CS 4204 Computer Graphics

OpenGL Basics

Yong Cao

Virginia Tech

References:

2001 Siggraph, "An Interactive Introduction to OpenGL Programming", Dave Shreiner,Ed Angel, Vicki Shreiner

Official Presentation from Text book "Computer Graphics using OpenGL", chapter 2

OpenGL and GLUT Overview

What is OpenGL & what can it do for me?

OpenGL in windowing systems

Why GLUT

A GLUT program template

What Is OpenGL?

Open Graphics Standard Specification for INTERACTIVE 3D Graphics

Graphics rendering API

- high-quality color images composed of geometric and image primitives
- window system independent
- operating system independent

What is it for us?

Open Graphics Standard

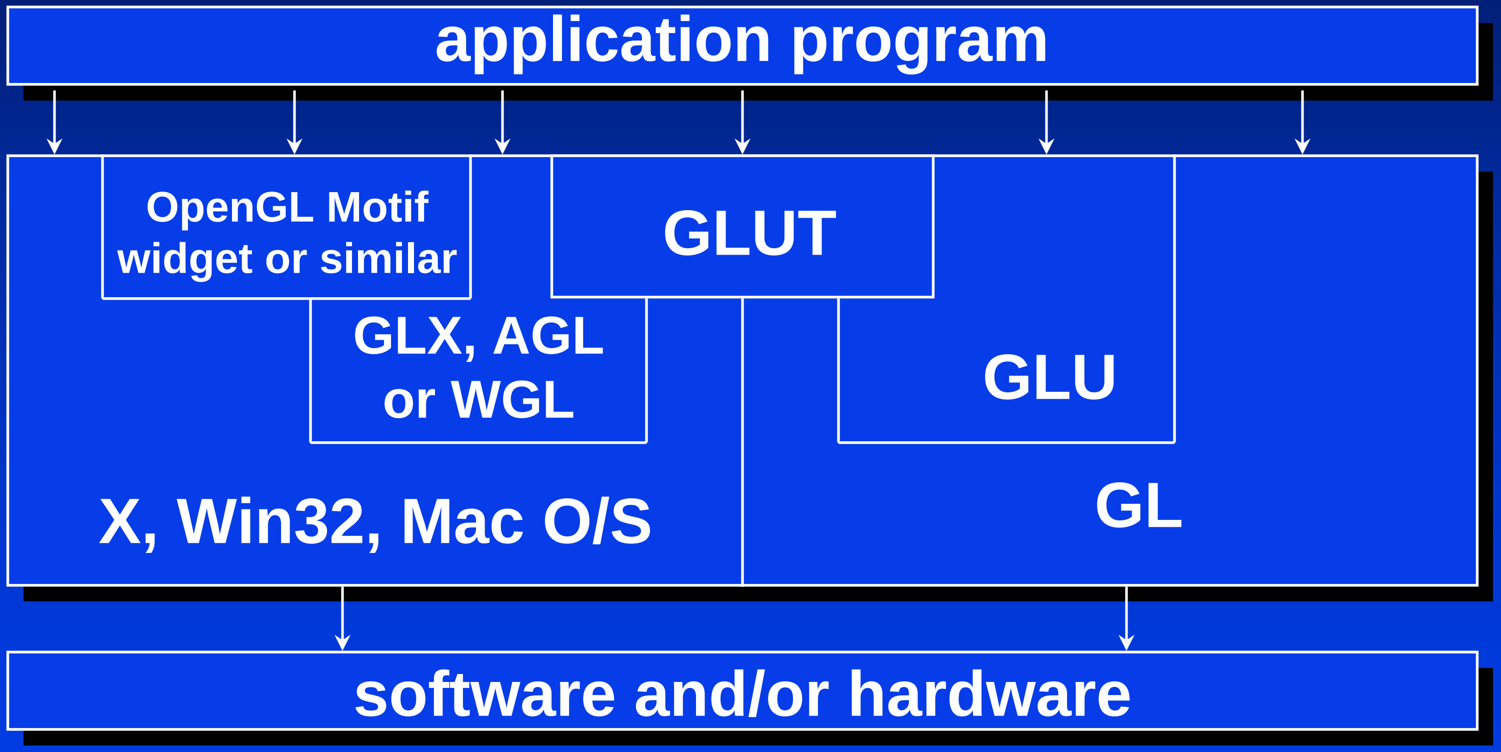
- API
- Library
- State Machine
- Pipeline

GL: Core

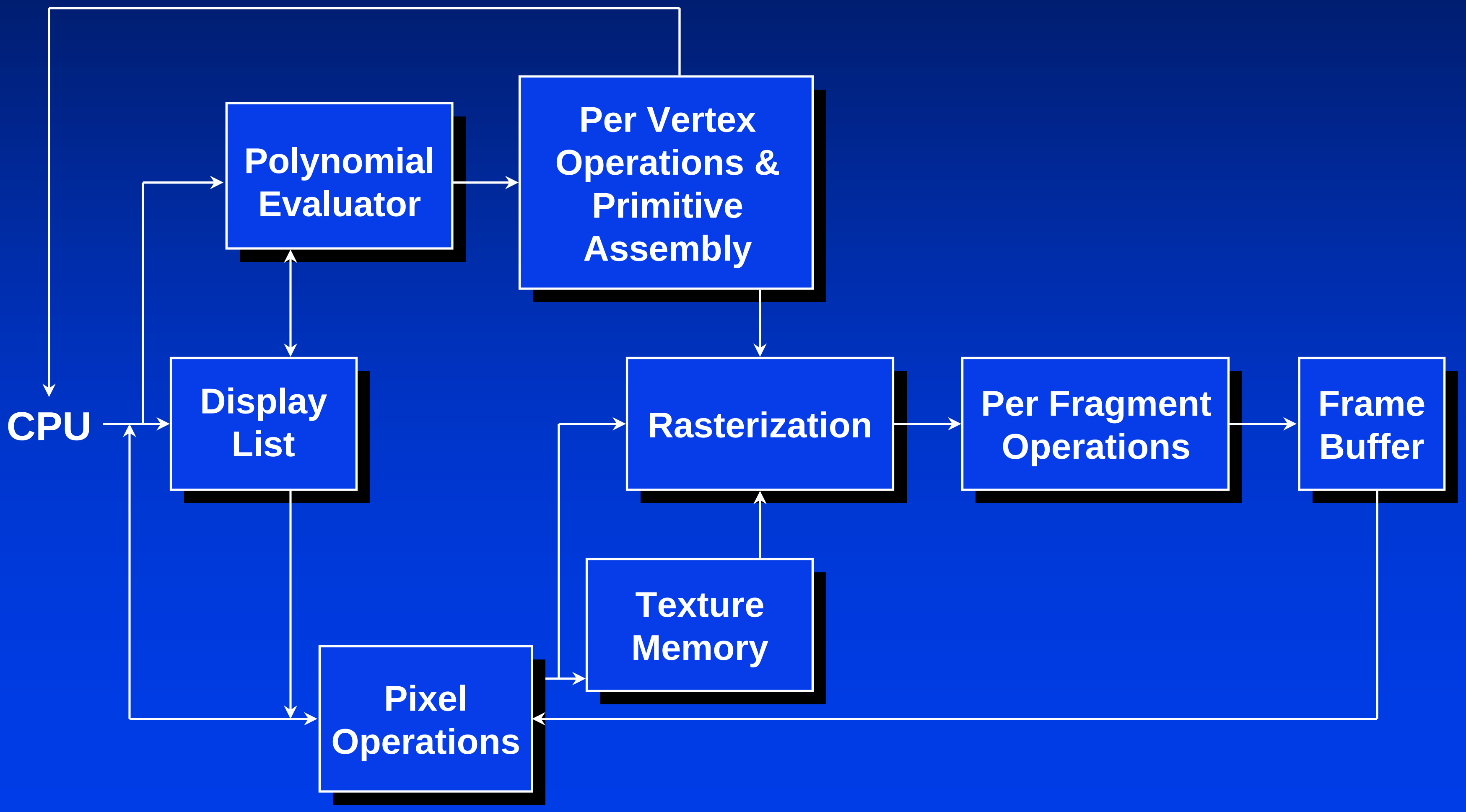
GLU: Higher level utilities

GLUT: Windowing and interaction

OpenGL and Related APIs



OpenGL Architecture (Pipeline)



Preliminaries

Headers:

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

Libraries:

- glut32.lib,
- opengl32.lib,
- glu32.lib

Dynamic libraries

- glut32.dll

Let’s setup a OpenGL project in Visual Studio 2003.

Setting up a GLUT Window

```
int main(int argc, char** argv)
{
  glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowPosition (0, 0);
  glutInitWindowSize(640,640);
  glutCreateWindow(argv[0]);

  // register callbacks
  glutReshapeFunc (myReshapeCB);
  glutKeyboardFunc(myKeyboardCB );
  glutMouseFunc(myMouseCB) ;
  glutMotionFunc(myMotionCB) ;
  glutDisplayFunc(display);

  myinit() ;          // initialize
  glutMainLoop();     // start the main  loop
  return 0;           // never reached
}
```

Mouse callbacks

```
void myMouseCB(int button, int state, int x, int y) {        // start or end interaction
      if( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN ) {
              printf("Left button down\n") ;
      }
      if( button == GLUT_LEFT_BUTTON && state == GLUT_UP ) {
              printf("Left button up\n") ;
      }
      glutPostRedisplay() ;            // Tell the system to redraw the window
}
void myMotionCB(int x, int y) {                     // interaction (mouse motion)
      printf("Moving the mouse\n") ;
      glutPostRedisplay() ;
}
```

Keyboard callback

```
void myKeyboardCB(unsigned char key, int x, int y) {
    switch (key) {
        case 'q':
        case 27:
                exit(0);
                break;
    }
}
```

Display function

```
void display(void)  {
        glMatrixMode(GL_PROJECTION) ;
        glLoadIdentity() ;
        glMatrixMode(GL_MODELVIEW) ;
        glLoadIdentity();

         glClearColor(0.0f,0.0f,0.0f,0.0f);   // set the background colour
        // OK, now clear the screen with the background colour
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glColor3f(0.5,0,0) ;                // set the current color
        glutWireSphere( 1.0, 10, 10 ) ;                 // draw a sphere
        glutSwapBuffers();                  // swap the buffers (show the image)
}
```

Elements of a scene in OpenGL

Geometric Primitives

Material properties

Light sources

Copyright Pixar

Primitives in OpenGL

Points

Lines

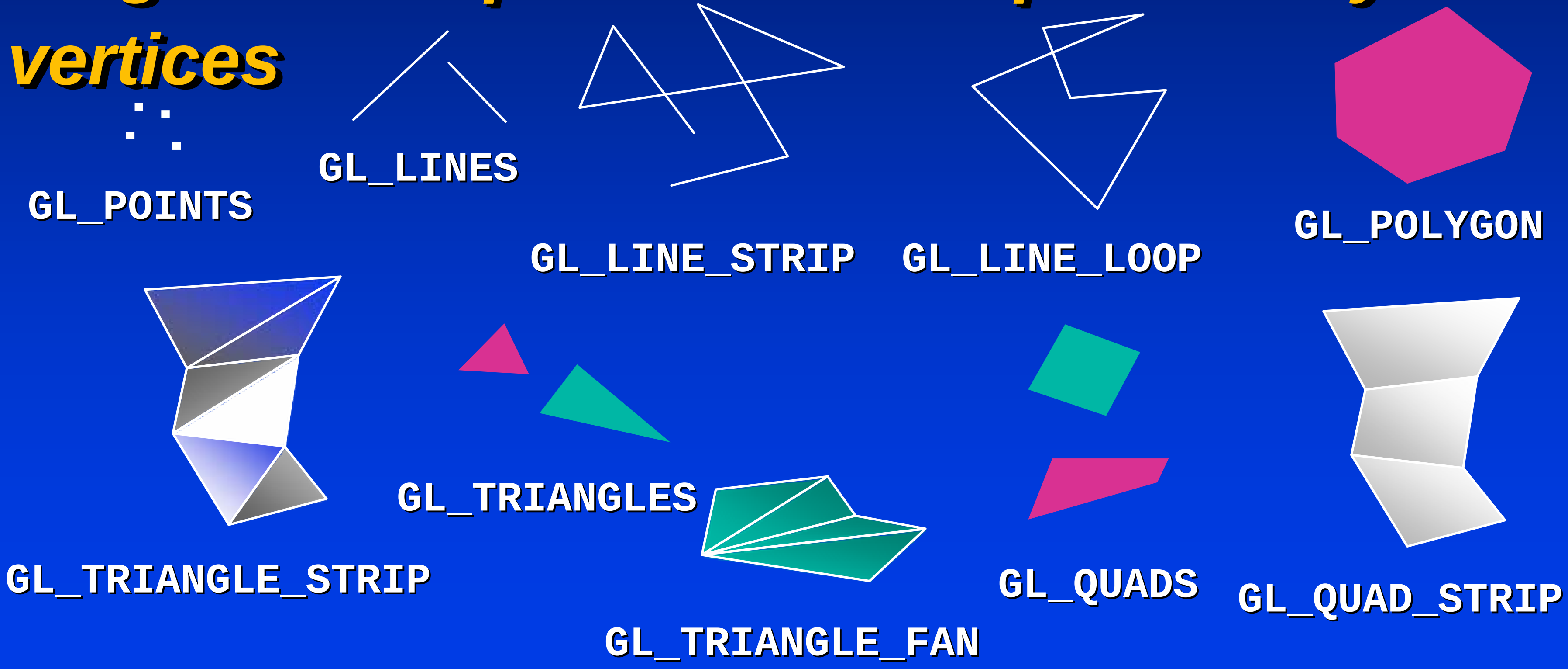
Curves (piece-wise linear approximation)

Polygons

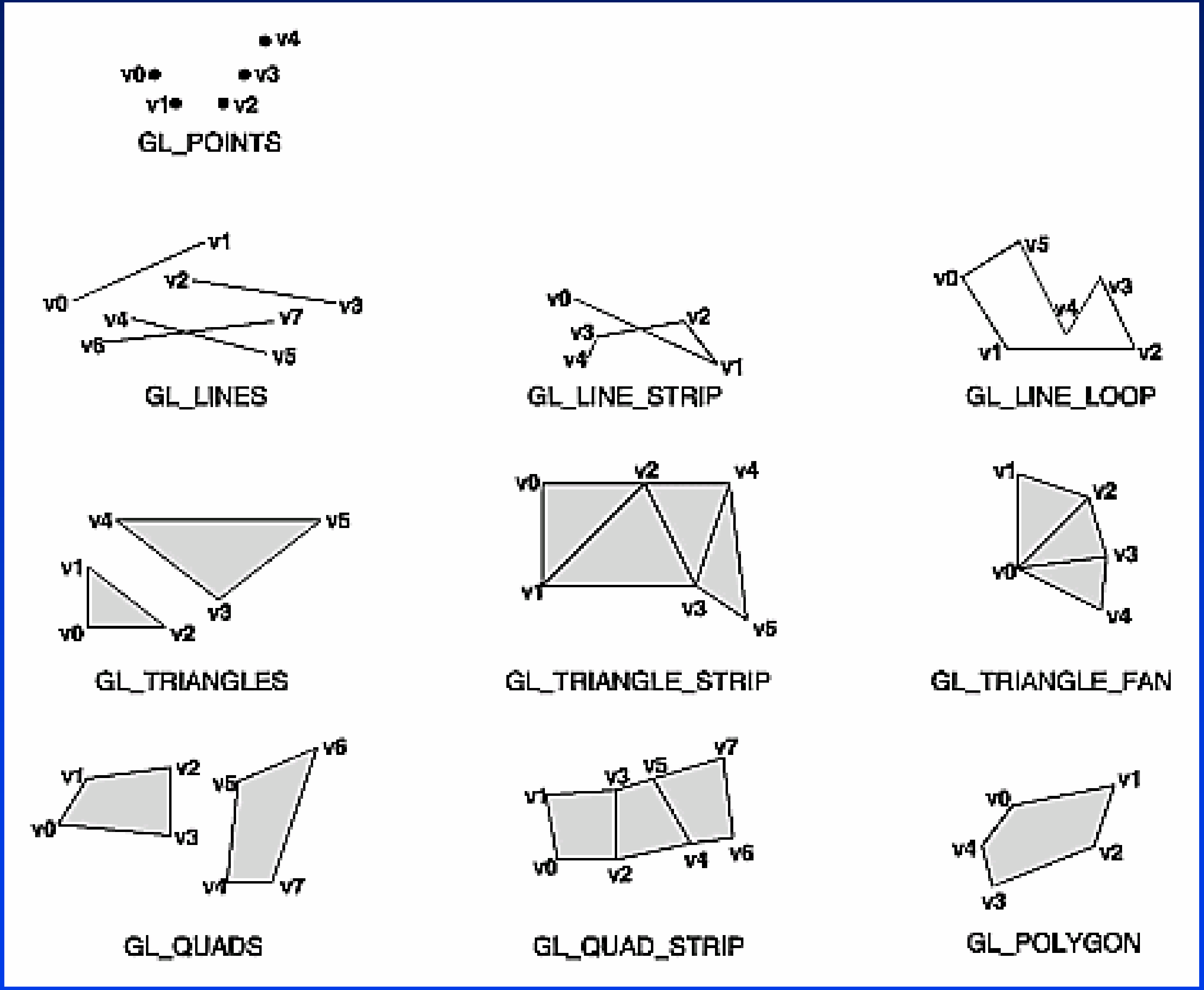
Surfaces (polygonal approximation)

OpenGL Geometric Primitives

All geometric primitives are specified by vertices



OpenGL Geometric Primitives (Contiune)



Simple Example

```
void drawRhombus( GLfloat color[] )
{
   glBegin( GL_QUADS );
   glColor3fv( color );
   glVertex2f( 0.0, 0.0 );
   glVertex2f( 1.0, 0.0 );
   glVertex2f( 1.5, 1.118 );
   glVertex2f( 0.5, 1.118 );
   glEnd();
}
```

OpenGL Command Formats

```
glVertex3fv( v )
```

Number of components

```
2 - (x,y)
3 - (x,y,z)
4 - (x,y,z,w)
```

Data Type

```
b  - byte
ub - unsigned byte
s  - short
us - unsigned short
i  - int
ui - unsigned int
f  - float
d  - double
```

Vector

omit “v” for scalar form

```
glVertex2f( x, y )
```

Specifying Geometric Primitives

Primitives are specified using

glBegin(*primType*);

glEnd();

- *primType* determines how vertices are combined

```
GLfloat red, green, blue;
Glfloat coords[3];
glBegin( primType );
for ( i = 0; i < nVerts; ++i ) {
  glColor3f( red, green, blue );
  glVertex3fv( coords );
}
glEnd();
```

Types

GLint

GLfloat

GLdouble

Points

```
glBegin(GL_POINTS)
  glVertex3f(GLfloat x, GLfloat y, GLfloat z) ;
  glVertex2i(GLint x, GLint y) ;
  glVertex3dv(GLdouble p[3] ) ;
glEnd() ;
```

Point details

glPointSize(float size) ;

glColor3f(GLfloat r, GLfloat g, Glfloat b) ;

Lines

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

Line strip

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

Line loop

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

Line details

glLineWidth(GLfloat w) ;

glColor3f(GLfloat r,GLfloat g,GLfloat b) ;

glLineStipple(Glint factor, GLushort pattern) ;

glEnable(GL_LINE_STIPPLE) ;

Polygons in OpenGL

```
glPolygonMode(GL_FRONT,GL_FILL) ;
glPolygonMode(GL_BACK,GL_LINE) ;
glColor3f(red,green,blue) ;
glBegin(GL_POLYGON)
      glNormal3f(v1,v2,v3) ;
      glVertex3f(x1,y1,z1) ;
      ….
      glNormal3f(v1n,v2n,v3n) ;
      glVertex3f(xn,yn,zn) ;
glEnd()  ;
```

Higher Primitives in GLUT

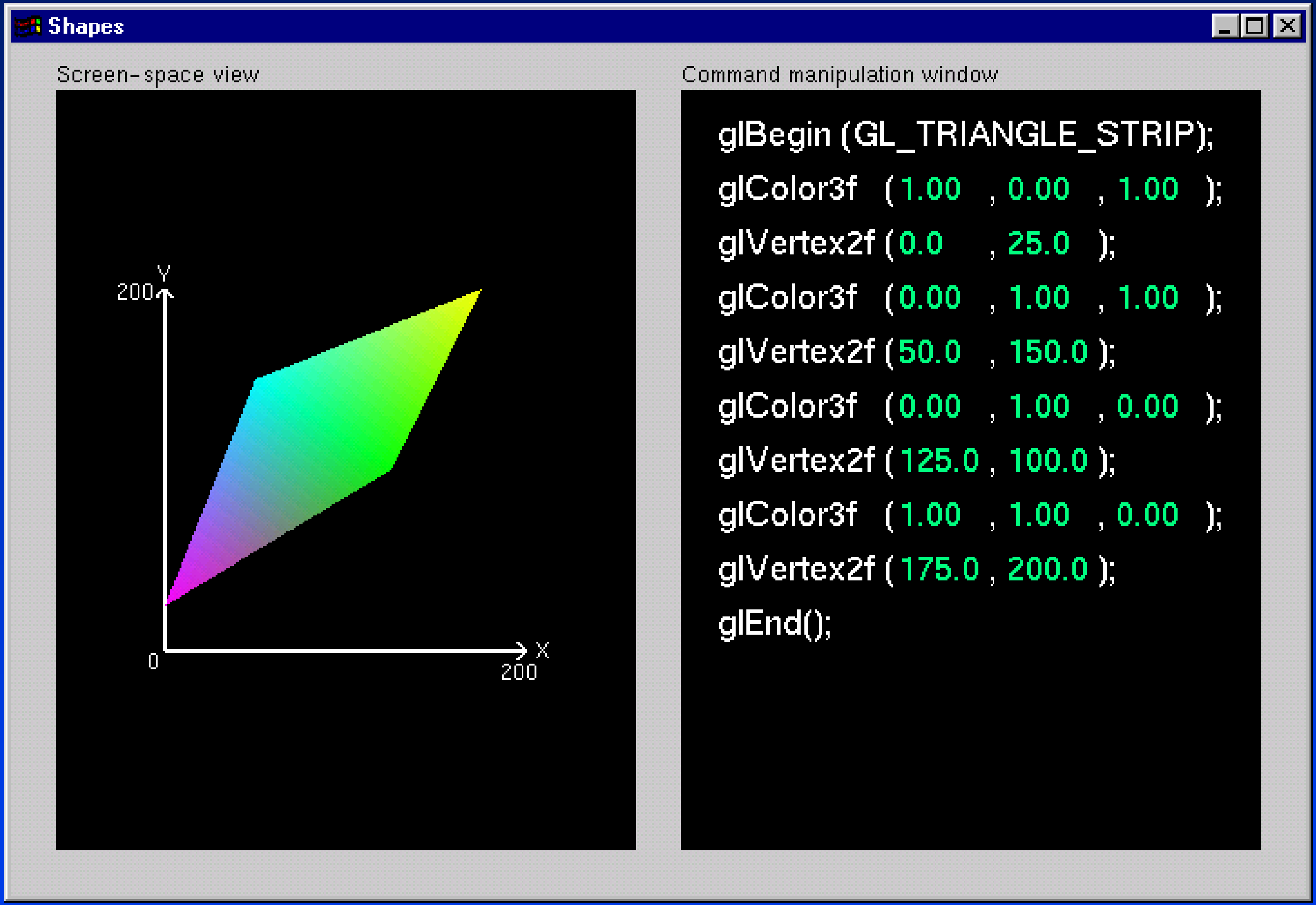
glutSolidShere() ;

glutSolidCube();

glutSolidCone() ;

glutSolidTeapot() ;

Shapes Tutorial



```
glBegin (GL_TRIANGLE_STRIP);
glColor3f  (1.00  , 0.00  , 1.00  );
glVertex2f (0.0   , 25.0  );
glColor3f  (0.00  , 1.00  , 1.00  );
glVertex2f (50.0  , 150.0 );
glColor3f  (0.00  , 1.00  , 0.00  );
glVertex2f (125.0 , 100.0 );
glColor3f  (1.00  , 1.00  , 0.00  );
glVertex2f (175.0 , 200.0 );
glEnd();
```

Reading Material and Homework

- ***Check Website.***
- ***Quiz on reading materials next class.***
- ***Homework due on next Monday midnight***
- ***Send homework to TA via email***