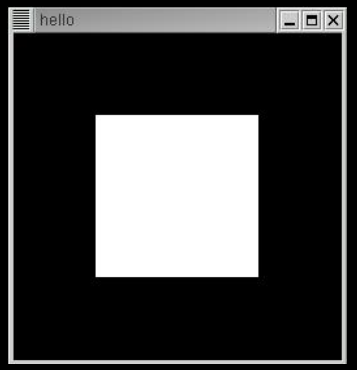
A look at the OpenGL Code

OpenGL Programming Guide:
Chapter 1:
Introduction to OpenGL

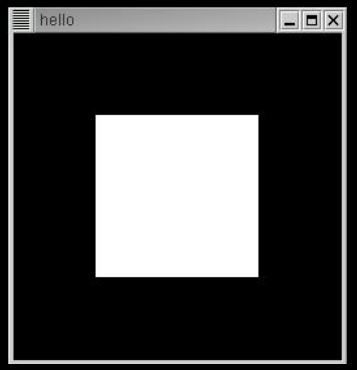
http://www.opengl.org/resources/code/basics.html

Sample code (simplistic)




```
#include <whateverYouNeed.h>

main() {

    InitializeAWindowPlease();

    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}
```

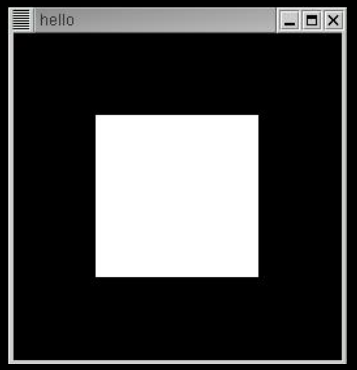
Sample code (simplistic)



OpenGL does not do it!

```
#include <whateverYouNeed.h>

main() {

   InitializeAWindowPlease();

   glClearColor (0.0, 0.0, 0.0, 0.0);
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (1.0, 1.0, 1.0);
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f (0.25, 0.25, 0.0);
      glVertex3f (0.75, 0.25, 0.0);
      glVertex3f (0.75, 0.75, 0.0);
      glVertex3f (0.25, 0.75, 0.0);
   glEnd();
   glFlush();

   UpdateTheWindowAndCheckForEvents();
}
```

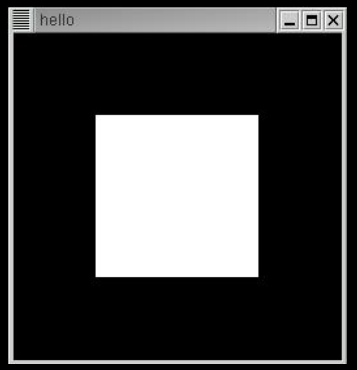
Sample code (simplistic)




```
#include <whateverYouNeed.h>

main() {

    InitializeAWindowPlease();

    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}
```

Clear color = black

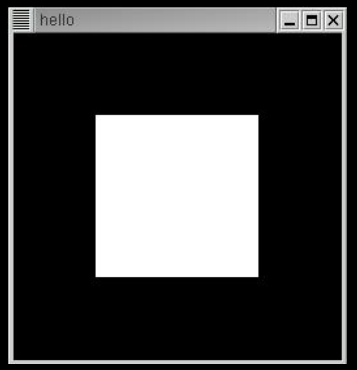
Sample code (simplistic)




```
#include <whateverYouNeed.h>

main() {

    InitializeAWindowPlease();

    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}
```

Clear color = black
Do the clearing

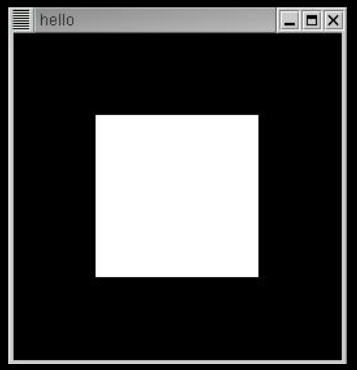
Sample code (simplistic)




```
#include <whateverYouNeed.h>

main() {

   InitializeAWindowPlease();

   glClearColor (0.0, 0.0, 0.0, 0.0);
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (1.0, 1.0, 1.0);
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f (0.25, 0.25, 0.0);
      glVertex3f (0.75, 0.25, 0.0);
      glVertex3f (0.75, 0.75, 0.0);
      glVertex3f (0.25, 0.75, 0.0);
   glEnd();
   glFlush();

   UpdateTheWindowAndCheckForEvents();
}
```

Current color = white

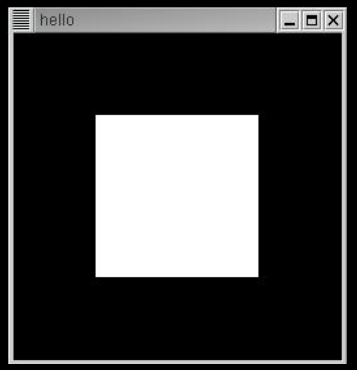
Sample code (simplistic)




```
#include <whateverYouNeed.h>

main() {

   InitializeAWindowPlease();

   glClearColor (0.0, 0.0, 0.0, 0.0);
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (1.0, 1.0, 1.0);
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f (0.25, 0.25, 0.0);
      glVertex3f (0.75, 0.25, 0.0);
      glVertex3f (0.75, 0.75, 0.0);
      glVertex3f (0.25, 0.75, 0.0);
   glEnd();
   glFlush();

   UpdateTheWindowAndCheckForEvents();
}
```

Specify 3D view

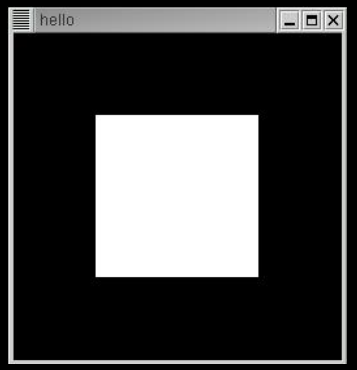
Sample code (simplistic)




```
#include <whateverYouNeed.h>

main() {

    InitializeAWindowPlease();

    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}
```

Start polygon definition

Sample code (simplistic)

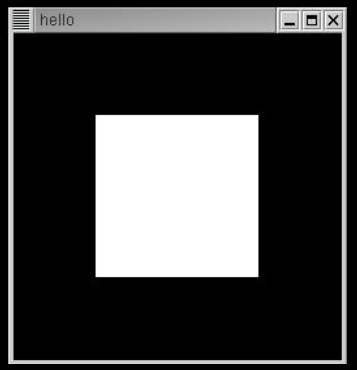



```
#include <whateverYouNeed.h>

main() {

    InitializeAWindowPlease();

    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}
```

Start polygon definition

List vertices

Sample code (simplistic)



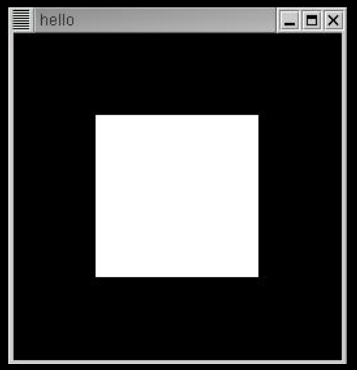

```
#include <whateverYouNeed.h>

main() {

    InitializeAWindowPlease();

    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}
```

Start polygon definition

List vertices

End polygon definition

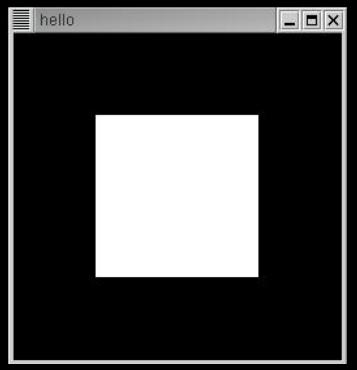
Sample code (simplistic)




```
#include <whateverYouNeed.h>

main() {

    InitializeAWindowPlease();

    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}
```

Flush the buffer

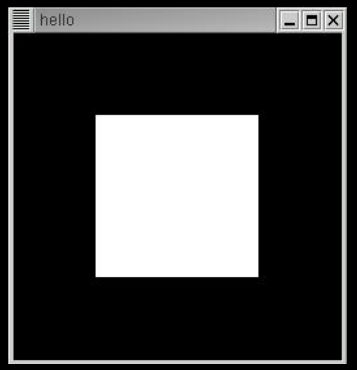
Sample code (simplistic)




```
#include <whateverYouNeed.h>

main() {

    InitializeAWindowPlease();

    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}
```

E.g., wait a while

Sample code (simplistic)




```
#include <whateverYouNeed.h>

main() {

    InitializeAWindowPlease();

    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush();

    UpdateTheWindowAndCheckForEvents();
}
```

OpenGL does not do it!

Clear color = black
Do the clearing
Current color = white
Specify 3D view
Start polygon definition

List vertices

End polygon definition
Flush the buffer

E.g., wait a while

Callbacks

A callback is a reference to executable code, or a piece of executable code, that is passed as an argument to other code. This allows a lower-level software layer to call a subroutine (or function) defined in a higher-level layer.

(Wikipedia)

Sample code (2)

Uses GLUT (GL utility toolkit)

```
/*
 * Declare initial window size, position, and display mode
 * (single buffer and RGBA).  Open window with "hello"
 * in its title bar.  Call initialization routines.
 * Register callback function to display graphics.
 * Enter main loop and process events.
 */
int main(int argc, char** argv)
{
   glutInit(&argc, argv);          <-- Initialize GLUT + OpenGL stuff
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (250, 250);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("hello");
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glClear (GL_COLOR_BUFFER_BIT);
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
   glutDisplayFunc(display);       <-- Register callback
   glutMainLoop();                 <-- Enter main loop, process events
   return 0;   /* ANSI C requires main to return int. */
}
```

Sample code (1)

```
/* hello.c: introductory OpenGL program. */
#include <GL/glut.h>

void display(void)
{
/* clear all pixels  */
   glClear (GL_COLOR_BUFFER_BIT);

/* draw white polygon (rectangle) with corners at
   (0.25, 0.25, 0.0) and (0.75, 0.75, 0.0) */
   glColor3f (1.0, 1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f (0.25, 0.25, 0.0);
      glVertex3f (0.75, 0.25, 0.0);
      glVertex3f (0.75, 0.75, 0.0);
      glVertex3f (0.25, 0.75, 0.0);
   glEnd();

/* don't wait! */
   glFlush ();
}
```

Introducing more callbacks

```
/*
 * Introduce callbacks on reshape and mouse events
*/

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (250, 250);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("hello");
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glClear (GL_COLOR_BUFFER_BIT);
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
   glutReshapeFunc(reshape);   <-- Register reshape callback
   glutMouseFunc(mouse);       <-- Register mouse event callback
   glutDisplayFunc(display);   <-- Register display callback
   glutMainLoop();             <-- Enter main loop, process events
   return 0;
}
```

Modifying the display() function

```
void display(void)
{
/* clear all pixels  */
   glClear (GL_COLOR_BUFFER_BIT);

/* draw white polygon (rectangle) with corners at
   (0.25, 0.25, 0.0) and (0.75, 0.75, 0.0) */
   glColor3f (1.0, 1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f (0.25, 0.25, 0.0);
      glVertex3f (0.75, 0.25, 0.0);
      glVertex3f (0.75, 0.75, 0.0);
      glVertex3f (0.25, 0.75, 0.0);
   glEnd();

   printf("I am drawing!\n");

   glFlush ();
}
```

Defining the mouse() function

```
void mouse(int button, int state, int x, int y)
{
   if (state == GLUT_DOWN)
      printf("Button %d pressed at x=%d, y=%d\n",
         button, x, y);
   if (state == GLUT_UP)
         printf("Button %d released at x=%d, y=%d\n",
         button, x, y);
}
```

Defining the reshape() function

```
void reshape(int w, int h)
{
   glViewport(0, 0, w, h);
   printf("New window size w=%d, h=%d\n", w, h);
}
```

callbacks.c

- Show code
- Run demo

Fundamentals of OpenGL programming

OpenGL Programming Guide: Chapter 2

Specifying vertices

Syntax

void **glVertex** {234} {sifd} [v] (TYPEcoords);

number of arguments

Specifying vertices

Syntax

void **glVertex** {234} {sifd} [v] (TYPEcoords);

type of arguments

Specifying vertices

Syntax

void **glVertex** {234} {sifd} [v] (TYPEcoords);

if vector

Specifying vertices

Syntax

void **glVertex** {234} {sifd} [v] (TYPEcoords);

the arguments

Specifying vertices

Syntax

void **glVertex** {234} {sifd} [v] (TYPEcoords);

Examples

```
glVertex2s(2, 3);
glVertex3d(0.0, 0.0, 3.1415926535898);
glVertex4f(2.3, 1.0, -2.2, 2.0);

GLdouble dvect[3] = {5.0, 9.0, 1992.0};
glVertex3dv(dvect);
```

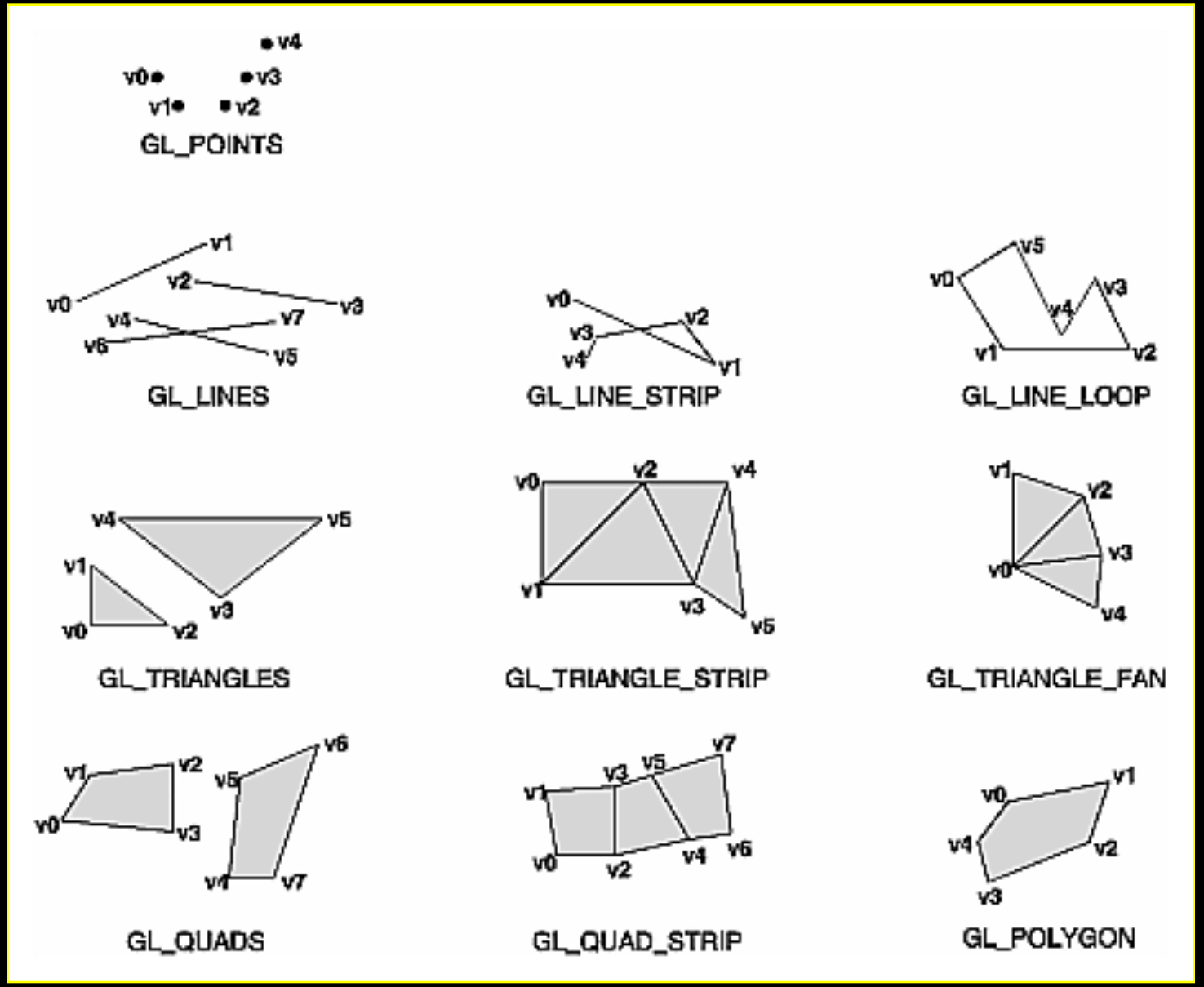
OpenGL drawing primitives


```
glBegin(GL_POLYGON);
   glVertex2f(0.0, 0.0);
   glVertex2f(0.0, 3.0);
   glVertex2f(4.0, 3.0);
   glVertex2f(6.0, 1.5);
   glVertex2f(4.0, 0.0);
glEnd();
```

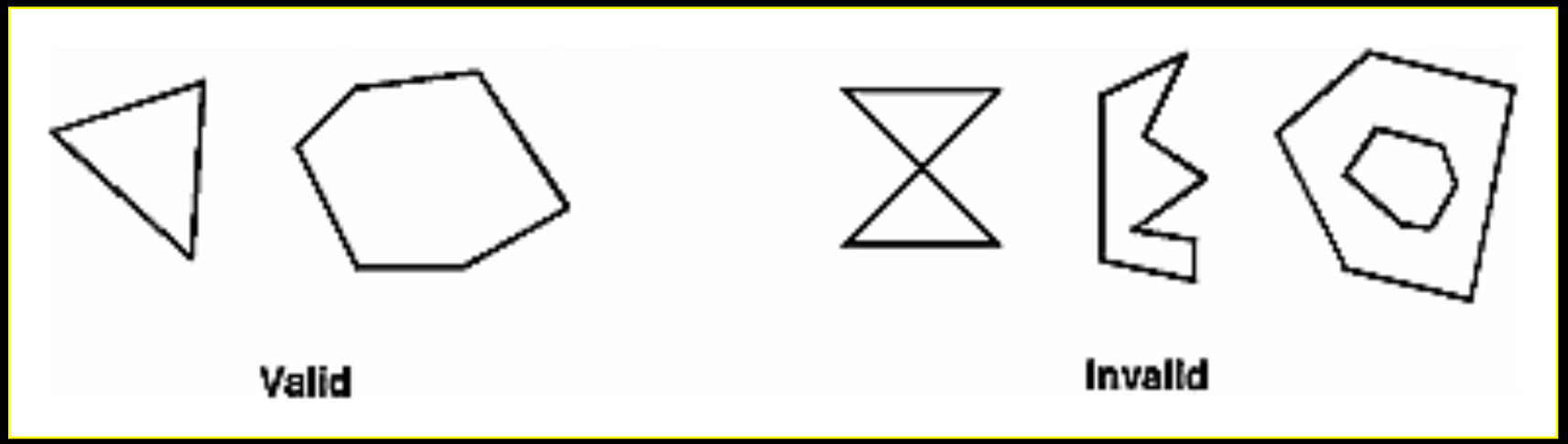
end of vertex-data list

```
glBegin(GL_POINTS);
   glVertex2f(0.0, 0.0);
   glVertex2f(0.0, 3.0);
   glVertex2f(4.0, 3.0);
   glVertex2f(6.0, 1.5);
   glVertex2f(4.0, 0.0);
glEnd();
```

OpenGL drawing primitives



Valid and invalid polygons



A shortcut for rectangles

void glRect{sifd}(TYPEx1, TYPEy1, TYPEx2, TYPEy2);

void glRect{sifd}v(TYPE*v1, TYPE*v2);

Valid commands between glBegin() and glEnd()

glVertex*()	set vertex coordinates
glColor*()	set current color
glIndex*()	set current color index
glMaterial*()	set material properties
glEdgeFlag*()	control drawing of edges
glArrayElement()	extract vertex array data
glNormal*()	set normal vector coordinates
glTexCoord*()	set texture coordinates
glEvalCoord*()	generate coordinates
glEvalPoint*()	generate coordinates
glCallList()	execute display list
glCallLists()	execute display lists

A sample valid construct

```
#define PI 3.1415926535898
GLint circle_points = 100;
glBegin(GL_LINE_LOOP);
for (i = 0; i < circle_points; i++) {
   angle = 2*PI*i/circle_points;
   glVertex2f(cos(angle), sin(angle));
}
glEnd();
```

The end