Pro Java™ 6 3D Game Development
Java 3D™, JOGL, JInput, and JOAL APIs

Andrew Davison
To Supatra and John
Contents at a Glance

About the Author .......... xxix
About the Technical Reviewers .. xxi
Acknowledgments .......... xxiii
Introduction ................. xxv

PART 1 — Java 3D

CHAPTER 1 Introducing Java 3D............................... 3
CHAPTER 2 Get a Life (in 3D).................................. 15
CHAPTER 3 Get a Life (the Java 6 Way)....................... 49
CHAPTER 4 The Colliding Grabbers.......................... 83
CHAPTER 5 When Worlds Collide............................ 121
CHAPTER 6 A Multitextured Landscape.................... 145
CHAPTER 7 Walking Around the Models.................... 179
CHAPTER 8 More Backgrounds and Overlays.............. 203

PART 2 — Nonstandard Input Devices

CHAPTER 9 Webcam Snaps...................................... 233
CHAPTER 10 Navigating a 3D Scene by Waving Your Arm.. 251
CHAPTER 11 Building a Gamepad Controller with JInput... 273
CHAPTER 12 Gamepad Grabbers.............................. 303
CHAPTER 13 3D Sound with JOAL............................ 327
CHAPTER 14 The P5 Glove...................................... 349

PART 3 — JOGL

CHAPTER 15 Two JOGL Programming Frameworks......... 377
CHAPTER 16 Touring the World............................... 409
CHAPTER 17 Loading Models................................. 439

INDEX .......................................................... 475
## Contents

About the Author ........................................................................................................... xix
About the Technical Reviewers ................................................................................... xxi
Acknowledgments ........................................................................................................... xxiii
Introduction ................................................................................................................... xxv

### PART 1  Java 3D

#### CHAPTER 1  Introducing Java 3D ................................................................. 3

- Overview of Java 3D .................................................................................................. 3
- Overview of the Scene Graph .................................................................................... 4
  - Some Java 3D Scene Graph Nodes ....................................................................... 5
  - The HelloUniverse Scene Graph .......................................................................... 5
- Java 3D Strengths ...................................................................................................... 7
  - The Scene Graph .................................................................................................. 7
  - Performance ........................................................................................................ 7
  - Unique Features .................................................................................................. 8
  - Java Integration ................................................................................................... 8
  - Documentation and Examples .............................................................................. 8
- Criticisms of Java 3D for Games Programming .................................................... 9
  - Java 3D’s Level of Abstraction .......................................................................... 9
  - Java 3D Games .................................................................................................... 10
  - Java 3D Support .................................................................................................. 12
- Alternatives to Java 3D ............................................................................................ 13
- Summary ................................................................................................................. 14

#### CHAPTER 2  Get a Life (in 3D) ................................................................. 15

- The Game of Life .................................................................................................. 15
- Running Life3D ....................................................................................................... 16
  - Configuring Life3D ............................................................................................... 16
  - A Life3D Screensaver ....................................................................................... 17
- An Overview of the Life3D Classes ....................................................................... 18
- Deciding How to Start ............................................................................................ 19
CHAPTER 3  Get a Life (the Java 6 Way) ................................................. 49

Life3D Basics ............................................................................. 49
An Overview of the Life3D Classes ........................................... 50
Making a Splash ................................................................. 52
  Animating a Clock ............................................................. 53
  Drawing onto a Splash ....................................................... 54
  Drawing the Clocks ............................................................ 54
JAR Packaging ....................................................................... 55
Adding ClockAnimation to Life3D ........................................... 55
CHAPTER 4

The Colliding Grabbers

Building Articulated Arms
Class Diagrams for Arms3D
Creating the Application Window
Drawing the 3D Scene
Processing the Keys
Monitoring Grabber Joints
Managing the Grabbers
   Adding the Grabbers to the Scene Graph
   Processing Keyboard Commands
   Translating the Grabbers
   Rotating a Grabber Joint
   Collecting the Collision Joints
   Collision Detection and Recovery
   Why Bother with JointsBehavior?
   Touching the Floor
CHAPTER 5 When Worlds Collide

Odejava and ODE ................................................................. 121
  Installing Odejava ......................................................... 123
  Documentation, Examples, and Online Help .................. 123
Bouncing a Ball ............................................................... 124
  Three-Stage Simulation ................................................. 125
  Initializing the Engines ............................................... 126
  Initializing Static Objects ............................................. 126
  Initializing Dynamic Objects ....................................... 127
  Executing the Simulation ............................................. 127
  Performing a Simulation Step .................................... 128
  Examining the Contact Points ..................................... 129
  Cleaning Up ............................................................... 129
Visualizing Balls in a Box ................................................ 129
  Creating the Scene ..................................................... 131
  The Box ................................................................. 133
  Managing the Spheres ............................................... 137
  A Sphere ............................................................... 139
  I’m Steppin’ Out .......................................................... 142
A Note of Application Development ................................. 143
Summary ........................................................................ 144
A Multitextured Landscape

Surveying the Landscape
Building the Scene
  Creating the Floor
  Start the Balls Moving
  Moving Around the Scene
The Multitextured Floor
  Heights Map Generation
  The Floor’s Geometry
  The Floor’s Appearance
The Splash Shape
  The Splash Shape’s Geometry
  TheSplash Shape’s Appearance
Moving Balls
  Positioning the Ball
  Moving About
Driving the Balls
More Multitexturing
  Bump Mapping
  Shaders
Summary

Walking Around the Models

Populating a Scene
Class Diagrams for ObjView3D
A Spherical Background
A Skybox Background
A Skybox Built with Quads
  Terragen and NConvert
  From Images to Textures
  Making the Skybox
  Texturing a Plane
  The Size of the Skybox
Loading Models
  Loading OBJ Models
  The OBJ Model Loader
  Positioning a Model
CHAPTER 12 Gamepad Grabbers .......................................................... 303

Example Overview ................................................................. 303
Playing Sounds ................................................................. 305
   Background Music with SoundsPlayer .................................. 307
   Obstacle Noises with SoundsPlayer .................................. 307
Managing Obstacles .............................................................. 308
   Making an Obstacle ......................................................... 309
   Making the Boxes ......................................................... 310
Collision Detection ............................................................... 311
Sending Input to the Grabbers .................................................. 312
   Processing Keyboard Input .............................................. 313
   Building a Grabber Operation for a Key Press ..................... 314
   Processing Gamepad Input ............................................. 316
   Building a Grabber Operation for the Gamepad ................... 318
The Grabbers .............................................................................. 319
   Connecting the Grabbers to the Camera ......................... 319
   Constructing the Grabbers ............................................. 320
   Processing an Operation ................................................. 321
The Grabber Arms .................................................................... 326
Summary .................................................................................. 326

CHAPTER 13 3D Sound with JOAL ................................................... 327

Why JOAL and Java 3D? .......................................................... 327
Background on OpenAL and JOAL ........................................... 328
   What About JOAL? .......................................................... 328
   Installing JOAL .............................................................. 329
Visiting the Musical Cow .................................................. 365
The Glove Behavior ......................................................... 366
Adding a Musical Cow ..................................................... 370
Initializing the Viewpoint .............................................. 371
Cleaning Up ................................................................. 371
The P5 Glove and JInput .................................................. 372
Summary ................................................................. 373

PART 3 ■ ■ ■ JOGL

CHAPTER 15 Two JOGL Programming Frameworks ......................... 377

What Is JOGL? .......................................................... 377
Installing JOGL ........................................................ 378
The Callback Framework ............................................. 379
   Using GLCanvas ................................................. 379
   Using GLJPanel .................................................. 380
Rotating a GLCanvas Cube with Callbacks ............................. 381
   Building the Top-Level Window ................................. 382
   Connecting the Canvas ......................................... 383
   Building the Listener ........................................... 383
   Measuring FPS Accuracy ....................................... 388
Rotating a GLJPanel Cube with Callbacks ............................... 389
   Building the Panels ........................................... 391
   The Background Panel ........................................ 391
   Making the 3D Background Transparent ........................ 392
   Timing the GLJPanel .......................................... 393
   More Visual Effects with GLJPanel ............................. 393
Callback Summary ....................................................... 393
The Active Rendering Framework ...................................... 393
Rotating a Cube with Active Rendering ................................. 395
   Building the Application ...................................... 396
   Dealing with Window Events .................................. 397
   Preparing the Canvas ......................................... 397
   Thread Rendering .............................................. 398
   The Game Life Cycle Methods ................................. 404
   Statistics Reporting ............................................ 405
Java 3D and JOGL ....................................................... 405
More Information on JOGL and OpenGL ............................... 406
Summary ................................................................. 407
CHAPTER 16  Touring the World .................................................. 409

Application Details .............................................................. 409
Constructing the Canvas ....................................................... 411
The Camera Position ............................................................ 412
Responding to Key Presses ..................................................... 413
Let the Rendering Commence .............................................. 416
Rendering Initialization ......................................................... 417
  Loading Textures ............................................................... 418
  Lighting the Scene ............................................................. 419
  Creating the Ground Shapes .............................................. 420
The Rendering Loop ............................................................ 421
Updating the Game .............................................................. 421
Rendering the Scene ............................................................ 423
Drawing the Tree Billboard ................................................... 423
Drawing the Ground Shapes .................................................. 426
The Planet Earth ................................................................. 428
The Skybox .................................................................... 430
Adding a 2D Overlay ............................................................ 433
Drawing the Floor ............................................................... 435
  Drawing the Tiles ............................................................... 435
  Drawing the Axes ............................................................... 437
Summary ..................................................................... 438

CHAPTER 17  Loading Models .................................................. 439

Overviews of the Examples .................................................. 439
The OBJ File Format ............................................................. 442
  Shape Statements ............................................................... 442
  Grouping Statements .......................................................... 443
  Material Use Statements ...................................................... 443
The MTL File Format ............................................................. 443
The OBJ File Loader ............................................................. 444
  Reading in the OBJ File .......................................................... 446
  Reading a Face .................................................................. 447
  Reading in an MTL File ......................................................... 449
  Recording Material Use ....................................................... 450
  Centering and Resizing a Model ......................................... 451
  Creating a Display List for the Model ................................... 452
  Finding a Material .............................................................. 452
  Rendering with a Material .................................................. 453
  Rendering a Face ............................................................... 455
  Drawing a Model ............................................................... 457
  When to Create the Display List ......................................... 457
Andrew Davison received his PhD from Imperial College in London in 1989. He was a lecturer at the University of Melbourne for six years before moving to Prince of Songkla University in Thailand in 1996. He has also taught in Bangkok, Khon Kaen, and Hanoi.

His research interests include scripting languages, logic programming, visualization, and teaching methodologies. This latter topic led to an interest in teaching games programming in 1999.

His O'Reilly book, *Killer Game Programming in Java*, was published in 2005 and is accompanied by a web site at http://fivedots.coe.psu.ac.th/~ad/jg/.

One of Andrew’s favorite hobbies is reading to his son, John, (currently they’re on *The Iron Man* by Ted Hughes). He has also been known to play PC and mobile phone games, but purely for research purposes.
About the Technical Reviewers

**CHIEN YANG** is a software engineer at Sun Microsystems, Inc., where he has worked since 1996. He works on graphics technologies for the Java Client Group. He has worked on the design and implementation of Java 3D API since its inception. He has also done work on graphics features simulation and graphics pipeline software. He holds a BS degree in computer science from the University of Iowa and an MS in computer science from the University of California at Berkeley.

**SHAWN KENDALL** has developed cutting-edge Java and Java 3D–based game technology demos for Full Sail Real World Education and Sun Microsystems, displayed at various conferences such as GDC and SIGGRAPH since 1999. In 2002, Shawn founded Immediate Mode Interactive (http://www.imilabs.com), a game technology company dedicated to the use of Java in games. Shawn has been developing in Java since 1995, starting with JDK 1.0 on SGI machines and Java 3D since 1998. Shawn graduated from the University of Central Florida with a BS degree in liberal science in 1995 and a computer science BS in 2002. Shawn maintains several Java 3D content loaders, as well as a host of Java 3D game demo projects.
My best regards to the many people at Apress who helped me get this book finished. They include Steve Anglin (Java editor), Denise Santoro Lincoln (project manager), Jennifer Whipple (copy editor), and Laura Cheu (production editor). Thanks also to the astute technical reviewers, Chien Yang and Shawn Kendall. I should also express my gratitude to the numerous people who sent me comments about the early drafts of this book, which are online at http://fivedots.coe.psu.ac.th/~ad/jg2/. Any remaining technical errors or poorly explained gobbledygook are my fault.

I must acknowledge my department (Computer Engineering), faculty (Engineering), and university (Prince of Songkla University) for being so supportive. They’ve always been understanding and have offered every encouragement. I recommend Thailand as a great place to live and work.

None of this would have been possible, or even worth attempting, without my family, Supatra and John, who I love very dearly.
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 straightaway, and purchase several copies for your friends. They’ll thank you profusely.

If you’re not persuaded yet, 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 (gosh, you’re a tough customer), 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, nonstandard 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 that 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 users navigate through the scene, bumping into things as they go.

I examine three nonstandard input devices: the webcam, the gamepad, 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 topics similar to those for Java 3D: landscapes, skyboxes, billboards, picking, fog, overlays, and building and loading models.

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 nonstandard 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 that already do that. 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 an FPS (first-person shooter). Instead I describe game elements, building blocks that can be used in a lot 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, I recommend *Developing Games in Java* by David Brackeen, Bret Barker, and Laurence Vanhelswue (New Riders Games, 2003).

As I explain JOGL, I 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 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: A New Edition* 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 are 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 sixteen chapters on Java 3D, while this book has eight (roughly half the book), and also covers nonstandard input devices and JOGL. Of those eight chapters, four are on topics not covered in *KGPJ*, namely Java SE 6 integration, physics modeling, multertexturing, 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 to 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, 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?

Java is a great games programming language. When you learned Java, I’m sure its 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, skipping over the advantages, have the following criticisms:

- 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% to 25% faster than J2SE 5.

Memory leaks can be avoided with good programming and techniques such as 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 gamepads. This is nonsense, as shown in Chapters 9–14.

Installation isn’t a nightmare. Java applets can be delivered via the Web or downloaded using Java Web Start. There are 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*, *Tiltillation*, and *Wurm Online*. Many are written entirely in Java, others employ Java in subcomponents such as game logic.

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

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 more than 100 million units sold, dwarfing its competitors such as the Xbox 360, Xbox, Wii, and GameCube. 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 512MB of RAM, a large hard drive, and Linux support, and use an extended version of OpenGL. The PS3’s Cell Broadband Engine essentially consists of a central 64-bit PowerPC-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 256MB of RAM is available.

There are currently (March 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/, and information on Fedora and Gentoo is 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 PowerPC chip (the PPE), it should be possible to install the 32-bit or 64-bit PowerPC version of Java for Linux offered by IBM (at http://www-128.ibm.com/developerworks/java/jdk/linux/download.html; select J2SE5.0 for the 32-bit or 64-bit pSeries). As of March 2007, I’ve heard from one person who has done this (with YDL and the 32-bit J2SE 5.0) and got JBoss running as a test Java application.

A good ongoing 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 says there were 661 million PC users in 2006 and that 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 gameplay. There are many more PC games, particularly in the area of multiplayer online games.
The Java 3D API, a scene graph API developed by Sun Microsystems, provides a collection of high-level constructs for creating, rendering, and manipulating a 3D scene graph. A scene graph makes 3D programming much easier for novices (and even for experienced programmers) because it emphasizes scene design, rather than rendering, by hiding the graphics pipeline. The scene graph supports complex graphical elements such as 3D geometries, lighting modes, picking, and collision detection.

This chapter gives an overview of the main features and strengths of Java 3D, leaving program examples aside for the moment, and addresses the common complaints about the API (which are unfounded).

URLs are included that lead to more information, games, model loaders, games-related libraries, and alternative scene graph systems.

Overview of Java 3D

Prior to the most recent release, version 1.5, there were two Java 3D variants: one implemented on top of OpenGL, the other above DirectX Graphics. OpenGL (the Open Graphics Library) is a cross-language, cross-platform API for 3D (and 2D) computer graphics. The DirectX Graphics API supports a rendering pipeline quite similar (in concept) to OpenGL, describing all geometry in terms of vertices and pixels. It's part of DirectX, a collection of related gaming APIs aimed at Microsoft Windows (http://www.microsoft.com/directx). The other APIs support 3D audio, networking, input device integration, multimedia, and installation management.

Java 3D on Windows uses the OpenGL renderer by default and requires OpenGL 1.3 or later. DirectX rendered can be switched on by the user with a -Dj3d.rend=d3d command-line argument, and requires DirectX 9.0 or later.

A JOGL rendering pipeline was added to Java 3D 1.5, making it easier to develop future Mac versions. JOGL is a thin layer of Java over OpenGL, effectively hiding some of the low-level variations in the OpenGL API across different OSES. The JOGL pipeline also offers a lightweight JCanvas3D class as an alternative to the heavyweight Canvas3D class. Canvas3D is utilized as a drawing surface for rendering a 3D scene but can be tricky to combine with lightweight Swing GUI components; I explain how to safely use Canvas3D in Chapter 2.

One of the main aims of Java 3D 1.6 (due out by the summer of 2008) is to use the JOGL binding to combine OpenGL and Java 3D rendering more closely.

The principal Java 3D web site is https://java3d.dev.java.net/, where Java 3D can be downloaded as a binary installation for various platforms; for example, I retrieved the final release version 1.5 for Windows. Java 3D should be installed after Java SE, with Java SE 5 or later the recommended version. The API documentation and examples are separate (but essential) downloads from the same site.
The Java 3D roadmap site
(http://wiki.java.net/bin/view/Javadesktop/Java3DRoadmap) details plans for versions 1.5.1, 1.6, and beyond. For instance, 1.5.1 will mainly add support for Microsoft Vista.

**Overview of the Scene Graph**

Java 3D uses a scene graph to organize and manage a 3D application. The underlying graphics pipeline is hidden, replaced by a treelike structure built from nodes representing 3D models, lights, sounds, the background, the camera, and many other scene elements.

The nodes are typed, the main ones being Group and Leaf nodes. A Group node has child nodes, grouping the children so that operations such as translations, rotations, and scaling can be applied en masse. Leaf nodes are the leaves of the graph (did you guess that?), which often represent the visible things in the scene, such as 3D shapes, but may also be nontangible entities, such as lighting and sounds. Additionally, a Leaf node may have node components, specifying color, reflectivity, and other attributes of the Leaf.

The scene graph can contain behaviors, represented by nodes holding code that affects other nodes in the graph at runtime. Typical behavior nodes move shapes, detect and respond to shape collisions, and cycle lighting from day to night.

The term scene graph is used, rather than scene tree, because it’s possible for nodes to be shared (i.e., have more than one parent).

Before looking at a real Java 3D scene graph, look at Figure 1-1 that shows how the scene graph idea can be applied to defining the contents of a living room.

![Scene graph for a living room](image)

**Figure 1-1. Scene graph for a living room**

The room Group node is the parent of Leaf nodes representing a sofa and two chairs. Each Leaf utilizes geometry (shape) and color node components, and the chair geometry information is shared. This sharing means that both chairs will have the same shape but be different colors.

The choice of symbols in Figure 1-1 comes from a standard symbol set (shown in Figure 1-2), used in all of this book’s Java 3D scene graph diagrams. I explain the VirtualUniverse and Locale nodes and the Reference relationship in the “HelloUniverse Scene Graph” subsection.
Some Java 3D Scene Graph Nodes

The Java 3D API can be viewed as a set of classes that subclass the Group and Leaf nodes in various ways. The Leaf class is subclassed to define different kinds of 3D shapes and environmental nodes (i.e., nodes representing lighting, sounds, and behaviors).

The main shape class is called Shape3D, which uses two node components to define its geometry and appearance; these are represented by classes called Geometry and Appearance.

The Group class supports basic node positioning and orientation for its children, and is subclassed to extend those operations. For instance, BranchGroup allows children to be added or removed from the graph at runtime, while TransformGroup permits the position and orientation of its children to be changed.

The HelloUniverse Scene Graph

The standard first code example for a Java 3D programmer is HelloUniverse (it appears in Chapter 1 of Sun’s Java 3D tutorial at http://java.sun.com/developer/onlineTraining/java3d and in the Java 3D examples collection at https://java3d.dev.java.net/). The HelloUniverse program displays a rotating colored cube, as shown in Figure 1-3.

![Scene graph symbols](image-url)
Figure 1-3. A rotating colored cube

The scene graph for this application is given in Figure 1-4.

VirtualUniverse is the top node in every scene graph and represents the virtual world space and its coordinate system. Locale acts as the scene graph’s location in the virtual world. Below the Locale node there are always two subgraphs.

Figure 1-4. Scene graph for HelloUniverse
The left branch is the content branch graph, holding program-specific content such as geometry, lighting, textures, and the world’s background. The content branch graph differs significantly from one application to another. The ColorCube is composed from a Shape3D node and associated Geometry and Appearance components. Its rotation is carried out by a Behavior node, which affects the TransformGroup parent of the ColorCube’s shape.

The right branch below Locale is the view branch graph, and specifies the user’s position, orientation, and perspective as he looks into the virtual world from the physical world (e.g., from in front of a monitor). The ViewPlatform node stores the viewer’s position in the virtual world; the View node states how to turn what the viewer sees into a physical world image (e.g., a 2D picture on the monitor). The Canvas3D node is a Java GUI component that allows the 2D image to be placed inside a Java application or applet.

The VirtualUniverse, Locale, and view branch graph often have the same structure across different applications, since most programs use a single Locale and view the virtual world as a 2D image on a monitor. For these applications, the relevant nodes can be created with Java 3D’s SimpleUniverse utility class, relieving the programmer of a lot of graph construction work.

Java 3D Strengths

The core strengths of Java 3D are its scene graph, performance, collection of unique features, Java integration, and extensive documentation and examples.

The Scene Graph

The scene graph has two main advantages: it simplifies 3D programming and speeds up the resulting code. The scene graph hides low-level 3D graphics elements and allows the programmer to manage and organize a 3D scene. The scene graph supports a wide range of complex graphical elements.

At the Java 3D implementation level, the scene graph is used to group shapes with common properties and carry out view culling, occlusion culling, level of detail selection, execution culling, and behavior pruning—all optimizations that must be coded directly by the programmer in lower-level APIs. Java 3D utilizes Java’s multithreading to carry out parallel graph traversal and rendering, both very useful optimizations.

Performance

Java 3D is designed with performance in mind, which it achieves at the high level by scene graph optimizations, and at the low level by being built on top of OpenGL or DirectX Graphics.

Some programmer-specified scene graph optimizations are available through capability bits, which state what operations can/cannot be carried out at runtime (e.g., prohibiting a shape from moving). Java 3D also permits the programmer to bypass the scene graph, either totally, by means of the immediate mode, or partially, via the mixed mode. Immediate mode gives the programmer greater control over rendering and scene management, but isn’t often required. Mixed mode “mixes” the immediate and retained modes so a program can utilize lower-level rendering and a scene graph together. Retained mode programs (the default Java 3D coding style) only use the scene graph API. Almost all of my Java 3D examples employ retained mode, except in Chapter 9, which looks at mixed mode rendering.
Unique Features

Java 3D's view model separates the virtual and physical worlds (through the ViewPlatform and View nodes). This makes it straightforward to reconfigure an application to utilize a range of output devices, from a monitor to eyeglasses with stereo displays to CAVEs (rooms with projected images covering every wall).

Virtual world behavior is coded with Behavior nodes in the scene graph and triggered by events. Among other things, Behavior nodes offer a different style of animation based on responding to events instead of the usual update-redraw cycle used in most games programs.

The core Java 3D API package, javax.media.j3d, supports basic polygons and triangles within a scene graph, while the com.sun.j3d packages add a range of utility classes, including ColorCube and SimpleUniverse, mouse and keyboard navigation behaviors, audio device handling, and loaders for several 3D file formats.

Geometry compression is possible, often reducing size by an order of magnitude. When this is combined with Java's NIO (new I/O capabilities present since version 1.4) and networking, it facilitates the ready transfer of large quantities of data between applications, such as multiplayer games.

Java 3D supports both of the popular programmable shader languages, OpenGL's GLSL (the default), and NVIDIA's Cg. This allows the programmer to create specialized rendering effects—such as bump mapping and shadows—very easily.

Java 3D offers 2D and 3D audio output, with ambient and spatialized sound. Unfortunately, there are bugs in the sound system, so spatialized sound isn't available by default in Java 3D 1.5. Version 1.6 will probably include a JOALMixer class instead, which will act as a programming interface to a JOAL-based audio device. JOAL is a Java binding for a 3D audio API called OpenAL, which is supported by many sound cards. As of January 2007, there's a partially completed version of JOALMixer at the j3d-incubator site (https://j3d-incubator.dev.java.net/). In Chapter 13, I develop a JOAL sound class that can be used with Java, Java 3D (in Chapter 14), and JOGL (Chapter 17).

Java Integration

Java 3D is Java, so it offers object orientation (classes, inheritance, polymorphism), threads, exception handling, and more. Java 3D can easily make use of other Java APIs, such as Java Media Framework (JMF) and Java Advanced Imaging (JAI). The JMF includes mechanisms for playing audio and video segments and can be extended to support new forms or audio and video (http://java.sun.com/products/java-media/jmf). JMF is utilized alongside Java 3D in Chapters 9 and 10. JAI provides many additional image processing features, including more than 100 imaging operators, tiling of large images, network-based capabilities, and the means to add new forms of image processing (JAI can be found at http://java.sun.com/products/java-media/jai).

Documentation and Examples

The Java 3D distribution doesn't come with any programming examples, which are a separate download at https://java3d.dev.java.net/; the 40-plus small to medium examples are a great help, but somewhat lacking in documentation.

Sun's Java 3D tutorial, available at http://java.sun.com/developer/onlineTraining/java3d, is quite old, dating from 2002 and focusing on version 1.2, but still useful for understanding Java 3D's core features.

Ben Moxon has a very nice introductory Java 3D tutorial based around getting a 3D figure to move over a hilly terrain (http://www.benmoxon.info/Java3d/); it's called "The Little Purple Dude Walks."