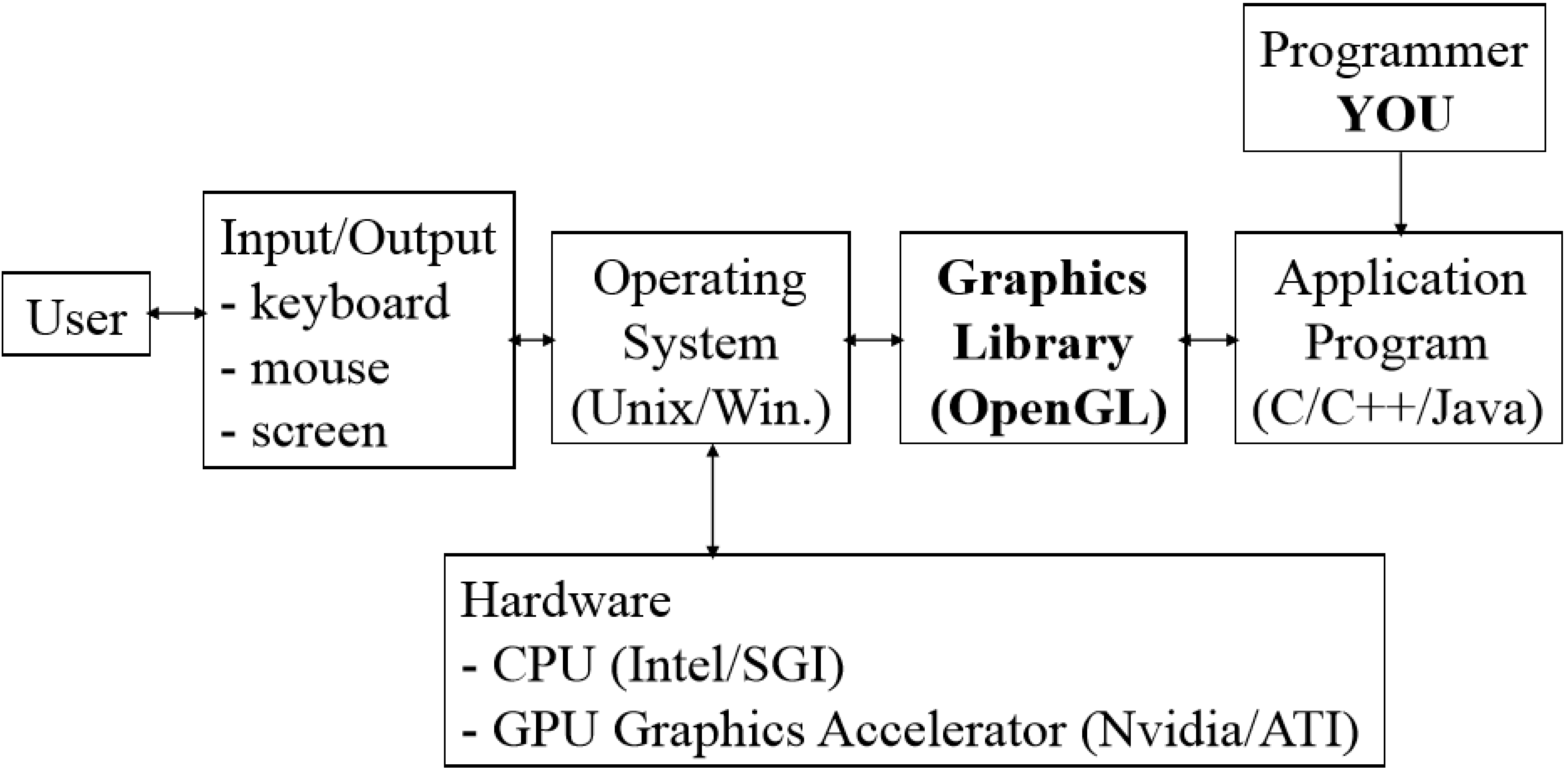
# Introduction to Computer Graphics with OpenGL/GLUT

# What is OpenGL

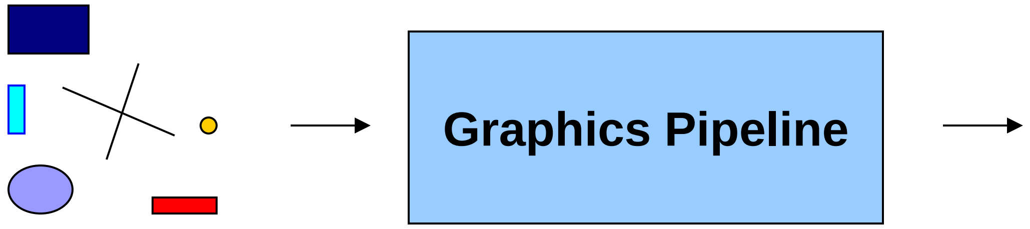

Programmer
YOU
User
Input/Output
- keyboard
- mouse
- screen
Operating System (Unix/Win.)
Graphics Library (OpenGL)
Application Program (C/C++/Java)
Hardware
- CPU (Intel/SGI)
- GPU Graphics Accelerator (Nvidia/ATI)

## What Is OpenGL?

- OpenGL is a computer graphics rendering *application programming interface,* or API (for short)
  - With it, you can generate high-quality color images by rendering with geometric and image primitives
  - It forms the basis of many interactive applications that include 3D graphics
  - By using OpenGL, the graphics part of your application can be
    - operating system independent
    - window system independent

# OpenGL Basics

- Rendering
  - Typically execution of OpenGL commands
  - Converting geometric/mathematical object descriptions into frame buffer values
- OpenGL can render:
  - Geometric primitives
    - Lines, points, polygons, etc…
  - Bitmaps and Images
    - Images and geometry linked through texture mapping



## OpenGL and GLUT

- GLUT (OpenGL Utility Toolkit)
  - A supporting library
    - A portable windowing API
    - Easier to show the output of your OpenGL application
  - Handles:
    - Window creation,
    - OS system calls
      - Mouse buttons, movement, keyboard, etc…
    - Callbacks

## Sample Program

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

void main(int argc, char** argv)
{
        int mode = GLUT_RGB|GLUT_DOUBLE;
   glutInitDisplayMode( mode );
   glutInitWindowSize( 500,500 );
   glutCreateWindow( “Simple” );
   init();
   glutDisplayFunc( display );
   glutMainLoop();
}
```

## Sample Program

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

void main(int argc, char** argv)
{
        int mode = GLUT_RGB|GLUT_DOUBLE;
   glutInitDisplayMode( mode );
   glutInitWindowSize( 500,500 );
   glutCreateWindow( “Simple” );
   init();
   glutDisplayFunc( display );
   glutMainLoop();
}
```

**Specify the display Mode – RGB or color Index, single or double Buffer**

## Sample Program

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

void main(int argc, char** argv)
{
        int mode = GLUT_RGB|GLUT_DOUBLE;
   glutInitDisplayMode( mode );
   glutInitWindowSize( 500,500 );
   glutCreateWindow( "Simple" );
   init();
   glutDisplayFunc( display );
   glutMainLoop();
}
```

**Create a window Named "simple" with resolution 500 x 500**

## Sample Program

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

void main(int argc, char** argv)
{
        int mode = GLUT_RGB|GLUT_DOUBLE;
    glutInitDisplayMode( mode );
    glutInitWindowSize( 500,500 );
    glutCreateWindow( “Simple” );
    init();                              ←    Your OpenGL initialization
    glutDisplayFunc( display );               code (Optional)
    glutMainLoop();
}
```

## Sample Program

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

void main(int argc, char** argv)
{
        int mode = GLUT_RGB|GLUT_DOUBLE;
   glutInitDisplayMode( mode );
   glutInitWindowSize( 500,500 );
   glutCreateWindow( “Simple” );
   init();
   glutDisplayFunc( display );
   glutMainLoop();
}
```

$\longleftarrow$ **Register your call back functions**

## glutMainLoop()

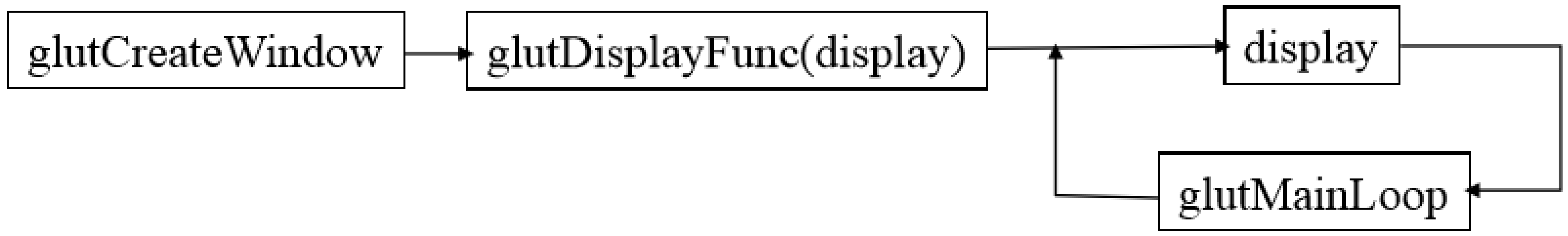
```
#include <GL/glut.h>
#include <GL/gl.h>

int main(int argc, char** argv)
{
        int mode = GLUT_RGB|GLUT_DOUBLE;
   glutInitDisplayMode(mode);
   glutInitWindowSize(500,500);
   glutCreateWindow("Simple");
   init();
   glutDisplayFunc(display);
   glutKeyboardFunc(key);
   glutMainLoop();
}
```

**The program goes into an infinite loop waiting for events**

## Summarization of Main

- glutCreateWindow() - creates a window of a pre-specified size.
- glutDisplayFunc(display) - calls a user specified function “display” whenever window needs to be drawn
- glutMainLoop() - enter an event processing loop so that graphics application continues to run & respond to user input until exited



## OpenGL Syntax

- All OpenGL commands have the prefix ‘gl’
  - glClear()  glColor3f()  glVertex3f()
- Constants are defined with prefix ‘GL’ & use ‘_’ to separate words   GL_COLOR_BUFFER_BIT

- American spelling: Color

# GLUT Callback Functions

- **Callback function** : Routine to call when an event happens
  - Window resize or redraw
  - User input (mouse, keyboard)
  - Animation (render many frames)
- “Register” callbacks with GLUT
  - glutDisplayFunc( my_display_func );
  - glutKeyboardFunc( my_key_events_func );
  - glutMouseFunc ( my_mouse_events_func );

# Rendering Callback

- Callback function where all our drawing is done
- Every GLUT program must have a display callback

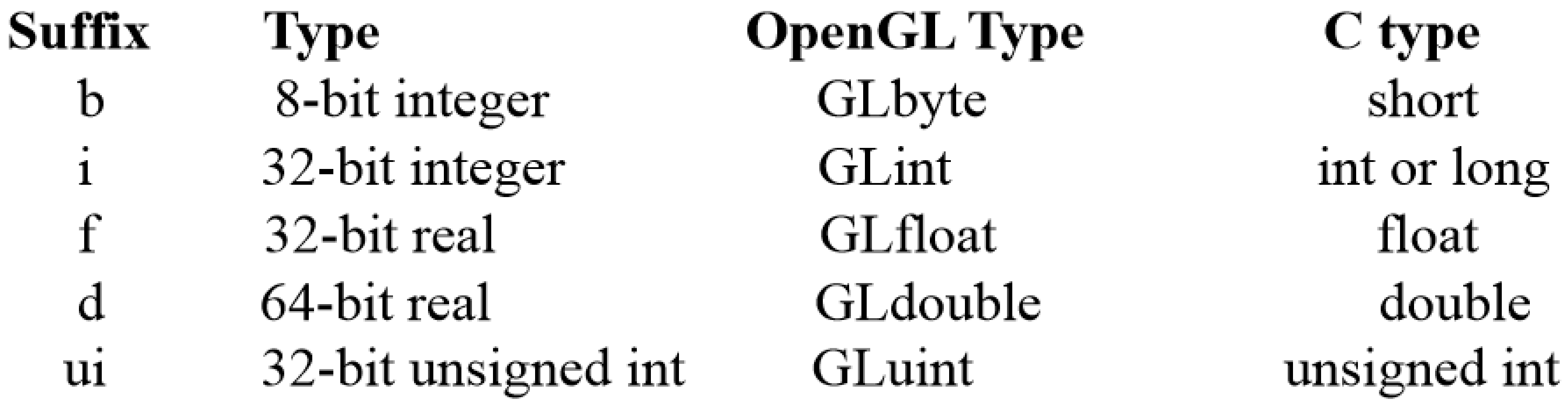
- glutDisplayFunc( ***my_display_func*** ); /* this part is in main.c */

```
void my_display_func (void )
{
   glClear( GL_COLOR_BUFFER_BIT );
   glBegin( GL_TRIANGLE );
     glVertex2f( 0.0, 0.0);
     glVertex2f( 0.5,4.5);
      glVertex2f( 1.0, 0.0);
   glEnd();
   glFlush();
}
```

## OpenGL Variable Types

- Type information is appended to the end of the command
- glColor3f(r,g,b)   -  a colour of 3 floating point components
- glVertex3f(x,y,z) - a vertex with 3 floating point coordinates
- glVertex2f(x,y)    - a vertex with 2 floating point coordinates

- Different versions of the same function exist for different types  glVertex2i(p,q)
  - - vertex with 2 integer coordinates



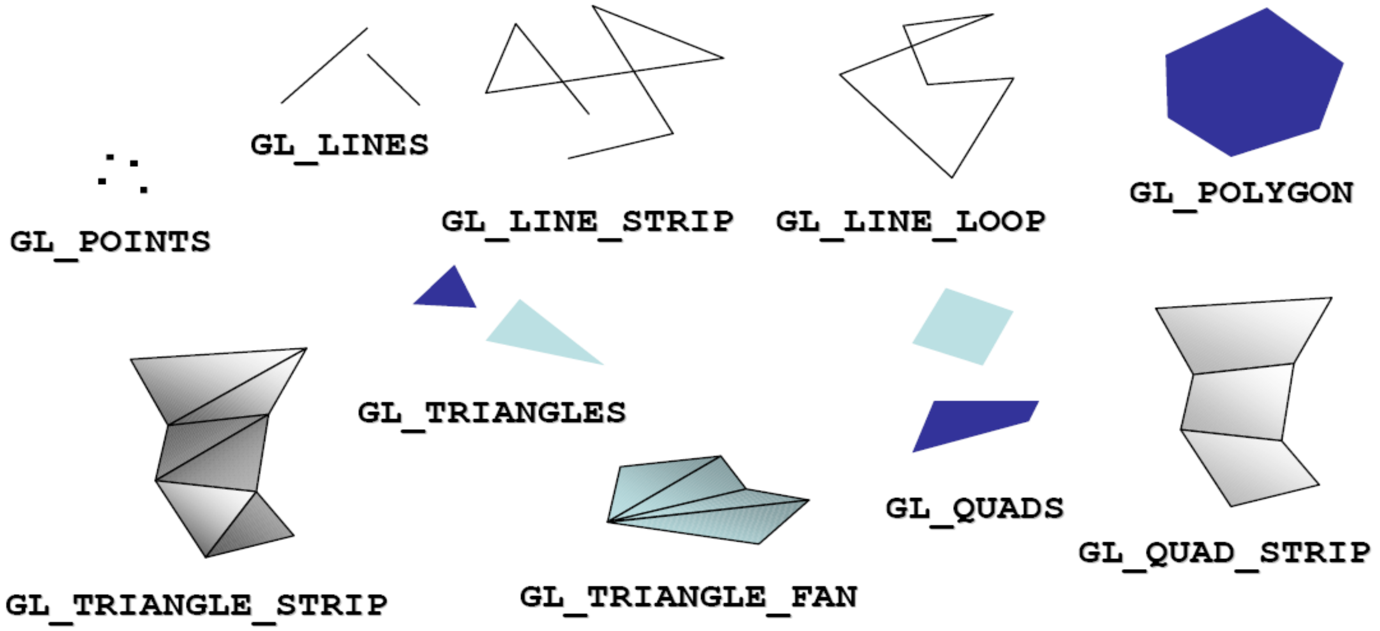
| Suffix | Type | OpenGL Type | C type |
|---|---|---|---|
| b | 8-bit integer | GLbyte | short |
| i | 32-bit integer | GLint | int or long |
| f | 32-bit real | GLfloat | float |
| d | 64-bit real | GLdouble | double |
| ui | 32-bit unsigned int | GLuint | unsigned int |

## Color in OpenGL

- It’s just not “color”!
- RGB Three-component color model: red + green + blue
- OpenGL color components are in the range [0.0,1.0]
- Each component represents the intensity of that color
  glColor3f(0.1,0.4,0.7);    /* r,g,b colour intensities */
- Alpha channel - represents the opacity or transparency - RGBA colour model
- glColor4f(1.0,0.0,0.0,0.5);  /* red semi-transparent */

# OpenGL Geometric Primitives

- The geometry is specified by vertices.



# Vertices and Primitives

- Primitives are specified using

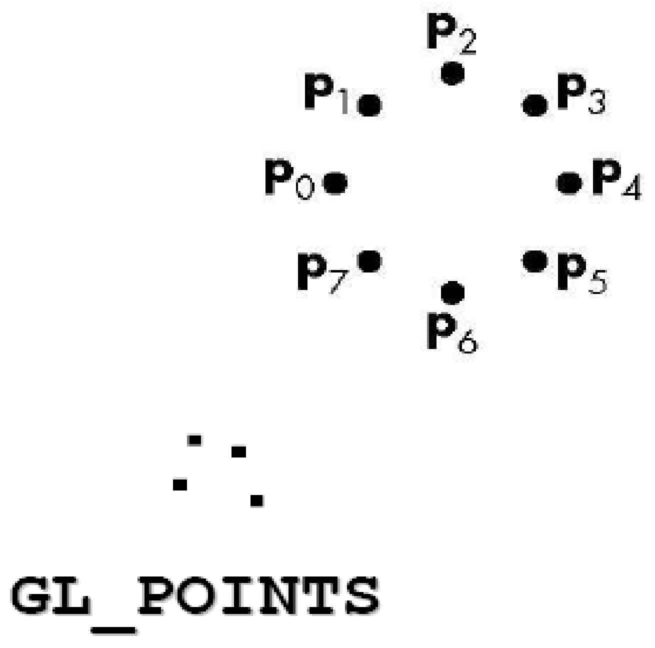
```
glBegin( primType );
…
glEnd();
```

  - `primType` determines how vertices are combined

# Vertices and Primitives

- Points, **`GL_POINTS`**
  - Individual points
  - Point size can be altered
    - ***glPointSize (float size)***

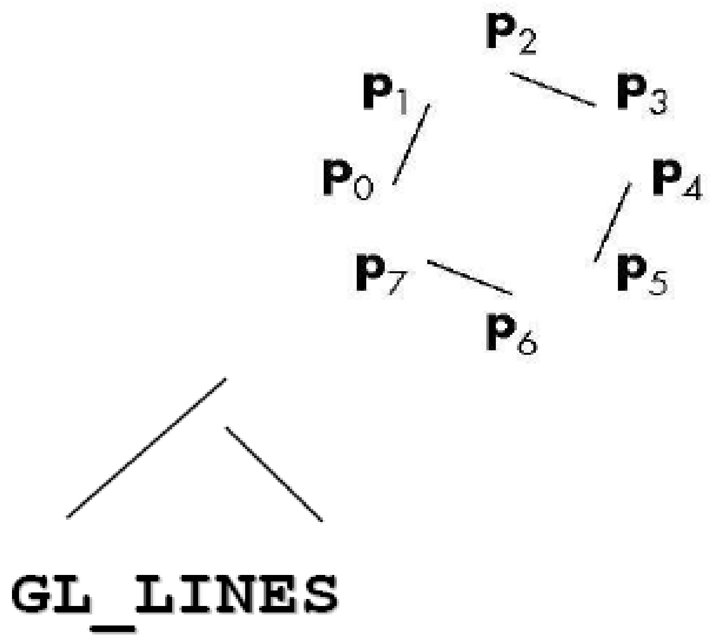
```
glBegin(GL_POINTS);
glColor3f( color );
glVertex2f( P0.x, P0.y );
glVertex2f( P1.x, P1.y );
glVertex2f( P2.x, P2.y );
glVertex2f( P3.x, P3.y );
glVertex2f( P4.x, P4.y );
glVertex2f( P5.x, P5.y );
glVertex2f( P6.x, P6.y );
glVertex2f( P7.x, P7.y );
glEnd();
```



# Vertices and Primitives

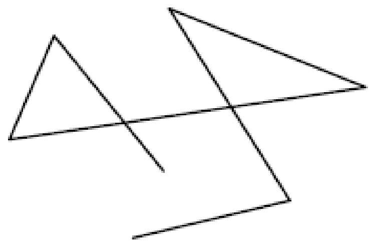
- Lines, **`GL_LINES`**
  - Pairs of vertices interpreted as individual line segments
  - Can specify line width using:
    - ***glLineWidth (float width)***

```
glBegin(GL_LINES);
glColor3f( color );
glVertex2f( P0.x, P0.y );
glVertex2f( P1.x, P1.y );
glVertex2f( P2.x, P2.y );
glVertex2f( P3.x, P3.y );
glVertex2f( P4.x, P4.y );
glVertex2f( P5.x, P5.y );
glVertex2f( P6.x, P6.y );
glVertex2f( P7.x, P7.y );
glEnd();
```

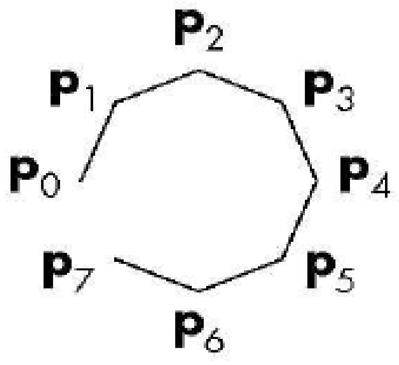


# Vertices and Primitives

- Line Strip, **`GL_LINE_STRIP`**
  - series of connected line segments

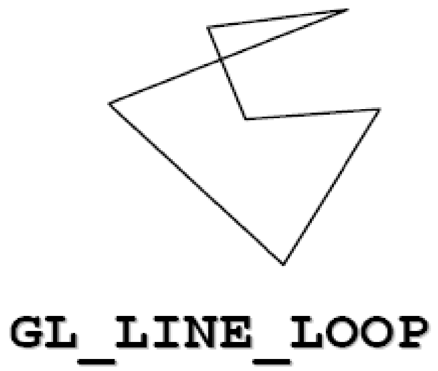


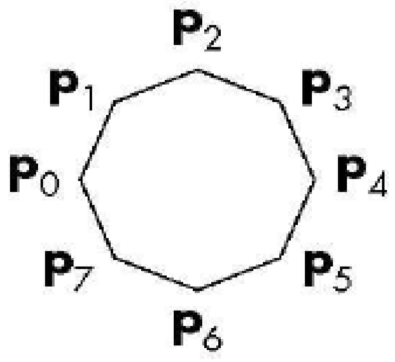
**`GL_LINE_STRIP`**



# Vertices and Primitives

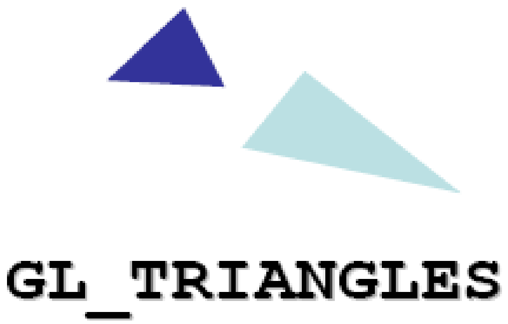
- Line Loop, **`GL_LINE_LOOP`**
  - Line strip with a segment added between last and first vertices

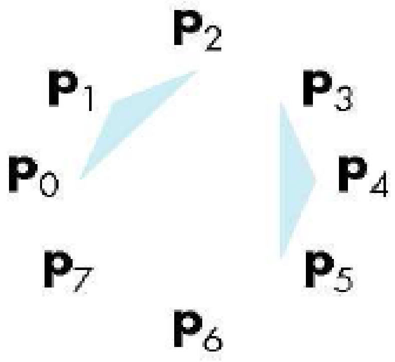




## Vertices and Primitives

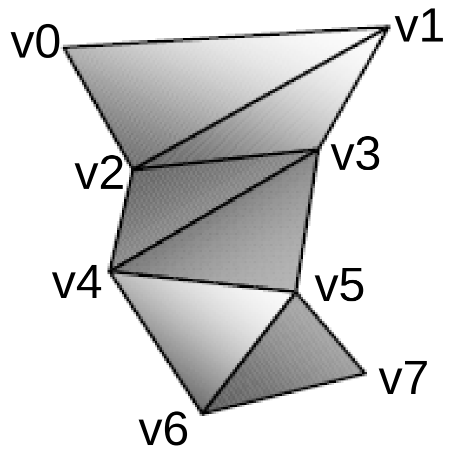
- Triangles , **`GL_TRIANGLES`**
  - triples of vertices interpreted as triangles





## Vertices and Primitives

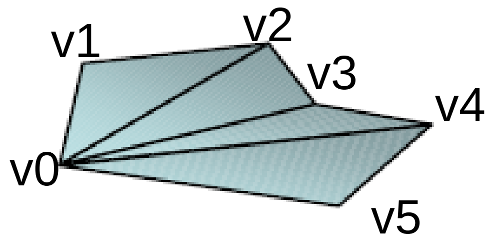
- Triangle Strip , **`GL_TRIANGLE_STRIP`**
  - linked strip of triangles



**`GL_TRIANGLE_STRIP`**

# Vertices and Primitives

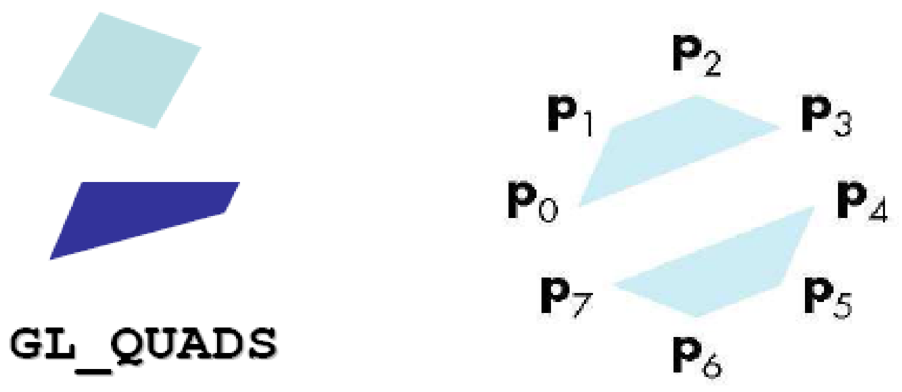
- Triangle Fan , **`GL_TRIANGLE_FAN`**
  - linked fan of triangles



**`GL_TRIANGLE_FAN`**

## Vertices and Primitives

- Quads , **`GL_QUADS`**
  - quadruples of vertices interpreted as four-sided polygons



## Vertices and Primitives

- Between glBegin/ glEnd, those opengl commands are allowed:
  - glVertex*() : set vertex coordinates
  - glColor*() : set current color
  - glIndex*() : set current color index (Later)
  - glNormal*() : set normal vector coordinates (Light.)(Later)
  - glTexCoord*() : set texture coordinates (Texture) (Later)

## Program Structure

- Most OpenGL programs have a similar structure that consists of the following functions
  - **`main()`**:
    - defines 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
  - callbacks
    - Display function
    - Input and window functions

```
#include<windows.h>
#include<GL/Glut.h>
void MyInit(void)
{
                glClearColor(0.0,0.0,0.0,0.0); //Set the background color
                glColor3f(1.0,1.0,0.0); //set the color
                glPointSize(10.0);//Set the point
                glMatrixMode(GL_PROJECTION);//projection matrix before drawing the objects in
your scene to set the view volume.
                gluOrtho2D(0.0,640.0,0.0,480.0);  // define a 2D orthographic projection matrix
Left,right,bottom,top)
}
void MyDisplay()
{
                glClear(GL_COLOR_BUFFER_BIT);//Clear the buffer
                glBegin(GL_POINTS);
                glVertex2i(260,230);
                glVertex2i(270,240);
                glVertex2i(280,250);
                glEnd();
                glFlush(); //force execution of GL commands in finite time
}
```

```
int main(int argc, char **argv)
{
                glutInit(&argc,argv); //OPengl toolkit to invoke
                glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
                 glutInitWindowSize(640,480);
                glutInitWindowPosition(100,150);
                glutCreateWindow("point");
                glutDisplayFunc(MyDisplay);
                MyInit();
                glutMainLoop();
}
```