Software

OpenGL Introduction

Software

- Link between applications and hardware
- Levels:
  - Low Level
  - Mid Level
  - High Level

Software – Low Level

- OS dependant hardware drivers
- Directly interacts with the specific hardware through its hardware interface
- Provided by the manufacturer

Software – Mid Level

- General purpose graphics libraries
- Specified by API (Application Programming Interface)
- Functions provided
  - Primitive functions: *lines, polygons, text*
  - Attribute functions: *color, fill pattern*
  - Viewing functions: *camera parameters*
  - Transformation functions: *translation, rotation*
  - Input functions: *keyboard, mouse*
  - Control functions: *communicate with window system*
  - Inquiry functions: *hardware specific differences*

Software – Mid Level (2)

- General: *GKS, PHIGS (mid 80s)*
- Mac: *QuickDraw GX (80's)*
- Windows: *DirectX (late 90's)*
- Unix: *Xlib, Motif (80's)*
- HP: *Starbase (90's)*
- SGI: *IrisGL (early 90's)*
- Industry standard: *OpenGL (mid 90's)*


Software – High Level

- Application specific packages
  - Virtual Reality: *SGI Performer*
  - Animation/Rendering: *Postscript, Povray*
  - Document Layout: *Acrobat, LaTeX*
  - Scientific: *IBM Data Visualizer, VRML*

OpenGL - Introduction

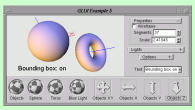
- Standard library for 2D/3D graphics
- Mid level
- Originally by SGI but has cross-platform support
- Independent of OS: (Windows, Mac, Linux, OS/2)
- Independent of windowing system
- Routines for creating, handling and rendering geometrical primitives
- State based
- Shipped as a part of Windows & Mac
- Mesa3D: A freeware implementation for multiple platforms

OpenGL - Scope

- Capabilities
  - Primitives functions: line, polygon, text, …
  - Z-buffer
  - Lightning
  - Texture mapping
- Limitations
  - windowing system/user interface
  - Input
  - Limited support for mirrors, shadows, indirect lightning, curved surfaces, …

OpenGL – Additional Libraries

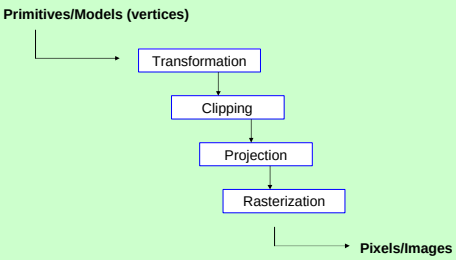
- GLU: OpenGL Utility Library
  - Uses the base OpenGL library to provide higher-level drawing routines
  - Additional primitives for OpenGL
- GLUT: OpenGL Utility Toolkit
  - System level I/O with the host OS
  - Window and event management
- GLUI: OpenGL User Interface Library
  - Uses GLUT to provide standard user interface controls: buttons, checkboxes, spinners, …
  - Support for multiple user interface windows



OpenGL - State

- Is always a global current state initialized with default values. e.g.
  - *current drawing color, line width, point size*
  - *coordinate system*
  - *extra features: double buffering, alpha bending, …*
- Changing the state changes applies to all subsequent operations

OpenGL - Pipeline



OpenGL- A typical program

- System dependant initialization
  - Setup a window
  - Bind OpenGL to the window
- OpenGL initialization
  - Setup coordinate system
  - Setup initial system values
- Event loop
  - Wait for an event: refresh window, mouse move, …
  - Figure out the change to be made on the display
  - Use OpenGL functions to update the scene

OpenGL – Naming convention

gl[u/ut/ui]Func[234][dfis][v] (arguments)

- function name
- dimension
- data type: *double, float, integer, short*
- Pointer / Values

Examples:

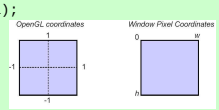
glVertex3fv (vertex1)

glColor3f (r, g, b)

OpenGL – Setting up the View

- We need to define the coordinate system
- Specify how to map vertex coordinates onto window: gluOrtho2D(left, right, bottom, top)
- e.g.:

```
gluOrtho2D(-1, 1, -1, 1);
```



OpenGL – Geometric Primitives

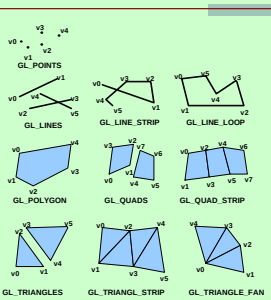
glBegin (TYPE);

define vertices:

e.g. glVertex3f

glEnd();

TYPE



OpenGL – Finally a Sample!

```
#include <GL/glut.h>                          // also includes <GL/gl.h>
int main(int argc, char** argv)
{
    glutInit(&argc, argv);                    // initialize the toolkit
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);  // set display mode
    glutInitWindowSize(400,400);              // set window size
    glutInitWindowPosition(100, 100);         // set window position on screen
    glutCreateWindow("The First Attempt");    // open the screen window
    glutDisplayFunc(myDisplay);               // register redraw function
    gluOrtho2D(-1, 1, -1, 1);
    glutMainLoop();                           // go into a perpetual loop
    return 0;                                 // never reached
}

void myDisplay(void)
{
    glClear(GL_COLOR_BUFFER_BIT);             // clear the screen
    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();                                // send all output to display
}
```