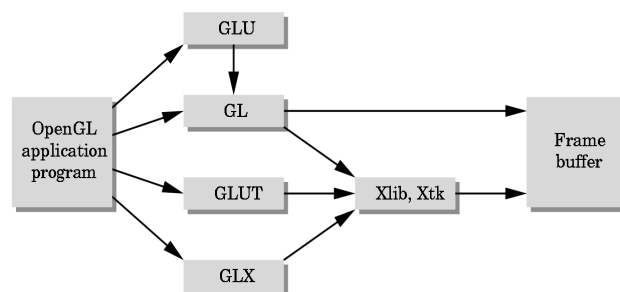


OpenGL Primer

OpenGL Interface



- o All (core) OpenGL functions (macros) are in GL (*gl**)
- o Common functions are in GLU, graphics utility library (*glu**)
- o Interface to Window Systems is in GLUT, GL utility toolkit (*glut**)
- o Interface to specific window system

"White Rectangle"

```
void display(void) {
    /* draw unit square polygon */
    glBegin(GL_POLYGON);
        glVertex2f(-0.5, -0.5);
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, -0.5);
    glEnd();

    /* flush GL buffers */
    glFlush();
}
```

"White Rectangle"

```
#include <windows.h>
#include <gl/glut.h>
#include <stdlib.h>

int main(int argc, char** argv) {
    /* Initialize mode and open a window in upper left corner of screen */
    glutInit(&argc,argv);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("My White Rectangle");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

OpenGL Types & Vertex*()

- o OpenGL basic types:
 - #define GLshort short #define GLint int
 - #define GLfloat float #define GLdouble double
 - #define GLsizei unsigned long
 - GLclampf : floats in [0.0, 1]
- o Arrays:
 - 2D point: GLint point2[2];
 - 3D point type: typedef GLfloat point3[3];
- o glVertex*() where *= nt or ntv
 - n captures dimensions (2,3,4)
 - t captures data type of arguments (s,i,f,d)
 - v passing arguments by reference rather than by value

CS1566, Panos K. Chrysanthis – University of Pittsburgh

13

Objects

- o glBegin() ...glEnd() : group vertices together

```
glBegin(geometric type);
glVertex2i(GLint x1, GLint y1)
glVertex2i(GLint x2, GLint y2)
glVertex2i(GLint x3, GLint y3)
glVertex2i(GLint x4, GLint y4);
glEnd();
```

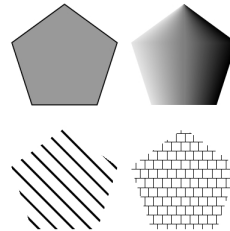
- o **Geometric type:** how vertices are interpreted
 - GL_POINTS
 - GL_LINES
 - GL_LINE_STRIP
 - GL_LINE_LOOP

CS1566, Panos K. Chrysanthis – University of Pittsburgh

14

Polygons

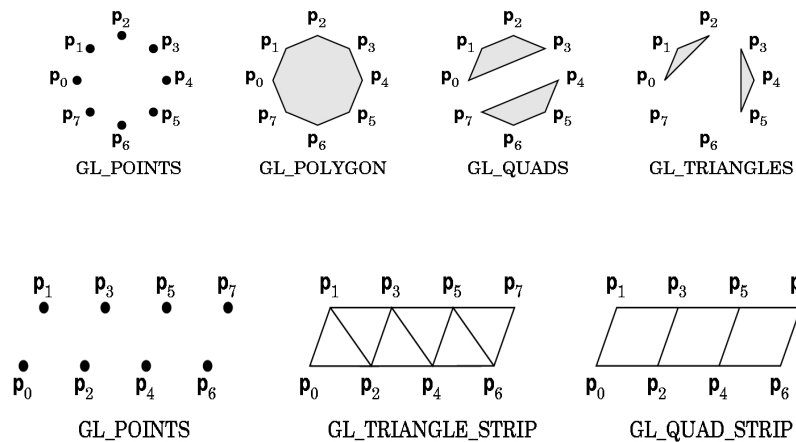
- o **Polygon** (fill area) is a closed objects that has an *interior*
- o It must be *simple*, *convex* and *flat*
 - *Simple*
 - *Convex*
 - *Flat*
- o Polygon geometric Types:
 - GL_POLYGON
 - GL_TRIANGLE
 - GL_QUADS
 - GL_TRIANGLE_STRIP
 - GL_QUAD_STRIP
 - GL_TRIANGLE_FAN



CS1566, Panos K. Chrysanthis – University of Pittsburgh

15

Polygon Types



CS1566, Panos K. Chrysanthis – University of Pittsburgh

16

Primitives and Attributes

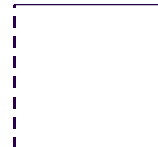
- o Primitives define what (object) to display
- o Attributes determine how an object is display
- o Conceptually attributes are associated with objects but in OpenGL are associated with the state of drawing.
- o The size of a point: `glPointSize(GLfloat width);`
- o Attributes for a line segments:
color, thickness, and (stipple) pattern
- o `void glLineWidth(GLFloat width)` default 1.0 pixel
- o `void glLineStipple(GLint factor, GLUshort pattern)`
 - 1 in pattern means pixel is drawn, 0 means pixel is not drawn
 - Factor (1-256) repeats successive groups of 1s and 0s in pattern
 - E.g., `glLineStipple(2, 0xACAC);`

CS1566, Panos K. Chrysanthis – University of Pittsburgh

17

"White Rectangle"

```
glBegin(GL_POLYGON);  
    glVertex2f(-0.5, -0.5);  
    glVertex2f(0.5, -0.5);  
    glLineWidth(2.0);  
    glLineStipple(2, 0xCCCC);  
    glVertex2f(0.5, 0.5);  
    glLineWidth(1.0);  
    glLineStipple(1, 0xFFFF);  
    glVertex2f(-0.5, 0.5);  
    glLineWidth(2.0);  
    glLineStipple(2, 0xCCCC);  
glEnd();
```



CS1566, Panos K. Chrysanthis – University of Pittsburgh

18

"White Rectangle?"

```
glLineWidth(1.0);  
glLineStipple(1, 0xFFFF);  
glBegin(GL_LINES);  
glVertex2f(-30, -30);  
glVertex2f(30, -30);  
glEnd();
```

```
glLineWidth(1.0);  
glLineStipple(1, 0xFFFF);  
glBegin(GL_LINES);  
glVertex2f(30, 30);  
glVertex2f(-30, 30);  
glEnd();
```



```
glLineWidth(2.0);  
glLineStipple(2, 0xCCCC);  
glBegin(GL_LINES);  
glVertex2f(30, -30);  
glVertex2f(30, 30);  
glEnd();
```

```
glLineWidth(2.0);  
glLineStipple(2, 0xCCCC);  
glBegin(GL_LINES);  
glVertex2f(-30, 30);  
glVertex2f(-30, -30);  
glEnd();
```

Enabling OpenGL Features

- o It is not sufficient to set the parameters of a feature such as stippling, lighting, hidden-surface removal etc.
- o Features must be enable and disable using:
 void glEnable(GLenum feature)
 void glDisable(GLenum feature)
- o E.g., line stippling is enabled by
 glEnable(GL_LINE_STIPPLE)