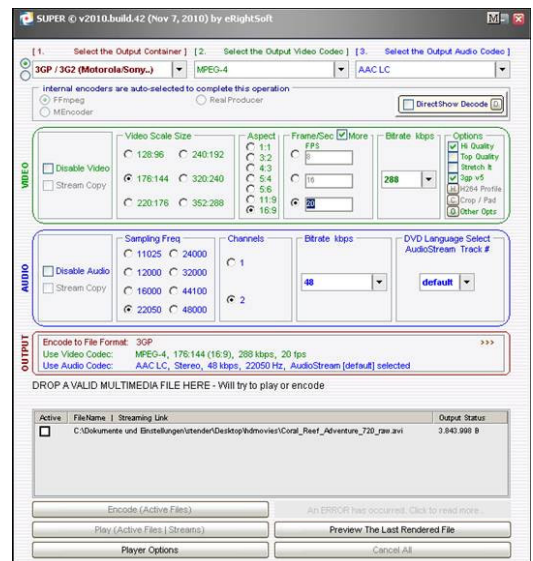


FIGURE 11-23: The SUPER video converter

With this:

```
<resource>LOCATION OF THE .3GP FILE ON YOUR SERVER</resource>
```

Since you have left the tracking image alone this time, you should see a video of a rather bemused Hanna played. I previously said that 3D models have textures; in most cases, textures are PNG files that wrap around an image as skin. For example, a soldier could have different PNG files representing different clothing. The `movieplane.md2_enc` file is a rectangle that has no texture; instead the video is used as the texture and applied in much the same way you would apply a PNG texture file to a 3D object. Also, notice that the `scale` tag was applied. This affects the MD2 file, not the video. The purpose of changing the scale is to ensure that the video and the rectangle plane are the same size. So are the three components:

This code sets the resource plane the video will be applied to:

```
<mainresource>http://www.junaio.com/publisherDownload/
tutorial/movieplane.md2_enc</mainresource>
```

This code sets the scale of the MD2:

```
<s>.3</s>
```

And this code sets the 3PG-encoded video to play over the plane:

```
<resource>http://www.junaio.com/publisherDownload/
tutorial/insideAR.3gp</resource>
```

Perhaps you have just had a light-bulb moment and are wondering if video can be applied to other objects? For example, can you apply video to a 3D model? The answer is, of course, yes! It can be used in this way but you have no control over how the video is mapped. In Figure 11-24, I have used the `hanna.mp4` video and wrapped it around the `fighter.md2` model used previously. The video plays, but as you can see from the image, it is very difficult to make sense of the video because it follows the contours of the model.



FIGURE 11-24: Video wrapped around a 3D model

If you want to try this, the code is shown in Listing 11-6.



Available for
download on
Wrox.com

LISTING 11-6: Mapping video to complex 3D objects

```
<?php
echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
  <results trackingurl=\"LOCATION OF YOUR TRACKING FILE\">
    <poi id=\"3\" cosid=\"1\" interactionfeedback=\"click\">

      <o>0.0,2.0,0.0</o>
      <l>0.0,0.0,0.0</l>
      <translation>0,0,0.0</translation>

      <mime-type>model/md2</mime-type>
      <mainresource>LOCATION OF YOUR FIGHTER.MD2_ENC
    </mainresource>
    <s>.3</s>
    <force3d>>true</force3d>
    <resources>
      <resource>LOCATION OF YOUR .3PG FILE</resource>
    </resources>
  </poi>
</results>";

?>
```

IMAGE REQUIREMENTS FOR NATURAL-FEATURE TRACKING

As we come to the end of the natural-feature tracking section, we should explain what makes for a good tracking pattern. The reference image that you use for your natural-feature tracking pattern should be a mix of high contrasts and colors to help the junaio engine identify the image. When choosing a tracking image for the cat demo, a picture of Ray sitting on a beige couch (see Figure 11-25) wouldn't have been a good image to use because there is not enough contrast or sharp edges to aid identification.

Other examples of poor images include pictures of plain clouds or empty pitchers of water, each of which lacks any focal object. Other bad examples are pictures that are either too dark or too light. A far better reference image is a complex image like that shown in Figure 11-26. Other good reference images are book covers or magazines; they often contain high-contrasting information, such as bold titles and captivating images.

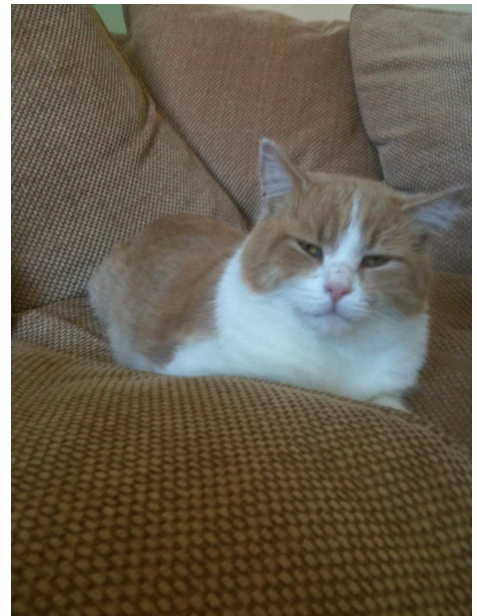


FIGURE 11-25: Ray on beige background



FIGURE 11-26: A good tracking image

The tracking image you use should also be presented in the same aspect as it will be used when printed. Don't crop out key parts. For example, don't upload your tracking image and then crop the image into a circle — most of the tracking information will be lost and you will achieve poor results. A good rule of thumb is to assume that the image you create and use for the XML tracking encoded file is the final product.

For best results, take a picture of the image with your smartphone camera and use this as your reference image. This is often better than using a high-quality image because it's closer to what the smartphone will actually see. You can also reduce the size of the image used for the tracking image to around 300 pixels. This reduces the overall size of the tracking image that needs to be downloaded to the client and improves recognition results.

metaio's recommendations for tracking images are as follows:

- Use a pattern that is highly structured (includes a lot of visual hints with different colors, high contrasts, and sharp edges).
- Put your pattern in a “common” format (for example, a square format or rectangle format in 3:2 or 4:3 or similar).
- Text is usually hindered and uses flat-shaded surfaces.
- Crop to a highly structured area of your image and use this as a pattern.

- Make sure the image is not too dark and there are no reflection points on your pattern.
- The shortest side of the image sent to www.junaio.com/publisher/trackingxml can be approximately 150-200 pixels.
- Make sure that your object, as shown in the real world, looks the same as the pattern.
- Make sure images share the same format (for example, they have the same aspect ratios and are both landscape/portraits).
- Make sure there are no missing parts.
- Use diffuse, non-reflective materials for the pattern.

Once captured, consider that image untouchable. If you don't consider that image suitable for print, then retake the picture. Spending a little bit of extra time getting the image right will ultimately pay huge dividends.

SUMMARY

At the start of this chapter, you probably thought natural-feature tracking would be one of the most difficult technologies to master. As you have learned, however, metaio makes it easy enough that even non-developers can build natural-feature tracking solutions. Even when you use code, natural-feature tracking solutions are extremely simple to create. In my opinion, the most difficult tasks are reserved for the graphics developer, who has to create the 3D models. As developers, our job is reduced to just a few lines of code.

Armed with your new skills, why not experiment with your company's business card and amaze your work colleagues by building a channel that links your company logo with your company's web site?

PART V

The Next Steps

- ▶ **CHAPTER 12:** Adding Advanced Functionality
- ▶ **CHAPTER 13:** Taking Your Application to Market
- ▶ **CHAPTER 14:** The Future of AR

12

Adding Advanced Functionality

WHAT'S IN THIS CHAPTER?

- ▶ How to work with dedicated XML files
- ▶ How to create advanced interactions
- ▶ How to use LLA markers
- ▶ How to retrieve data from databases

In this chapter, you will complete your junaio skills by learning how to include advanced functionality, such as separating the XML from the PHP file and storing your POIs in either a dedicated XML file or in a database. You will learn about Latitude, Longitude, Altitude (LLA) markers and how they can be used in environments where no GPS signal can be received. You will also learn how interaction events can be created to provide click events to perform actions when POIs have been selected. Before you get started, make sure you've read Chapters 10 and 11.

WORKING WITH DEDICATED XML FILES

In the previous junaio chapters, you have created a PHP file that hosts the XML. Because this file contained a mix of PHP and the XML that represents the POIs, it can be problematic to maintain — especially when you start to add several POIs to the file. Typical errors include missing an opening or closing angled bracket or including unintentional extra characters. Such errors can be extremely difficult to track down since errors are exposed only at runtime.

A better approach to managing your POIs is to use a dedicated XML file and add POIs via a dedicated tool, such as XML Notepad. This enables the tool to report any errors with the XML and display where the error has occurred. Keeping your POIs in a database is an even better solution and we will examine that later in this chapter. For now, however, let's examine what changes are necessary to separate the PHP and XML. Locate the `index.php` file in the `src` folder on your web server. Inside the file, locate the following lines of code:

```
if(in_array('pois', $aUrl))
{
    if(in_array_substr('search', $aUrl))
    {
        include '../src/search.php';
        exit;
    }
}
```

Alter this code as shown:

```
$xmlFilePoisSearchPath = "resources/pois_search.xml";

if(in_array('pois', $aUrl))
{
    if(in_array_substr('search', $aUrl))
    {
        echo file_get_contents($xmlFilePoisSearchPath);
        exit;
    }
}
```

In the previous code, the `$xmlFilePoisSearchPath` variable points towards an xml file named `pois_search.xml`. This is located in a `resources` directory that you must create in the `html` folder.

You also must make a few changes to XML. For example, the PHP file that hosts the POI will currently resemble the following:

```
<?php

echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<results>
  <poi id=\"1\" interactionfeedback=\"none\">
    <name><![CDATA[A generic POI]]></name>

    <description><![CDATA[This is simple POI.]]></description>

    <1>51.589581,0.079794,0</1>
```



```

    <o>0,0,0</o>

    <mime-type>text/plain</mime-type>

    <thumbnail>http://www.myserver.com/logo.jpg</thumbnail>
    <icon>http://www.myserver.com/icon.jpg</icon>

  </poi>
</results>";

?>

```

Because you're extracting the XML, you need only the following code:

```

<?xml version="1.0" encoding="UTF-8"?>
<results>
  <poi id="1" interactionfeedback="none">
    <name><![CDATA[A generic POI]]></name>

    <description><![CDATA[This is simple POI.]]></description>

    <l>51.589581,0.079794,0</l>
    <o>0,0,0</o>

    <mime-type>text/plain</mime-type>

    <thumbnail>http://www.myserver.com/logo.jpg</thumbnail>
    <icon>http://www.myserver.com/icon.jpg</icon>

  </poi>
</results>

```

Notice in the previous listing that I edited the line of code that read:

```

  <poi id="1" interactionfeedback="none">

```

To become:

```

  <poi id="1" interactionfeedback="none">

```

Notice the removal of several slashes (\). Once you make these changes, save the file as `pois_search.xml` and upload the file to the `/html/resources/` folder.

The final step is to change the Callback URL in the junaio dashboard for your channel to now point to the `/html/index.php` file. Now when the channel is loaded, junaio will load the `pois_search.xml` file. Now you can add new POIs with an XML tool, which helps you maintain a valid structure. Figure 12-1 shows the simple POI in XML Notepad. By using this tool, you can enforce the structure. Upload the files you have amended to your server and test your channel.

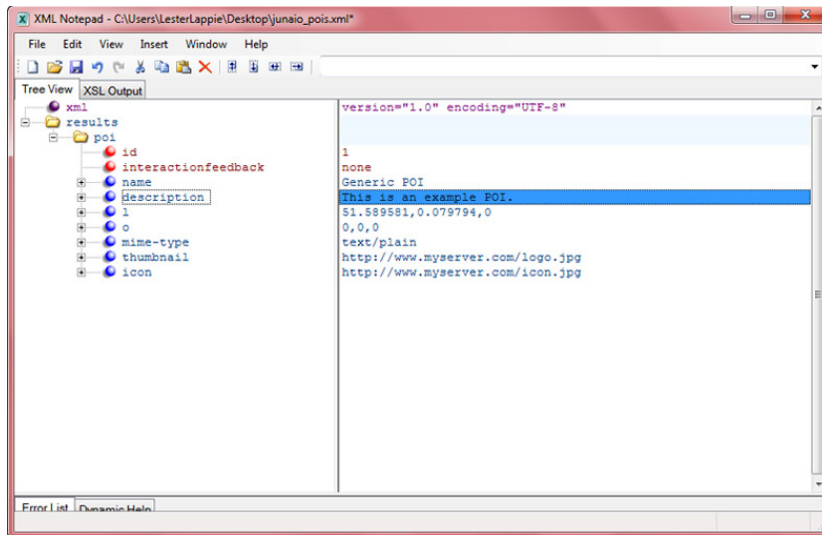


FIGURE 12-1: A simple junaio POI shown in XML Notepad

CREATING ADVANCED INTERACTIONS

In previous chapters you have enabled the user to interact with your channel by clicking POIs, reading descriptions, playing videos, playing sounds, or by visiting web sites. In this section, you learn how existing POIs can be changed or removed and how you can add new POIs when the user has interacted.

These interactions enable you to build compelling channels. For example, you can build a game where the user has to travel to a location on the map, and once they reach that destination and interact with the POI, that POI is removed and a new POI displayed.

Adding Interactions

To add interactions, you must create two XML files:

1. One file represents the POI search results.
2. The other file represents the action taken once the event for the POI has been triggered.

Listing 12-3 creates a channel that displays a 3D UFO that, when clicked, is replaced with a 3D balloon. You will create the XML search file that contains the UFO, an event file that contains the balloon, and then re-edit the `index.php` file to support events.

In the `index.php` file, you have already added support for separating the XML from the PHP and have created the code that will load the `pois_search.xml` file. Locate the code you added; you will now need to add support for events:

```

$xmlFilePoisSearchPath = "resources/pois_search.xml";

if(in_array('pois', $aUrl))
{
    if(in_array_substr('search', $aUrl))
    {
        echo file_get_contents($xmlFilePoisSearchPath);
        exit;
    }
}

```

Amend the code to reflect the following:

```

$xmlFilePoisSearchPath = "resources/pois_search.xml";
$xmlFilePoisEventPath = "resources/pois_event.xml";

if(in_array('pois', $aUrl))
{
    if(in_array_substr('search', $aUrl))
    {
        echo file_get_contents($xmlFilePoisSearchPath);
        exit;
    }
    else if(in_array_substr('event', $aUrl))
    {
        echo file_get_contents($xmlFilePoisEventPath);
        exit;
    }
}

```

These additions create a variable for the `pois_event.xml` file and extend the `if` statement to call the file when an event is triggered.

Working with the `pois_search.xml` File

The `pois_search.xml` file contains the location of each UFO. Use one of the latitude, longitude coordinates you have harvested from earlier chapters. As shown in Listing 12-1, you can use the UFO and Balloon models directly from metaio's server (and use the URLs shown).



LISTING 12-1: `pois_search.xml`

Available for
download on
Wrox.com

```

<?xml version="1.0" encoding="UTF-8"?><results>
<poi id="1" interactionfeedback="click">
<name><![CDATA[UFO]]></name>

<l><your latitude/longitude>,0</l>
<o>0,0,0</o>

<mime-type>model/md2</mime-type>
<mainresource>http://www.junaio.com/publisherDownload/tutorial/ufo3.md2_enc
</mainresource>
<route>>false</route>

```

continues

LISTING 12-1 (continued)

```

<s>300</s>

<force3d>true</force3d>
<resources>
  <resource>http://www.junaio.com/publisherDownload/tutorial/texture_ufo.png
  </resource>
</resources>
<behaviours>
  <behaviour type="idle">
    <length>0</length>
    <node_id>idle</node_id>
  </behaviour>
  <behaviour type="click">
    <length>0</length>
    <node_id>explosion</node_id>
  </behaviour>
</behaviours>
</poi>
</results>

```

Once you have created this file, upload it to the \html\resources folder on your server.

Working with the pois_event.xml File

The pois_events.xml file will be called when an event is triggered, as shown in Listing 12-2.



Available for
download on
Wrox.com

LISTING 12-2: pois_event.xml

```

<?xml version="1.0" encoding="UTF-8"?><results>
<poi id="1" interactionfeedback="click">
<name><![CDATA[Ballon]]></name>

<l>51.589581,0.0797947,0</l>
<o>0,0,0</o>
<mime-type>model/md2</mime-type>
<mainresource>http://www.junaio.com/publisherDownload/tutorial/balloon.md2_enc
  </mainresource>
<route>false</route>
<s>300</s>
<force3d>true</force3d>
<resources>
  <resource>http://www.junaio.com/publisherDownload/tutorial/balloon.png
  </resource>
</resources>
<behaviours>
  <behaviour type="idle">

```

```
<length>0</length>
  <node_id>action</node_id>
</behaviour>
</behaviours>
</poi>
</results>
```

In this example, both POIs share the `<poi id=1>`. Thus, as the POI with the `id` of 1 is clicked, the corresponding POI in the `pois_event.xml` file is displayed. Figure 12-2 shows the UFO from the `pois_search.xml`. Figure 12-3 shows the Balloon that forms the action in the `pois_event.xml` file.



These models are available for download at the book's companion web site.



FIGURE 12-2: The UFO POI



FIGURE 12-3: The Balloon POI

USING LLA MARKERS

Latitude, Longitude, Altitude (LLA) markers are a way to enhance the GPS accuracy of a location, particularly when an accurate satellite fix is not available. Simply put, LLA markers temporarily override the location reported by the smartphone's GPS device.

Configuring an LLA

The first step in creating an LLA is to edit your channel in the junaio dashboard. Midway down the list of channel options you will see a check box for Support LLA Marker. If this check box is selected, the camera actively scans for LLA markers when the channel is used. If a marker is detected, the GPS position will be updated to the coordinates provided by the marker. Once support has been enabled, you must create the LLA maker and provide the latitude, longitude and optional altitude parameters.

Configuring LLA makers is very easy. Simply visit <http://www.junaio.com/publisher/11amarker> and you'll be presented with a configuration form similar to that shown in Figure 12-4.

The screenshot displays a web interface for creating an LLA marker. On the left is a Google Map of San Francisco with a 'Go' button and a search bar. On the right is a configuration form with the following elements:

- Marker Name:** A text input field containing the word "Marker".
- Latitude in deg (-90 - 90):** An empty text input field.
- Longitude in deg (-180 - 180):** An empty text input field.
- Altitude in m (-1696 - 6496):** An empty text input field.
- Create Marker:** A blue button.
- Your LLA Markers:** A section header.
- Marker (March 12th, 2011):** A dropdown menu.
- Open Marker:** A blue button.
- Show on Map:** A blue button.

At the bottom of the map area, there is a copyright notice: ©2011 GOOGLE, MAP DATA ©2011 TELE ATLAS.

FIGURE 12-4: Configuring an LLA marker

Provide a name for your marker along with the relevant coordinates. If necessary, you can search for the coordinates on the map provided. Once you have created your LLA maker, it can be saved and printed for use. Typically, the marker will be located at the coordinates they represent. For example, in a museum, you can create an LLA marker for a particular exhibit. Because you know the location of that exhibit, you can provide directions to other areas of the museum even if there is no GPS signal.

To illustrate how LLA markers over-ride the GPS position, Figure 12-5 shows an LLA marker, which resembles a QR code. Like QR codes, which contain encoded data (such as a URL), LLA markers encode the latitude and longitude coordinates. LLA markers cannot be read by QR code readers since they are not designed to decode coordinates.

The LLA marker shown in Figure 12-5 contains the coordinates of an area called Leopoldina in Sao Paulo, Brazil — close to where my family lives. Scanning the LLA marker will place you in that location.

Update the pois_search.xml file with the code shown in Listing 12-3. When you add the code, leave the latitude and longitude coordinates as shown since you want to place this POI in its real location. The coordinates in Listing 12-6 represent the Leopoldina train station located in close proximity to the LLA marker coordinates.



LISTING 12-3: Leopoldina coordinates

Available for
download on
Wrox.com

```
<?xml version="1.0" encoding="UTF-8"?>
<results>
  <poi id="1" interactionfeedback="none">
    <name><![CDATA[Hello from Sao Paulo]]></name>

    <description><![CDATA[Vila Leopoldina Station]]></description>

    <l>-23.523924,-46.737864,0</l>
    <o>0,0,0</o>

    <mime-type>text/plain</mime-type>

    <thumbnail></thumbnail>
    <icon></icon>

  </poi>
</results>
```

Assuming you are not in Leopoldina when you load the channel, you will not see any POIs. However, if you point your smartphone camera at Figure 12-5, you will see a message indicating that an LLA marker has been detected. junaio will ignore the real latitude and longitude coordinates as reported by your phone's GPS in favor of the coordinates of the LLA marker. After a few seconds, the POI will appear.

While you are viewing the Leopoldina POI, select the View icon in the top right of the junaio UI and then select the Map icon. When the map is displayed, you will notice that instead of your real location, the Google Map is positioned on Leopoldina. It's as if you have been magically transported to Leopoldina and you are standing by the side of the street. Only junaio assumes you are in Leopoldina; if you open another mapping or navigation application, your real latitude and longitude coordinates will be used. LLA markers will remain valid in junaio until either you close the channel or you scan another LLA marker to change your position.

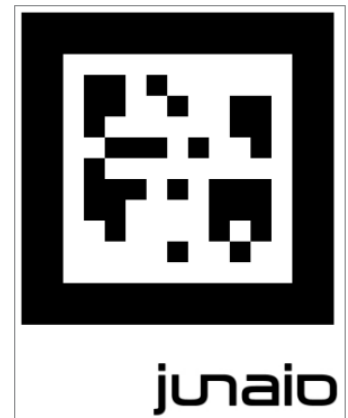


FIGURE 12-5: A sample LLA marker

RETRIEVING DATA FROM A DATABASE

In most cases, you want to access data that is stored in a database. In some cases, you will create a custom database to hold your junaio channel POIs; in other cases, will access data where the format has already been created.

1. Download and extract the junaio SQL starter package from www.junaio.com/publisher-Download/junaioSQL.zip
2. Extract the files and copy them to your server.
3. Locate the `config.php` file located in the `config` folder on your server:

```
//General channel stuff
define('JUNAIIO_API_HOST', 'http://api.junaio.com'); // URL to the junaio API
define('AUTH_DATE_TOLERANCE', 15 * 60 * 1000); // Set time period for
valid requests: 15 minutes
define('JUNAIIO_KEY', '<your junaio developer key>'); // Junaio developer key

//insert your sql information here
define("HOST", "localhost");
define("USER", "username");
define("PASSWORD", "thePassword");
define("DATABASE", "database");
```

4. Add your MySQL login details to the four `define` statements.
5. Save and upload the file back to your server.
6. In the new `search.php` file that was copied to your server as part of the SQL starter package (located in the `src` folder), locate line 24 (`$tablename = "";`) and update it to:

```
$tablename = "your_table_name";
```

This enables junaio to access your table data. Further down the `search.php` file, you will find a section that maps the contents from the table specified in the `$tablename` to an array:

```
$informationColumns = array(
    'id' => "", //unique alphanumeric value expected from this column
    'name' => "", //string for the name of the poi
    'description' => "", //string description
    'author' => "", //string author
    'latitude' => "", //number latitude
    'longitude' => "", //number longitude
    'mimetype' => "text/plain", //mime-type
    'mainresourcepath' => "", //only needed if mime-type != text/plain
    'thumbpath' => "", //url to thumbnail
    'iconpath' => "", //url to icon
    'phonenumber' => "", //phone number
    'mail' => "", //email address
    'homepage' => "", //homepage
    'enablerouting' => "", //expects "true" or "false"
);
```


7. To indicate it as a constant, rather than as a table field, use an exclamation mark (for example, !text/plain).

By using mapping in this way, you can maintain a single database for both your Laya and junaio POIs (although you may need to build a specific query to ensure that all the data is part of the data set).

8. To build a custom query, locate the following line of code and enter the relevant SQL that retrieves the data:

```
$query = "SELECT * FROM $tablename";
```

SUMMARY

In this final junaio chapter, you learned how to separate your XML and PHP to simplify the management of multiple POIs. You were presented with two choices: separating the XML into independent XML files or hosting the POIs in a database. You also saw examples of how POIs can have interaction events that change 3D objects into new objects how you can change the latitude and longitude to change the position once an event has been triggered.

Finally, you also learned how to use LLA markers in environments where no reliable GPS signal can be received. These LLA markers are an ideal solution for building indoor navigation applications.

13

Taking Your Application to Market

WHAT'S IN THIS CHAPTER?

- ▶ How to create the perfect listing
- ▶ Ideas for marketing your application
- ▶ Strategies for making money for your content

About a year ago, I was looking for an iPhone developer to build an application. The application was very simple in nature, so simple I could have written it myself given the time and access to a Mac. However, I didn't want to invest in buying a Mac so I decided that it would be cheaper/more efficient to pay a developer for a day's work. In my quest to find a developer, I found my way into an Apple developer's forum and came across a post from a marketing guy looking for help. His proposition went along the lines of:

"I have a great idea for an application but I don't have any development skills. The application is simple and will only take a day, maximum two days to complete. If a developer is interested in writing the application I will do all the marketing and we'll split the profits 50%."

I thought the opportunity sounded interesting. At the very least, if I was a developer, I would want to know more. What the marketing guy received, however, was 120 responses — all giving him abuse. Some developers pointed out that they wouldn't develop the application for 99 percent of the profits; other responses pointed out in more colorful language where the marketing guy could take his offer.

The temptation is for developers to think they are the most important part of the process. They often think all that is required of them is to build the product and then money will begin arriving in their bank accounts on a regular basis. In reality, building the application is only the beginning. It gives you a product, and you need additional expertise to tell people about

that product and encourage them to buy it or to install it. The days of the “build it and they will come” mentality are over. Marketing — like software development — is a full-time job and requires you to be just as creative. Every moment you spend trying to get users to install or buy your content is time away from coding and improving your product.

I have no idea what this marketing guy’s application idea was, or even if it ever got built. Maybe it was a multi-million dollar idea that could have made a developer very rich? I’m not advocating that, as a developer, you give away half your profits or agree to work for nothing. But you should accept the fact that development is only part of the process. Once you have built your application, the hard work comes with selling it and building a large user base. In this chapter, you’ll learn how to promote your applications.

MARKETING YOUR CONTENT

Now you have created your AR content, what is your next step? Will you simply publish to the respective store and sit back and wait for users to flock to your content? If that’s your plan, be prepared to be disappointed. The better step is to invest a little time and effort on a few simple techniques that will inform your audience about your content.

Marketing need not be difficult, so we are not going to focus on market segmentation, mature markets, customer-relation management, or any of the more complicated matters of marketing. We are just going to focus on some practical ways to tell people about your content. Some of these tips will be completely obvious, perhaps others not so obvious. As with all marketing techniques, not all work effectively in all situations. The best approach is to try a technique, measure its success, and then refine (if necessary).

Listing Your Content

The first place you should spend your time is with your *content listing*, which is the description of your application as it will appear in the Layar, junaio, or Wikitude client page. Every potential user — regardless of how they find your content — will see this page. This is your ultimate selling page, so it must be compelling enough to entice users to take the time to try your content. How you describe your content here will ultimately make or break whether users try your content.

Write a Catchy Title

The title of your content should be catchy. Think of this as a newspaper headline; it should grab the reader’s attention and demand he reads the description to learn more. Before you publish your listing, review the titles your competitors have used for their content. Are the listings shown alphabetically? If so, think about how your title will affect where your application will be displayed in the listing.

Write a Clear, Concise Description

The short description is your hook to get the user to load your content. Explain what your application does and explain how your audience will benefit from using it. This description should be three or four lines only, and be sure to use related keywords. It needs to be short and snappy

enough to whet the appetite of your audience. Your listing won't just consist of three or four lines; this will just be an introductory paragraph to hook potential users and entice them to read more.

Include Any Awards and Testimonials

If any user has sent you positive feedback about your content, use it. If one of your users sends you some praise, or if a blogger praises your content, ask them if you can include this praise in your description. Of course, praise should be descriptive. Simply adding, "Gary says it's a nice layer," is hardly worth making a song and dance over. A quote such as, "One of the best examples of visualisation of the London tube system available today," is much more interesting. Similarly, if your content gets reviewed or talked about on a review web site (such as augmentedplanet.com), add it to your description. For example, include text such as, "Featured on augmentedplanet.com!"

Put testimonials near the top of the description so it entices users to learn more about your app. Negative feedback shouldn't be ignored. If a user leaves comments that you find unfair, address them in a positive way (perhaps with a blog post). Don't get into a flame war with the user because you'll make it look as if your users are not important. If you fix problems or add features based on user comments, make a statement that says something along the lines of "we have listened to your requests and have added the feature that . . ." This is a great way to show that your users are important!

Use Bullet Points in the Full Description

The short description provides a brief overview of your application. It lays out the basic functionality of your application and generates the readers' interests in learning more. The awards and testimony section tells readers that other people find your application great and suggests that they might also find it to be great.

By the time readers begin reading the full description, they should already know about the benefits of using your application and should be ready to press the Install button. The full description is your opportunity to provide more details about the features and benefits of your application. You should use bullet points rather than long paragraphs because bulleted points are easier to read. No one wants to read a lot of text on his phone. If you make them work hard here, they're likely to think your app might make them work hard as well. With a well-formed description, readers will know exactly what your application does and why yours is better than all the others that show up in the search results.

Include a Call to Action

Okay, they have read your description, they are impressed with your testimonials, and they agree with your entire marketing blurb. You have a potential user here. What do you want them to do next? Don't just assume they will install your application; provide a call to action that tells them what to do next. This can be as simple as, "Now try it for yourself and discover what's around you!"

Provide Compelling Screenshots

Visual representation of your content in action is just as important as the written description, so include screenshots where possible. You should pay careful consideration to the screenshots you use. Don't just lean out the window and take picture of the house opposite you, with a logo indicating that there is a fast-food restaurant two miles in that general direction. Take the time to travel to a location relevant to your content.

As you previously learned, POIs have a tendency to move around (depending on various conditions), so it can be difficult to get the POI bubble to appear in the right space. But it's worth your perseverance!

Provide Feedback/Support

When you include an email address where your users can get in contact with you, you show that you have an interest in the continued development of the content. It's also a great way for users to submit bugs, ask for assistance, or even send you feedback that you can use to make your listing even more compelling.

It goes without saying that you should always review your description in a program like Word or Google Docs and run a spelling check. I find it very helpful to read the text aloud or, if you have a text-to-speech application installed, have the text read back to you. You'll be surprised how the eye sees what it wants to see and not necessarily what is written. Don't publish a description that is full of grammar or spelling errors; it tells your reader your content is not polished.

THE RIGHT TOOLS FOR THE 3-D JOB

Figure 13-1 shows two examples of the same fictitious London Tube station finder. The description on the left is the half-baked effort with a simple description and a screenshot taken from my office window. The description and screenshot show what the layer can do but is not compelling enough to entice the user to install and try the content. In contrast, the screenshot on the right shows that I have applied a little more care and attention in writing the listing. Which one would you install?

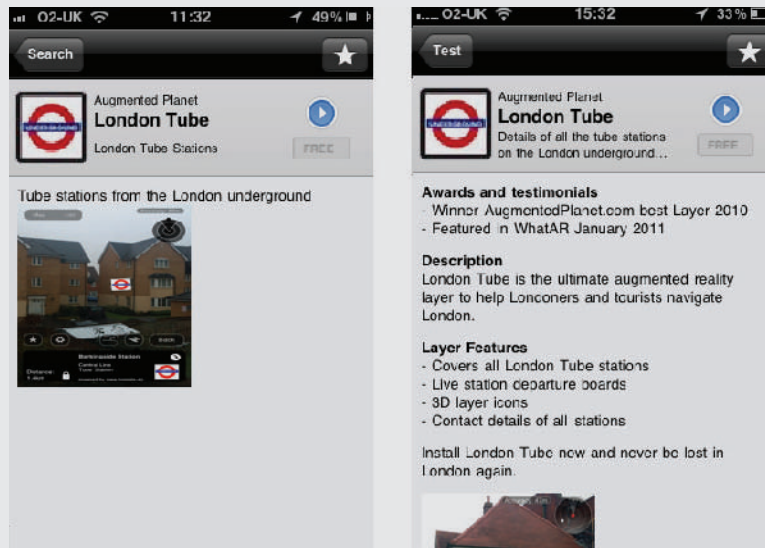


FIGURE 13-1: A comparison of marketing information

Generating Excitement

Now that you have created a compelling listing, it's time to tell people about your content and drive them to test it. If you ignore these steps, it means you have the best AR content in the world but no one will know about it. All your hard work will go unnoticed!

Create a Blog/Product Page

At the very least, you should have a product page that describes your AR content in more detail. To keep things simple, this can be a WordPress blog that you use to update with screenshots of your content in action or to announce new features.

Since blogs can be easily updated, consider publishing hints and tips about how users can use your content. Even if your content is comprised of the location of fast-food restaurants, make sure you take updated pictures of restaurants that you have discovered and upload them to the blog. This enables you to easily create new content and drive others to the page. You may find that people will post pictures of restaurants that they have discovered or they may comment about features that they would like to see in future versions of your content. Your blog will give you a way to communicate with your users and engage in discussions.

Blogs also provide you with an existing audience to which you can communicate the release of new AR content. If you have 2,000 users using your London Tube layer and they read your blog, you already have an audience of 2,000 potential new users of your new restaurant layer.

Don't forget to also tell them how to find your content. On your blog, you can do this easily by including a QR code that users can scan with their cameras to load your content instantly. Don't be concerned if you don't get a lot of hits immediately; your blog should steadily gain traction as you build the content.

Use Twitter

A Twitter presence can inform potential users of your AR content. Twitter is limited to 140 characters, and that's not a lot when you want to inform people about your new AR content. To master the art of tweeting, use tweeting in conjunction with your blog. The tweet becomes the driving force behind sending users to your blog or web site, where they then learn more about your AR content.

The art of a good tweet is to ensure that it reaches the right users. If they are interested, they will re-tweet the content to their followers, and those followers are likely to have similar interests. A single tweet can easily reach tens of thousands of people. I am not known to be an active tweeter, but I do make a point of tweeting the fact that I have posted a new blog post on augmentedplanet.com. A single tweet announcing a blog post can generate as many as 1,500 hits in just a few hours.

A good practice is to tweet when your audience is online. If you tweet about UK Tube stations at midnight (when your audience is asleep), you're not likely to generate a lot of traffic. Similarly, if you have content that is applicable across different countries and time zones, consider tweeting again at the later in the day when your second target audience comes online. You'll need to figure out the perfect time to use tweets that bring users to your blog.

So what makes a good tweet?

- A good description or headline that grabs the reader's attention is essential.
- Include a URL that provides more detail. Since long URLs can use a lot of those valuable 140 characters, shorten them with a service such as <http://ow.ly>, <http://bit.ly/> or <http://www.tinyurl.com>.
- Make sure your tweet reaches your audience using both hashtags and @.

Include Useful Hashtags

It can be difficult to search Twitter for relevant content, but thanks to hashtags, related information can be grouped together and easily found by searches. In addition, most Twitter clients let you subscribe to hashtags so you can stay informed with what the Twitter community is talking about. Think of hashtags as searchable keywords that enable you to search and find relevant content.

Some useful hashtags you may want to include with your tweets include the following:

- #augmentedreality
- #ar
- #layar
- #junaio
- #wikitude
- #technology
- #iphoneapps
- #androidapps

Hashtags can be found at <http://hashtags.org> along with a graph that shows how popular each hashtag is. A well-used hashtag may drive your content to thousands of potential users.

Use the @ Symbol

At the time of this writing, there were 175 million Twitter users and 95 million tweets were written every day. It has quickly become the new way to communicate with large groups of people. The object of tweeting is to gain as much exposure as possible, so in addition to using a hashtag to target the wider community, use tweeting directed at the platform provider that you have used for your content. For example, if you have built a new junaio channel, tweet @junaio to let them know your new junaio channel has been published. Why? Because you hope they will re-tweet your tweet to their community, which exposes your channel to an even bigger junaio audience.

Of course, some tweets might be of interest to different audiences. Other tweets might be of interest to the Twittersphere at large.

Figure 13-2 shows how I tweet from the Twitter account for my Augmented Planet blog. Notice that I include a headline to describe the article and a URL where the user can click to read more. In addition, I use a mix of hashtags to bring the blog post to the attention of the audience who will be

most interested in the content. I also @ the relevant person or company to give them the opportunity to promote the article to their communities.



© 2011 TWITTER

FIGURE 13-2: The Twitter account for Augmented Planet

Here are the Twitter accounts for the AR platform providers you have built content for:

- @layar
- @junaio
- @wikitude

In addition, Augmented Planet can be reached at @AugmentedPlanet and I can be reached at @lestermadden.

Twitter is a tool you'll either love or hate. Too many Twitter users seem obsessed with telling the world they are about to eat their tuna and tomato sandwich. But as long as you use Twitter responsibly, you'll have one of the most powerful marketing tools in your hands.

Create a YouTube Video

Creating a video is a time-consuming exercise, but if you are serious about enticing users, video is well worth the time and effort. YouTube videos are not only available to the casual YouTube viewer, they can be embedded on your web site or, better yet, a blogger's web site. Think of a YouTube video as your own mini commercial to create awareness. The first thing I do when I blog about an AR product is scan YouTube for a relevant video.

Write Press Releases

Press releases can be a useful way to get publicity in the major tabloids and web sites. Press releases are an art form and should be entrusted only to professional writers. A professional will also help you distribute the press release to the right channels. There is a cost involved with writing and distributing press releases, so make sure your news is indeed news worthy. If you think that your content is compelling and enough to warrant a press release, contact the platform provider to see

what assistance they can provide. Many are eager to provide you with a testimonial and some guidance. Your publicity is their publicity.

If you are on a limited budget, consider contacting a local college that offers writing or journalism classes. Sometimes students are able to earn extra credit or fulfill course requirements by working on ‘real-world’ projects where they can gain experience. Often they’ll be cheaper or perhaps even free, and you’ll have someone who’ll want to do a great job so they can include their work on their CV.

Once you have a completed press release, there are a number of web sites that will distribute your press release on your behalf. Some will charge for their services, others host a free service where members of the media can scan a web site for content.

Also, search for AR-related blogs and send them your press release. That will help you reach the right audience and begin building a relationship with AR enthusiasts. I get 20-25 press releases a week. If you do send a press release to a blogger, include any discount codes for the application so it can be tested. Very few reviewers spend money just to write a review on an application. In addition, don’t be afraid to follow up with the blogger to see if they received your email. Some may not be interested in blogging about your solution; others (like myself) just need a reminder from time to time to help them prioritize.

Contacting Bloggers

It is amazing just how influential bloggers can be. When I started blogging about AR in 2009, I would have laughed if I would have told me that it would lead to television appearances on the BBC, radio interviews, and quotes appearing in *The Guardian* and the *Wall Street Journal*. These days, however, some bloggers are almost on par with paid journalists; you should try to get to know them so they can become advocates of your solution. Read the various AR-related blogs, connect with a few bloggers or industry leaders on LinkedIn, and, if possible, arrange to meet at industry events for face time. At any given time, I have 70 unread emails from companies looking to announce news. Those who I know personally will get preferential treatment; I will get to their emails sooner.

In addition to just getting to know bloggers, ask how they can help you with your launch and ask them if they are interested in running a competition on your behalf. A blogger’s priority is to generate traffic for his blog; your priority is to find users for your content. Having a blogger run a competition on your behalf is a win-win.

Why is all this good for you? Because bloggers are advocates of the technology. When a blogger writes about your AR content, be sure to include positive snippets in your content listing so potential users know just how good your AR content is. Email bloggers to inform them what you are working on. Bloggers like to be the first (and often only) to know about an exciting new app, ping them a week in advance of your product release date to give them advance notice. That gives them a chance to ask questions and prepare a timely blog post. You’re more likely to get coverage that way than if they just discover your content on another web site. Also ask the blogger if he will provide you with a testimonial that you can include with your listing. Most bloggers I know would only be too pleased.

Create a Newsletter

Creating a newsletter is not an easy task. Finding content and ensuring that the newsletter is updated and distributed on a regular basis takes a lot of time and effort. Many companies are eager to receive content that they can include in their newsletters to help make their newsletters original and news

worthy. Once your content is ready to go, email companies and ask them if they would like an article written by you for their newsletter. Most providers will take you up on your offer. They will likely use only a few lines from your article, referring readers to your web site to read more.

Purchase Advertising

Advertising is one way to drive awareness of your content. The high-end advertising sites (such as Google AdWords) are great, but only if you have the budget for them. If you have a limited budget, or if you want to reach only the AR community (rather than reach out to hundreds, if not thousands, of non-relevant locations), consider advertising on an AR-related blog.

Most bloggers have advertising space and it's often filled with default Google AdSense ads which earn bloggers just a few dollars. Advertising on an AR-related blog is beneficial because that audience is exactly the audience you are trying to reach. If you find a site that doesn't have space for advertising, don't fret; just send the site owner an email to see if she might accept a paid advertisement for a short period of time.

Also, don't be put off by advertising prices that seem too high. Advertising pricing is often negotiable, so feel free to haggle a bit. From a blogger's perspective, a 468x60 banner ad that earns only ten cents a day is not nearly as lucrative as a customer paying a \$75 fixed fee for a month. That blogger is interested in keeping you as a customer, the following month, so she'll often be eager to give you a two-month rate of, say, \$125. Now you've paid essentially \$62.50 for two months, instead of \$150 for two months.

You'll negotiate more effectively with smaller niche blogs; they'll be more willing to work with you than larger, more established blogs — but they also won't have an audience as large as the latter blogs. You do tend to get what you pay for here. However, because of the viral nature of the internet, a good app that is marketed well and supported with your marketing and customer-service efforts might very well take off after hitting only a smaller niche blog.

When working with bloggers, remember they are not being paid to blog. They're likely blogging for the love of technology; any money generated by their blogging likely goes toward non-essential expenses. When negotiating an advertising discount with bloggers, make sure you sell them on the fact it's a win-win for both parties.

Making Money from AR

Your AR application has been created and it's available on smartphones for consumers to enjoy. What are your options for making money? While there might be a lot of interest in your app, it doesn't necessarily translate to revenue. Quite frankly, your money-making options are somewhat limited.

Selling Your Apps

If you have your own launch icon, you can sell your application via the Apple App Store or Android Market. Typically, this route offers the industry's standard 70/30 split where the developer receives 70 percent of every purchase. When building AR browsers, your options are currently limited to selling content via Layar or building a standalone application outside of the browser using the Wikitude API or another custom AR API engine. The latter route is simply not that rewarding. In July 2010, we analyzed every iPhone application from the App Store, recording its

respective price and category. We found 207 AR browsers for the iPhone platform alone. As shown in Figure 13-3, the majority of AR browsers were free.

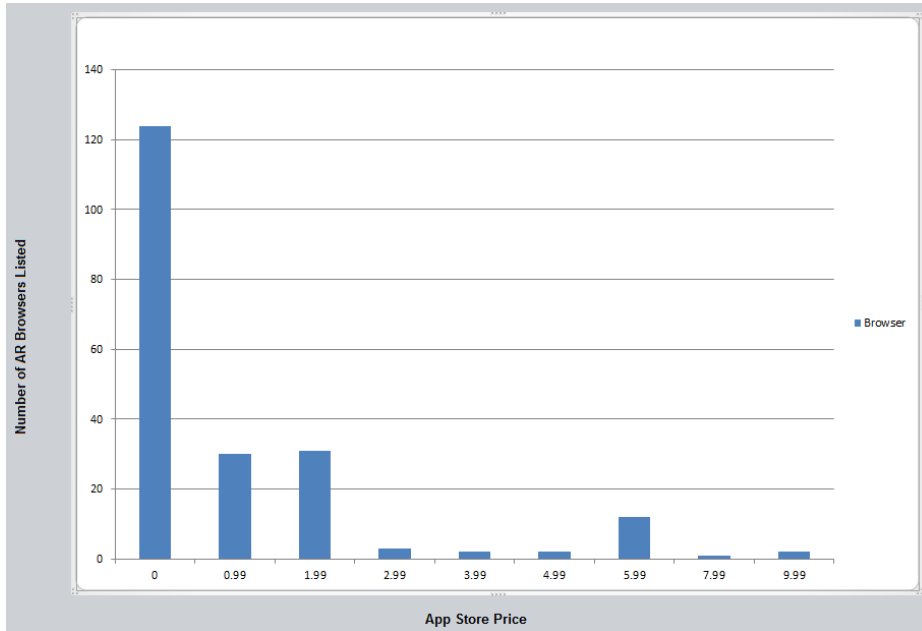


FIGURE 13-3: iPhone browsers and price (Source: augmentedplanet.com)

Free is always going to be a difficult price point to beat, especially if you want cross-device compatibility. If you are selling your application via an application store, you will have more success when you're simply using AR to extend your existing application. For example, if you have an existing application that provides users with route-planning tools for the London Underground network and the application links live departure boards to display train status information, adding AR browser functionality is clearly a benefit that extends your application. On the other hand, if your AR application is based around showing the position of the London Tube stations, then there is no additional value to the user. There are already plenty of applications that will do that free, so you are less likely to be successful if you are trying to sell an application that is simply an AR browser.

Selling Layers

If you don't want to create your own application, you can simply sell access to your POIs. But Layar is the only browser platform that supports access to paid content. Consumers from nearly 200 countries can purchase your Layar content using PayPal.

With Layar, the prices are set in tiers and content can be priced between \$0.99 and \$20.59. This gives you ample opportunity to earn serious money from selling your content. The price structure created by Layar is subject to change, so for the latest pricing tiers, you should refer to the Pricing link in your Layar Dashboard. With each sale, you earn 60 percent of the revenue (minus any tax). When you charge for layers, you should understand there are high standards for quality and

uniqueness. Layar does, however, represent a good way for you to reach an audience of more than one million AR consumers.

When building content for a foreign market — particularly if you intend to sell it — you should seriously consider localizing the content for the target countries' spoken languages. This ensures that your layer appeals to the intended audiences. While there are many web sites that provide free translation services, you should use professional translation services so the layer is translated correctly. You can inspect the `lang` parameter from the query string to determine the language of the Layar client and then programmatically determine the appropriate language.

Becoming a Publisher

If nothing else, I hope this book has inspired a desire in you to become an expert on the various AR browsers. At this point in the book, you probably have a favorite platform and have learned just how easy it is to create AR content. Armed with your new found skills, why not become a publisher who creates content on behalf of others? Many brands want to cash in on the AR craze and simply don't have the skills or staffing necessary to create content. As you have learned in this book, AR is nothing to be scared of.

Once you have built and released content, you're eligible to join the various platform certification or development programs. As a member of these programs, you'll be among those who are contacted when companies are in search of AR content.

Layar Programs

Layar maintains a list of developers with public profiles at www.layar.com/publishing/developers/list. Any developer who has created a layer and selected the relevant box for his profile to be public will be listed here.

In addition, Layar has created the Layar Partner Network, a program that offers monthly webinars and access to information that the general developer community doesn't have access to. Additionally, Layar recommends Layar Partner Network members to companies seeking help with developing AR content.

To join, you must meet the following criteria:

- You have developed at least one published layer of high quality.
- You have at least one person on your staff who is dedicated to Layar development.
- You're willing to develop layers for third parties.

In return, you receive the following benefits:

- You can use the Layar Partner Network logo (see Figure 13-4) on your web site.
- You receive business-development requests from companies seeking Layar developers.
- You gain access to early and exclusive Layar-related information.



FIGURE 13-4:
The Layar Partner
Network logo



More information about the Layar Partner Network can be found at <http://site.layar.com/create/partner-network/>

junaio Programs

metaio runs the junaio Certified Developer Program, an invitation-only program that helps junaio developers who have proven their understanding of the junaio platform find business-development leads for their new found AR development skills. Typically, these developers have published several quality junaio channels and show a willingness to participate in the community.

The benefits of the junaio program include:

- You are listed in the public directory for all junaio developers.
- You receive business-development requests from companies seeking junaio developers.
- You can use the junaio Certified Developer Program logo (see Figure 13-5) on your web site or in your applications.
- You are invited to take part in beta programs.



FIGURE 13-5: The junaio Certified Developer Program logo

You can join both the Layar and junaio programs. In fact, many developers are already a member of both programs.



More information on the junaio Certified Developer Program can be found at: <http://www.junaio.com/publisher/certifieddeveloper>

SUMMARY

Building your AR content is only the beginning; the next step — and possibly the biggest challenge — is to get users to interact with your content. To do this effectively, you must market your application. As you learned in this chapter, your marketing starts with a compelling listing that grabs the user's attention and demands to be installed and tested. You also learned that Twitter and blogs are great resources for spreading the word about your content, particularly when you use hashtags or send your tweet to the right person.

Finally, you learned that when it comes to making money, AR is not a get-rich-quick scheme. As you gain experience, you can take advantage of the various programs put in place by the platform providers to help you generate paid development opportunities.

14

The Future of AR

WHAT'S IN THIS CHAPTER?

- How AR is being used in marketing
- How AR is being used for translation
- How AR is being used for interactive TV
- How AR is being used in diminishing reality
- How AR is being used in advertising
- How AR is being used in books and print
- How AR is being used in hardware

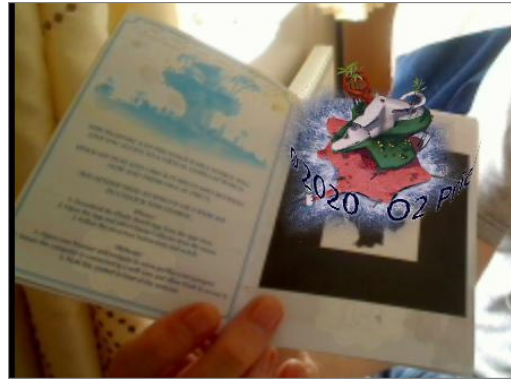
So you are almost at the end of the book and what a ride it has been. You now know how to create AR applications for junaio, Layar, and Wikitude. You can create location-aware POI browsers, you can utilize 3D, and you can build channels that recognize images and that overlay 3D. You are on your way to becoming a certified AR content creator and perhaps earn money from your new skills. In this chapter, you learn about other developments in AR, what other developers a building in the AR world as a whole, and what the future likely holds for AR.

USING AR IN MARKETING

In the last year, AR has been used in a variety of marketing campaigns. I have even seen AR used on the back of breakfast cereal boxes and I've seen it used as to enhance gaming experience. Many examples, however, require users to interact with an AR-enabled object via their PC webcams. As smartphones become increasingly more powerful, we will see more AR marketing campaigns that can be activated through mobile devices.

When the pop band Gorillaz launched their *Plastic Beach* album in 2010, the advertising campaign featured a unique passport that was distributed by *NME* magazine. The passport enabled readers to use their smartphones to gain access to exclusive content, such as competition to win Gorillaz branded prizes, get the concert ticket information, and even interactive 3D content. This campaign featured a marker that was used as the tracking image for the content that was subsequently overlaid. The campaign, developed by [PIAS], won the Augmented Planet Readers Choice Awards for best Augmented Reality Campaign of 2010. Like all successful campaigns, the AR functionality was not included just for the sake of being included. There was an effective strategy for how AR could be used. Campaigns like these provide good examples of how AR is becoming mainstream. Figure 14-1 shows an example of the campaign's 3D interactivity.

junaio is the only AR browser that offers natural feature tracking, so it's leading the way with how consumers interact with their smartphones and objects. In the United Kingdom in 2010, junaio was used to create the world's first AR postage stamp. The stamps are conventional postage stamps created to celebrate the 50th anniversary of the building of British Rail's last steam locomotive, *Evening Star*. When the user views the stamps with the junaio browser, they are taken to a short video of Bernard Cribbins reciting the poem entitled *Night Mail*.



© [PIAS] 2010

FIGURE 14-1: Plastic Beach's 3D interaction

Natural feature tracking opens up so many possibilities with how we interact with everyday objects. In previous chapters, you saw how you can already use AR to purchase books by using your phone to recognize the cover. In the future, you will be able to experience many products in much the same way. Imagine shopping in your local supermarket and wanting to know about a particular bottle of wine. Solutions already exist for the iPhone and Android that work in much the same way as the book example and will give you textual-based information (and, of course, price comparisons), but the future will enable you to see the winemaker talking about the wine and telling you why you should buy it. As we move towards that magical fabled day of device convergence, the camera is going to play an increasingly important role.

USING AR FOR TRANSLATION SERVICES

Have you ever travelled to a country where you don't speak the language and found yourself thumbing your way through a translation device while trying to understand some simple instructions? Those days are coming to an end.

There have been many attempts at combining AR and translation, but most of the solutions have not been effective. With any computerized translator, you are at the mercy of the optical character recognition (OCR) engine and its ability to convert written or typed input into something the

computer can translate. If the OCR fails to understand a word, or if it understands incorrectly, the rest of the translation is doomed to fail. As they say, garbage in, garbage out. There is also the problem with context. In the English language, the word *cool* can mean any number of things — ranging from the temperature to slang for how much we might approve of something. In Brazilian Portuguese, the word *legal* (pronounced *leh-gal*) is slang similar to the English slang of *cool*. Depending on the context on how it is used, it can affect the entire meaning of translation. “Your new car is really freezing!” doesn’t quite convey the same sentiment that you probably intended to convey. Similarly, if someone told me my car was *legal*, I would probably agree — after all, I do have all the necessary legal documentation for the car. So for computerized translation to be successful, you need a good OCR and a translation engine that is able to understand context.

Most of the AR translation applications I have seen are quite cumbersome to use. Typically, you must take a picture of the text to be translated. Then you must highlight the textual area and submit that image to the OCR engine. The image is converted into text and then sent to the translation engine to be translated. Finally, the result is displayed to the user.

The new generation of AR translation applications are quite impressive. They translate in the actual camera window and display the results immediately. You saw this in action in Chapter 2. AR also is on the cusp of offering voice-activated translation in real-time with mobile devices. Google recently added live audio translator functionality to the Google Goggles mobile application for the Android and a standalone iPhone application called Translate. The technology is still in the alpha stage, so it will be some time before it is accurate enough to hold a real-time conversation. However, it’s a glimpse of how AR will influence the technology world.

USING AR FOR INTERACTIVE TV

Interactive TV and AR may not seem like an obvious coexistence, but I do think there are tremendous opportunities here.

Have you ever voted for your favorite star on the latest reality TV program? Generally, this requires you to text your vote to a certain messaging number. These messages might come at a cost to you, you have to wait a couple of hours for all the votes to be counted before the results are known. Imagine if there was a way to use your phone to interact with your TV and gain instant feedback!

One of the latest junaio channels created for a popular German TV show, Galileo, does exactly that. At key points during the show, a question is placed in the corner of the TV screen. Users with junaio browsers can load the junaio channel on their smartphones and point their camera’s phones at the TV screen. The question is recognized in the exact same way you learned in the junaio chapters, and then it’s displayed on the users’ phones, which they can then use to provide their answers. Figure 14-2 shows the Galileo channel in action. Since the smartphone has a connection to the server, users can answer the question and receive real-time feedback.

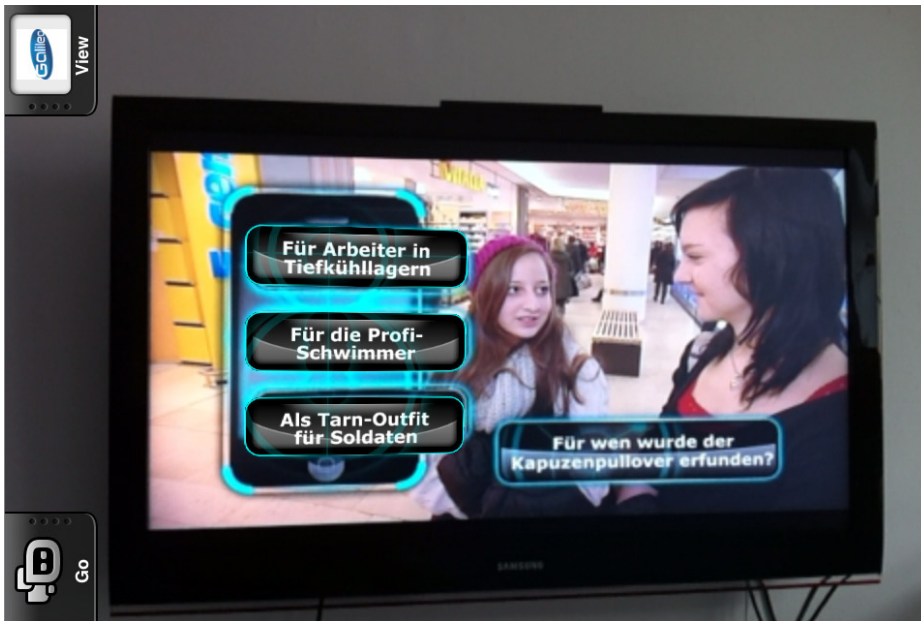


FIGURE 14-2: Interactive questions that can be used in the junaio channel

The possibilities with AR an interactive TV are both endless and exciting. Imagine taking part in live versions of your favorite quiz show from the comfort of your own home. As the questions appear, you simply scan the TV screen and use your phone to provide your answers, see your scores, and see how many people are left in the competition.

You can also watch a cooking show and scan a special icon in the corner of the TV screen that sends the recipe directly to your smartphone. Depending on your point of view, this is clearly straying into the visual search field. Even so, AR and TV are an exciting combination of technologies that will become more and more utilized.

USING AR IN DIMINISHING REALITY

The new reality catch phrase is *diminishing reality*. Where AR is the process of adding computer graphics to a live video feed, diminishing reality is the process of removing objects from a live video feed. Imagine a closed-circuit TV image of a criminal in a crowd of a few hundred people. It can be quiet difficult to track the individual in the crowd. But if you could digitally remove every individual except the criminal, tracking that individual becomes a lot simpler. Diminishing reality does exactly that.

Simply draw a circle around the object you want to exclude from the video. Then the software removes the object from the image by reducing the pixels and filling in the colors with pixels from

the surrounding space until the object ultimately disappears. As the object is removed in real-time, you can move the camera around the object while it remains invisible. Figure 14-3 shows an example of diminishing reality in action. This image was taken from the following YouTube video, which is worth watching if you want to learn more about the technology:

<http://tinyurl.com/3agbum1>

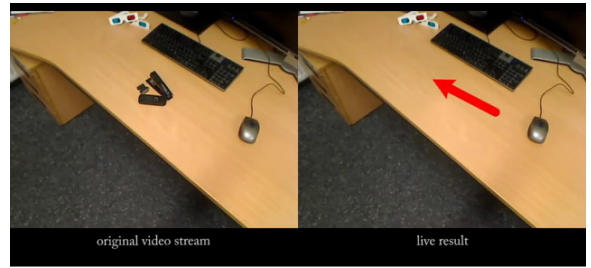


FIGURE 14-3: Diminishing reality in action

My guess is it will be a while before we see diminishing reality in smartphone applications. It's a technology that is more suitable for military or policing applications than it is for consumer applications. However, the technology will ultimately find its way onto smartphones so users can take live video or pictures and exclude people or objects in the background. If you have ever taken the almost perfect picture and found that it has been ruined by an unwanted person in the background, you can use diminishing reality to seamlessly remove the individual from the photo.

USING AR IN ADVERTISING

In London, the advertising displayed in the public underground system is moving towards digital signage. The printed billboards located alongside the escalators and on the platforms are quickly becoming an advertising medium of the past. As these printed billboards are being replaced with more interactive advertising, what role might AR play in the future of advertising? Consider the following anecdote an example of AR's capabilities here:

I recently needed to buy a new laptop, so I Googled for new laptops. Then I noticed that when I visited my favorite sports web site, I was shown advertisements for laptops. After my new laptop arrived, I decided that a tablet PC would be nice for browsing. You guessed it — as I browsed my favorite sports sites, I saw advertisements promoting iPads and associated tablet devices. Of course, this technology is nothing new. Many sites place cookies on your system to track the sites you visit and display relevant advertising content.

The possibility of combining advertising and AR makes for an interesting future. Face detection — the art of detecting the presence of a face, not necessarily recognizing you as a specific individual — is already a powerful tool used in digital signage systems. The software determines if your face is male or female, young or old, even happy or sad. Then it plays content that is relevant to the information it now knows about you. A young woman might see an advertisement for a particular brand of makeup; a young man might see an advertisement for the latest shaving product. Ultimately, this kind of technology enables advertisers to reach their target audiences with their messages tailored to fit those precise audiences.

The scary part of this? When this technology evolves to the point at which face recognition is used to determine exactly who you are and what products you have purchased in the past. Maybe your face will become the equivalent of a web browser cookie? Now combine this knowledge with

your browsing habits. Next time you stand at a train station waiting for a train, the video camera hidden behind the digital signage board may know a lot more about you than you think!

USING AR IN BOOKS AND PRINT

Imagine if you could bring books to life by holding your book next to your PC's webcam and seeing a 3-D image of how dinosaurs actually looked. Books are already beginning to appear on Amazon with exactly this kind of functionality. *Dinosaurs Alive* is one such book. Install the software from the CD that accompanies the book, present some of the special pages to the webcam, and you can interact with the dinosaurs.

As AR becomes more widely adopted, many books will feature AR functionality. Cookbooks will walk you through the recipe. Car manuals will show you how to perform maintenance on your vehicle. Children will create their own interactive stories that are different every time. Two companies that are specializing in this area today are www.barnstormbooks.com and <http://zooburst.com/> (see Figure 14-4), each of which lets users create their own stories by uploading images.

With AR, books can be constantly updated with new information (for example, an encyclopaedia can always reflect the latest scientific discoveries). This isn't necessarily a new concept. In 2009, Brazilian bookstore Editoras Online wanted to connect with younger readers. As part of its campaign, Editoras Online placed 4,000 stickers in random locations throughout Sao Paulo. The stickers represented 200 QR codes with messages of love or hate from Editoras Online's Twitter feed. Brazilians scanned the QR codes with their smartphones to view the messages.

At the end of the campaign, a book was produced with the 200 QR codes, each of which reflected a message received via Twitter — half the book dedicated to the love messages and half the book dedicated to hate messages. On a regular basis, the messages are updated with messages from Editoras Online's Twitter feed so the book always has something new to discover.

USING AR IN GAMING

There are some interesting AR games available on the iPhone and Android, but for those who like gaming with a little more realism, let's talk about robots and helicopters.

The Wowwee Rovio Drone (www.wowwee.com) is a mobile robot equipped with a VGA webcam that can be controlled from anywhere in the world. Resembling a high-tech, remote-controlled car (see Figure 14-5), the Rovio contains a wireless receiver and a sensor to automatically help it avoid obstacles. It can be programmed to patrol your house while you are away so you can remotely



COURTESY OF WWW.ZOOBURST.COM

FIGURE 14-4: Telling stories the ZooBurst way

connect and view the webcam video feed to ensure your house is safe and secure. Via the web interface, you can even take control of the unit and send it to new areas of the house. Thanks to its two-way microphone system, you can even drive up to your beloved pet and tell it how much you miss it. When the battery gets low, Rovio finds its way back to its charging unit and automatically charges.

If that was all there was to the Rovio, it would just be an exciting toy. What gives Rovio a place in the future of AR gaming is the fact developers can use an SDK to build Rovio-enabled games. Using the SDK, developers can recognize AR markers which can be used to create an interactive games environment. The markers can represent any 3D object you care to build. For example, markers could be 3D tanks or anti-tank missiles. Because you can connect to the Rovio and control it via the web, you can hide markers around the house and attempt to navigate your Rovio to a target while destroying or avoiding any obstacles. For an example of how the Rovio enables an AR game experience, see the video produced by LinceoVR, the creators of the SDK, at <http://tinyurl.com/4vg87rj>. The video also contains links to where you can install the SDK.

AR Drone from Parrot was probably the most eagerly anticipated AR gadget of 2010. Unlike Rovio, the AR Drone (www.ar-drone.com) was designed for AR gaming and supports multi-player



WWW.WOWWEE.COM

FIGURE 14-5: The Wowwee Rovio

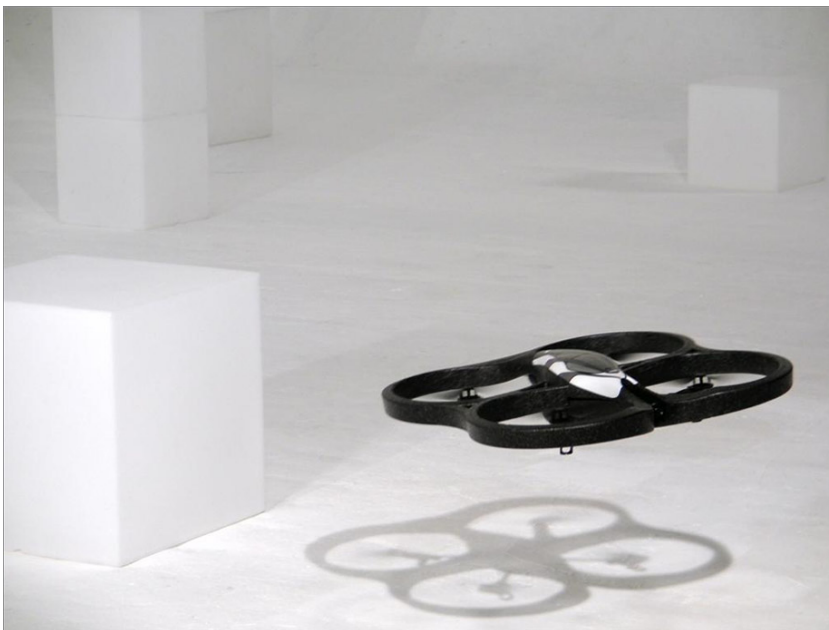


FIGURE 14-6: The AR Drone

functionality. With credit due to Frederic Simon, the drone can best be described as a Wi-Fi-enabled helicopter (see Figure 14-6) that is controlled by an iPhone, an iPad, or an iPod. More devices will be added in the near future.

The drone contains two cameras: a forward-facing camera that provides a cockpit view and a downward-facing camera that shows users what is below the device. The display from both the cameras is shown on the iPhone, which also provides the controls. The drone has been designed with AR gaming in mind. Markers can be interpreted in real-time, providing in-game objects such as buildings, missiles, and more. The AR Drone is the most innovative 3D AR platform released to date.

The AR Drone currently has five games available, two of which are multi-player and can be played with other drone owners. You can see the AR Drone in action at <http://tinyurl.com/yehx7vk>

USING AR IN HARDWARE

If you have seen the Terminator or Iron Man, you have seen the AR vision of the future — a future in which you're equipped with a pair of AR glasses that provides context on everything you see. Imagine looking at any individual and face-recognition software instantly tells you who they are. Imagine a world in which it is impossible to get lost or lose your way because your personal navigation system in your heads-up display (HUD) unit guides you to your destination. YouTube features plenty of glimpses into this future, including these two:

- <http://tinyurl.com/starkhud>. This is somewhat of a tongue-in-cheek promotional campaign of Iron Man 2, but the services are similar to those we'll be able to access some day.
- <http://tinyurl.com/holographics>. This is a more realistic vision of the future with an interesting focus on gaming. The video shows two guys playing air hockey in the street. Of course, it's an AR version of air hockey, so only they can see the game. In the future, this could be quite a sight.

In AR we often debate just how soon these services will become a reality. Last year, I wrote a blog post saying that I thought these types of services were around 20 years out. A lot of people disagreed with me, claiming the technology is only 5-10 years out. I still think that timeframe is unrealistic for such a technology to become universally adopted, but perhaps a follow-up book in a year so will be entitled *Building AR Applications for Your Personal AR HUD*.

In the meantime, research on producing AR glasses is being conducted by number of large electronics companies and work on producing viable headsets is surprisingly advanced. One of the leaders in producing consumer-ready glasses is Vuzix (www.vuzix.com). Today, you can pick up a Vuzix AR glasses system for \$500-\$2,000, depending on the system. However, before you rush out hoping that you'll be able to experience the world via a Layar-type interface, the Vuzix systems are designed for use with the desktop. The Vuzix Wrap 920 AR, for example, features a pair of glasses with two VGA video cameras. These cameras provide the input into the PC which is in then able to perform the recognition action. Using glasses in this way enables the PC to see where the wearer is looking and provides much steadier results than trying to keep a webcam focused on an object.

In industry, companies such as BMW are using similar headset systems to train their servicing engineers. Wearers simply put on the glasses and the system guides the wearer on how to service the

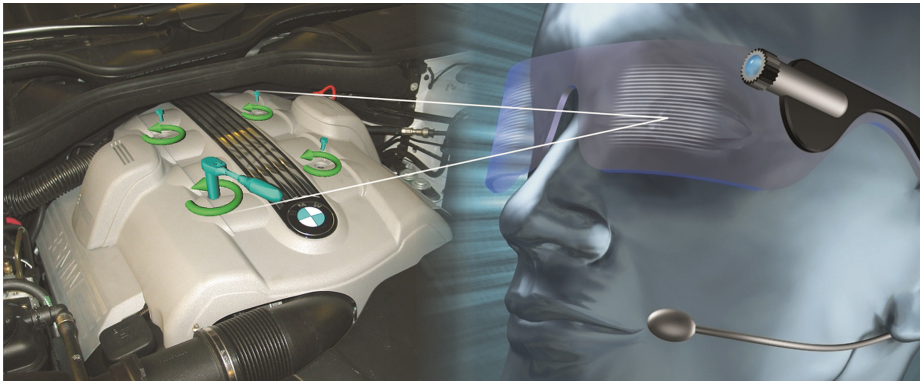


FIGURE 14-7: The BMW servicing application

car (see <http://tinyurl.com/bmwservice>). Such systems can save manufactures enormous sums of money because they reduce mistakes and the amount of training required. Figure 14-7 shows the BMW AR concept.

Of course, for these systems to be available to everyone, they can't be attached to such a powerful PC. That means the headwear of the future will need to be just as powerful but much lighter. The Japanese company NTT DoCoMo (among many others) is experimenting with building lightweight glasses that use the smartphone as the processor; the glasses connect via a cable or via Bluetooth. Figure 14-8 shows an NTT DoCoMo and Olympus system based on a 2008 prototype called AR Walker; it is very lightweight, very desirable, and has the potential to become a true mobile AR platform of the future. While the system is still very much a concept and not something that is production-ready, it's expected to offer personal navigation and e-mail solutions.



FIGURE 14-8: The NTT DoCoMo / Olympus lightweight mobile AR prototype (Photo courtesy of www.crunchgear.com)

In my original blog post, I said that it would be 20 years before these systems were mainstream products. In hindsight, however, perhaps 20 years isn't accurate. I'm willing to reduce my estimate to 10-15 years. But whatever happens, the next few years for AR are certainly going to be interesting.

SUMMARY

In this final chapter, you have learned how AR will change our lives. We live in a time where AR is almost ready to influence everything we do and see. In the future, every piece of text that you read or every sound you hear could be translated instantly into your native language. How you interact with your TV could change forever; every member of your household will be able to interact in real-time with their favorite shows.

Face recognition will instantly provide information on who is around you and, ultimately, what is on their minds—thanks to integration with Twitter and Facebook. In the immediate future, mobile will continue to provide the catalyst for AR to become an everyday technology. In the distant future, uture dedicated wearable AR systems will become as common as today’s mobile phones.

It’s not a question of if — it’s only a matter of when.



Wikitude Support and ARML Parameters

This appendix provides information you might find helpful when working with Wikitude.

SUPPORT

Here are some URLs that you will find useful in your Wikitude development:

- ▶ Wikitude development site: www.wikitude.me
- ▶ To contact Wikitude for general inquiries: content@wikitude.org
- ▶ E-mail for developers looking to join Wikitude content developer network: developer@wikitude.org
- ▶ ARML specification: www.openarml.org/wikitude4.html

ARML PARAMETERS

ARML (augmented reality markup language) is a proposed specification that allows content developers create content for the Wikitude 4 World Browser. ARML is built on a subset of KML. The parameters detailed in Table A-1 describe the full set of tags supported by Wikitude 4.

TABLE A-1: ARML Parameters

NAME	REQUIRED	DESCRIPTION
xmlns	no	www.opengis.net/kml/2.2: Standard KML namespace
xmlns:ar	no	www.openarml.org/arml/1.0: Standard ARML namespace
xmlns:wikitude	no	www.openarml.org/wikitude/1.0: Public Wikitude namespace
xmlns:wikitudeInternal	no	www.openarml.org/wikitude/1.0: Public Wikitude namespace
ar:provider	yes	Identifies a content provider or content channel. Wikipedia, YouTube, or Twitter are popular separate providers. Each provider must have a unique identifier.
ar:name	yes	Name of the content provider. This name will be used to display the content provider in the settings and bookmarks menu of the browser.
ar:description	no	Name of the content provider. This name will be used to display the content provider in the settings and bookmarks menu of the browser.
wikitude:providerurl	no	Link to the content provider. If the content provider adds an own logo the user will be redirected to the provider URL when clicking on the logo.
wikitude:tags	no	Logo displayed on the left bottom corner on Wikitude when an icon is selected. Format: 96 × 96 pixel, transparent PNG
wikitude:logo	no	The icons are displayed in the cam view of Wikitude to indicate a point of interest (POI). Format: 32 × 32 pixel, transparent PNG
wikitude:icon	yes	The icons are displayed in the cam view of Wikitude to indicate a point of interest (POI). Format: 32 × 32 pixel, transparent PNG
placemark		Placemark describes one point of interest (POI) in Wikitude.
ar:provider (within placemark)	yes	Reference to the content provider definition.
name	yes	Name of the POI. Displayed as POI title.

NAME	REQUIRED	DESCRIPTION
description	no	Description of the POI. Currently no HTML formatting is allowed.
wikitude:info	no	Additional information about a POI that is displayed in the bubble.
wikitude:thumbnail	no	Specific POI image that is displayed in the bubble. This could be for instance a hotel picture for a hotel booking content provider. Format: 64 × 64 pixel, PNG
wikitude:phone	no	When a phone number is given, Wikitude displays a “call me” button in the bubble. You can directly call the person/organization behind the POI. E.g. call a restaurant to reserve a table for dinner.
wikitude:url	no	Link to a web page that contains additional information about the POI.
wikitude:email	no	Write the person/organization an email directly from Wikitude.
wikitude:address	no	The address of the POI.
wikitude:attachment	no	Can be a link to a resource (image, PDF file, etc.). You could use this to issue coupons or vouchers for potential clients that found you via Wikitude.
point/coordinates	yes	The coordinates are entered in the format longitude, latitude, altitude. Altitude is optional. Altitude must be given in meters.

B

Layar Support and Parameters

Layar has one of the best websites for developers available. It is filled with useful hints, tips, How-Tos, publishing guides, and much more. Many of the samples used in this book have been written and developed by Xuan Wang from Layar. Xuan is also very active in supporting developers via the official Layar support forum.

SUPPORT

Here are some URLs that you will find useful in your Layar development:

- ▶ Layar development Wiki: <http://layar.pbworks.com>
- ▶ Layar support forum: <http://groups.google.com/group/layar-developers>
- ▶ Layar support e-mail: support@layar.com
- ▶ Feature requests for Layar can be submitted via a link found on the Layar support forum. Layar regularly publishes a report on feedback requests at: <http://layar.pbworks.com/w/page/27254513/Feature-Request-Feedback-Form>

REQUEST PARAMETERS

The parameters shown in Table B-1 are passed to the end point function. Table B-2 shows optional parameters that can be passed to the endpoint function via the query string. Table B-3 shows optional parameters that are available if you want to use authentication in your layer.

To see an example query string passed to the endpoint function, visit www.myserver.com/myendpointfunction.php?lang=en&countryCode=AF&lon=4.887339

The query string parameters are assigned to the `$value` array.

TABLE B-1: Standard Request Parameters

PARAMETER	FORMAT	USAGE	DESCRIPTION	API VERSION
userId	string	userId=123456789	Unique ID of the end-user, anonymized. It is a hashed code. In most cases, the IMEI is used to create the hash, otherwise another unique ID (for example, UDID for iPhone or other serial number on other devices).	v2.1
layerName	string	layerName=mylayer	Identifier of the layer. The unique name for a layer.	v2.1
version	string	version=3.1	This indicates the version of the API that the phone is using. Note that for older versions, strings like “ip3.0” might be used. However, this version string cannot be used any more to differentiate clients. (Android and iPhone clients use the same version string, as it’s the API version.) Current list of versions: “2”, “2.1”, “ip2.0”, “2.2”, “ip2.1”, “2.2.1”, “2.2.2”, “3.0”, “ip3.0”, “3.1”, “3.5”, “3.6”, “4.0”.	v2.1
lat	decimal	lat=52.360112	Latitude of current position (GPS coordinate).	v2.1
lon	decimal	lon=-4.895665	Longitude of current position (GPS coordinate).	v2.1
countryCode	string	countryCode=NL ISO country code identifier See www.iso.org/iso/english_country_names_and_code_elements for a complete list	This is the home country of the user, as determined by the network provider of the SIM card in the phone. If this is not available (for example, it’s an iPhone or no SIM card is inserted), it will return the locale value set on the phone. NOTE: If you have development mode on, the country setting in developer settings will overrule the values from SIM card and Locale setting.	v2.1

PARAMETER	FORMAT	USAGE	DESCRIPTION	API VERSION
lang	string	lang=EN	Language used on the client (locale setting on the phone). Currently using ISO-639-1.	V3.0
action	string	action=update	This tells the POI provider whether the client is requesting a full refresh of the layer or just an update: This can be caused by the new refresh parameter (<code>fullRefresh</code>) or by an intent containing the <code>action=update</code> command. Default <code>action=refresh</code>	V4.0

TABLE B-2: Optional Parameters

PARAMETER	FORMAT	USAGE	DESCRIPTION	API VERSION
accuracy	integer	accuracy=350	The accuracy of the current location, as given by the device. Note that accuracy may not be given if a fixed location is used (developer feature).	v2.1
pageKey	string	pageKey=aldfj9348dk	If the results are spread over more pages, the Laya Server can request the next page using this <code>pageKey</code> (if received in the previous response).	v2.1
RADIOLIST	string	RADIOLIST=1001	The optionid corresponding to the value of the radio button list option selected by the user (or default value if not changed).	v2.1
SEARCHBOX, SEARCHBOX_n	string	SEARCHBOX=bar, SEARCHBOX_3=foo	The search term entered by the user. When multiple search boxes are defined (up to 3), they will be numbered <code>_2</code> , <code>_3</code> .	v2.1; v3.0 for <code>_n</code>
CUSTOM_ SLIDER, CUSTOM_ SLIDER_n	float/ integer	CUSTOM_ SLIDER=45.6	Optional: the value for the custom slider selected by the user. When multiple sliders are defined (up to 3), they will be numbered <code>_2</code> , <code>_3</code> .	v2.1; v3.0 for <code>_n</code>

continues

TABLE B-2 (continued)

PARAMETER	FORMAT	USAGE	DESCRIPTION	API VERSION
radius	integer	radius=2500	The value for the search radius selected by the user in the RANGE_SLIDER (unit in meters). From v3 on, radius is not mandatory and flexible radius can be used (the client will automatically adapt the radius to the POIs returned).	v2.1
CHECKBOXLIST	string	CHECKBOXLIST=4,7	The value(s) selected for the checkbox: multiple values get passed using comma-separated values.	v3.0
localCountryCode	string	localCountryCode=JP	Country code for the current country the device is located in, based on the country of the current mobile network provider the user is roaming in. If not available (for example, it's an iPhone or no SIM card is inserted), the app will try to determine the current country based on the GPS location (reverse geo-coding lookup). However, this may be unsuccessful and therefore incorrect.	v3.0
alt	integer	alt=325	The current altitude of the user. This is not always known on the client and will only be passed when there is a GPS fix.	v3.0
requestedPoiId	string	requestedPoiId=42114sddga94	Text string identifying the POI that triggered the request for this layer in Layar Stream (in Nearby view). If possible, this POI should be included in the response regardless of the filter settings.	v3.5

TABLE B-3: Authentication Parameters

PARAMETER	FORMAT	USAGE	DESCRIPTION	API VERSION
oauth_consumer_key	string (see API structure)	oauth_consumer_key=f43f3p214k3103	oauth consumer key: The developer can submit this key using the Layer Provisioning web site.	v2.1
oauth_signature_method	string	oauth_signature_method=HMAC-SHA1	This will always be HMAC-SHA1	v2.1
oauth_timestamp	integer	oauth_timestamp=1191242096	Timestamp for the request.	v2.1
oauth_nonce	string	oauth_nonce=k11o9940pd9333jh	Unique nonce to rule out replay attacks.	v2.1
oauth_version	string	oauth_version=1.0	oauth version used. Will always be 1.0	v2.1
oauth_signature	string	oauth_signature=tR3+Ty811MeYar/Fid0kMTYa/WM=	HMAC-SHA1 signature using the Signature Base String as text and the Consumer Secret (no token secret) as key. The consumer secret is submitted by the developer on the Layer Provisioning web site.	v2.1



When using authentication with your layer, all the parameters listed here are required.

C

junaio Support and Parameters

This appendix provides information you might find helpful when working with junaio.

SUPPORT CHANNELS

metaio has put together an excellent resource for developers building junaio channels. It is filled with useful hints, tips, How-Tos, publishing guides, and much more. Many of the samples used in this book have been written and developed by Frank Angerman of metaio. Frank is also very active in supporting developers via the official junaio support forum.

Here are some useful URLs that you will find useful in your junaio development:

- junaio development site: www.junaio.com/publisher/main
- junaio support forum: <http://groups.google.com/group/junaio-developer/>
- junaio support e-mail (e-mails are published to the junaio support forum and are not private): junaio-developer@googlegroups.com
- General junaio feedback: developer@junaio.com

JUNAIO CERTIFICATION PROGRAM

Once you have mastered junaio development and released several channels, you might consider joining the junaio Certified Developer Program. The junaio Certified Developer Programs gives you the opportunity to work on exiting AR projects and also presents you with the opportunity to:

- Become listed in the public directory of all junaio Certified Developers at www.junaio.com

- Receive business requests submitted to metaio by clients looking for professionals to develop AR channels on their behalf.
- Display the official junaio Certified Developer logo on your home page or web site.
- Be among the first to receive information about program updates, new features, as well as opportunities to become a beta tester.
- Receive information about upcoming updates, new features, and opportunities to beta-test major junaio releases.

Of course, before you can apply, you'll need to demonstrate your junaio development skills by releasing some high-quality channels. The conditions for entry into the program are:

- All applicants who can show, through their submitted channels, that they have the skills and capabilities to develop a channel of the desired quality and complexity can apply to become a junaio Certified Developer.
- Your ranking and level of certification as a junaio Certified Developer will be influenced by the type, number, complexity, and/or attractiveness of the created channels.

You will find more information about the program at www.junaio.com/publisher/certifieddeveloper

JUNAIO PARAMETERS

Table C-1 provides an overview of all parameters which can be returned with your xml and the consequences on the client application. Parameters are used in the following format:

- `<id>1</id>`
- `<interactionfeedback>click</interactionfeedback>`

TABLE C-1: Junaio Parameters

PARAMETER	FORMAT	DESCRIPTION	API VERSION
id	string	Unique alphanumeric identifier for your POI.	
interaction feedback	none/click	Currently there are two possible values: none: The client does not expect any feedback. A poi/event will be sent out upon clicking the POI, but no feedback will be processed. click: The client expects an update of POI information after a user clicks on a POI. The poi/event request will be triggered and the client awaits return values to process.	

PARAMETER	FORMAT	DESCRIPTION	API VERSION
removed	true	Attribute to be returned upon pois/event to remove a loaded POI. Set to true to remove a POI, which is displayed.	1.1
trackingurl	string (url)	Attribute of results. For junaio GLUE only! Path to the tracking XML.	
name	string	Sets the title of the POI.	
description	string	Sets a message for a POI (up to 1024 characters). It is mandatory for mime-type text/plain but optional otherwise.	
date	date/time (optional)	Sets a date when the POI was created. This tag can be empty.	
l	lat:Float long:Float alt:Float	Determines the location of a POI and needs to be given with latitude, longitude, altitude (WGS84). The string has to represent all three values. If in doubt, set altitude to "0".	
translation	x:Float y:Float z:Float	For junaio GLUE only. It sets the position of the poi/model. The parameter is needed with every POI to determine its location on the image. Note there should be note spaces in the translation tag.	
o	x:Float y:Float z:Float	Sets the orientation of a POI with euler angles in radians around x-, y- and z-Axis. Positive and negative values are possible. Rotates a model according to the values given. For non-3D models, only the z value effects the model.	
minaccuracy	integer	Determines a minimum accuracy the mobile client has to provide before the 3D model will be shown. The client application will not render any POIs where <code>minaccuracy</code> is lower than the current position accuracy of the mobile client.	
maxdistance	integer	The client application will not render any POIs where <code>minaccuracy</code> is lower than the current position accuracy of the mobile client. The client application will not render any POIs where <code>maxdistance</code> is lower than the current distance to the model.	

continues

TABLE C-1 (continued)

PARAMETER	FORMAT	DESCRIPTION	API VERSION
mime-type	string (valid mime-type)	Depending on the mime type, different representations will be chosen for the information. The following mime-types are supported: image/png image/jpeg video/quicktime ^{1,1} video/mp4 ^{1,1} audio/mp3 ^{1,1} text/plain model/md2 - not standardized model/obj - not standardized	
mainresource	string	Depending on the mimetype, this tag will hold a URL to an image, to an encrypted 3D model file or an audio or video file. For mime-type text/plain, this is optional.	
thumbnail	string (url)	Holds a URL to an image file which will be displayed with the POI in live view. Shows a thumbnail image with a POI in the live view. The thumbnail can be PNG or JPG format and should not exceed the size of 150 × 150 pixels.	
icon	string (url)	Holds a URL to an image file which will be displayed with the POI on the map. Shows an icon on the map for a given POI. The icon can be PNG or JPG format and should not exceed the size of 40 × 40 pixels.	1.1
force3d	boolean	Force the display of the 3D model. Only valid in combination with mime-type model/md2 or model/obj. Displays the 3D model, without showing the thumbnail first.	
relativetoscreen	x:Float y:Float	Displays the 3D model without showing the thumbnail first. Fixes a model to the screen (for example, for additional GUI elements). The lower-left corner of the screen is 0,0; the upper-right is 1,1.	

PARAMETER	FORMAT	DESCRIPTION	API VERSION
showcorrect perspective	boolean	For usability reason we adjust model distances in the world, so they can be seen by users. Set to “true” to avoid view correction.	
homepage	string (URL)	Provides a URL to be linked to a POI. If this parameter is set, an “Open Web” button will be displayed.	
route	Boolean	Determines whether a user should be able to be navigated to that POI. If this parameter is set, a Directions button will be displayed.	
s	Float	For 3D models only! Sets the scale of a model.	
behaviour-type	idle,click	For animated 3D models only! Behaviors determine possible animations of a 3D model. It sets the type of the animation. If type is set to “idle” the stated animation of the model will be started, after the model is done loading, “click” animations will be started upon clicking.	
behaviour-length	Float	For animated 3D models only! Behaviors determine possible animations of a 3D model. Sets the length of an animation in frames. Set length in frames to 0 means looping the animation.	
behaviour-node_id	string	For animated 3D models only! Behaviors determine possible animations of a 3D model. Sets the name of the animation that shall be started. This parameter determines the animation that will be started after loading the model.	
customization-name	string	Name of the customization.	
customization-type	video/sound/url	Type of value given in the customization.	
customization-node_id	click/idle/ onTargetDetect	Type of value given in the customization. <code>click</code> is valid for all customization types, <code>idle</code> for sounds only and <code>onTargetDetect</code> is only valid for videos and URLs.	

continues

TABLE C-1 (continued)

PARAMETER	FORMAT	DESCRIPTION	API VERSION
customization-value	string (url)	Provides a URL to a web site, video file or sound file). Depending on the node_id, at a certain point or user interaction, the value will be loaded or started.	
resources	string (url)	For 3D models only! Defines a URL to a texture or material file for a 3D model. The image file pointed to within this tag, will define the look of a 3D model.	

TROUBLESHOOTING GUIDE

Always validate your XML to ensure that it is working correctly. It is always a good idea to empty the junaio cache while testing. This section provides potential solutions to validation failures.

Failure of Validation Test 1 - Check Callback URL

Double-check whether the underscore of `_.htaccess` has been removed, leaving `.htaccess` on your server.

Make sure your URL really is available by typing in the URL in your browser. If your browser shows a page not found or similar, please correct your Callback URL. A blank page is usually a good sign.

Perhaps your server does not support `mod_rewrite`. Please add at the end of your Callback URL `"?path="`. For more information, look at the `Readme.pdf` in your `Getting Started Package`.

Failure of Validation Test 2 - Check pois/search

For certain servers, this test will output an unauthorized error.

Make sure you have edited the `config.php` and added your correct API key. (Your API key can be found in the junaio Dashboard)

Certain servers will not forward the authorization header. If your server supports `mod_rewrite`, please add the following line to your `.htaccess` in your html folder:

```
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization},L]
```

Make sure that your server time is correct. Requests to your server that are older than 15 minutes will be refused with a server unauthorized message. You can extend that time period in your `config.php`.

Try to comment out the authorization to your server (line 13 - 16 in your `index.php`) and the `require_once junaio.class.php` statement (line 9 in your `index.php`). If the error still occurs,

there is a server reason why HTTP Status Code 401 - Unauthorized is sent. Please ask your server provider for more information.



It is not recommended to comment out the authorization because there will be no authentication. Only junaio can access information on your server.

Test 3 - Check pois/search return value error

A request to the pois/search interface was sent. The return XML will be checked whether it is a valid XML file and whether it is empty or not. Possible warning or error causes can be invalid xml creation and/or a return of wrongly received parameters (be aware that this is a GET method).

*Expected return: HTTP Code 200 and valid XML.
Return: 200 - OK*

This error means you have an error in your XML file. Common causes include:

- Misspelled tag names
- Space between the latitude/longitude
- Invalid MIME Type

Failure of Validation Test 5 - Check pois/search

A request to the pois/event interface was sent. The return XML will be checked whether it is a valid XML file and whether it is empty or not. Possible warning or error causes can be invalid xml creation and/or a return or wrongly received parameters (be aware that this is a POST method).

*Request:
- your url
Expected return: HTTP Code 200 - OK and valid XML.
Return: 200 - OK
An invalid XML was returned. 3 error(s) and 0 warning(s).
Error: POI #0/[poi_id:string]: POI ID ([poi_id:string]) has invalid characters.
Error: POI #0/[poi_id:string]: Missing or invalid interactionfeedback (<poi interactionfeedback=??>). It must be set to "none" or "click"
Error: POI #0/[poi_id:string]: Unknown MIME Type.*

The application has been configured for event notifications but none are being used. (The default when using the sample packages.) The easiest way to fix this is to edit the callback function to remove a call to the event.php file. To prevent Test 5 failing, comment out the following line. The line will need to be uncommented when you want to include event notifications.

```
include '../src/event.php';
```

Status Codes

Your channel must provide HTTP status codes in every response. Table C-2 displays the status codes.

TABLE C-2: Status Codes

CODE	PHASE	RESULT
200	OK	Request successful.
400	Bad Request	Request invalid (for example, request did not pass every required parameters or a parameter was invalid).
401	Unauthorized	Request was unauthorized; the requested method requires authentication.
403	Forbidden	Request was authorized, but user is not allowed to execute this request.
404	Not Found	Request was authorized, but user is not allowed to execute this request.
500	Internal Server Error	An internal error on the server side occurred, due to extraordinary conditions.
503	Service Unavailable	API is temporarily not available (for example, due to maintenance).

INDEX

Symbols & Numbers

- ?>, php tag, 121, 137
- @ (at symbol), Twitter, 284–285
- ! (exclamation mark), constants, 277
- # (hashtags), 284
- 200, HTTP status code, 318
- 400, HTTP status code, 318
- 401, HTTP status code, 318
- 403, HTTP status code, 318
- 404, HTTP status code, 318
- 500, HTTP status code, 318
- 503, HTTP status code, 318

A

- <a href>, 93
- accelerometers, 23
- accuracy, 307
- action, 307
- actions, Layar, 135–144
 - adding, 140–142
 - audio, 142–143
 - fetching, 137–139
 - triggers, 143–144
 - video, 142–143
- actions table, Layar, 135–137
- ACTION_Table
 - contentType, 141
 - triggers, 144
- activityType, 136
 - icons, 141
 - iPhone, 141
- advertising
 - future, 295–296
 - marketing, 287
 - QR, 7
 - tags, 8
- affiliate programs, 253

- alt, 308
- altitude. *See* latitude, longitude, altitude
- Amazon, 253
- Android
 - <ar:name>, 89
 - Augmented ID, 9
 - browsers, 21–22
 - fiduciary markers, 5
 - Google Goggles, 18, 293
 - gravimetric AR, 5
 - junaio, 43
 - Layar Shortcut Tool, 179, 180
 - marketing, 292
 - QR, 7
 - Red Laser, 7
 - SDK, 180
 - SnapTell, 17
 - Space InvadAR, 16
 - video, 214
 - Wikitude World Browser, 25, 77–79, 99
- angle, 169
- animation
 - Idle-Animation, 246–247
 - junaio, 45, 225–229
 - OnClick-Animation, 246–247
- Apache, 190–192
- API endpoint URL, 106
- APIs
 - junaio, 189, 191–192
 - testing, 198–199
 - junaio Dashboard, 316
 - Layar, 146
 - Wikitude World Browser, 28
- app store, 25
- Apple App Store, 287–288
- applications
 - marketing, 19, 279–290
 - sale of, 287–288

AR Beatles Tour, 40–41
 AR Drone, 297–298
 archINFORM, 46
 ar:description, 302
 <ar:description>, 89
 ARGirl, 5
 ARML. *See* Augmented Reality Markup Language
 ar:name, 302
 <ar:name>, 89
 ar:provider, 302
 <ar:provider>, 86, 89–91, 92
 POIs, 95
 <ar:provider id=>, 95
 Art is all around you, 42
 ASP.NET, 190
 audio
 junaio, 211–212
 Layar, 38
 actions, 142–143
 <wikitude:attachment>, 94
 Augmented ID
 Android, 9
 facial recognition, 9
 Augmented Planet, 26, 30, 69
 Layar, 106
 logos, 253
 QR, 7
 Twitter, 284
 Augmented Reality Markup Language (ARML)
 Google Earth, 86
 KML, 86
 latitude/longitude, 86
 tags, 86–87
 Wikitude World Browser, 27, 85–100,
 301–303
 creating worlds, 87–91, 98–99
 email address, 93
 metadata, 90–91
 phone numbers, 93
 POIs, 86, 92–94, 99
 testing, 99
 thumbnails, 93
 XML Notepad, 96–97
 XML Notepad, 86, 88
 Wikitude World Browser, 96–97
 authentication, Layar, 38
 Author
 GLUE channels, 240, 243

 junaio Dashboard, 243
 Wikitude World Browser, 74
 autoTrigger, 136
 POIs, 137
 autoTriggerOnly, 136, 137, 144
 autoTriggerRange, 136, 143–144
 awards, 281

B

Bada, 25
 barcodes, 6–7
 baseURL, 166, 168
 Beatles, 40–41
 <behavior type>, 226
 idle, 227
 behaviour-length, 315
 behaviour-node_id, 315
 behaviour-type, 315
 bird's eye view, Layar, 37
 Blender, 230
 blogs, 283
 bloggers, marketing, 286
 BMW, 298–299
 Booking.com, 33
 books, 296
 branding, 9
 browsers, 5, 21–52
 accuracy of, 47–51
 anatomy of, 24–26
 Android, 21–22
 app store, 25
 discoverability, 10
 GPS, 22
 accuracy of, 47–50
 growth of, 23–24
 info bubbles, 25
 iPhone, 21–22
 junaio, 43–47
 Layar, 36–43
 maps, 25
 accuracy of, 50–51
 radar, 25
 range, 25
 Wikitude World Browser, 25–36
 BuildAR, 184–185
 BuildAR Dashboard, 184
 bullet points, 281

C

call to action, 281

callback functions

- junaio, 189
 - Getting Started Package, 194
 - validation, 197–198
- Layar, 116

Callback URL

- junaio, 194, 316
- junaio Dashboard, 269

Category, Layar Listing & indexing, 130

ChangetoIntLoc(), 119

channels. *See also* junaio

- GLUE, 237–247
 - Author, 240, 243
 - building from scratch, 250–253
 - creating, 248–250
 - Description, 244
 - E-mail, 244
 - images, 241–242
 - MD2 Model, 244
 - Phone, 244
 - Position on the Tracking Image, 245
 - Scale, 245
 - scaling, 245
 - testing, 242
 - Title, 240, 243
 - Tracking Image button, 241
 - Upload Video, 241
 - video, 241–242

Channel Categories, 194

Channel description, 193

Channel Name, 193

Channel refresh time, 194

Channel Selection list, 238

Channel Short Name, 193

Channel Visibility, 194

Checkbox, 155

CHECKBOXLIST, 308

checkboxlist, 160–161

CityGrid, 64

closeBiw, 136

color

- Layar POIs, 131–132
 - reference images, 15

Commission Junction, 253

comparison image, 14

compass, 49–50

Compass & Navigation AR, 39

Conquar, 41

constants, 277

content

- description, 280–281
- marketing, 280–290
- title, 280

content listing, 280

contentType, 136

- ACTION_Table, 141

contrast, 15

<coordinates>, 61, 83

- <wikitude:address>, 94

cosid, 253

countryCode, 306

Create Tracking XML, 251

Cribbins, Bernard, 292

cropping, reference images, 262

customization

- Layar, 128–134
 - databases, 119–121
 - tags, 8
- customization-name, 315
- customization-node_id, 315
- customization-type, 315
- customization-value, 316

CUSTOM_SLIDER, 307

CUSTOM_SLIDER_n, 307

D

dashboards. *See also* junaio Dashboard; Layar

- Dashboard
- BuildAR Dashboard, 184
- Hoppala Dashboard, 182–183
- Wikitude Dashboard, 70–71, 98

databases

- CityGrid, 64
- Foursquare, 65
- GeoNames, 64
- Hoovers, 65
- junaio, 276–277
- Layar, 108–128
 - creation of, 148–149
 - customization, 119–121
 - mylayar.php, 121–125
 - MySQL, 148–153
 - phpMyAdmin, 113–114

databases (*continued*)

- POIs, 112–114
- query string, 116–117
- SQL, 113
- tables, 109–111
- testing, 125–128
- POIs, 63–65
 - Layar, 112–114
- Trulia, 65
- Wikipedia, 63–64
- Yahoo, 65
- Yelp, 64
- Zvents, 65

databases_name, 114–115

databases_password, 114–115

databases_username, 114–115

Date, 313

DBpedia, 64

debugging, junaio 3D objects, 216–220

decimal notation, latitude/longitude, 56–57

dedicated XML files, 267–270

Delete, 196

Description

- content, 280–281
- GLUE channels, 244
- junaio Dashboard, 244
- Layar, 107
- Wikitude World Browser, 74, 89

description, 303, 313

<description>, 83, 92

Detail description, 130

dimensions

- junaio, 202
- Layar 2D objects, 173–176

diminishing reality, 294–295

Dinosaurs Alive (book), 296

discoverability, 10

</Document>, 82

E

Eclipse, 180–181

Edit, 196

E-mail

- GLUE channels, 244
- junaio Dashboard, 244
- Wikitude World Browser, 75

email address, Wikitude World Browser ARML, 93

Enable in-channel search, 194

eRightSoft, 259

errorCode, 117

Expiration dates, 130

F

Facebook, 256

- markerless AR, 9

facial recognition

- Augmented ID, 9
- Viewdle, 9–10

feedback, 282

fiduciary markers. *See* markers

filters

- Layar, 147–165
 - Checkbox, 155
 - connecting, 157–158
 - creating, 153–157
 - POIs, 173–174
 - query string, 158
 - Radiobutton List, 155
 - Slider, 154
 - SQL, 158–163
 - testing, 163
 - Textbox, 154
 - troubleshooting, 163–164
- Layar Dashboard, 147
- 500, HTTP status code, 318
- 503, HTTP status code, 318

Flickr

- browsers, 23
- Wikitude World Browser, 32–33

force3d, 314

<force3d>, 215, 217

foreach, 139

400, HTTP status code, 318

401, HTTP status code, 318

403, HTTP status code, 318

404, HTTP status code, 318

Foursquare, 65

Foursquare Venues, 31–32

FROM, 162

FTP

- junaio, 189
- Layar, 103

full, 166

future, 291–300
 advertising, 295–296
 books, 296
 diminishing reality, 294–295
 games, 296–298
 hardware, 298–299
 interactive TV, 293–294
 language translation, 292–293
 marketing, 291–292
 print, 296

G

Galileo, 293–294
 games
 future, 296–298
 Layar, 41–42
 natural feature tracking, 16
 GeoNames, 64
 GET, 136
 Get London Reading, 10–11
 Gethotspot, 162
 Gethotspots(), 117
 Gethotspots
 foreach, 139
 Layar 2D objects, 173
 GetJar, 179
 GetObject, 170–171
 Getting Started Package, junaio, 190–191
 callback functions, 194
 Gettransform, 171–172
 GIF, 94
 GLUE, 47
 channels, 237–247
 Author, 240, 243
 building from scratch, 250–253
 creating, 248–250
 Description, 244
 E-mail, 244
 images, 241–242
 MD2 Model, 244
 Phone, 244
 Position on the Tracking Image, 245
 Scale, 245
 scaling, 245
 testing, 242
 Title, 240, 243
 Tracking Image button, 241
 Upload Video, 241
 video, 241–242
 Idle-Animation, 246–247
 junaio, 236–253
 MP4, 237
 natural feature tracking, 236, 237–247
 OnClick-Animation, 246–247
 reference images, 237, 238
 3D objects, 237
 images, 242–247
 rotation, 245
 Rotation/Orientation in Degrees, 245
 Visibility, 238
 GoDaddy, 108
 Google, 4
 Google AdSense, 287
 Google Earth, 81–84
 ARML, 86
 junaio, 202
 KML, 71–74, 86
 POIs, 59–62, 73–74, 202
 XML, 59–61
 Google Goggles, 18, 293
 Google Local, 32
 Google Maps
 BuildAR, 184–185
 latitude, 201
 latitude/longitude, 58–59
 longitude, 201
 POIs, 58–59
 triggers, 143
 Wikitude World Browser, 27
 Google Marketplace, 179
 Google Search, 23
 Gorillaz, 292
 GPS
 browsers, 22
 accuracy of, 47–50
 junaio, 45
 POIs, 22
 Toozla, 22
 gravimetric AR, 5

H

hardware, 298–299
 Harvesine formula, 117
 hashtags, 284

- heads-up display (HUD), 298
- homepage, 315
- Homepage URL, 194
- Hoovers, 65
- Hoppala, 182–183
 - MySQL, 182
 - Woomba Mania, 182
- Hoppala Augmentation, 182–183
- Hoppala Dashboard, 182–183
- horizontal plane
 - info bubbles, 36
 - Layar, 37
- _.htaccess, 316
- HTTP status codes, 318
- http://m_layar.com/open/your_unique_layer_name, 178
- HUD. *See* heads-up display
- hyperlinks
 - Layar Intent, 179
 - natural feature tracking, 15
 - QR, 7

I

- IBM Seer, 28
- icon, 314
- <icon>, 203
- Icon, OBJECT_Table, 167
- icons
 - activityType, 141
 - POIs, Layar, 132–134
 - Wikitude World Browser, 75
- <icons>, 205
- ID, 136
 - OBJECT_Table, 166
 - TRANSFORM_Table, 169
- id, 312
- idle, 227
- Idle-Animation, 246
- images. *See also* reference images
 - GLUE channels, 241–242
 - junaio, 210–211
 - maps, natural feature tracking, 14
 - natural feature tracking, junaio, 261–263
 - recognition
 - junaio, 44
 - natural feature tracking, 14
 - 3D objects, GLUE, 242–247

- i-MetrO, 40
- import, junaio, 231–233
- info bubbles
 - browsers, 25
 - horizontal plane, 36
 - POIs, 25, 36–37, 132
- information bar, 25
- INSERT INTO, 132
- insideAR, 257
- interactionfeedback, 312
- interactive TV, 293–294
- iPhone
 - activityType, 141
 - ARGirl, 5
 - <ar:name>, 89
 - browsers, 21–22
 - compass, 49–50
 - fiduciary markers, 5
 - Get London Reading, 10–11
 - Google Goggles, 18, 293
 - junaio, 43, 195
 - marketing, 292
 - Red Laser, 7
 - SnapTell, 17
 - Translate, 293
 - Upsies, 16
 - video, 214
 - Wikitude World Browser, 25, 79–80, 99
- ISO standard, 7

J

- Java, 180
- JPEG, 94
- JSON, 103
- JSON, 120
- \$JsonResponse, 120
- junaio, 21, 43–47
 - advanced functionality, 267–277
 - animation, 45, 225–229
 - Apache, 190–192
 - API, 189, 191–192
 - testing, 198–199
 - ASP.NET, 190
 - audio, 211–212
 - callback functions, 189
 - Callback URL, 316
 - channels, 45–46, 189–233

- creating, 192–196
- databases, 276–277
- development choices, 45
- dimensions, 202
- FTP, 189
- Galileo, 293–294
- Getting Started Package, 190–191
 - callback functions, 194
- GLUE, 236–253
- Google Earth, 202
- GPS, 45
- images, 210–211
 - natural feature tracking, 261–263
 - recognition, 44
- iPhone, 195
- latitude, 189, 205
- LLA, 45
 - markers, 274–275
- longitude, 189, 205
- MD2, 225–229
 - importing, 231–233
- movie textures, 257–261
- MP3, 211–212
- MP4, 213, 259
- [name:string], 208–209
- natural feature tracking, 45, 236–253
- OBJ, 229–230
 - importing, 231–233
- parameters, 312–316
- PHP, 189, 190
- POIs, 45, 199–206
 - <icon>, 203
 - <icons>, 205
 - <name>, 203
 - <thumbnail>, 205
 - XML, 267–268
 - XML Notepad, 269–270
- proximity triggers, 45
- PSPad, 189
- publishing, 290
- requirements, 190
- SDKs, 44
- support, 311
- tags, 202
- testing, 47, 196–199
- troubleshooting, 316–318
- 3D objects, 45, 202, 214–225
 - accuracy of, 225

- creating, 230–233
- debugging, 216–220
 - <maxdistance>, 222–225
 - <minaccuracy>, 225
- rotating, 218–220
- scaling, 220–225
- testing, 232–233
- validation
 - failure, 317
 - testing, 197–198
- video, 212–214, 257–261
- visual search, 253–257
- XML, 202–203, 206–207
- junaio Certified Developer Program, 290, 311–312
- junaio Dashboard
 - API, 316
 - Author, 243
 - Callback URL, 269
 - Description, 244
 - E-mail, 244
 - MD2 Model, 244
 - Model Texture, 244
 - Phone, 244
 - Title, 243
 - Tracking Image, 244

K

- Keyhole Markup Language (KML)
 - ARML, 86
 - Google Earth, 86
 - limitations, 84
 - Wikitude World Browser, 27, 69–84
 - creating worlds, 74–77
 - Google Earth, 71–74, 81–84
 - XML, 61
- KML. *See* Keyhole Markup Language
- KML/KMZ file, 75

L

- L, 313
- label, 136
- lang, 307
- Language, 75
- language translation
 - future, 292–293
 - visual search, 17–18

- LastFM, 9
- Lat, 306
- lat, 116
- <lat>, 152
- latitude, 54–56
 - Google Maps, 201
 - junaio, 189, 205
 - search.php, 201
- latitude, longitude, altitude (LLA)
 - junaio, 45
 - markers
 - junaio, 274–275
 - QR, 274
- <latitude value>, 113
- latitude/longitude, 53–65
 - ARML, 86
 - decimal notation, 56–57
 - Google Maps, 58–59
 - Layar, 103
 - POIs, 57–65
 - Wikitude World Browser, 69
- Layar, 21
 - account creation, 104–105
 - actions, 135–144
 - adding, 140–142
 - audio, 142–143
 - fetching, 137–139
 - triggers, 143–144
 - video, 142–143
 - actions table, 135–137
 - API endpoint URL, 106
 - APIs, 146
 - audio, 38
 - Augmented Planet, 106
 - authentication, 38
 - bird's eye view, 37
 - browsers, 36–43
 - BuildAR, 184–185
 - callback functions, 116
 - customization, 128–134
 - databases, 108–128
 - creation of, 148–149
 - customization, 119–121
 - mylayar.php, 121–125
 - MySQL, 148–153
 - phpMyAdmin, 113–114
 - POIs, 112–114
 - query string, 116–117
 - SQL, 113
 - tables, 109–111
 - testing, 125–128
 - Description, 107
 - development choices, 37
 - filters, 147–165
 - Checkbox, 155
 - connecting, 157–158
 - creating, 153–157
 - POIs, 173–174
 - query string, 158
 - Radiobutton List, 155
 - Slider, 154
 - SQL, 158–163
 - testing, 163
 - Textbox, 154
 - troubleshooting, 163–164
 - FTP, 103
 - functionality, 38
 - games, 41–42
 - Google Marketplace, 179
 - Hoppala, 182–183
 - horizontal plane, 37
 - JSON, 103
 - latitude/longitude, 103
 - launching, 177–179
 - layers, 36, 38–42, 103–144
 - creating, 42–43, 105–107
 - sale of, 288–289
 - Listing & indexing, 128–130, 164
 - Category, 130
 - Detail description, 130
 - Expiration dates, 130
 - Layar Stream options, 130
 - Minimum API version, 130
 - Short description, 130
 - Tags, 130
 - Time to live, 130
 - Title, 130
 - Look & feel, 164
 - MySQL, 103, 108
 - parameters, 305–309
 - PHP, 103
 - POIs, 106
 - color, 131–132
 - icons, 132–134
 - MySQL, 133–134
 - parsing, 119
 - ports, 107
 - proximity triggers, 38

- PSPad, 103
- Publisher name, 106
- publishing, 289–290
- security, 118
- Skaloop, 185–186
- support, 305
- testing, 107, 158
- Title, 106
- tools, 177–186
- 2D objects, 165–176
 - dimensions, 173–176
 - Gethotspots, 173
 - GetObject, 170–171
 - Gettransform, 171–172
 - OBJECT_Table, 166–170
 - scaling, 176
- 3D objects, 38, 165–166
- URL, 105, 106
- Wikitude, 36
- layar://, 179
- Layar Dashboard, 104–106, 146
 - filters, 147
 - PHP, 147
 - testing, 126, 164
- Layar Intent, 178–179
- Layar Launcher Creator Tool, 180
 - nonsupport for, 182
- Layar Partner Network, 289
- Layar Shortcut Tool, 179–182
 - Android, 179
 - SDK, 180
 - development environment, 180
 - Eclipse, 180–181
 - Java, 180
 - USB, 180
- Layar Stream options, 130
- layar://your_unique_layer_name, 178
- layerName, 116, 306
- layers, Layar, 36, 38–42, 103–144
 - creating, 42–43, 105–107
 - sale of, 288–289
- <length>, 226
- Leopoldina, 275
- LinkedIn, 255–256
 - markerless AR, 9
- links. *See* hyperlinks
- Listing & indexing, Layar, 128–130, 164
 - Category, 130
 - Detail description, 130
 - Expiration dates, 130
 - Layar Stream options, 130
 - Minimum API version, 130
 - Short description, 130
 - Tags, 130
 - Time to live, 130
 - Title, 130
- LLA. *See* latitude, longitude, altitude
- Local Search Web Service, 65
- localCountryCode, 308
- localhost, 114–115
- location simulation, 80–81
- logos
 - Augmented Planet, 253
 - Wikitude World Browser, 75, 90
 - wikitude:logo, 302
 - <wikitude:logo>, 90
- Lon, 306
- lon, 116
- <lon>, 152
- longitude, 54. *See also* latitude, longitude, altitude; latitude/longitude
 - Google Maps, 201
 - junaio, 189, 205
 - search.php, 201
- <longitude value>, 113
- Look & feel, Layar, 164

M

- magazines, 7
- magnetometers, 23
- main resource, 314
- <mainresource>, 211, 215
 - junaio
 - audio, 211
 - video, 212
- maps
 - browsers, 25
 - accuracy of, 50–51
 - images, natural feature tracking, 14
 - POIs, 25
- markers, 5–6
 - LLA
 - junaio, 274–275
 - QR, 274
 - natural feature tracking, 13
 - markerless AR, 9–10
 - natural feature tracking, 13–14

marketing

- advertising, 287
- Android, 292
- applications, 19, 279–290
- blogs, 283
- bloggers, 286
- content, 280–290
- future, 291–292
- iPhone, 292
- newsletter, 286–287
- press releases, 285–286
- product page, 283
- Twitter, 283–285
- YouTube, 285

maxdistance, 313

<maxdistance>, 222–225

Maze, 42

MD2, junaio, 225–229

- importing, 231–233

MD2 Model, 244

meridians. *See* longitude

Message Central Layer, 42–43

metadata, 90–91

method, 136

Microsoft Paint, 130

Microsoft tags, 8–9. *See* tags

MimeType, 317

mime-type, 314

<mime-type>, 210–211, 215

- junaio channel audio, 211
- junaio channel video, 212

minaccuracy, 313

<minaccuracy>, 225

Minimum API version, 130

minutes, 56

Model Texture, 244

model/md2, 215

movie textures, 257–261

MP3, 211–212

MP4

- GLUE, 237
- junaio, 213, 259

MPEG, 94

My Channels, 203

mylayer.php, 121–125

MySQL

- Hoppala, 182
- Layar, 103, 108

- databases, 148–153
- POIs, 133–134
- localhost, 115
- URL, 115

N

name, 302, 313

<name>, 83, 92

- junaio channels POIs, 203

Name, Wikitude World Browser, 89

[name:string], 208–209

natural-feature tracking, 13–16

- comparison image, 14
- games, 16
- GLUE, 236, 237–247
- hyperlinks, 15
- images
 - junaio, 261–263
 - maps, 14
 - recognition, 14
- junaio, 45, 236–253
- markerless AR, 13–14
- markers, 13
- PHP, 236
- reference images, 14–15
- 3D objects, 15

New Channel, 192

newsletters, 286–287

Night Mail (poem), 292

<node_id>, 226

NTT DoCoMo, 299

O

o, 313

oauth_consumer_key, 309

oauth_nonce, 309

oauth_signature, 309

oauth_signature_method, 309

oauth_timestamp, 309

oauth_version, 309

OBJ, junaio, 229–230

- importing, 231–233

objects. *See also* 3D objects; 2D objects

- recognition, 4
- visual search, 18

- tracking
 - fiduciary markers, 5–6
 - gravimetric AR, 5
- OBJECT_Table, 166–170
- OCR. *See* optical character recognition
- OnClick-Animation, GLUE, 246
- online XML creator tool, 251
- online-converter.com, 242
- optical character recognition (OCR), 18, 292–293

P

- pageKey, 307
- parallels. *See* latitude
- params, 136
- parsing, LayaR POIs, 119
- PDF, <wikitude:attachment>, 94
- PDO::execute(), 118
- PDO::prepare(), 118
- PDOStatement, 162
- Phone
 - GLUE channels, 244
 - junaio Dashboard, 244
- phone numbers, Wikitude World Browser ARML, 93
- PHP
 - junaio, 189, 190
 - LayaR, 103
 - LayaR Dashboard, 147
 - natural feature tracking, 236
 - video, 257
 - XML, 267
- php tag, 121, 137
- phpMyAdmin, 113–114
- placemark, 302
- <Placemark>, 83, 88, 92
 - POIs, 95
 - XML Notepad, 94
- <Placement>, 86
- PNG
 - POIs, 132
 - URL, 211
 - <wikitude:attachment>, 94
- \$poi, 170
- <poi>, 205
- <poi id>, 205, 252
- poiID, 136
 - OBJECT_Table, 166
 - TRANSFORM_Table, 169
- Point/coordinates, 303
- points of interest (POIs), 9
 - action table, 135
 - <ar:provider>, 95
 - autoTrigger, 137
 - color, LayaR, 131–132
 - databases, 63–65
 - Google Earth, 59–62, 73–74, 202
 - GPS, 22
 - Hoppala Augmentation, 182–183
 - icons, LayaR, 132–134
 - info bubbles, 25, 36–37, 132
 - information bar, 25
 - junaio, 45, 199–206
 - <icon>, 203
 - <icons>, 205
 - <name>, 203
 - <thumbnail>, 205
 - XML, 267–268
 - XML Notepad, 269–270
 - latitude/longitude, 57–65
 - LayaR, 106
 - action table, 135
 - databases, 112–114
 - filters, 173–174
 - MySQL, 133–134
 - parsing, 119
 - map, 25
 - <Placemark>, 95
 - PNG, 132
 - Qype, 22–23
 - radar, 25
 - range, 25
 - Skaloop, 185–186
 - SQL, 112, 117–118
 - Wikipedia, 62–63
 - Wikitude, 22–23
 - Wikitude World Browser, 69
 - ARML, 86, 92–94, 99
 - XML, 24
- POIs. *See* points of interest
- pois_event.xml, 272–273
- pois_search.xml, 271–272
- POI_Table, 136
- Polar Rose, 9
- ports, LayaR, 107
- Position on the Tracking Image, 245
- POST, 136

press releases, 285–286
 prime meridian, 54
 print, 296
 product page, 283
 Provider URL, 74, 89
 proximity triggers
 junaio, 45
 Layar, 38
 PSPad
 junaio, 189
 Layar, 103
 Publisher name, 106
 publishing
 junaio, 290
 Layar, 289–290

Q

QR. *See* quick response codes
 query string
 Layar databases, 116–117
 Layar filters, 158
 quick response codes (QR)
 hyperlinks, 7
 Layar Intent, 179
 LLA markers, 274
 Woomba Mania, 179
 Qype
 POIs, 22–23
 Wikitude World Browser, 31

R

radar
 browsers, 25
 POIs, 25
 Radiobutton List, 155
 RADIOLIST, 307
 radiolist, 159–160
 radius, 116, 308
 range
 browsers, 25
 POIs, 25
 real-time, 4
 real-time tracking, gravimetric AR, 5
 Red Laser, 7
 Reduced, 166

reference images
 color, 15
 contrast, 15
 cropping, 262
 GLUE, 237, 238
 natural feature tracking, 14–15
 rel, 169
 relativetoscreen, 314
 Removed, 313
 requestedPoiId, 308
 <resource>, 215
 resources, 316
 </results>, 205, 252
 <resultstrackingurl>, 252
 rotation
 GLUE 3D objects, 245
 junaio 3D objects, 218–220
 Rotation/Orientation in Degrees, 245
 Route, 315

S

s, 315
 scale, 169
 Scale, GLUE channels, 245
 scaling
 GLUE channels, 245
 junaio 3D objects, 220–225
 Layar 2D objects, 176
 screenshots, 281
 SDKs
 Android, Layar Shortcut Tool, 180
 junaio, 44
 SEARCHBOX, 307
 searchbox, 158–159
 SEARCHBOX_n, 307
 search.php, 200–201
 seconds, 56
 security, Layar, 118
 shopping, visual search, 17
 Short description, Layar Listing & indexing,
 130
 showActivity, 136
 showcorrectperspective, 315
 Size, 167
 Skaloop, 185–186
 Slider, 154

- SMS, 7
 - SnapTell, 17
 - social media. *See also* Facebook; LinkedIn;
 - Twitter
 - markerless AR, 9
 - Soulbit7, 9
 - sounds. *See* audio
 - Space InvadAR, 16
 - Spots, 143
 - SQL. *See also* MySQL
 - injection attacks, 118
 - Layar databases, 113
 - Layar filters, 158–163
 - POIs, 117–118
 - State of the channel, 193
 - Submit, 196
 - SUPER, 259
 - Support LLA Marker, 274
 - Support Visual Search, 248
 - Supported Features, 194
 - Symbian
 - fiduciary markers, 5
 - Wikitude World Browser, 25
-
- T**
- \$tablename, 276
 - tables
 - actions table, Layar, 135–137
 - ACTION_Table
 - contentType, 141
 - triggers, 144
 - Layar databases, 109–111
 - OBJECT_Table, 166–170
 - POI_Table, 136
 - TRANSFORM_Table, 168–170
 - Tags
 - Layar Listing & indexing, 130
 - Wikitude World Browser, 74
 - tags
 - advertising, 8
 - ARML, 86–87
 - branding, 9
 - customization, 8
 - hashtags, 284
 - junaio, 202
 - php, 121, 137
 - Wikitude, 301–303
 - Wikitude World Browser, 90
 - testimonials, 281
 - testing
 - 3G connection, 214
 - GLUE channels, 242
 - junaio, 47, 196–199
 - API, 198–199
 - 3D objects, 232–233
 - validation, 197–198
 - Layar, 107, 158
 - databases, 125–128
 - filters, 163
 - Layar Dashboard, 164
 - triggers, 143
 - Wikitude World Browser, 77–80, 84
 - ARML, 99
 - Textbox, 154
 - 3D objects
 - GLUE, 237
 - images, 242–247
 - rotation, 245
 - Rotation/Orientation in Degrees, 245
 - images, GLUE, 242–247
 - junaio, 45, 202, 214–225
 - accuracy of, 225
 - creating, 230–233
 - debugging, 216–220
 - <maxdistance>, 222–225
 - <minaccuracy>, 225
 - rotation, 218–220
 - scaling, 220–225
 - testing, 232–233
 - Layar, 38, 165–166
 - natural feature tracking, 15
 - wireframe, 216
 - 3G connection, 214
 - 3G2, 259
 - 3GP, 259
 - 3PG, 260
 - thumbnail, 314
 - <thumbnail>, 205
 - thumbnails, 93, 194
 - wikitude:thumbnail, 303
 - <wikitude:thumbnail>, 92
 - Time to live, 130

Title

- content, 280
- GLUE channels, 240, 243
- junaio Dashboard, 243
- Layar, 106
 - Listing & indexing, 130
- Wikitude World Browser, 74
- Toozla, 22
- Total Immersion, 19
- Tracking Image
 - GLUE channels, 241, 244
 - junaio Dashboard, 244
- tracking images. *See* reference images
- Trackingurl, 313
- TRANSFORM_Table, 168–170
- Translate, 293
- Translation, 313
- triggers
 - ACTION_Table, 144
 - autoTrigger, 136
 - POIs, 137
 - autoTriggerOnly, 136, 137, 144
 - autoTriggerRange, 136, 143–144
 - Google Maps, 143
 - Layar actions, 143–144
 - proximity
 - junaio, 45
 - Layar, 38
 - testing, 143
- troubleshooting
 - junaio, 316–318
 - Layar layer filters, 163–164
- Trulia, 65
- Twitter
 - @ (at symbol), 284–285
 - Augmented Planet, 284
 - browsers, 23
 - markerless AR, 9
 - marketing, 283–285
- 2D objects, Layar, 165–176
 - dimensions, 173–176
 - Gethotspots, 173
 - GetObject, 170–171
 - Gettransform, 171–172
 - OBJECT_Table, 166–170
 - scaling, 176
- 200, HTTP status code, 318
- TXT, 94

U

- Upload Video, 241
- Upsies, 16
- uri, 136
- URL
 - API endpoint URL, 106
 - baseURL, 166, 168
 - Callback URL
 - junaio, 194, 316
 - junaio Dashboard, 269
 - Homepage URL, 194
 - Layar, 105, 106
 - Layar Intent, 178
 - MySQL, 115
 - PNG, 211
 - Provider URL, 74, 89
 - QR, 7
 - <resultstrackingurl>, 252
 - Trackingurl, 313
 - Wikitude World Browser ARML, 93
 - wikitude:providerUrl, 302
 - <wikitude:providerURL>, 89
 - wikitude:url, 303
 - <wikitude:url>, 92
- USB, 180
- userId, 306

V

- Validate it, 196
- validation, junaio
 - failure, 317
 - testing, 197–198
- \$value, 116, 305
- version, 306
- video
 - Android, 214
 - GLUE channels, 241–242
 - iPhone, 214
 - junaio, 212–214, 257–261
 - Layar actions, 142–143
 - PHP, 257
 - 3G connection, 214
 - <wikitude:attachment>, 94
- Viewdle, 9–10
- virtual reality, 4
- Visibility, GLUE, 238

visual search, 16–18
 Google, 4
 junaio, 253–257
 language translation, 17–18
 object recognition, 18
 shopping, 17
 Vuzix AR glasses, 298

W

Webcams.travel, 30
 WebServices, 27
 WHERE, 162
 Wikipedia
 browsers, 23
 databases, 63–64
 junaio, 46
 POIs, 62–63
 Wikitude World Browser, 29
 Wikitude, 21
 Layar, 36
 POIs, 22–23
 support, 301
 tags, 301–303
 Wikitude Dashboard, 70–71, 98
 Wikitude Drive, 26
 Wikitude World Browser, 25–36
 Android, 25, 77–79, 99
 APIs, 28
 ARML, 27, 85–100, 301–303
 creating worlds, 87–91, 98–99
 email address, 93
 metadata, 90–91
 phone numbers, 93
 POIs, 86, 92–94, 99
 testing, 99
 thumbnails, 93
 XML Notepad, 96–97
 Booking.com, 33
 creating worlds, 34–36
 Flickr, 32–33
 Foursquare Venues, 31–32
 Google Local, 32
 Google Maps, 27
 iPhone, 25, 79–80, 99
 KML, 27, 69–84
 creating worlds, 74–77
 Google Earth, 71–74, 81–84

latitude/longitude, 69
 location simulation, 80–81
 POIs, 69
 Qype, 31
 testing, 77–80, 84
 Webcams.travel, 30
 WebServices, 27
 Wikipedia, 29
 XML, 69
 XML Notepad, 69
 YouTube, 29–30
 wikitude:address, 303
 <wikitude:address>, 94
 wikitude:attachment, 303
 <wikitude:attachment>, 94
 <wikitude:email, 92
 wikitude:icon, 302
 wikitude:info, 303
 wikitude:logo, 302
 <wikitude:logo>, 90
 wikitude:phone, 303
 <wikitude:phone>, 92
 wikitude:providerUrl, 302
 <wikitude:providerUrl>, 89
 wikitude:tags, 302
 <wikitude:tags>, 90
 wikitude:thumbnail, 303
 <wikitude:thumbnail>, 92
 wikitude:url, 303
 <wikitude:url>, 92
 wiktude:email, 303
 wireframe, 3D objects, 216
 Wood, David, 255
 Woomba Mania, 39–40
 Hoppala, 182
 QR, 179
 Word Lens, 17–18
 WorkSnug, 46
 World Status, 75
 Wowwee Rovio Drone, 296–297

X

xmins, 302
 xmins:ar, 302
 xmins:Wikitude, 302
 xmins:wikitudeInternal, 302

XML

- ARML, 86
 - dedicated files, 267–270
 - Google Earth, 59–61
 - junaio, 202–203, 206–207
 - POIs, 267–268
 - KML, 61
 - online XML creator tool, 251
 - PHP, 267
 - POIs, 24
 - junaio, 267–268
 - Wikitude World Browser, 69
- XML Notepad, 83–84
- ARML, 86, 88
 - junaio POIs, 269–270
 - <Placemark>, 94
 - Wikitude World Browser, 69
 - ARML, 96–97

Y

- Yahoo, 65
- Yelp, 64
- <your layer name>, 152
- YouTube
 - marketing, 285
 - Wikitude World Browser, 29–30

Z

- Zend Framework, 190–191
- Zvents, 65