JOGL: JAVA AND OPENGL

Andrew Davison
Department of Computer Engineering
Prince of Songkla University
Hat Yai, Songkhla 90112
Thailand
E-mail: ad@fivedots.coe.psu.ac.th

EXTENDED ABSTRACT

This tutorial introduces JOGL (https://jogl.dev.java.net/), a Java interface to the APIs in the OpenGL 2.0 specification.

OpenGL is an extremely popular software API for writing 3D (and 2D) graphics applications across a wide range of hardware and OSes (http://www.opengl.org/). It's a low-level API based around a graphics pipeline for pixel and vertex manipulation.

JOGL has the same focus as OpenGL, on 2D and 3D rendering. It doesn't include support for gaming elements such as sound or input devices, but these are available through Java's standard libraries, or other external APIs, such as JOAL and JInput. JOGL can be combined with Java's AWT and Swing GUI components, and supports the shader languages, GLSL and Nvidia's Cg.

The OpenGL API is accessed via Java Native Interface (JNI) calls, leading to a very direct mapping between the API's C functions and JOGL's Java methods. As a consequence, it's extremely easy to translate OpenGL examples into JOGL.

Most of the features in the OpenGL GLU and GLUT libraries are present in JOGL. GLU (the OpenGL Utility library) includes support for rendering spheres, cylinders and disks, camera positioning, tessellation, and texture mipmaps. The JOGL version of GLUT (OpenGL Utility Toolkit) doesn't include its windowing functionality, which is handled by Java, but does offer geometric primitives (both in solid and wireframe mode). JOGL's utility classes include frame-based animation, texture loading, file IO, and screenshot capabilities.

I'm not going to explain all of OpenGL in this tutorial (that would require several weeks!), but I will introduce all the concepts necessary for understanding my JOGL examples. I'll supply pointers to sources of further information about JOGL and OpenGL.

I'll start by examining two different programming frameworks for JOGL: one using *callbacks*, the other utilizing *active rendering*. I'll describe a very simple example, a rotating multi-colored cube, using both approaches. I'll compare them by seeing how well they handle different frame rates for the cube's animation.

In the second part of the tutorial, I'll utilize the active rendering framework to build a more substantial application: a small 3D world, with textured objects, a skybox, billboards, overlays, a splash screen, a game-over message, and keyboard navigation. The application automatically suspends when the window is iconified or deactivated, and can be resized.

Various OpenGL (JOGL) techniques are illustrated in this example, including transparent images as textures, 2D overlays, and bitmap and stroke fonts.

BIBLIOGRAPHY

ANDREW DAVISON received his Ph.D. 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.

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, accompanied by a website at http://fivedots.coe.psu.ac.th/~ad/jg/.