Introduction to OpenGL

CS 351-50

OpenGL: Open Graphics Library

- Graphics API
  ( Application Programming Interface)
  - Software library
  - Layer between programmer and graphics hardware (and other software)
- Several hundred procedures and functions

Programmer’s View

Output Device  Input Device  Input Device

What is OpenGL

- Configurable state machine
  - Input is 2D or 3D data
  - Output is framebuffer
  - Modify state to modify functionality
### What is OpenGL
- Widely used and supported
- “the” choice for Linux developers
- Very well documented
- Easy to use
- Supports high-end graphics features
- Geometric and pixel processing

### OpenGL History
- Originally developed by SGI in early 90’s
- No longer SGI proprietary
- License free

---

### OpenGL History
- Evolution controlled by OpenGL ARB (architecture review board)
  - One vote per company
    - Includes Microsoft, SUN, SGI, nVidia, ATI

### What is OpenGL used for
- Real-time applications
- Fast preview for visualizations
- Interactive virtual environments
- Video games (Quake, by id Software)
What OpenGL is not used for

• Quality rendering
  – OpenGL uses scan-line rasterization
  – Use ray-tracing or radiosity for quality

How does OpenGL do it?

• Client-server interpretation
  – Program (client) issues commands
    • Eg. Enable lighting, render triangle, etc.
  – Commands interpreted and processed by server
    • the “GL”

OpenGL

• Does not provide a means of modeling complex objects
  – Requires a higher level API
• Does not provide support for peripherals
  – I.e. mouse, sound, etc
  – Requires other libraries
• Does not provide windowing or a GUI
  – For this we use GLUT
    • (Graphics Library Utility Toolkit)

OpenGL: In a nut shell

• 2D, 3D data goes in
• Framebuffer comes out
  MAGIC!
What does this mean to you? (Why lean thru OpenGL)
- High quality rendering
- Easy to program
- Portable code (hopefully)

What you need to know
- How GL works
- How to interface with it
  - How to configure it
  - How to pass data to it
- How to know what went wrong

Libraries
- #include <GL/gl.h>
- #include <GL/glu.h>
- #include <GL/glut.h>

Window System Interaction
- OpenGL is completely platform dependent
- Need a windowing system for things like
  - Interaction
  - Opening/Closing Windows
  - Handling events
- Options:
  - GLX (*nix)
  - WGL (windows)
  - GLUT (window-system independent)
Event Driven Interaction

- OpenGL does not dictate any particular model of interaction
- Applications respond to events generated by devices (ie mice) and window system events (ie window resized)
- Events usually placed in a queue awaiting action
- **Callbacks** let you associate a function with a particular type of event
  - Mouse callback

Create a window with GLUT

- glutInitWindowSize
- glutInitDisplayMode
- glutCreateWindow

How do I render a geometric primitive?

- To Framebuffer
- **OpenGL primitives**
  - A group of one or more vertices
  - Vertex defines:
    - A point
    - An endpoint of an edge
    - A corner of a polygon where two edges meet

OpenGL Rendering

- Data consists of
  - Positional coordinates
  - Colors
  - Normals
  - Texture Coordinates
- Each vertex is processed
  - independently
  - In order
  - In the same way
OpenGL Primitives

OpenGL drawing

- To draw a primitive, call `glBegin()`
- `glEnd()` encloses a list of vertices and their attributes
- Coordinates of a primitive are given counter-clockwise order

Function calls to draw a primitive

```cpp
glBegin(GL_POINTS);
glVertex3f(0.0f, 0.0f, 0.0f);
glEnd();
```

Draw a triangle:

```cpp
glBegin(GL_TRIANGLES);
glVertex3f(0.0f, 1.0f, 0.0f);
glVertex3f(-1.0f, -1.0f, 0.0f);
glVertex3f(1.0f, -1.0f, 0.0f);
glEnd();
```
Draws a triangle with different colors at each vertex

```c
glBegin(GL_TRIANGLES);
    glColor3f(1.0f, 0.0f, 0.0f); //pure red
    glVertex3f(0.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 1.0f, 0.0f); //pure green
    glVertex3f(-1.0f, -1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f); //pure blue
    glVertex3f(1.0f, -1.0f, 0.0f);
glEnd();
```