OpenGL Research

We are planning on using openGL to visualize our simulation. The reasoning behind this selection became more obvious to us after the research we have conducted about the subject. OpenGL's website explains why openGL is the industry standard graphics library today.

Most Widely Adopted Graphics Standard

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

High Visual Quality and Performance

Any visual computing application requiring maximum performance-from 3D animation to CAD to visual simulation-can exploit high-quality, high-performance OpenGL capabilities. These capabilities allow developers in diverse markets such as broadcasting, CAD/CAM/CAE, entertainment, medical imaging, and virtual reality to produce and display incredibly compelling 2D and 3D graphics.

Developer-Driven Advantages

- **Industry standard**
- An independent consortium, the OpenGL Architecture Review Board, guides the OpenGL specification. With broad industry support, OpenGL is the only truly open, vendor-neutral, multiplatform graphics standard.
- **Stable**
- OpenGL implementations have been available for more than seven years on a wide variety of platforms. Additions to the specification are well controlled, and proposed updates are announced in time for developers to adopt changes. Backward compatibility requirements ensure that existing applications do not become obsolete.

- **Reliable and portable**
- All OpenGL applications produce consistent visual display results on any OpenGL API-compliant hardware, regardless of operating system or windowing system.
- **Evolving**
- Because of its thorough and forward-looking design, OpenGL allows new hardware innovations to be accessible through the API via the OpenGL extension mechanism. In this way, innovations appear in the API in a timely fashion, letting application developers and hardware vendors incorporate new features into their normal product release cycles.
- **Scalable**
- OpenGL API-based applications can run on systems ranging from consumer electronics to PCs, workstations, and supercomputers. As a result, applications can scale to any class of machine that the developer chooses to target.
- **Easy to use**
- OpenGL is well structured with an intuitive design and logical commands. Efficient OpenGL routines typically result in applications with fewer lines of code than those that make up programs generated using other graphics libraries or packages. In addition, OpenGL drivers encapsulate information about the underlying hardware, freeing the application developer from having to design for specific hardware features.
- **Well-documented**
- Numerous books have been published about OpenGL, and a great deal of sample code is readily available, making information about OpenGL inexpensive and easy to obtain.

We are also going to use java so we are going to need a Java wrapper library. JOGL is one such wrapper. The wikipedia page of the library talks about why it can be a nice tool for us to use.

Java OpenGL (**JOGL**) is a wrapper library that allows OpenGL to be used in the Java programming language.[1][2] It was originally developed by Kenneth Bradley Russell and Christopher John Kline, and was further developed by the Sun Microsystems Game Technology Group. Since 2010, it has been an independent open source project under a BSD license. It is the reference implementation for Java Bindings for OpenGL (JSR-231).

JOGL allows access to most OpenGL features available to C language programs through the use of Java Native Interface (JNI). It offers access to both the standard GL* functions along with

the GLU* functions; however the OpenGL Utility Toolkit(GLUT) library is not available for window-system related calls, as Java has its own windowing systems: Abstract Window Toolkit (AWT), Swing, and some extensions.

We also checked the web for tutorials and roadmaps and it seems like we won't run into any problems using the library for our goals. Some of the such helpful links are ;

http://www.land-of-kain.de/docs/jogl/

http://opengl.j3d.org/tutorials/

http://ogldev.atspace.co.uk/

http://www.cs.umd.edu/~meesh/kmconroy/JOGLTutorial/