HACKS

Jared St. Jean

Kinect Hacks

Tips & Tools for Motion and Pattern Detection

O'REILLY®

Make:
makezine.com

Kinect Hacks

Jared St. Jean

Published by O'Reilly Media, Inc.

Preface

The way we interact with machines is always changing. As technology evolves, new ways of interacting with computers become available to us, one innovative breakthrough after the next. If we go back 10 years, RIM was just starting to implement phone capabilities into their line of Blackberry mobile devices. Now we have touch screens capable of delivering a full computing experience in the palm of our hands. Voice recognition software is finally becoming mainstream with the introduction of Siri on the iPhone 4S. We are rapidly entering an age in which being tethered to an accessory or peripheral, such as a mouse or keyboard, will be considered an archaic way of getting things done.

Touch screen interfaces are all the rage right now, but the next true evolution in human/computer interaction won't require you to physically touch a thing. You've seen it before in a number of sci-fi films: some guy in a futuristic get up is waving his hands around, barking orders at a computer that seamlessly tracks his movement and executes every command with flawless accuracy. The proper name for this type of computer interaction is called a Natural User Interface (NUI), which describes the ability to issue commands to a device using nothing more than your own natural body movements gestures, and voice. You'll no longer need to clean those germ-infested keyboards and touch screens pick up new batteries for your wireless mouse or gaming controller. The possibilities are truly endles and we're already starting to see deployments of NUI interactivity all over the world. Soon you'll see NUIs in store windows, bus stations, malls, as well as many other places that could benefit from adding natural human interaction to the process of selling and providing information.

One great piece of tech that has ushered in this new wave of creative development is the Microsoft Kinect. The Kinect has introduced itself to the masses at the best possible time. In its short existence it has proven to be a great option for those looking to spice up the way we work with computers. For starters, the Kinect is a fraction of the cost of most professional 3D depth sensing cameras. A wide assortment of programs allow you to use Kinect 3D depth data with Windows, Linux, and Mac machines. Whether you're an innovative professional interested in bleeding edge tech, or a part time hobbyist with a great new idea, the Kinect is easy to set up, it's cheap, and it's loaded with potential t change the way we interact with machines.

My intention from the start was to write a book that everyone could take something from. Whether you're completely new to the scene or have been tinkering around with the Kinect since its debut, there's something in here for everyone. Some of the hacks range from basic installation of tool sets o programs to detailed code write-ups relating to a wide range of available IDEs such as Processing an openFrameworks.

There are gesture-based solutions that work with a quite a few diverse applications in this book. For music related application, there is information on how to set up Ableton Live with Kinectar. If moca is your thing, there are tips and tricks for using Blender or Animata with NI mate. HTML5 and JavaScript integration is included using Zigfu's ZDK along with full-on 3D object and scene recreati written in C++ using PCL and OpenCV. If visual effects are your thing, be sure to check out all the diverse Kinect hacks using open source IDEs such as Processing and openFrameworks.

Whether you're looking for some fun projects to work on in your spare time or finding the perfect jump off spot to get started on your next big project, I believe you'll be able to find something in here that will appeal to you. Whatever is your proverbial cup of tea, there's definitely something in here fo everyone to enjoy working on.

The Story

Writing this book was one of the best experiences of my life, hands down. I had a pretty good idea of what was out there and how to set things up, but what I learned going through the amount of diverse hacks that would eventually compose the content of this book truly made me aware of the great work people are doing with the Kinect. I got to know many of the talented people involved in the scene fro around the world and learned an incredible amount in the process.

I started off by writing the guides necessary to get you started on your preferred OS. The great thing about the Kinect scene is that it was born out of the open source community. This ensured that driver would be released across all platforms. From there, various libraries were released allowing people t start playing around with Kinect data in other open source IDEs such as Processing, openFrameworks and Cinder. Getting the Kinect up and running and installing the proper libraries for your IDE of choice is the first step. After that, the real fun begins.

After completing the first two chapters, things really started to get interesting. A wide assortment of new tools became available to play around with. For example, working with Chris Vik's amazing Kinectar app, which allows you to be your very own hands-free composer and musician, and Stephan Howell's Kinect2Scratch, which uses the Scratch IDE, was an unbelievable experience. I had a blast settings things up and then playing around with the end results. I can honestly tell you that I loved working on this book from beginning to end.

Acknowledgments

A big huge juicy thank you to everyone at O'Reilly Media for taking a chance on this lowly blogger. I never thought in my wildest dreams that putting a bit of work into a niche scene like Kinect hacking would result in something like this. In my opinion, they took a huge chance with me and I hope I was able to deliver a great product to add to their impeccable track record in the world of technical writing.

Even though O'Reilly Media made this thing you're holding in your hands a reality, I would have never had the opportunity if it wasn't for Greg Mackenzie and the Dashhacks network. His passion fo hacking and modding electronics eventually produced the best network of hacking sites on the Internet. The journey seemed short in hindsight, but what I got out of it has changed my life forever. Mad props for letting me do my own thing, GregCube.

Last but certainly not least, the amazing folks who took time out of their lives to develop such incredible projects using Kinect's 3D depth sensing capabilities. You are an inspiration and also responsible for making us think beyond the keyboard and mouse, ushering in a new age of interacting with machines. In no particular order, here is my "I'm not worthy!" list of incredible contributors to

both this book and the scene in general. Without the efforts of each and all of you—I can't complete this sentence because I don't even want to think about it!

Thanks to Daniel Shiffman, Joshua Blake, Kyle McDonald, Nicolas Burrus, Shawn Wallace, Brian Jepson, Ryan Challinor, Chris Vik, Julius Tuomisto, Peter Nash, RJ Durran, Ning Ma, Kris Temmerman, Mike Newell, Andrew Berg, Felix Endres, Stephen Howell, Jordi Llobet Torrens, Anna Fusté Lleixà, Jeremy Archer, Taylor Veltrop, Takashi Nishibayashi, Daniel Ho, Javier Graciá Carpio, Theodore Blackman, Amir Hirsch, and Stefan Stegmueller.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`

Shows commands or other text that should be typed literally by the user.

`Constant width italic`

Shows text that should be replaced with user-supplied values or by values determined by context.

> **TIP**
>
> These lines signify a tip, suggestion, warning, caution, or general note.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "Kinect Hacks by Jared St. Jean (O'Reilly). Copyright 2013 Jared St. Jean, 978-1-449-31520-7."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

Safari® Books Online

NOTE

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference book and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at http://my.safaribooksonline.com.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

800-998-9938 (in the United States or Canada)

707-829-0515 (international or local)

707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://oreil.ly/kinect_hacks

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, courses, conferences, and news, see our website at http://www.oreilly.com.

Find us on Facebook: http://facebook.com/oreilly

Follow us on Twitter: http://twitter.com/oreillymedia

Watch us on YouTube: http://www.youtube.com/oreillymedia

Contributors

Anna Fusté Lleixà (hi@annafuste.com) is a Multimedia Engineering student at La Salle Barcelona

(Ramon Llull University), an Audiovisual Communication Graduate (Pompeu Fabra University), and an Intern at la Salle HCI Department. Find out more at http://www.annafuste.com.

Peter Nash is a creative technologist based in London. He thrives on variety, including taking on hardcore academic challenges (he achieved "top of the year" for researching Nanotechnology in College) and experimenting with interactive installations, programming mobile apps and web, and working with big high street retailers to keep with it. He is currently pioneering a start-up dedicated t making the lives of smartphone app developers and designers easier. Peter contributed Draw in the A with Processing.

Reach him via LinkedIn, Twitter, or his personal website.

RJ Duran is a Computational Artist & Engineer exploring the inherent properties and aesthetics of biological and emergent pattern formation through Media Arts & Technology. His fascination with natural systems, engineering, traditional art forms such as music and architecture, and philosophy inform and guide his artistic pursuits. His mission as a creator is to understand the "deep complexity" embedded within systems in order to develop engaging, interactive and experiential audio, visual, an physical tools and experiences for education, exploration, and enlightenment.

He holds a BS in Electrical Engineering from Colorado State University, a certificate in Audio Engineering from The Conservatory of Recording Arts & Sciences, and a certificate in Digital Cultur & Creative Technology from Boulder Digital Works. He is currently a Graduate Student Researcher i the Media Arts & Technology program at the University of California Santa Barbara. RJ contributed Create a Gravitational Particle Effect. He can be reached at his website or via email at rjduranjr@gmail.com.

Born in China, **Ning Ma** now lives in Germany and has received his Master's Degree in Computational Engineering in Faculty Civil and Environmental Engineering from Ruhr-Bochum University. He is currently a PhD candidate.

Apart from his studies, he is very interested in digital graphics. He is self-taught to work with design tools, playing around with images and animations. His recent experiments with Kinect extends the field of his hobby. He loves actually interacting with graphics by programming is fun and exciting. Ning contributed Make Objects Follow a Tracked User's Hand and Dynamically Change Colors and Background Perspective.

Mike Newell works for Goodby Silverstein & Partners. Mike contributed Create a Live 3D Rendered Effect with Processing.

He can be reached via his personal website, via email at mike@iwearshorts.com, on Twitter at @newshorts, or on Google+.

Stephen Howell is a Computing Lecturer in the Computing Dept. at the Institute of Technology Tallaght, Dublin. He lectures on Kinect software development in modules called Interactive Media Design & Interactive Media Development. Visit his website at http://scratch.saorog.com. Stephen contributed Create a Basic Tennis Game Using Scratch, Build a Defender-Style Shooter Game with Scratch, and Build a Kinect Space Shooter Game in Processing.

Stephen can be contacted by email at stephen.r.howell@gmail.com.

Jordi Llobet (jordi.llobet@me.com) is a Multimedia Engineering student at La Salle Barcelona (Ramon Llull University). He's also a team member at the Interactive Prototype Design Group (La Salle HCI Department). Find out more at http://www.jordillobet.es. Jordi contributed Build a Processing Fridge Magnet Game.

Jeremy Archer (https://github.com/fatlotus) is a student at the University of Chicago. He works on various open-source projects, primarily for scientific computing, at Chicago's Computation Institute (http://ci.uchicago.edu/). He is also co-founder of Carbonless Community, a green energy software company based in Chicago. Jeremy contributed Control a Robotic Arm.

Taylor Veltrop grew up near Chicago surrounded by Lego, Meccano, and computers. His first humanoid robot was made from wood (mostly by his father) at the age of 6. He studied Japanese and computer science at the engineering school in the University of Colorado and eventually found himself in Japan chasing robot dreams. There he came across Kinect-based teleoperation while exploring solutions for robot hand-eye coordination and navigation. Taylor currently resides in Paris working with Aldebaran Robotics. Taylor contributed Control a Robot's Navigation and Use Candescent NUI.

Taylor can be reached via email at taylor@veltrop.com, at his personal website, or on YouTube.

Stefan Stegmueller is a professional Software Architect/Developer from Switzerland (Master of Science in Computer Science). Candescent.ch is his private development project. I runs candescent.c to publish programs which he writes mainly for himself but could be useful for others too.

One of his current fields of interest is natural user interaction. Stefan believes that after the Comman Line Interface (CLI) and the Graphical User Interface (GUI) the Natural User Interface (NUI) is the next big thing in human-machine interaction. Stephan contributed Use Kinect Jesture for Mac.

Stefan can be reached via email at info@candescent.ch, at his website, blog, and on Twitter at @CandescentCH.

Winect is a final year project by **Daniel Ho** in National University of Singapore, School of Computing. Since graduation, he had continued this project during his free time at http://ixorastudios.com/software/winect/. Daniel contributed Use Winect for Windows.

Kris Temmerman contributed Create a Hairy Effect: Create a Hairy Effect.

Andrew Berg contributed Create Hand Tracking Trails Using Cinder: Create Hand Tracking Trails Using Cinder.

Chris Vik contributed Install Kinectar: Install Kinectar; Map Parameters in Ableton Live with Kinectar: Map Parameters in Ableton Live With Kinectar; Set Up a Drum Kit with Kinectar: Set Up Drum Kit with Kinectar; and Create a Dubstep Wobble Bassline with Kinectar: Create a Dubstep Wobble Bassline with Kinectar.

Javier Graciá Carpio contributed Use Processing to Create a 3D Scanner with Mesh Viewer: Use Processing to Create a 3D Scanner with Mesh Viewer.

Felix Endres contributed Set Up PCL and OpenCV: Set Up PCL and OpenCV; Display a Colored Point Cloud: Display a Colored Point Cloud; Use Features to Track Camera Image Motion: Use Features to Track Camera Image Motion; Fuse Point Clouds into a Consistent 3D Model: Fuse Point Clouds into a Consistent 3D Model; Add Convenience Functionality to a 3D Model: Add Convenienc Functionality to 3D Model; and Next Steps: SLAM, OctoMaps, Surface Reconstruction: Next Steps; SLAM, OctoMaps, Surface Reconstruction.

Takashi Nishibayashi contributed Use Linux Gesture-Based Mouse Control: Use Kinect Justure for Mac.

Chapter 1. Getting Up and Running

The Kinect was designed and marketed as an accessory to the Xbox 360 with the intention of breathi new life into Microsoft's gaming division without the company having to release a brand new consol The multimillion dollar idea was to offer a new and exciting way for Xbox 360 owners to play video games. Microsoft's marketing department hit the nail on the head with the catchy tag line "You are th controller"; in other words, the Kinect offers a natural user interface free of any cables, chargers, or controllers. When you move your hand, the game or dashboard interface responds accordingly. The Kinect was launched on November 4th in North America at a retail price of $150 and allowed users t simply plug the device into their Xbox 360 and start playing right away.

What Microsoft may not have anticipated was that its affordable gaming accessory was capable of many great things once placed in the hands of creative developers around the world. A mere six days after its launch, the Kinect was hacked and people started experimenting with it, shortly thereafter discovering what they could do with this affordable depth-sensing camera.

To truly get an idea of the amazing things you can do with the Kinect, you need to understand what it actually is and what it can do.

Hack #1. How the Kinect Works

The Kinect is a pretty impressive piece of tech. Sporting an RGB camera, multiarray microphones, and a depth sensor capable of full-body 3D motion capture along with facial and voice recognition capabilities, this video game accessory packs a serious punch. The Kinect uses both hardware and software simultaneously to capture and interpret depth data on the fly. The software, developed by a company called PrimeSense, is able to recognize humans based on their skeletal structure (Figure 1-1). People have the distinct advantage of, well, standing on two legs and having arms. This is how the Kinect is able to determine that a human being is present. It can then focus on capturing data from th movements of that recognized "player." Unless you have apes frequently interrupting your game time by busting into your living room, this is a pretty rock-solid means of isolating and tracking a human skeleton in the room and disregarding everything else.

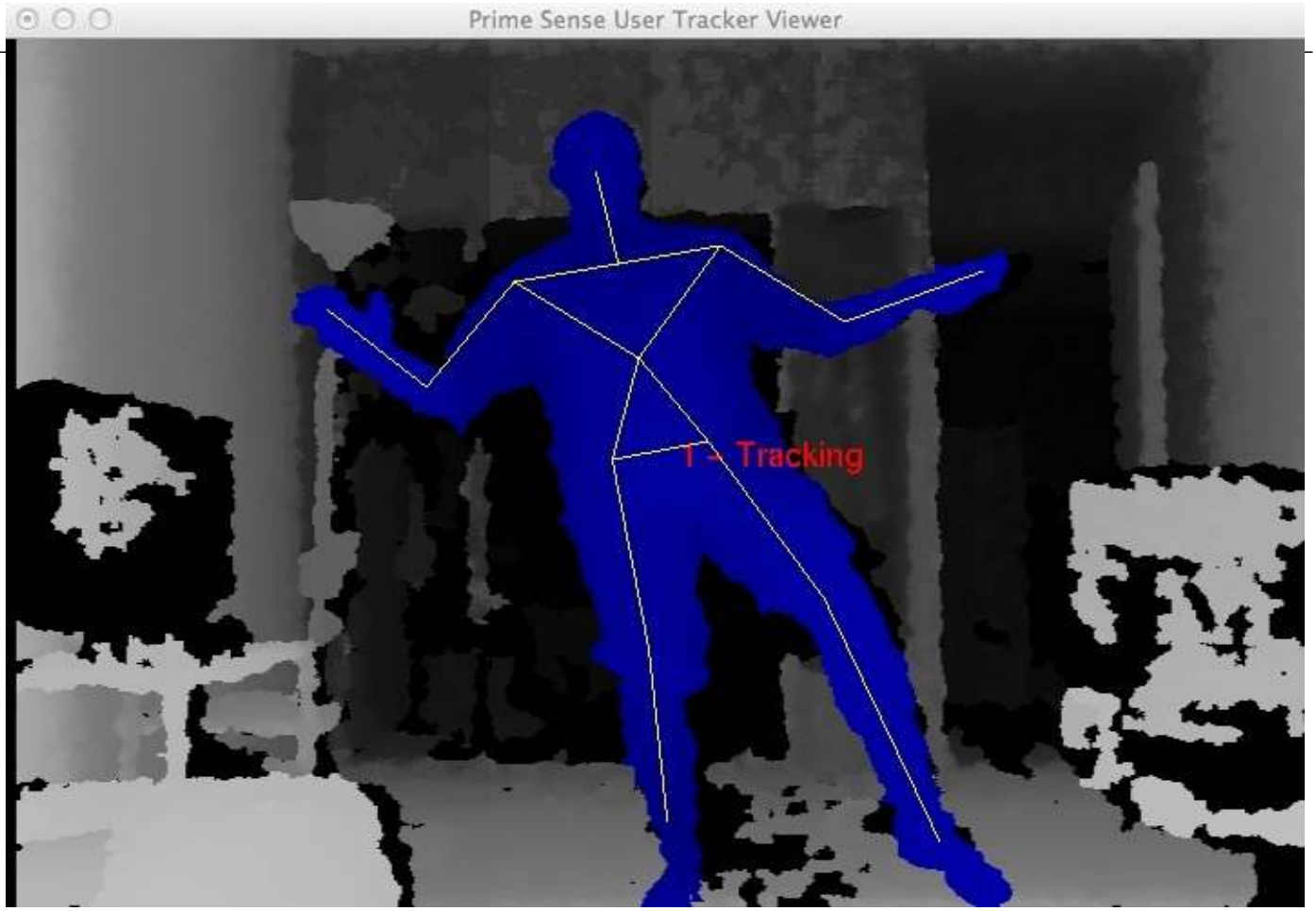

Figure 1-1. PrimeSense skeletal tracking viewer

At its core, the Microsoft Kinect's true innovation and technical prowess lie within its depth-sensing technology. An infrared (IR) cluster of light (also referred to as a point cloud) is produced and spread out across a room, carrying with it encoded information in the form of varying light patterns—pictur hundreds of little laser-point–size dots covering the room. The IR beams that are being emitted are undetectable to the naked eye, but when viewed in the dark through night vision goggles, they are on brilliant display (Figure 1-2).

Point cloud data is accurate only at a distance of approximately 1.2 to 3.5 m (3.9 to 11 ft) but within this range, data is collected and sent back, relaying information such as the distance of any detected objects based on any deformations in the IR light patterns. Two onboard complementary metal-oxide semiconductor (CMOS) cameras are used in individual capacities to further analyze any data that's been collected from the IR light patterns. The RGB camera collects 30 frames per second of actual real-time events at a 640x480 resolution, while the other handles the 3D depth imaging. An onboard processor then renders the collected data in the form of 3D images so everything is wrapped up in a nice little package—and all of this for a measly $150. If only it did things other than rate my pathetic dance moves or let me pet jungle cats. If only….

Figure 1-2. IR light pattern using night vision

Hack #2. How the Kinect Was Hacked

I think it's a pretty safe bet to say that the Kinect would have been hacked at some point or another. I potential was just too great to have people sit around idly waiting for Microsoft to release its own software development kit (SDK) for the device. For all we know, it may never have even released its SDK were it not for the efforts of the OpenKinect community. Actually, I'm getting a little bit ahead of myself. Let's back it up to the fledgling days of the Kinect's launch and talk about a little bounty put forth by the good folks over at Adafruit.

The bounty, called the X prize, was to be awarded to the first person able to produce open source drivers for the Kinect. The drivers could be functional on any operating system and had to be accompanied by an application that demonstrated their functionality by displaying a split window of the depth data and the RGB camera. Microsoft caught wind of this little competition, and its initial response was that of any company undergoing a major hardware launch. Since the Xbox 360 was hacked quite some time ago, this knee-jerk response was issued from a Microsoft PR rep:

> *Microsoft does not condone the modification of its products… With Kinect, Microsoft built in numerous hardware and software safeguards designed to reduce the chances of product tampering. Microsoft will continue to make advances in these types of safeguards and work closely with law enforcement and product safety groups to keep Kinect tamper-resistant.*

Most would cower in fear thinking of what Microsoft, with its deep pockets, could do to keep this sor of thing from catching the world's attention, but what actually happened was the complete opposite. Microsoft's response enticed Adafruit to increase the bounty of its X prize, not shy away. Fast-forwa six days, and a winner was announced! Héctor Martin took the prize and dubbed his open source Kinect drivers "libfreenect." With this, a new generation of open source Kinect development was under way, and excited developers interested in working with Kinect wasted no time getting involved The OpenKinect project was born, and a community of programmers and developers began building futuristic applications using the Kinect's depth-sensing capabilities.

On December 9, 2010, PrimeSense embraced the work being put forth by the open source community with the release of OpenNI and NITE. Things were really starting to get cooking at this point. With th backing of the developers responsible for the Kinect hardware, a wave of hacks began gaining a lot o attention all around the world. Wrappers started pouring in at this point, enabling people to toy aroun with their favorite programming language or framework in order to start experimenting with their ow Kinect-related projects.

If ever there were a case for presenting the term "hacker" in a positive light, the efforts of the

OpenKinect community would be it.

Hack #3. Choose a Framework and Driver

This section is all about getting your Kinect set up on your computer so you can get started on that next game changer. We'll be covering the installation process with a few different methods to get up and running. Step-by-step walkthroughs are available for Windows 7, Mac OS X, and Ubuntu that cover installing the libfreenect drivers, OpenNI, NITE, and SensorKinect, along with the Kinect SDK For the sake of consistency, the guides formulated in this section have been successfully tested on a Mac running OS X 10.7, Windows 7, and Ubuntu 11.10.

Although having a wide variety of options can be a great thing, at times, the potential of getting mixe up in things can always rear its nasty little head. So we'll focus on two main options. As I mentioned earlier, libfreenect was the first set of open source–compatible Kinect drivers made available to the public. It is maintained by the Open Kinect community of developers and can be downloaded at https://github.com/OpenKinect/libfreenect.

PrimeSense, the developers behind the Kinect's depth-sensing technology, released OpenNI, a derivative of the open source LGPL PrimeSense code. You'll need to also install the avin2 SensorKinect module, built specifically for the Kinect based on code from the PrimeSense sensor driver, if you want to use OpenNI.

So what we're left with is two completely viable options available for us to use when working with Kinect depth data. The question now is, which one is right for you?

Well, for starters, if you plan on releasing your program in some commercial form, OpenNI and libfreenect are both fine to use, so you can scratch that off of your list. If you're interested in motor control, however, go with libfreenect; OpenNI with SensorKinect does not support it. For higher-leve NUI support, OpenNI has its NITE middleware integration available. libfreenect has a much more complicated installation process, so if you're unfamiliar with compilers and are more comfortable dealing with binary installers, OpenNI/NITE involves a much less complicated installation process. Whichever route you decide to take, you'll be in good hands. Both organizations have a huge, supportive community with great online forums.

There are usually two different methods of installing the open source drivers required to capture data from the Kinect sensor. You can compile the latest builds from scratch, or just install the binaries if available. We'll cover both methods in case one or the other just doesn't quite work out the way it wa supposed to. It's always good to have a plan B, right?

Install OpenNI, SensorKinect, and NITE for Windows 7

For the sake of simplicity, we'll start things off the nice and easy way. Since PrimeSense was kind enough to release its own binary installers, we'll begin with this route.

TIP

The title of this hack is a bit deceiving. You can apply these guidelines to a Windows XP or Vista machine, if that's all you have. As I mentioned before, however, I used a Windows 7 machine for all of these guides.

Download the OpenNI package installer from http://www.openni.org/Downloads/OpenNIModules.aspx.

Select OpenNI Packages from the first drop-down menu. From the next drop-down, select Stable, and then choose PrimeSense Package Stable Build for Windows-Development Edition.

TIP

Be sure to download the appropriate 32- or 64-bit versions depending on your system. If you don't know what you're running, go to Control Panel → System and Security → System, and see System Type.

Launch the executable file to begin installing OpenNI and NITE.

Download the Kinect sensor drivers from https://github.com/avin2/SensorKinect.

Once you've downloaded the package, extract the contents and install the driver mod by executing the binary located in the Bin folder. Be sure to choose the right 64- or 32-bit installer.

NOTE

When prompted, allow the unsigned driver from PrimeSense during the installation.

To see if everything was installed correctly, plug the Kinect into a USB port and make sure it is plugged into a power source as well. Go to your Start Menu → All Programs → OpenNI → NiViewer. After a few seconds, a window should open showing you a simple depth view from the Kinect sensor (Figure 1-3).

Figure 1-3. A NiSimpleViewer demo

To see if the NITE samples are working, you'll need to copy all of the sample XML files from the PrimeSense/NITE/Data folder to the PrimeSense/Sensor/Data folder. Once they've been copied over, go to Start → All Programs → PrimeSense → NITE 64 bit (for those who installed the 64-bit version → Samples → Sample-Box64 (Figure 1-4).

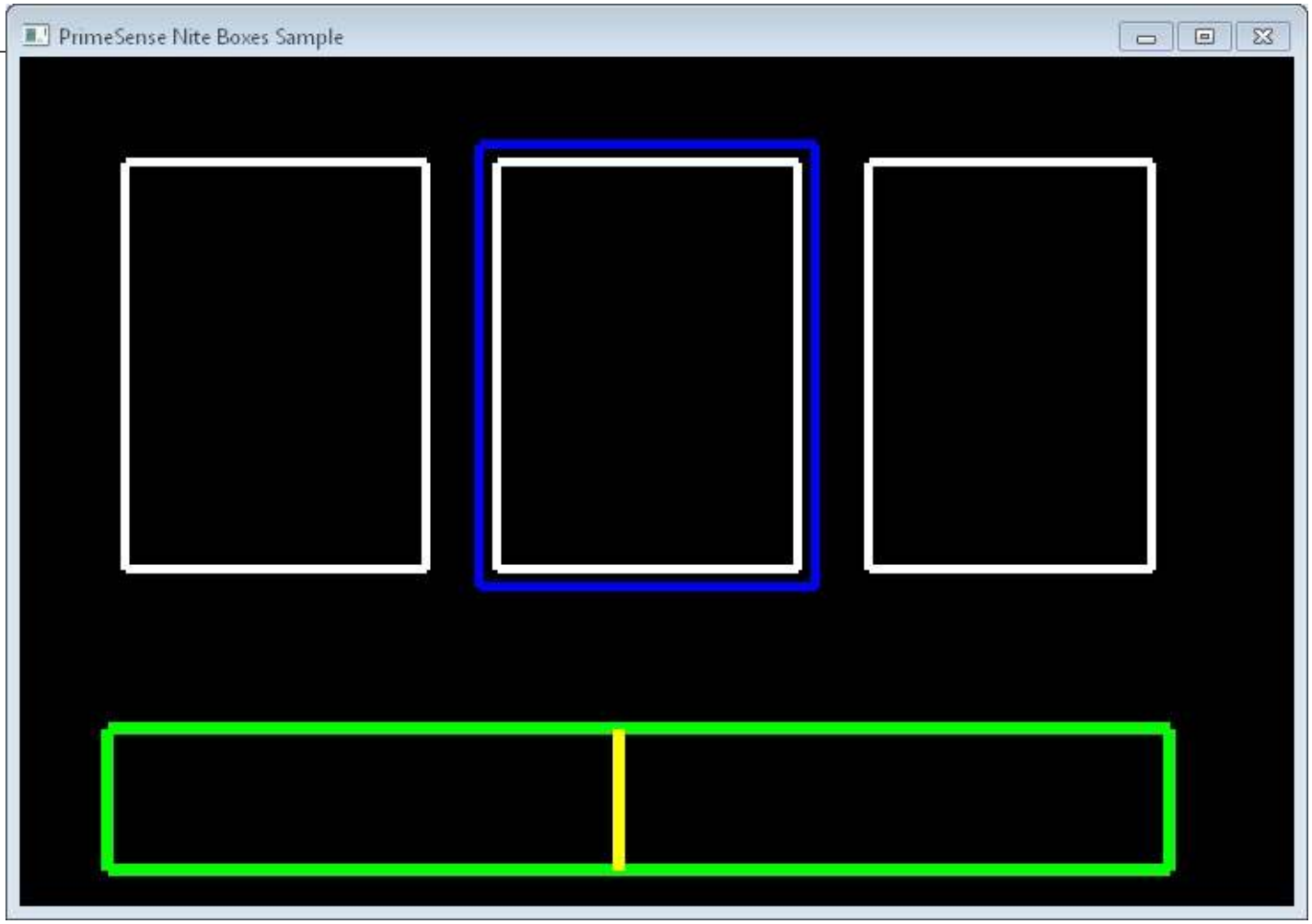

Figure 1-4. PrimeSense NITE SampleBox

Install OpenNI, NITE, and SensorKinect for OS X

Before we get started, you'll need the following applications installed and configured:

- Xcode
- CMake
- MacPorts
- Git

You'll need to install a few library dependencies as well. Using MacPorts, install libtool and libusb-devel + universal. Open up your Terminal application (Applications → Utilities → Terminal) and run the following command:

```
sudo port install libtool
```

Restart your Mac. Open up your Terminal again and install libusb-devel + universal as follows:

```
sudo port install libusb-devel +universal
```

Restart your computer once again. Create a directory in which you'll store all of our installers. We'll create a Kinect folder in the Home directory.

```
cd ~/
mkdir Kinect
cd Kinect
```

Download the latest version of OpenNI from its GitHub repository.

```
sudo git clone https://github.com/OpenNI/OpenNI.git
```

Change the working directory to OpenNI/Platform/Linux-x86/CreateRedist and run RedistMaker.

```
sudo ./RedistMaker
```

Once the build has finished, back out of that directory and make your current working directory Bin/Release.

```
cd ../Bin/Release
```

Run one of the examples to make sure everything is working properly.

```
sudo ./Sample-NiUserTracker
```

Next, we'll need to install the avin2 SensorKinect driver mod to retrieve data captured by the Kinect. Navigate back to the Kinect directory and use Git to download the latest version of SensorKinect.

```
cd ~/Kinect
sudo git clone https://github.com/avin2/SensorKinect.git
```

Change your working directory to ~/Kinect/SensorKinect/Bin and extract the contents of SensorKinect-Bin-MacOSX-v*..*.tar.bz2.

Navigate into the newly extracted directory and run the installer script.

```
sudo ./install.sh
```

Last but not least, we'll want to install NITE. Download the latest unstable build of NITE from http://www.openni.org and place it in your Kinect directory. (Go to http://openni.org/Downloads/OpenNIModules.aspx and select OpenNI Compliant Middleware Binaries → Unstable Release → PrimeSense NITE Mac OSX.)

I will refer to the NITE root directory as NITE to keep things consistent. If you'd like to rename it, by all means do so.

```
mv nite-bin-macosx-v*.*.*.* NITE
```

Extract the contents of the file, change the working directory to the NITE folder, and run the install script.

```
sudo ./install.sh
```

You will then be prompted to enter the PrimeSense license key, which is `0KOIk2JeIBYClPWVnMoRKn5cdY4=`.

We're almost done! If you want to run some of the examples, you'll need to move the sample XML files from the SensorKinect/Data directory over to NITE/Data. You can do this in the Finder or in a Terminal.

Cruise on over to the Samples directory and try one out for size! (See Figure 1-5.)

```
cd ~/Kinect/NITE/Samples/Bin/Release
sudo ./Sample-PointViewer
```

Figure 1-5. PrimeSense NiUserTracker sample

Install OpenNI, NITE, and SensorKinect for Ubuntu

Getting your Kinect working with Ubuntu is pretty straightforward. It will definitely help if you're comfortable working on a command line. We'll be using apt to download everything except the NITE middleware package, so things should go smoothly.

First, as always, we'll need to ensure that all of the required libraries and other dependencies are installed before we'll be able to download the required SDK, drivers, and middleware from the usual spots.

We'll install OpenNI first.

WARNING

The order in which we install each component is important, so try not to veer off course and install NITE or SensorKinect before OpenNI!

There are a few libraries that we need to install, so we'll tackle those before moving on. I've listed th requirement names along with the official sites that host the downloads. Instead of installing them al manually, apt commands you to install everything in one swift motion.

Open up your Terminal application and enter the following commands, allowing time for each install to finish successfully before moving on to the next one.

```
sudo apt-get install git-core cmake g++ python \
freeglut3-dev pkg-config build-essential \
libxmu-dev libxi-dev libusb-1.0-0-dev \
doxygen graphviz
```

Don't be surprised if this takes a lifetime to complete, especially if you're using a fresh Ubuntu installation.

You'll also need to install the Java Development Kit. I used a separate session of apt since I needed t build a repository.

```
sudo add-apt-repository "deb http://archive.canonical.com/ lucid partner"
sudo apt-get update
sudo apt-get install sun-java6-jdk
```

Hopefully everything installs properly without your having to cruise the backwaters of the Internet in search of dependencies for those dependencies. Always remember, in times of trouble, Copy, Google and Paste are your best friends in the whole wide Web. Moving on, create a Kinect directory whereve you'd like. I chose the Home directory.

```
mkdir ~/Kinect
cd ~/Kinect
```

Download OpenNI using Git and make sure you're dealing with the unstable version.

```
sudo git clone https://github.com/OpenNI/OpenNI.git
sudo git checkout unstable
```

To check out some of the samples, you'll need to build them first.

```
cd OpenNI/Platform/Linux-x86/Build
sudo ./RedistMaker
```

TIP

If you're unable to run the previous command, you may need to change the file permissions. Type `chmod 777 *` while in the Build directory.

After compiling everything, we can now move on to installing the SensorKinect driver.

```
cd ~/Kinect
sudo git clone https://github.com/avin2/SensorKinect
```

Change the working directory to CreateRedist to compile and create a redist package.

```
cd SensorKinect/Platform/Linux-x86/CreateRedist
```

Run the script RedistMaker.

```
sudo ./RedistMaker
```

Navigate to the newly created Redist directory located inside of Build and run the installation script.

```
cd ../Build/Redist
sudo ./install.sh
```

Change the working directory to Build and run `make` and then `make install`.

```
cd ~/Kinect/SensorKinect/Platform/Linux-x86/Build
sudo make && sudo make install
```

That's it for SensorKinect. The final step is to install the PrimeSense NITE Middleware package. Firs

download the version of PrimeSense NITE from http://www.openni.org/Downloads/OpenNIModules.aspx. In the first drop-down, select OpenNI Compliant Middleware Binaries. Then select the unstable version. Download the package that corresponds to your Ubuntu installation (32- or 64-bit).

> **TIP**
>
> I created a new directory called NITE in which I placed the contents of the extracted NITE package. This is just to keep things consistent, as version numbers may be different at the time this book is released.

If you want to try out some of the sample programs, you'll need to edit the XML files located in the Data directory to include the PrimeSense license key, as shown in Figure 1-6. To do this, you'll need to change the third line of the XML sample files from:

```
<License vendor="PrimeSense" key=""/>
```

to:

```
<License vendor="PrimeSense"
key="0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
```

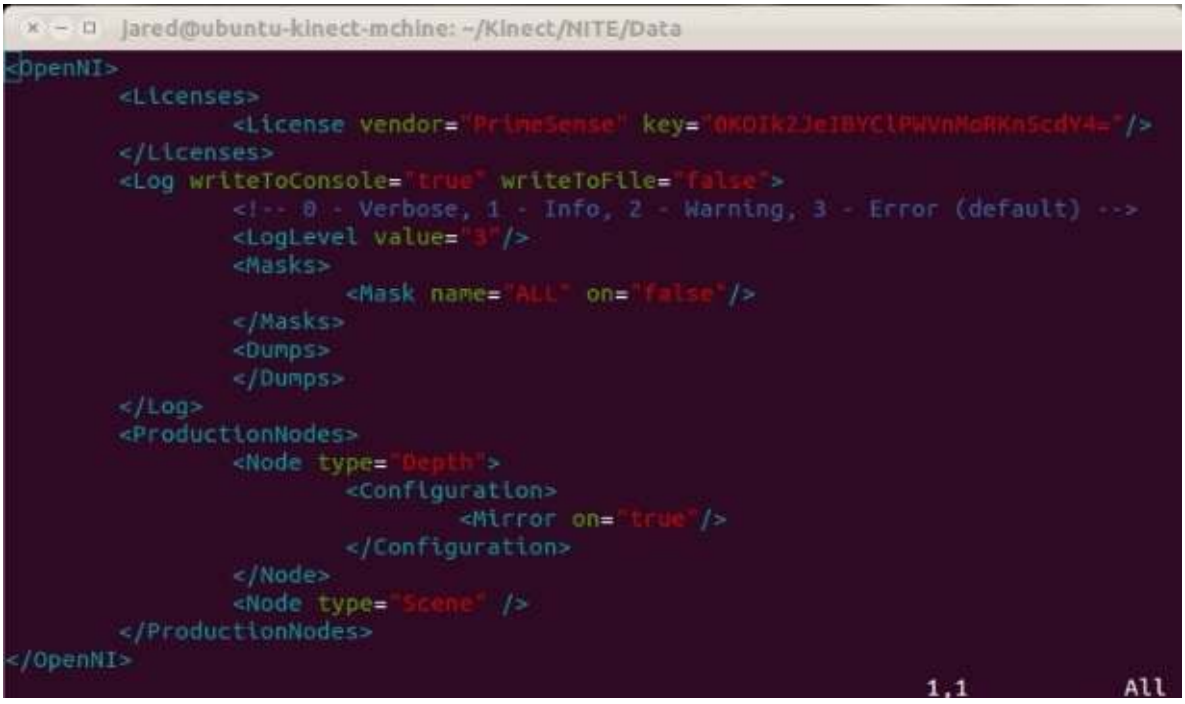

Figure 1-6. NITE XML sample key

Change the working directory to NITE and run the installation script.

```
cd ~/Kinect/NITE
./sudo install.sh
```

Right away, you'll be prompted to enter the PrimeSense license key. Enter the following key to complete the installation process.

```
0KOIk2JeIBYClPWVnMoRKn5cdY4=
```

NOTE

Newer versions do not require a license key.

```
At this point, you should be ready to see if the samples are working. Connect your Kinect to a USB
port and ensure that the power adapter is also plugged in.
```

Change the working directory to NITE Sample and execute one of the accompanying demos (Figure 7).

```
cd ~/Kinect/NITE/Samples/Bin/Release
sudo ./Sample-PointViewer
```

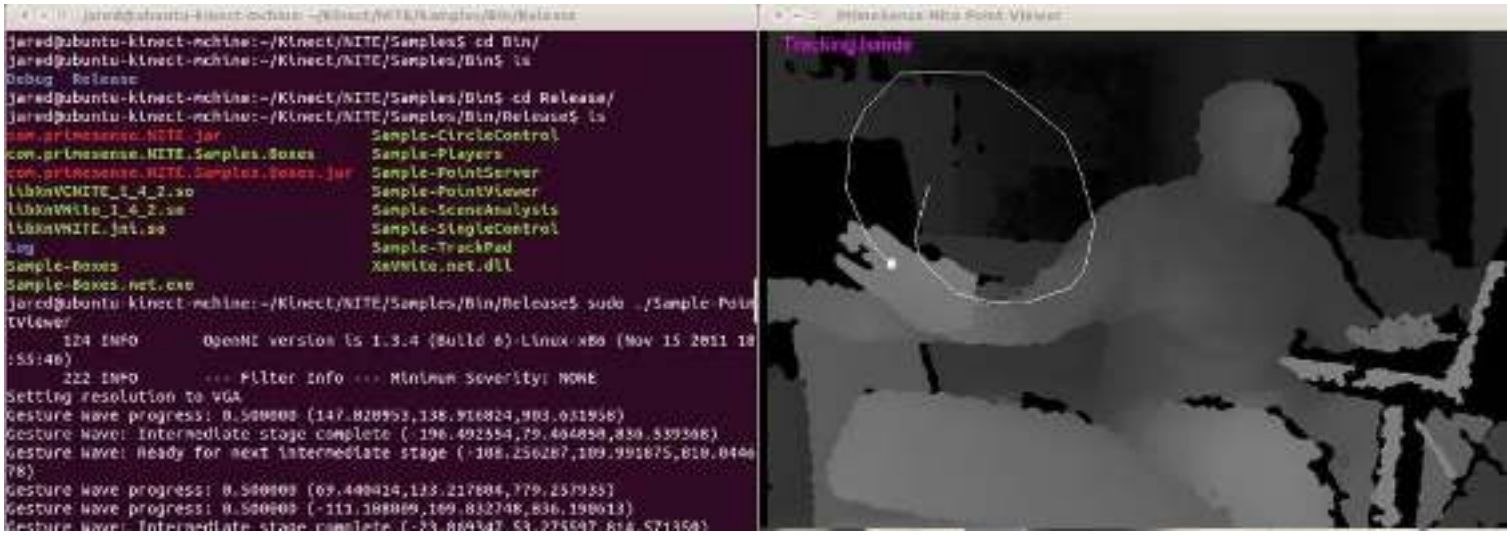

Figure 1-7. NITE sample point viewer running on Ubuntu

TIP

If you are continuously running into the "InitFromXML failed: Failed to set USB Interface! Error," shown in Figure 1-8, try the following (it solved my problem right away):

```
sudo rmmod gspca_kinect
```

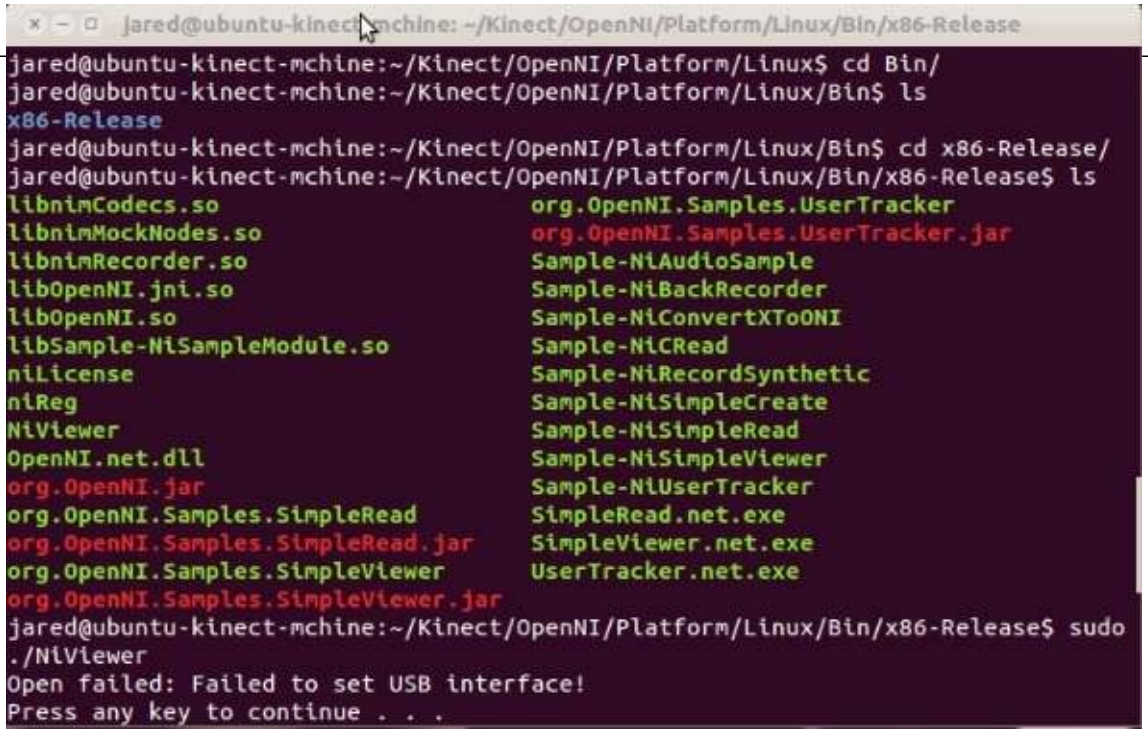

Figure 1-8. OpenNI USB interface error

Install libfreenect for Mac OS X

For this guide, we will be using MacPorts to download and configure the libraries along with other components. We'll perform the majority of the work using the Terminal application located in your Utilities directory: Applications/Utilities/Terminal. You will need to install Git to process the latest repository builds from GitHub and install the libtool and libusb-devel libraries. You'll need to issue the following commands to download and install the prerequisites and dependencies for compiling libfreenect:

```
sudo port install git-core
sudo port install
sudo port install libusb-devel
```

CAUTION

The `sudo` command requires the admin-level password for Mac, which grants you temporary root-level access.

You will also need to download and install CMake to compile libreenect. You can find the latest version at http://www.cmake.org/cmake/resources/software.html. Create a directory from which you would like to work. It can be named anything and placed anywhere. For the sake of simplicity, we wi create the directory OpenKinect located in your Home directory.

```
cd ~/
mkdir OpenKinect
cd OpenKinect
```

Download the latest libfreenect repository using the `git clone` command in your current working directory:

```
sudo git clone https://github.com/OpenKinect/libfreenect.git
```

Navigate to the libfreenect directory:

```
cd ~/OpenKinect/libfreenect
```

Create a new build directory, navigate into it, and compile the source code:

```
mkdir build
cd build
ccmake ..
```

When prompted, press the C key to begin compiling.

If the `ccmake` command fails, you will need to manually change the LIBUSB_1_INCLUDE_DIR path to usr/local/include/libusb-1.0.

Press C to restart the build.

Press the G key to generate and then type the following once everything has been completed:

```
ccmake ..
```

Once this process has completed and the cursor prompt has returned, run the following commands:

```
sudo make && sudo make install
```

The installation should be complete at this point. Navigate to the libfreenect/build/bin directory and run the glview demo to see if things are working properly.

```
cd ~/libfreenect/build/bin
sudo ./glview
```

Install libfreenect for Ubuntu

As with the Windows installation method described in Install OpenNI, SensorKinect, and NITE for Windows 7, there is a binary package available for Ubuntu installations of OpenKinect that allows us to easily install the drivers. You can find precompiled RPM and Deb packages by visiting http://bit.ly/TlFy5x.

These packages are rather old, so if you run into any installation problems, skip to the next section, which covers compiling libfreenect from source.

If you're using a fresh install of Ubuntu, you'll more than likely have to install quite a few dependencies on your machine to get started. A quick way to get things moving is to manually install everything at once by issuing the following command:

```
sudo apt-get install git-core cmake freeglut3-dev \
pkg-config build-essential libxmu-dev libxi-dev libusb-1.0-0-dev
```

After the library dependencies have all been successfully installed, we can now move on to compiling libfreenect from source. For the sake of this tutorial, I created a Kinect folder in my Home directory. This is where we'll place the extracted contents of libfreenect.

```
mkdir ~/Kinect
cd ~/Kinect
```

Download libfreenect using Git and then create and move into a build directory.

```
git clone https://github.com/OpenKinect/libfreenect.git
cd libfreenect
mkdir build
```

```
cd build
sudo cmake ..
```

Once CMake is done compiling, you can now run `make` and then `make install`.

```
sudo make
sudo make install
```

Then there's one last command before we can run some of the sample demos.

```
sudo ldconfig /usr/local/lib64/
```

Navigate to bin inside the build directory and run the glview demo. Make sure your Kinect sensor is connected via USB and that the power adapter is plugged in as well.

```
cd /libfreenect/build/bin
sudo ./glview
```

If glview opens properly, you should now be looking at the famous dual view of the RGB and depth camera output that the Kinect is currently capturing. See Figure 1-9.

Figure 1-9. libfreenect glview sample for Ubuntu

Install libfreenect for Windows 7

Get ready for quite the ride if you're interested in compiling and building libfreenect on a Windows machine. It is by far the most involved process, requiring a lot of tinkering around with library dependencies to properly compile and build libfreenect.

Before we get started, we'll need to download and install CMake and Visual Studio 2010 from the following places:

Visual C++ 2010 Express
: http://www.microsoft.com/visualstudio/eng/downloads

CMake Windows Binary Installer
: http://cmake.org/cmake/resources/software.html

Farther down the road, you'll need to manually assign the path for certain library files and includes to finish compiling libfreenect in CMake. I've added them all to a nice little ZIP file, which you can