COMP3201 – Computer Graphics

Module 1: Introduction and Graphics Primitives

1.1 Introduction To OpenGL

1.1.1 Origins

OpenGL stands for the Open Graphics Library. It started as the graphics library IRIS GL developed by SGI (Silicon Graphics Incorporated), and was made publicly available in 1992 when it became OpenGL.

The OpenGL Architecture Review Board (OpenGL ARB) controls development of the standard.

OpenGL is independent of the graphics hardware or windowing system. Operating systems that support OpenGL include IRIX, Linux, Solaris, MS Windows, OSX, amongst others. Languages that can be used with OpenGL include C, C++, Fortran, Java and Python amongst others.

There is on-line documentation available, namely

- OpenGL Programming Guide (The Red Book)
- OpenGL Reference Manual (The Blue Book)
- OpenGL Utility Toolkit (GLUT) Programming Interface
    - http://www.opengl.org/resources/libraries/glut.html

1.1.2 OpenGL Pipeline

The (simplified) OpenGL **pipeline** (see Wright and Sweet (2000)) is:

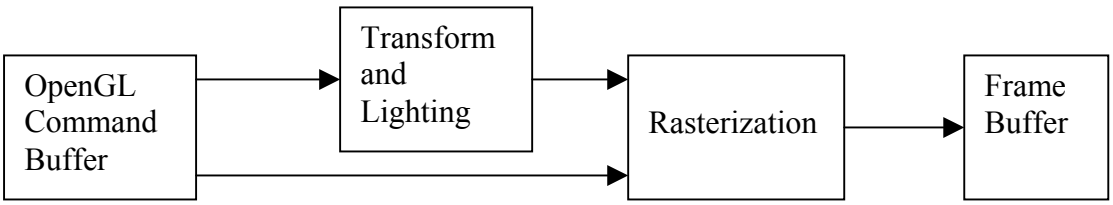

Figure 1.1

The OpenGL Command Buffer contains commands, vertex data, attributes, etc. When the buffer is flushed, this data is passed to the Transform and Lighting stage, which recalculates the data according to position, orientation and lighting definitions.

The data proceeds to Rasterization, which creates the colour image according to the geometric, colour and texture information. This image is placed in the Frame Buffer (i.e. the memory of the graphics display device) and displayed on the screen.

1.1.3 State Machine

OpenGL is a **state machine**, which means that when a mode is turned ON, it stays ON until turned OFF, and vice versa. For example, lighting is a mode (GL_LIGHTING) which is either ON (enabled) or OFF (disabled).

There are functions for testing the state of any mode. Note that some modes have an impact on performance, and so should only be enabled when required.

1.1.4 GL, GLU and GLUT Libraries

There are three main components to a usable implementation of OpenGL. These are

- GL – the OpenGL library of functions;
  - the functions in GL are used to render primitives (points, lines) as well as to specify colours, lighting and material properties;
  - there are also functions for view control;
  - defined constants commence with **GL_,** while function names commence **gl**
    - GL_COLOR_BUFFER_BIT, glClear( );
  - C-headers are included by **#include <GL/gl.h>**;
  - link libraries are **libGL.so** on Unix systems, and **opengl32.dll** on MS Windows;
  - OSX has a GL framework.

- GLU – the OpenGL Utility library of functions;
  - this is an ease-of-use layer built on top of GL;
  - it includes functions to draw more complicated objects (e.g. sphere, cone) as well as functions to handle errors and to set projection and viewing transformations;
  - defined constants begin with **GLU_**, while the function names commence **glu**
    - gluErrorString( )
  - C-headers are included by **#include <GL/glu.h>**;
  - link libraries are **libGLU.so** on Unix systems, and **glu32.dll** on MS Windows;
  - OSX has a GLU framework.

- GLUT – the OpenGL Utility Toolkit library;
  - this provides a cross-platform windowing environment for OpenGL programs; i.e., it is used to create windows and also to handle mouse and keyboard interactions;
  - constants are prefixed with **GLUT_**, while function names start with **glut**;
  - it is not necessary to use GLUT, and there are other alternatives, but we will use GLUT in this course;
  - C-headers are included by **#include <GL/glut.h>**;

- link libraries are **libglut.a** on Unix systems, and **glut32.dll** on MS Windows;
- OSX has a glut framework.

1.1.5 OpenGL Command Syntax

The most basic element in OpenGL is a vertex which defines a position in space. OpenGL works in 3D space, but vertices are specified with 4 coordinates (more on this later). Building up from a single vertex, two vertices are sufficient to define a line segment, three vertices are required for a triangle, and four vertices define a quadrilateral.

OpenGL uses a strict naming convention for each command. The command name is built up by defining the element type, dimension and the data type.

For a vertex, the base form is **glVertex***; the * is replaced by **ntv**, where

- **n** is the number of dimensions
- **t** is the data type (e.g. **i** for integer, **f** for float, **d** for double)
- **v** indicates if the data is in a vector (just omit the **v** if the data is not in vector format).

- **Examples:**
  - **glVertex2i(GLint x, GLint y) -** 2D vertex defined by integers
  - **glVertex3f(GLfloat x, GLfloat y, GLfloat z) -** 3D vertex defined by floating point numbers
  - **glVertex3fv(xyz) -** where **xyz** is a 3D array defined by GLfloat xyz[3].

Some of the OpenGL data types are given in the following table; see the OpenGL Reference Manual for more details.

OpenGL Data Type	Minimum Precision	Description
GLboolean	1 bit	Boolean value
GLbyte	8 bits	Signed 2's complement binary integer
GLubyte	8 bits	Unsigned binary integer
GLshort	16 bits	Signed 2's complement binary integer
GLushort	16 bits	Unsigned binary integer
GLsizei	32 bits	Non-negative binary integer size
GLbitfield	32 bits	Bit mask value
GLfloat	32 bits	Floating-point value
GLint	32 bits	Signed 2's complement binary integer
GLuint	32 bits	Unsigned binary integer
GLenum	32 bits	Enumerated type
GLdouble	64 bits	Floating-point value

Table 1.1

1.1.6 OpenGL States and Attributes

For a complete list of possible states, see Appendix B of the OpenGL Programming Guide. Some of these states are

- **GL_BLEND**
- **GL_DEPTH_TEST**
- **GL_COLOR_MATERIAL**
- **GL_LIGHTING**
- **GL_LINE_SMOOTH**
- **GL_POINT_SMOOTH**

A state is turned ON by using **glEnable("some state")**, for example **glEnable(GL_BLEND)**, while **glDisable("some state")** turns the state off.

A Boolean state can be queried by using **glIsEnabled**. For example,

GLboolean isBlending = glIsEnabled(GL_BLEND);

There are other **glGet*** functions for determining, for example, some system values (e.g. **GL_MAX_LIGHTS**) or current parameter values (for example, **GL_CURRENT_COLOR**), but the use of these functions can impact heavily on performance.

