Programming with OpenGL
Part 1: Background

Ed Angel
Professor of Computer Science,
Electrical and Computer Engineering, and Media Arts
University of New Mexico
Objectives

• Development of the OpenGL API
• OpenGL Architecture
  - OpenGL as a state machine
• Functions
  - Types
  - Formats
• Simple program
Early History of APIs

• IFIPS (1973) formed two committees to come up with a standard graphics API
  - Graphical Kernel System (GKS)
    • 2D but contained good workstation model
  - Core
    • Both 2D and 3D
  - GKS adopted as ISO and later ANSI standard (1980s)

• GKS not easily extended to 3D (GKS-3D)
  - Far behind hardware development
PHIGS and X

- Programmers Hierarchical Graphics System (PHIGS)
  - Arose from CAD community
  - Database model with retained graphics (structures)
- X Window System
  - DEC/MIT effort
  - Client-server architecture with graphics
- PEX combined the two
  - Not easy to use (all the defects of each)
SGI and GL

• Silicon Graphics (SGI) revolutionized the graphics workstation by implementing the pipeline in hardware (1982)
• To access the system, application programmers used a library called GL
• With GL, it was relatively simple to program three dimensional interactive applications
The success of GL lead to OpenGL (1992), a platform-independent API that was
- Easy to use
- Close enough to the hardware to get excellent performance
- Focus on rendering
- Omitted windowing and input to avoid window system dependencies
Open GL Evolution

• Controlled by an Architectural Review Board (ARB)
  - Members include SGI, Microsoft, Nvidia, HP, 3DLabs, IBM, ……
  - Relatively stable (present version 2.0)
    • Evolution reflects new hardware capabilities
      – 3D texture mapping and texture objects
      – Vertex programs
  - Allows for platform specific features through extensions
OpenGL Libraries

• OpenGL core library
  - OpenGL32 on Windows
  - GL on most unix/linux systems (libGL.a)

• OpenGL Utility Library (GLU)
  - Provides functionality in OpenGL core but avoids having to rewrite code

• Links with window system
  - GLX for X window systems
  - WGL for Windows
  - AGL for Macintosh
• OpenGL Utility Toolkit (GLUT)
  - Provides functionality common to all window systems
    • Open a window
    • Get input from mouse and keyboard
    • Menus
    • Event-driven
  - Code is portable but GLUT lacks the functionality of a good toolkit for a specific platform
    • No slide bars
Software Organization

application program

OpenGL Motif widget or similar

GLUT

GLX, AGL or WGL

GL

X, Win32, Mac O/S

GLU

software and/or hardware
OpenGL Architecture

- Immediate Mode
  - Polynomial Evaluator
  - Per Vertex Operations & Primitive Assembly
  - Display List
    - Rasterization
      - Texture Memory
        - Per Fragment Operations
          - Frame Buffer

- CPU
  - Pixel Operations
OpenGL Functions

• Primitives
  - Points
  - Line Segments
  - Polygons
• Attributes
• Transformations
  - Viewing
  - Modeling
• Control (GLUT)
• Input (GLUT)
• Query
OpenGL State

• OpenGL is a state machine

• OpenGL functions are of two types
  - Primitive generating
    • Can cause output if primitive is visible
    • How vertices are processed and appearance of primitive are controlled by the state
  - State changing
    • Transformation functions
    • Attribute functions
Lack of Object Orientation

• OpenGL is not object oriented so that there are multiple functions for a given logical function
  - `glVertex3f`
  - `glVertex2i`
  - `glVertex3dv`

• Underlying storage mode is the same

• Easy to create overloaded functions in C++ but issue is efficiency
OpenGL function format

- Function name: `glVertex3f(x,y,z)`
- Dimensions: `x, y, z` are floats
- Belongs to GL library

- Function name: `glVertex3fv(p)`
- `p` is a pointer to an array
OpenGL #defines

• Most constants are defined in the include files `gl.h`, `glu.h` and `glut.h`
  - Note `#include <GL/glut.h>` should automatically include the others
  - Examples
    - `glBegin(GL_POLYGON)`
    - `glClear(GL_COLOR_BUFFER_BIT)`

• Include files also define OpenGL data types: `GLfloat`, `GLdouble`, ....
A Simple Program

Generate a square on a solid background
#include <GL/glut.h>

void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POLYGON);
    glVertex2f(-0.5, -0.5);
    glVertex2f(-0.5, 0.5);
    glVertex2f(0.5, 0.5);
    glVertex2f(0.5, -0.5);
    glEnd();
    glFlush();
}

int main(int argc, char** argv){
    glutCreateWindow("simple");
    glutDisplayFunc(mydisplay);
    glutMainLoop();
}

Angel: Interactive Computer Graphics 4E © Addison-Wesley 2005
Event Loop

• Note that the program defines a display callback function named `mydisplay`
  - Every glut program must have a display callback
  - The display callback is executed whenever OpenGL decides the display must be refreshed, for example when the window is opened
  - The `main` function ends with the program entering an event loop
Defaults

- *simple.c* is too simple
- Makes heavy use of state variable default values for
  - Viewing
  - Colors
  - Window parameters
- Next version will make the defaults more explicit
Notes on compilation

• See website and ftp for examples

• Unix/linux
  - Include files usually in …/include/GL
  - Compile with –lglut –lglu –lgl loader flags
  - May have to add –L flag for X libraries
  - Mesa implementation included with most linux distributions
  - Check web for latest versions of Mesa and glut
Compilation on Windows

• **Visual C++**
  - Get glut.h, glut32.lib and glut32.dll from web
  - Create a console application
  - Add opengl32.lib, glut32.lib, glut32.lib to project settings (under link tab)

• **Borland C similar**

• **Cygwin (linux under Windows)**
  - Can use gcc and similar makefile to linux
  - Use –lopengl32 –lglu32 –lglut32 flags