Preface

I'll start by answering some questions that might occur to you while you're checking this book out.

Is this Book for Me?

Of course it is! Buy it straight away, and purchase several copies for your friends. They'll thank you profusely.

If you're not persuaded yet, then how about a managerial-style one sentence summary of the book? My aim is to describe the key building blocks needed to create fun, exciting 3D games in Java on a PC, with an emphasis on the construction of 3D landscapes that a player can explore.

If that's not enough yet (gosh, you're a tough customer), then cast your eyes over the next section (but really there's no need, this book was meant for you).

What's this Book About?

This book is divided into three main sections: Java 3D, non-standard input devices for game playing, and JOGL.

Java 3D is a high-level 3D graphics API, based around the construction of a scene graph data structure which contains the objects that appear in the 3D scene. Java 3D topics covered here include: how to build your own 3D models, load existing models, create detailed landscapes, display beautiful skies and backgrounds, and have the user navigate through the scene, bumping into things as they go.

I examine three non-standard input devices: the webcam, the game pad, and the P5 data glove, all fun alternatives to the rather boring PC keyboard and mouse.

JOGL is a Java wrapper around the popular OpenGL 3D graphics API, which offers a less high-level programming abstraction than Java 3D (in particular, there's no scene graph to build). JOGL's closeness to OpenGL means there's a wealth of existing OpenGL examples, tutorials, and programming tips and techniques that can be reused without much recoding. I look at similar topics to those for Java 3D: building and loading models, landscapes, skyboxes, billboards, picking, fog, and overlays.

Another theme of this book is the examination of games-related Java APIs that aren't part of the standard Java distribution (i.e. they're not in the software you get when you download Java SE). I've already mentioned Java 3D and JOGL. Other APIs include JInput (for interfacing Java to non-standard input devices), JOAL (a Java wrapper around the 3D sound API, OpenAL), JMF (for managing time-based multimedia, which I employ for rapidly taking webcam snaps), and OdeJava (a Java layer over the physics API, ODE).

This book examines the latest Java SE 6 features relevant to gaming, including splash screens, JavaScript scripting, and the desktop and system tray interfaces.

What's this Book not About?

I don't bother introducing Java; there are many books which do that already. Two worth checking out are *Head First Java* by Bert Bates and Kathy Sierra (O'Reilly, 2005), and *Thinking in Java* by Bruce Eckel (Prentice Hall, 2006). An older version of Eckel's book is available free at

http://www.mindview.net/Books/TIJ/. The sort of background you need for my book is what you'd learn in an introductory course on Java.

This isn't a book about developing a single massive application, such as a FPS (first person shooter). Instead I describe game elements, building blocks, that can be used in lots of different 3D games.

This book isn't about building a 3D rendering engine, I'm using Java 3D and JOGL for that. If you're interested in creating an engine from the ground up, then I recommend *Developing Games in Java* by David Brackeen, Bret Barker, and Laurence Vanhelswue (New Riders Games, 2003).

As I explain JOGL, I'll also explain the basic features of OpenGL. Unfortunately, I don't have the space to discuss all of OpenGL's wonderful capabilities, or the numerous extensions provided by different graphic card vendors. I'll supply you with pointers to more information when I start on JOGL in Chapter 15.

This isn't a games design book; two great resources are *Game Architecture and Design* by Andrew Rollings and Dave Morris (New Riders Games, 2003), and *Chris Crawford on Game Design*, by Chris Crawford (New Riders Games, 2003).

I won't be talking about J2ME games programming on mobile devices. There's some interesting 3D APIs for J2ME (the mobile 3D API and Java Bindings for OpenGL ES), but the emphasis of this book is on desktop applications.

Where's the CD/Code?

All the source code can be found at http://fivedots.coe.psu.ac.th/~ad/jg2/. I've also uploaded early drafts of the chapters there, including a few that didn't make it into the book.

How is this Book Different from KGPJ?

KGPJ is the abbreviation for my earlier book, *Killer Game Programming in Java*.

KGPJ is about 2D, 3D, and network games programming, but this book concentrates solely on 3D programming.

KGPJ has 16 chapters on Java 3D, while this book has eight (roughly half the book), *and* also covers non-standard input devices and JOGL. Of those eight chapters, four are on topics not covered in KGPJ, namely Java SE 6 integration, physics modeling, multitexturing, and mixed-mode rendering. The other four chapters overlap with KGPJ to some degree, but it means that this book is completely self-contained; there's no need for you to have read KGPJ before starting here. Also, all the example programs are entirely new.

KGPJ doesn't discuss the range of APIs covered here, such as JOGL, JInput, JOAL, OdeJava, and JMF.

KGPJ was published in May 2005, so focuses on J2SE 1.4 and Java 3D 1.3.1, while this book utilizes Java SE 6 and Java 3D 1.5.

If you're still not sure, then the best solution, at least the one most pleasing to me, is to *buy both books*. Thank you.

Java for Games Programming: Are you Joking?

No, Java is a great games programming language. When you learnt Java, I'm sure it's many advantages were mentioned: an elegant object-oriented paradigm, cross-platform support, code reuse, ease of development, tool availability, reliability and stability, good documentation, support from Sun Microsystems, low development costs, the ability to use legacy code (e.g. C, C++), and

increased programmer productivity. That list adds up to my personal reason for programming in Java – *it's fun*, especially when you're programming something inherently good-for-you, such as games.

Most Java-bashers tend to skip over advantages, preferring to concentrate on criticisms. Here's a typical list:

* Java is too slow for games programming;
* Java has memory leaks;
* Java is too high-level;
* Java application installation is a nightmare;
* Java isn't supported on games consoles;
* No one uses Java to write real games;
* Sun Microsystems isn't interested in supporting Java gaming.

Almost all of these objections are substantially wrong.

Java is roughly the same speed as C++, and has been since version 1.4. Many benchmarks indicate that Java SE 6 is 20-25% faster than J2SE 5.

Memory leaks can be avoided with good programming, and techniques like profiling.

Java is high-level, but also offers access to the graphics hardware and external devices. Many of the behind-the-scenes speed improvements in Java SE 6 are related to graphics rendering using OpenGL and DirectX.

A variant of the moaning about 'high-level' is that Java can't be connected to gaming peripherals such as game pads. This is nonsense, as shown in Chapters 9-14.

Installation isn't a nightmare. Java applets can be delivered via the Web, or with Java Web Start (JWS) can be utilized to download applications. There's numerous third-party installers, such as install4j (`http://www.ej-technologies.com/products/install4j/overview.html`).

There's a growing number of excellent commercial Java games, including *Tribal Trouble*, *Puzzle Pirates*, *Call of Juarez*, *Chrome*, *Titan Attacks*, *Star Wars Galaxies*, *Runescape*, *Alien Flux*, *Kingdom of Wars*, *Law and Order II*, *Ultratron*, *Roboforge*, *IL-2 Sturmovik*, *Galactic Village*, *Tiltilation*, and *Wurm Online*. Many are written entirely in Java, others employ Java in sub-components such as for game logic.

Java is used widely in the casual gaming market, where game-play is generally less complex and time-consuming. Implementation timelines are shorter, budgets smaller, and the required man-power is within the reach of small teams. By 2008, industry analysts believe the casual games market will surpass US $2 billion in the US alone.

There are numerous of Java gaming sites, including a showcase at Sun Microsystems (`http://www.java.com/en/games/`), community pages at `http://community.java.net/games/`, a collection of open-source gaming tools at `https://games.dev.java.net/`, the Java Games factory (`http://javagamesfactory.org/`), and many, very helpful forums at `http://www.javagaming.org/`.

What about Java on Games Consoles?

If you were paying attention in the last section, you'd have noticed that I didn't disagree with the lack of a games console version of Java. That's a bit embarrassing for a "write once, run anywhere" language.

The Sony PlayStation 2 (PS2) was the dominant games console at the end of 2006, with over 100 million units sold, dwarfing its competitors such as the Xbox 360, Xbox, Wii, and GameCube. Not

unsurprisingly, there have been many rumors over the years about a Java port for the PS2. In fact, it *is* possible to run Java on Sony's version of Linux, but the OS requires the PS2 to have a hard disk, and only offers limited access to the PS2's other hardware.

The good news is that the prospects for Java support on the PlayStation 3 (PS3) are much brighter. Both the basic and premium PS3 versions have 512 MB of RAM, a large hard drive, support Linux, and use an extended version of OpenGL. The PS3's Cell Broadband Engine essentially consists of a central 64-bit Power PC-based processor (the PPE) and nine data crunching support chips called SPEs.

Sony's software development chief, Izumi Kawanishi, has spoken of making it easier for individuals to create games on the PS3. One aspect of this is allowing third-party OSes to be installed, with the major restriction that they can't directly access the graphics hardware, which means that only 256 MB of RAM is available.

There are currently (January 2007) three versions of Linux known to run on the PS3 – Yellow Dog Linux (YDL) 5, Fedora Core 5, and Gentoo, with YDL officially supported by Sony. Installation details for YDL can be found at `http://www-128.ibm.com/developerworks/power/library/pa-linuxps3-1/`, with information on Fedora and Gentoo at `http://ps3.qj.net/PS3-Linux-The-void-has-been-filled-Full-install-instructions-for-Fedora-Core-5-/pg/49/aid/73144/` and `http://ps3.qj.net/Gentoo-Linux-on-your-PS3-With-full-install-instructions-/pg/49/aid/78739/`.

Since the PS3 uses a Power PC chip (the PPE), it should be possible to install the 64-bit Power PC version of Java for Linux offered by IBM (at `http://www-128.ibm.com/developerworks/java/jdk/linux/download.html`, and select J2SE5.0 SR3 for the 64-bit pSeries). No one has tried this yet, and the big question is whether the PPE will be powerful enough for running Java.

A good on-going thread about Java and the PS3 can be found at javagaming.org (`http://www.javagaming.org/forums/index.php?topic=15122.0`). It's also worth checking the ps3Forums and QJ.net sites (`http://www.ps3forums.com/` and `http://ps3.qj.net/`).

Java already has a presence on the PS3, as the software for its Blu-ray disc for high-definition video and data. All Blu-ray drives support a version of Java called BD-J for implementing interactive menus and other GUIs. Also, Blu-ray's network connectivity means that BD-J can be utilized for networking applications such as downloading subtitles, short movies, and adverts.

The present lack of Java on consoles is a serious issue, but the remaining PC market is far from miniscule. The Gartner Group believes there were 661 million PC users in 2006. The number will hit 953 million at the end of 2008, and cross over the billion mark in 2009.

Games on PCs benefit from superior hardware—such as video cards, RAM, and internet connections—so can offer more exciting game play. There are many more PC games, particularly in the area of multiplayer online games.