ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
VIETNAM - KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY
한-베정보통신기술대학교

Nhân bản – Phụng sự – Khai phóng

Programming CG in OpenGL

Computer Graphics

CONTENTs

- Graphics rendering API
- GLUT Basics
- Program Structure
- OpenGL Primitives

CONTENTs

- **Graphics rendering API**
- GLUT Basics
- Program Structure
- OpenGL Primitives

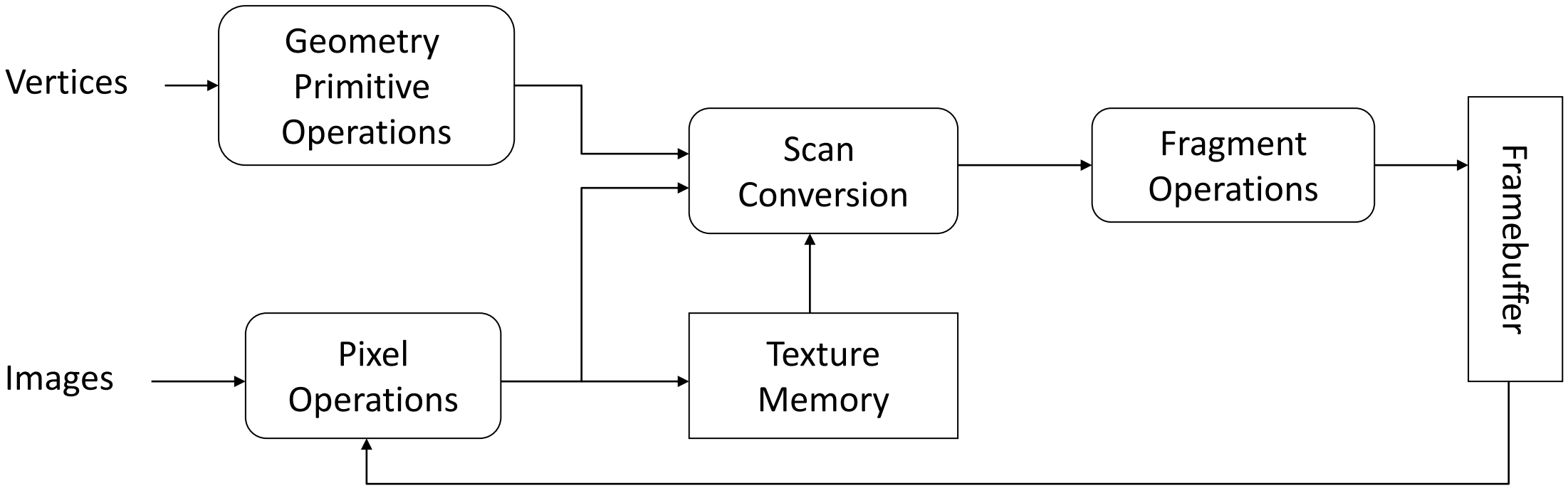
What Is OpenGL?

- **Graphics rendering API**
  - Fast, Simple
  - Window system independent
  - Operating system independent
  - Standard, available on many platforms
  - Geometric and pixel processing
  - High-quality color images composed of geometric and image primitives

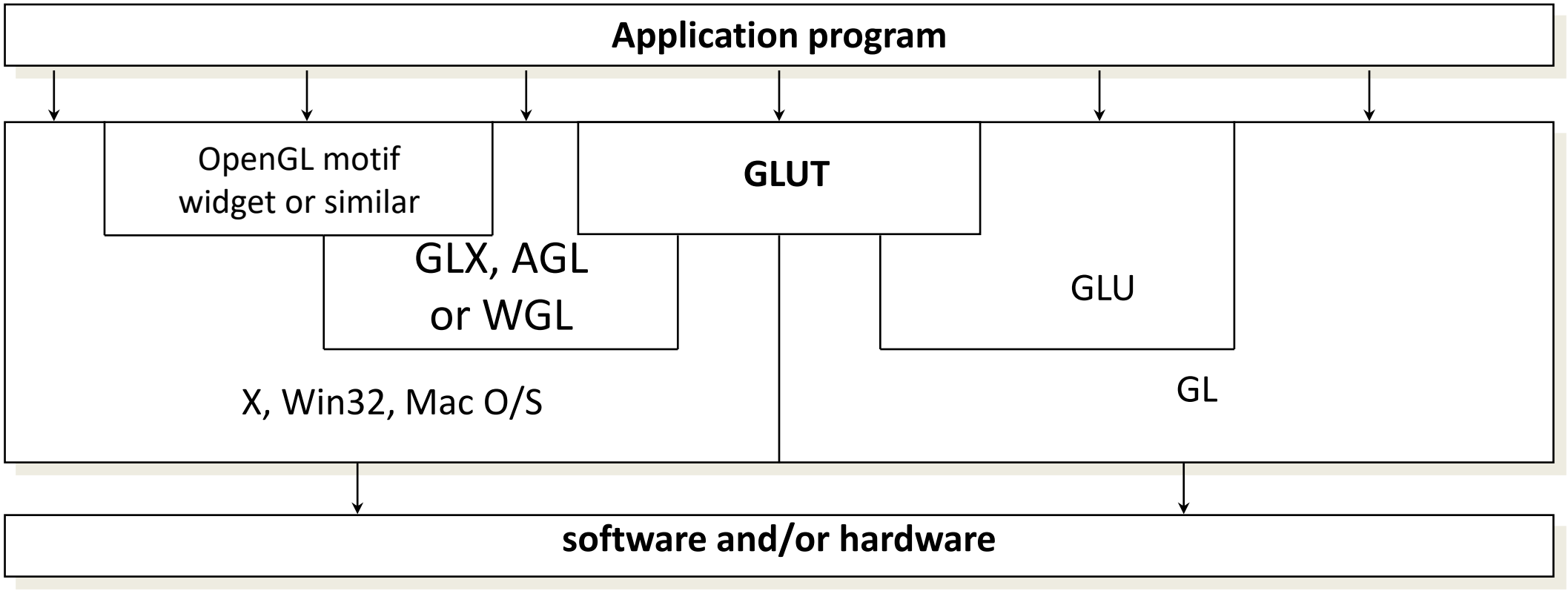
OpenGL as a Renderer

- **Geometric primitives**
  - points, lines and polygons
- **Image Primitives**
  - images and bitmaps
  - separate pipeline for images and geometry
    - linked through texture mapping
- **Rendering depends on state**
  - colors, materials, light sources, etc.

OpenGL Rendering Pipeline



Related APIs



Related APIs

- **AGL, GLX, WGL**
  - glue between OpenGL and windowing systems
- **GLU (OpenGL Utility Library)**
  - part of OpenGL
  - NURBS, tessellators, quadric shapes, etc.
- **GLUT (OpenGL Utility Toolkit)**
  - portable windowing API
  - not officially part of OpenGL

CONTENTs

- Graphics rendering API
- **GLUT Basics**
- Program Structure
- OpenGL Primitives

GLUT Basics

- **Provides functionality common to all window systems**
  - Simple, open-source library that works everywhere
  - Configure and open window
  - Input processing: keyboard, mouse, etc.
  - Register input callback functions
  - render
  - resize
  - Enter event processing loop

Preliminaries

- **Headers Files**

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
```

- **Libraries**
- **Enumerated Types**
  - OpenGL defines numerous types for compatibility
    - GLfloat, GLint, GLenum, etc.

CONTENTs

- Graphics rendering API
- GLUT Basics
- **Program Structure**
- OpenGL Primitives

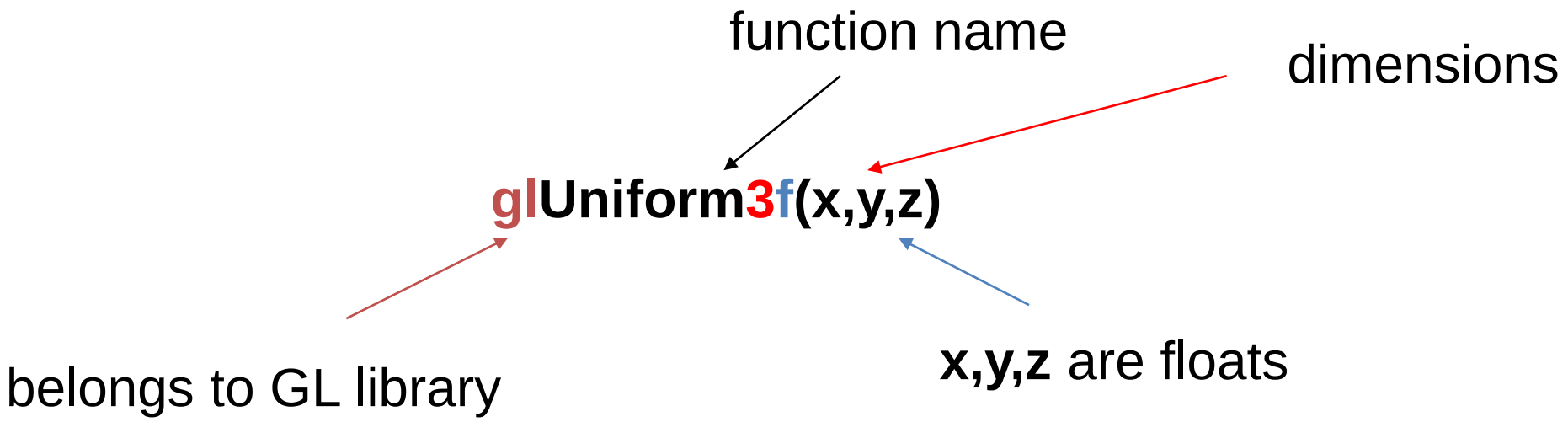
Program Structure

- **Most OpenGL programs have a similar structure that consists of the following functions**
  - main():
    - specifies the callback functions
    - opens one or more windows with the required properties
    - enters event loop (last executable statement)
  - init(): sets the state variables
    - Viewing, Attributes
  - initShader(): read, compile and link shaders
  - callbacks
    - Display function, Input and window functions

OpenGL Functions

- **Primitives**
  - Points, Line Segments, Triangles
- **Attributes**
- **Transformations**
  - Viewing, Modeling
- **Control (GLUT)**
- **Input (GLUT)**
- **Query**

OpenGL function Format



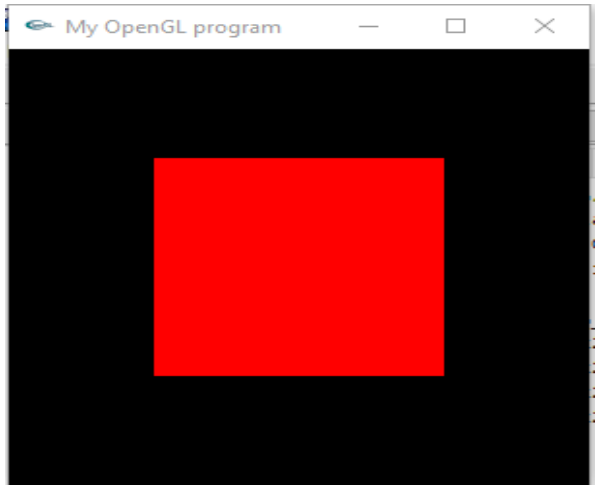
glUniform3fv(p)

p is a pointer to an array

A Simple Program

- Generate a red square on a solid background

```
#include <GL/glut.h>
void mydisplay(){
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0f, 0.0f, 0.0f);
        glBegin(GL_POLYGON);
                glVertex2f(-0.5f, -0.5f);
                glVertex2f( 0.5f, -0.5f);
                glVertex2f( 0.5f,  0.5f);
                glVertex2f(-0.5f,  0.5f);
        glEnd();
        glFlush();
}
int main(int argc, char** argv){
        glutCreateWindow("My OpenGL program");
        glutDisplayFunc(mydisplay);
        glutMainLoop();
}
```



Viewing in OpenGL

- Viewing consists of two parts
  - Object positioning: model view transformation matrix
  - View projection: projection transformation matrix
- OpenGL supports both perspective and orthographic viewing transformations
- OpenGL's camera is always at the origin, pointing in the –z direction
- Transformations move objects relative to the camera
- Matrices right-multiply top of stack.
  (Last transform in code is first actually applied)

Viewing in OpenGL

```
#include <GL/glut.h>
#include <stdlib.h>
int mouseoldx, mouseoldy ;    // For mouse motion
GLdouble eyeloc = 2.0 ;       // Where to look from; initially 0 -2, 2
void init (void) {
        glClearColor (0.0, 0.0, 0.0, 0.0);      // select clearing color
        glMatrixMode(GL_PROJECTION);      // initialize viewing values
        glLoadIdentity();

        glMatrixMode(GL_MODELVIEW) ;
        glLoadIdentity() ;
        gluLookAt(0,-eyeloc,eyeloc,0,0,0,0,1,1) ;
}
```

Callback functions for events

```
/* Defines what to do when various keys are pressed */
void keyboard (unsigned char key, int x, int y) {
        switch (key) {
           case 27:  exit(0) ;break ;          // Escape to quit
           default:  break ;
         }
}
/* Reshapes the window appropriately */
void reshape(int w, int h){
        glViewport (0, 0, (GLsizei) w, (GLsizei) h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(30.0, (GLdouble)w/(GLdouble)h, 1.0, 10.0) ;
}
```

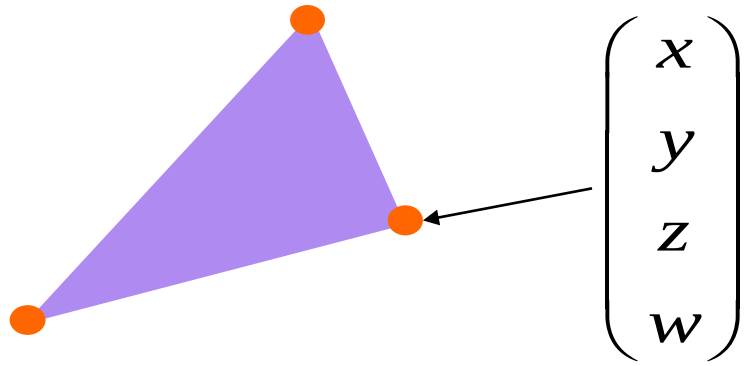
Callback functions for events: Mouse motion

```
/* Defines a Mouse callback to zoom in and out.  This is done by modifying
gluLookAt
   The actual motion is in mousedrag, mouse simply sets state for mousedrag */
   void mouse(int button, int state, int x, int y) {
        if (button == GLUT_LEFT_BUTTON) {
            if (state == GLUT_UP) {              // Do Nothing ;              }
        else if (state == GLUT_DOWN) {
            mouseoldx = x ; mouseoldy = y ; // so we can move wrt x , y
          }
        }
        else if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN){
            // Reset gluLookAt
           eyeloc = 2.0 ;
           glMatrixMode(GL_MODELVIEW) ;
           glLoadIdentity() ;
           gluLookAt(0,-eyeloc,eyeloc,0,0,0,0,1,1) ;
           glutPostRedisplay() ;
        }
   }
```

Callback functions for events: Mouse drag

```
void mousedrag(int x, int y) {
      int yloc = y - mouseoldy  ;              // the y coord to zoom in/out
      eyeloc  += 0.005*yloc ;                  // Where do we look from
      if (eyeloc < 0) eyeloc = 0.0 ;
      mouseoldy = y ;
      /* Set the eye location */
      glMatrixMode(GL_MODELVIEW) ;
      glLoadIdentity() ;
      gluLookAt(0,-eyeloc,eyeloc,0,0,0,0,1,1) ;
      glutPostRedisplay() ;
}
```

- Geometric objects are represented using **vertices**
- A vertex is a collection of generic attributes
  - positional coordinates
  - colors
  - texture coordinates
  - any other data associated with that point in space

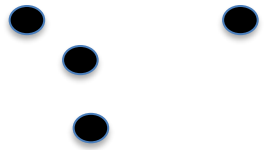


- Position stored in 4 dimensional homogeneous coordinates
- Vertex data must be stored in vertex buffer objects (VBOs)
- VBOs must be stored in vertex array objects (VAOs)

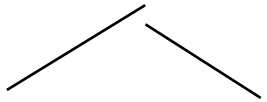
CONTENTs

- Graphics rendering API
- GLUT Basics
- Program Structure
- **OpenGL Primitives**

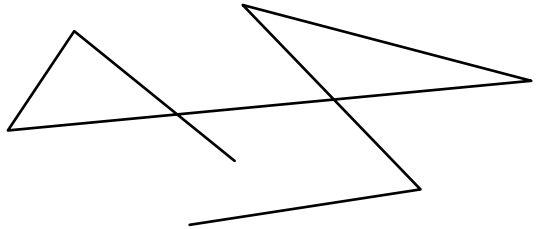
OpenGL Primitives



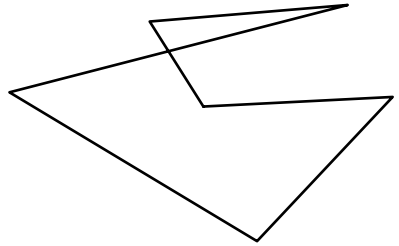
GL_POINTS



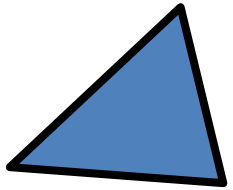
GL_LINES



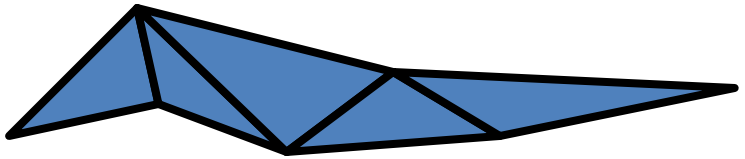
GL_LINE_STRIP



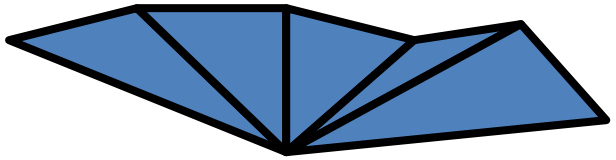
GL_LINE_LOOP



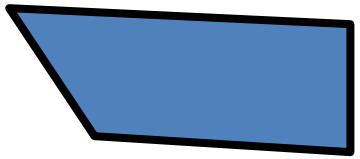
GL_TRIANGLES



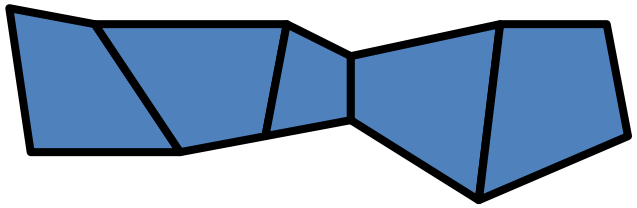
GL_TRIANGLE_STRIP



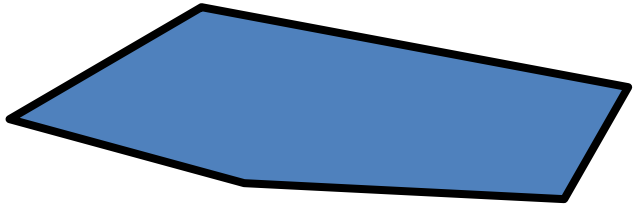
GL_TRIANGLE_FAN



GL_QUADS

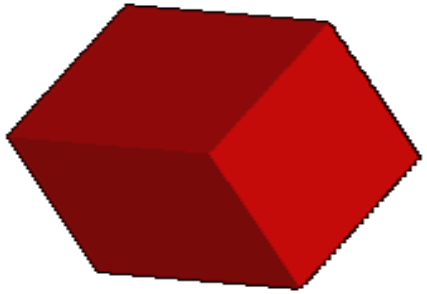


GL_QUAD_STRIP

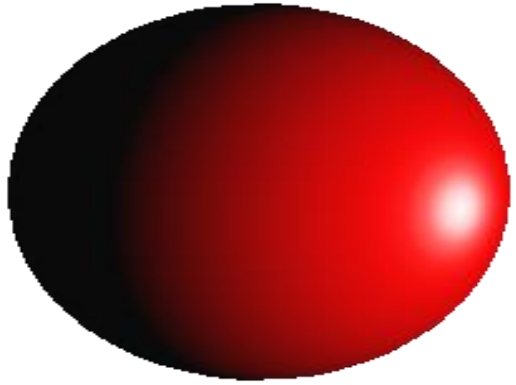


GL_POLYGON

GLUT 3D Primitives



Cube



Sphere

Teapot

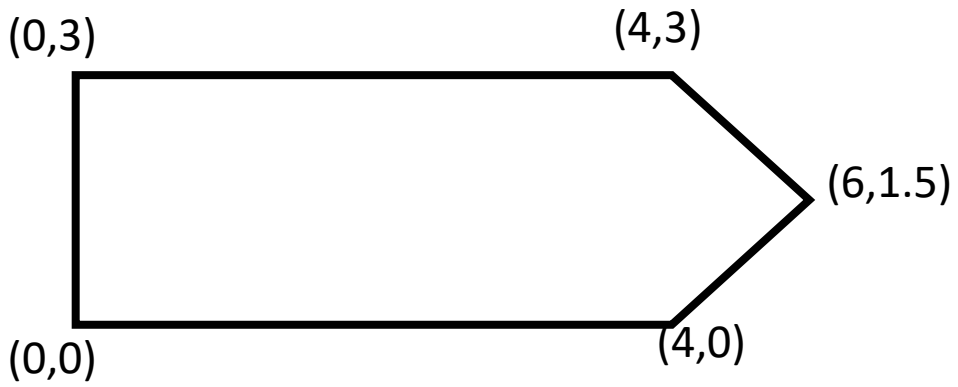
And others…

Geometry

- **Points (GL_POINTS)**
  - Stored in Homogeneous coordinates
- **Line segments (GL_LINES)**
- **Polygons**
  - Simple, convex (take your chances with concave)
  - Tessellate, GLU for complex shapes
  - Rectangles: glRect
- **Special cases:** strips, loops, triangles, fans, quads
- **More complex primitives (GLUT):** Sphere, teapot, cube,…

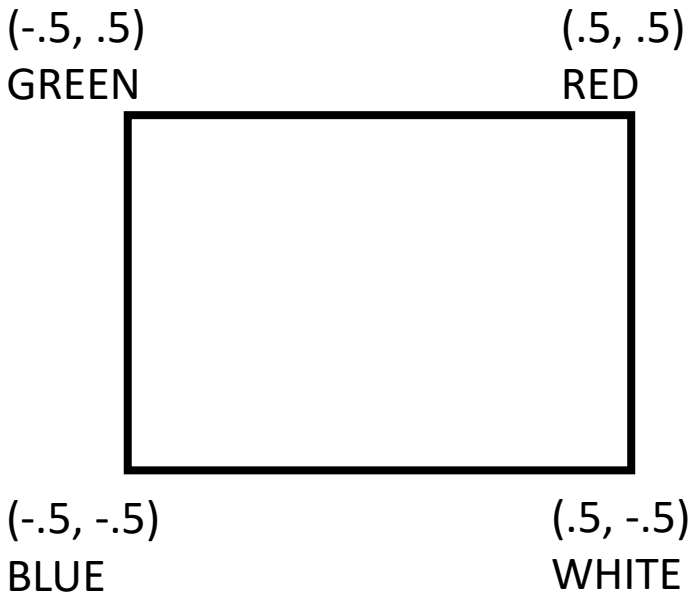
Specifying Geometry

```
glBegin(GL_POLYGON) ;
    glVertex2f (4.0, 0.0) ;
    glVertex2f (6.0, 1.5) ;
    glVertex2f (4.0, 3.0) ;
    glVertex2f (0.0, 3.0) ;
    glVertex2f (0.0, 0.0) ;
    // glColor, glIndex, glNormal, glMaterial…
    // Other GL commands invalid between begin and end
glEnd() ;
```



Drawing in Display()

```
void display(void){
  glClear (GL_COLOR_BUFFER_BIT);
   glBegin(GL_POLYGON);
     glColor3f (1.0, 0.0, 0.0);
     glVertex3f (0.5, 0.5, 0.0);
     glColor3f (0.0, 1.0, 0.0);
     glVertex3f (-0.5, 0.5, 0.0);
     glColor3f (0.0, 0.0, 1.0);
     glVertex3f (-0.5, -0.5, 0.0);
     glColor3f (1.0, 1.0, 1.0);
     glVertex3f (0.5, -0.5, 0.0);
  glEnd();
  glFlush () ;
}
```



SUMMARY

- Graphics rendering API
- GLUT Basics
- Program Structure
- OpenGL Primitives

Enjoy the Course…!