OPENGL

By Neal Patel

What is OPENGL?

- Short for Open Graphics Library
- OpenGL provides programmers with a software interface to graphics hardware.
- Low level rendering and modeling software library on all major hardware platforms.
  - Windows, Mac, Linux, Unix, etc....

OPENGL (cont.)

- Designed for use in any graphics applications
  - Games
  - Modeling
  - CAD
- Callable from C, C++, Fortran, Perl, Python, Java, as well as many other languages.
- Provides only low level rendering routines allowing the programmer to easily build high level rendering and modeling libraries.

History of OPENGL

- Originally developed by Silicon Graphics, Inc. (SGI)
- Currently on version 2.0 which was just released on September 7, 2004.
  - Specifications do not get updated often

OPENGL Architecture

- Collection of several hundred functions
  - Provide access to all features offered by the graphics hardware.
- Internally acts as a state machine
  - Collection of states that tell OpenGL what to do.
  - Using the API, easily set various aspects of the state machine.
  - Color, lighting blending, texture, etc....

OPENGL: The rendering pipeline
Before Getting Started

- The OpenGL Utility Library
  - Known as GLU, supplements OpenGL providing higher-level functions
  - Uses lower level OpenGL commands
  - Port of every OpenGL implementation

- Features
  - 2D image scaling
  - Rendering 3D objects
  - Spheres, cylinders, and disks
  - Automatic mipmap generation from a single image
  - Support for curve surfaces through NURBS
  - Handling simple nonconvex polygons
  - Special purpose transformation and matrices

What in the heck is GLUT

- OpenGL Utility Toolkit
  - Set of support libraries
- OpenGL does not directly support any form of windowing, menus, or inputs, such as reading events from the mouse or keyboard.
- GLUT provides basic functionality, while remaining platform independent
  - Multiple windows for OpenGL rendering
  - Callback driven event processing
  - Command line argument processing
  - A simple pop up menu facility

More on Glut

- Contains the more complex 3D objects
  - Sphere, torus, cone, cube, teapot

- Windows management
  - glutInit(&argc, argv);
  - initializes GLUT & processes any command line arguments
  - Must be called before any other GLUT routine

Using OpenGL in my program

- Header files contents
  - Defined types
  - Constant declarations
  - Function prototypes

- Header files you may or may not use
  - #include<GL/gl.h>
  - #include<GL/glut.h>
  - #include<GL/glu.h>
  - #include<GL/glx.h>

How do I start using OpenGL on my machine?

- First off make sure your graphics card supports OpenGL.

- Download the library, header, and dll files into the appropriate directory in your compiler.
  - gl.h  C:\(Program Files\)Microsoft Visual Studio\vC98\include\GL
  - glut.h  C:\(Program Files\)Microsoft Visual Studio\vC98\include\GL
  - gl\32.b  C:\(Program Files\)Microsoft Visual Studio\vC98\vC98i386\lib
  - glut\32.b  C:\(Program Files\)Microsoft Visual Studio\vC98\vC98i386\lib
  - glut\32.lib  C:\(Program Files\)Microsoft Visual Studio\vC98\vC98i386\lib
  - glut\32.lib  C:\(Windows\)system32
  - glut\32.dll  C:\(Windows\)system32

OpenGL command syntax

- glVertex3fv(.........)
  - 3 - Number of arguments (2,3,4)
  - f - data type
  - d 4-byte float
  - d 8-byte float
  - i signed 2-byte int
  - i signed 4-byte int
  - v - indicates vector format, if present
  - Absence of v is result in scalar format
**OPENGL States and Primitives**

- OpenGL state machine
  - Consists of hundred of settings that effect various aspects of rendering.
- glGet() used to query the state machine for its current settings:
  - void glGetBooleanv(Genum pname, GLfloat *params);
  - void glGetDoublev(Genum pname, GLdouble *params);
  - void glGetFloatv(Genum pname, GLfloat *params);
  - void glGetIntegerv(Genum pname, GLint *params);

**What are primitives?**

- Webster’s Dictionary
  - An unsophisticated person
- Basic geometric entities such as points, lines, triangles, etc...
- Draw points in 3D
  ```
  glBegin(GL_POINTS);
  glVertex3f(0.0, 0.0, 0.0);
  glEnd();
  ```

**Some data types you should be aware of**

<table>
<thead>
<tr>
<th>OpenGL data type</th>
<th>Minimum Precision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLboolean</td>
<td>1 bit</td>
<td>Boolean value</td>
</tr>
<tr>
<td>GLbyte</td>
<td>8 bits</td>
<td>Signed 2’s comp int</td>
</tr>
<tr>
<td>GLshort</td>
<td>16 bits</td>
<td>Signed 2’s comp int</td>
</tr>
<tr>
<td>GLint</td>
<td>32 bits</td>
<td>Unsigned int</td>
</tr>
<tr>
<td>GLdouble</td>
<td>64 bits</td>
<td>Floating point value</td>
</tr>
</tbody>
</table>

**Famous Games that use OpenGL**

- Call of Duty (Activision)
- Half Life (Siera)
- Jedi Knights II (Lucas Arts)
- Medal of Honor (Electronic Arts)
- Quake III Arena (ID)