Hochiminh city University of Technology
Faculty of Computer Science and Engineering

# COMPUTER GRAPHICS

# CHAPTER 02:

# Graphics Programming

tgson@hcmut.edu.vn

# OUTLINE

- Introduction
- OpenGL Libraries
- Windows-based programming
- A simple program
- Viewing
- Viewport
- Primitives
- Draw Object
- The gasket

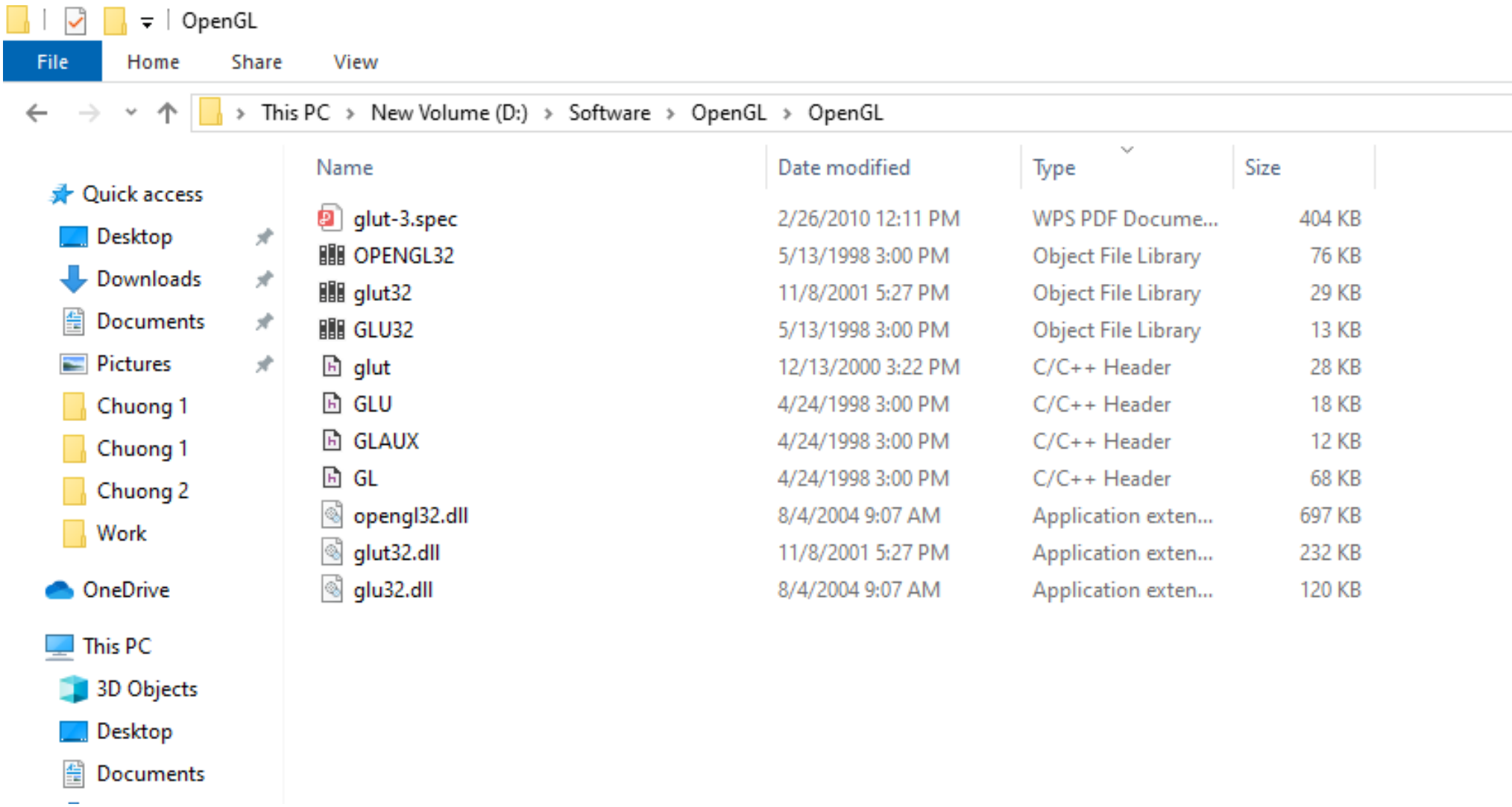
# Introduction

- Programming Environment
  - Hardware: display, graphics card
  - Software: OS (Windows), programming language (MS VC++), graphics library (OpenGL, DirectX)
- OpenGL
  - Platform-independent API
  - Easy to use
  - Close enough to the hardware to get excellent performance
  - Treat 2D and 3D in the same way

# OpenGL Libraries

- OpenGL core library
  - **OpenGL32 on Windows**
  - **GL on most unix/linux systems (libGL.a)**
- OpenGL Utility Library (GLU)
  - **Provides functionality in OpenGL core but avoids having to rewrite code**
- Links with window system
  - **GLX for X window systems**
  - **WGL for Windows**
  - **AGL for Macintosh**

# OpenGL Libraries

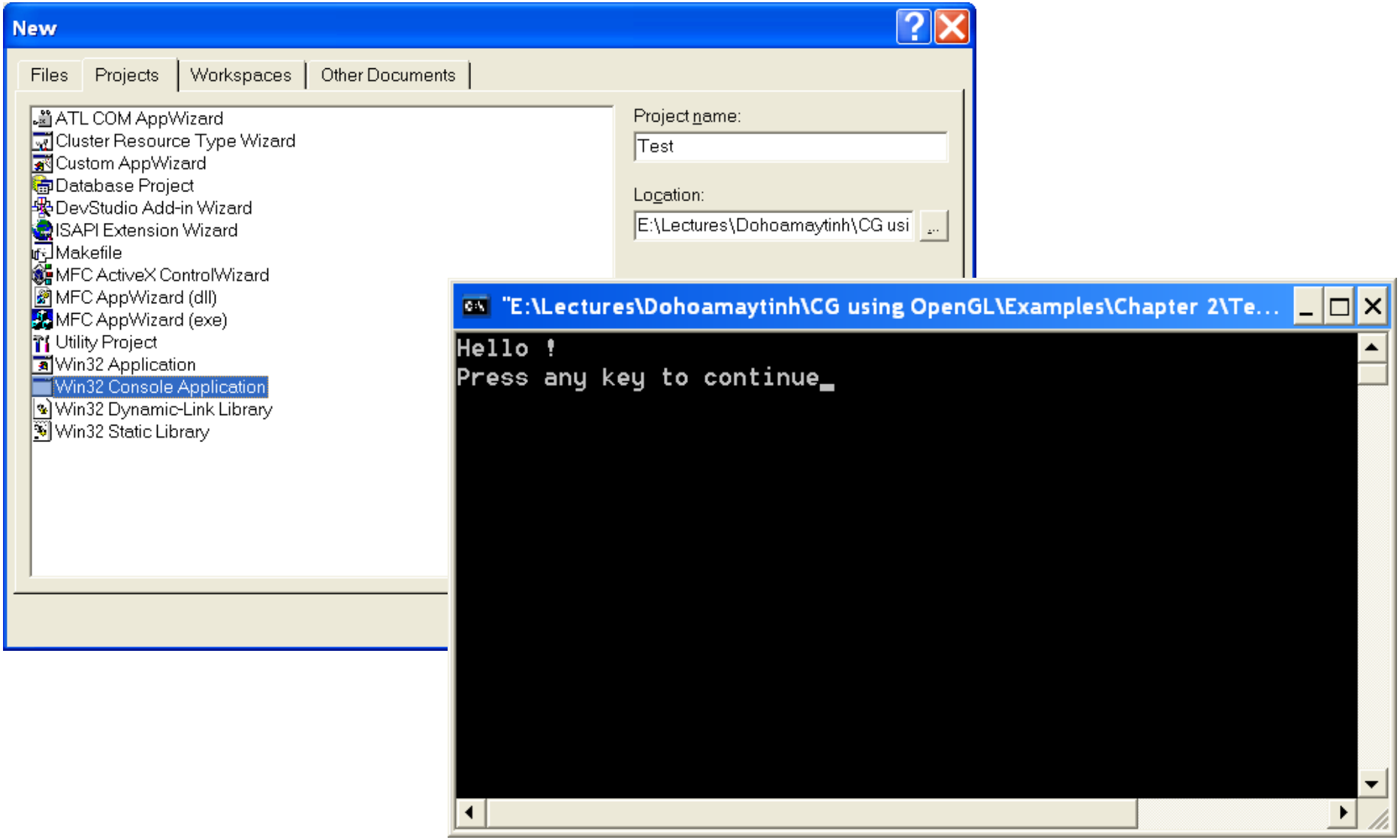



| Name | Date modified | Type | Size |
|---|---|---|---|
| glut-3.spec | 2/26/2010 12:11 PM | WPS PDF Docume... | 404 KB |
| OPENGL32 | 5/13/1998 3:00 PM | Object File Library | 76 KB |
| glut32 | 11/8/2001 5:27 PM | Object File Library | 29 KB |
| GLU32 | 5/13/1998 3:00 PM | Object File Library | 13 KB |
| glut | 12/13/2000 3:22 PM | C/C++ Header | 28 KB |
| GLU | 4/24/1998 3:00 PM | C/C++ Header | 18 KB |
| GLAUX | 4/24/1998 3:00 PM | C/C++ Header | 12 KB |
| GL | 4/24/1998 3:00 PM | C/C++ Header | 68 KB |
| opengl32.dll | 8/4/2004 9:07 AM | Application exten... | 697 KB |
| glut32.dll | 11/8/2001 5:27 PM | Application exten... | 232 KB |
| glu32.dll | 8/4/2004 9:07 AM | Application exten... | 120 KB |

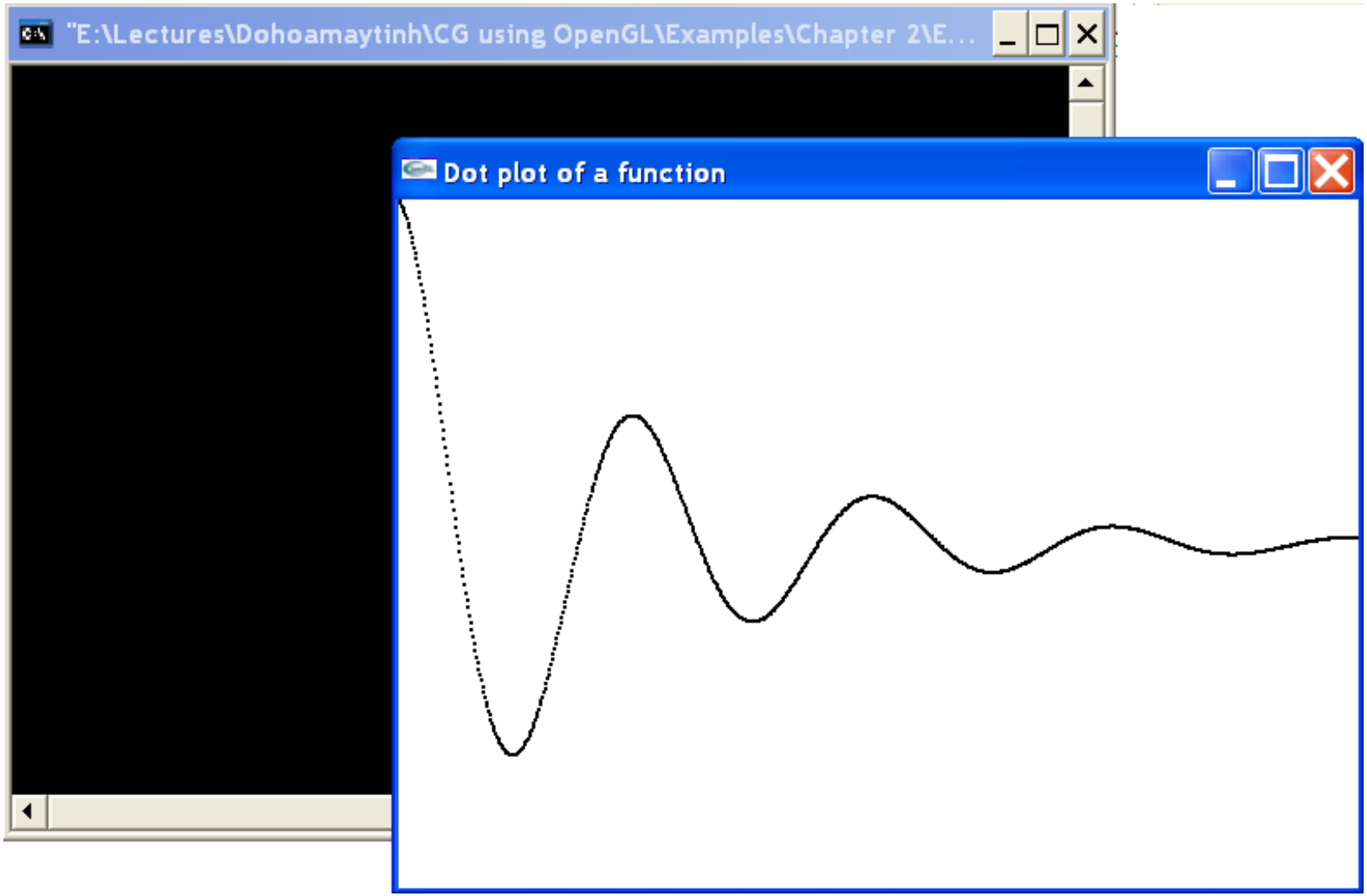
# OpenGL Libraries

- OpenGL Utility Toolkit (GLUT)
  - **Provides functionality common to all window systems**
    - Open a window
    - Get input from mouse and keyboard
    - Menus
    - Event-driven
  - **Code is portable but GLUT lacks the functionality of a good toolkit for a specific platform**
    - No slide bars

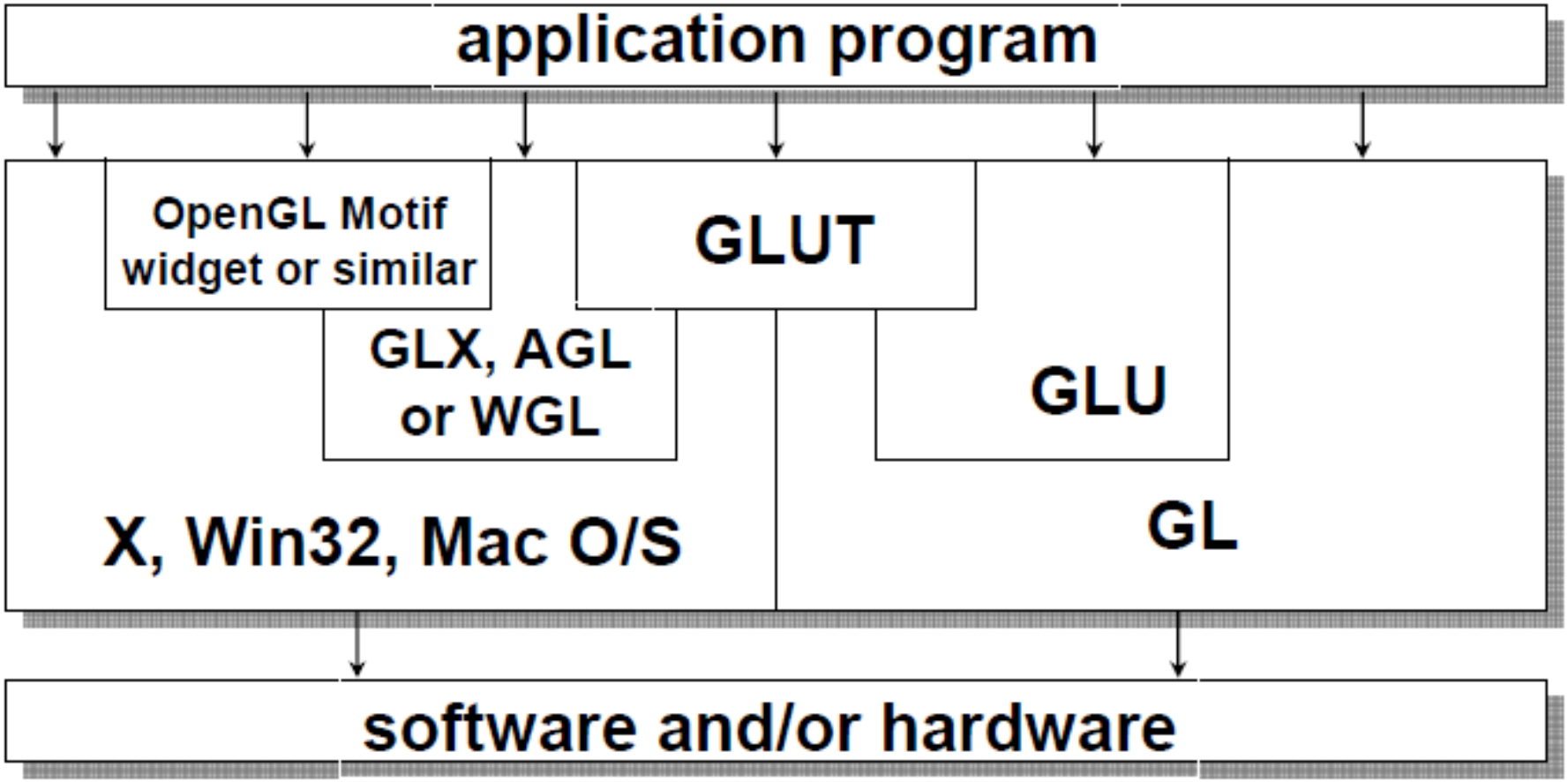
# OpenGL Libraries



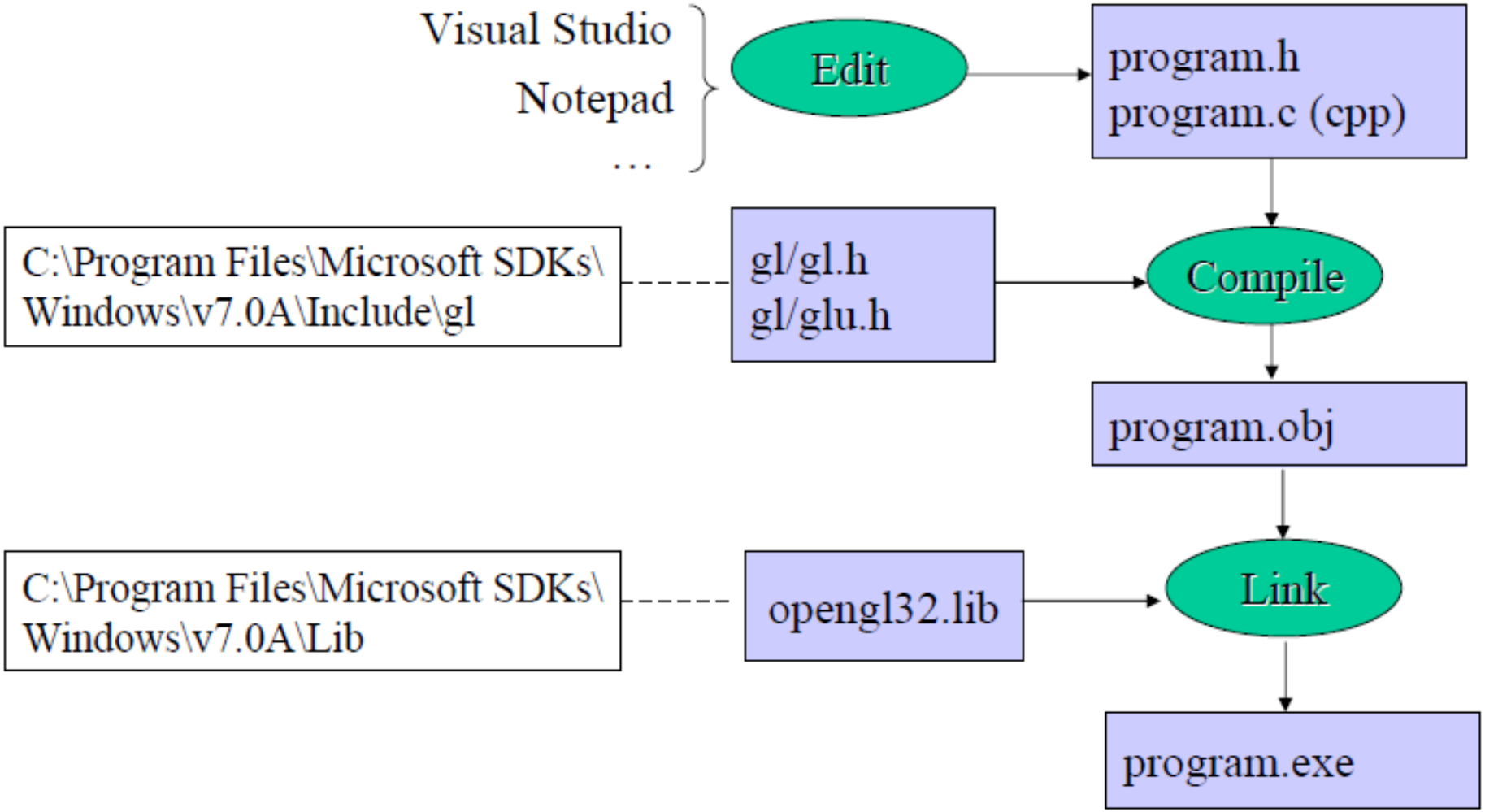
# OpenGL Libraries



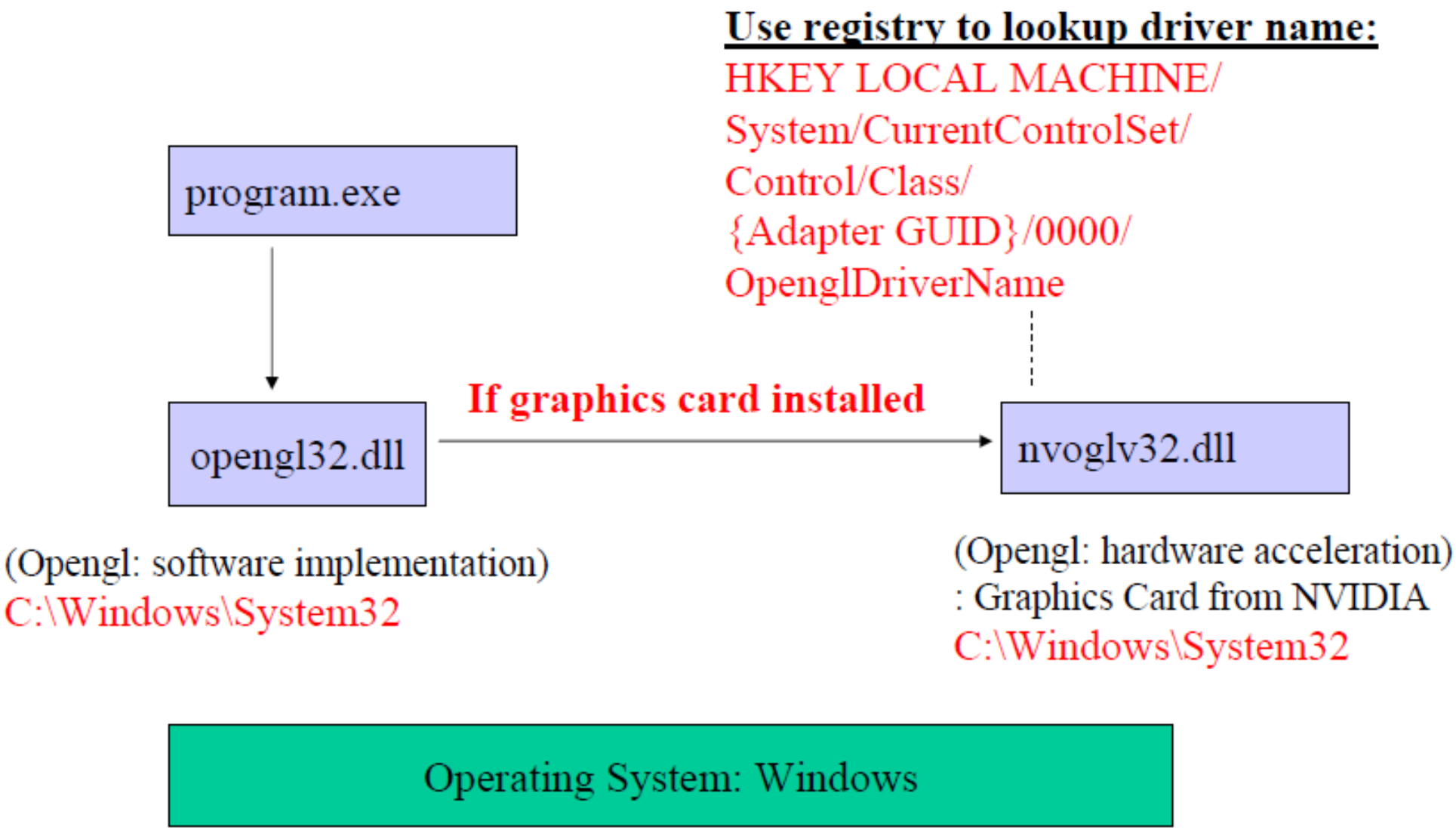
# OpenGL Libraries



# OpenGL Libraries



# OpenGL Libraries



# OpenGL Libraries

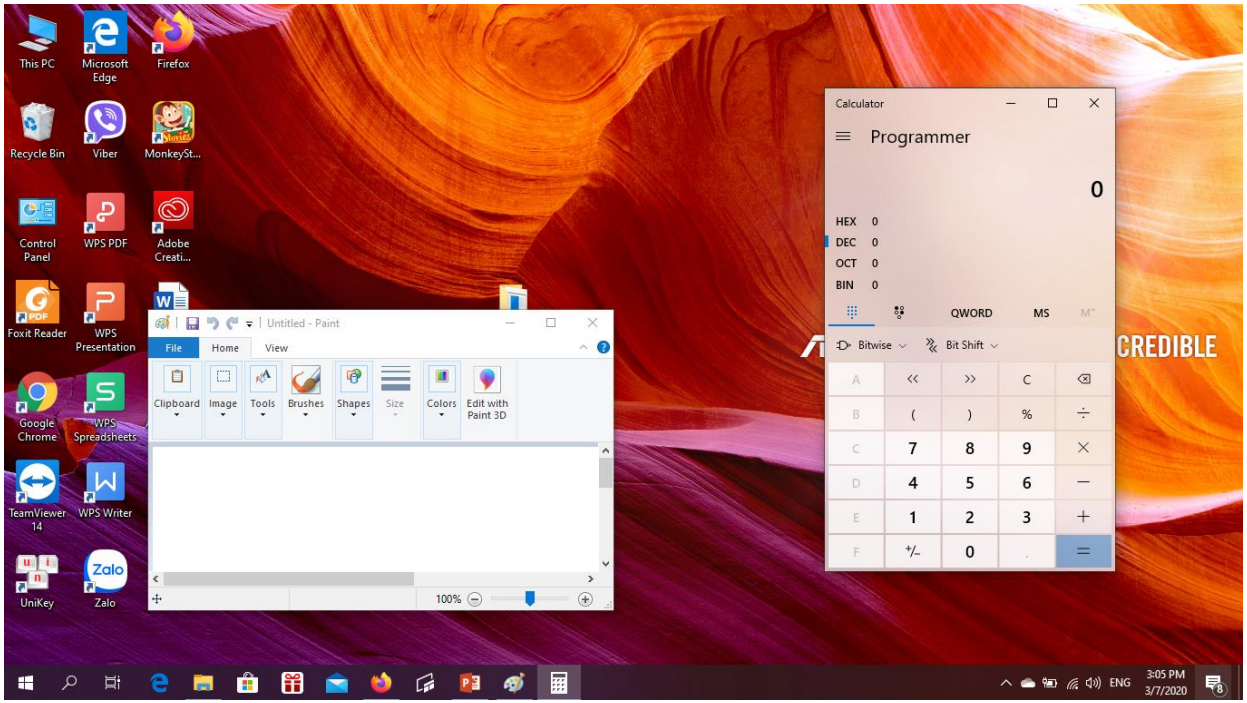
- OpenGL Functions
  - Primitives
    - **Points**
    - **Line Segments**
    - **Polygons**
  - Attributes
  - Transformations
    - **Modeling**
    - **Viewing**
  - Control (GLUT)
  - Input (GLUT)
  - Query

# Windows-based programming

- Event-driven programming
- Event queue
- Callback function
- Register callback function
  - glutDisplayFunc(myDisplay)
  - glutReshapeFunc(myReshape)
  - glutMouseFunc(myMouse)
  - glutKeyboardFunc(myKeyboard)

# Windows-based programming

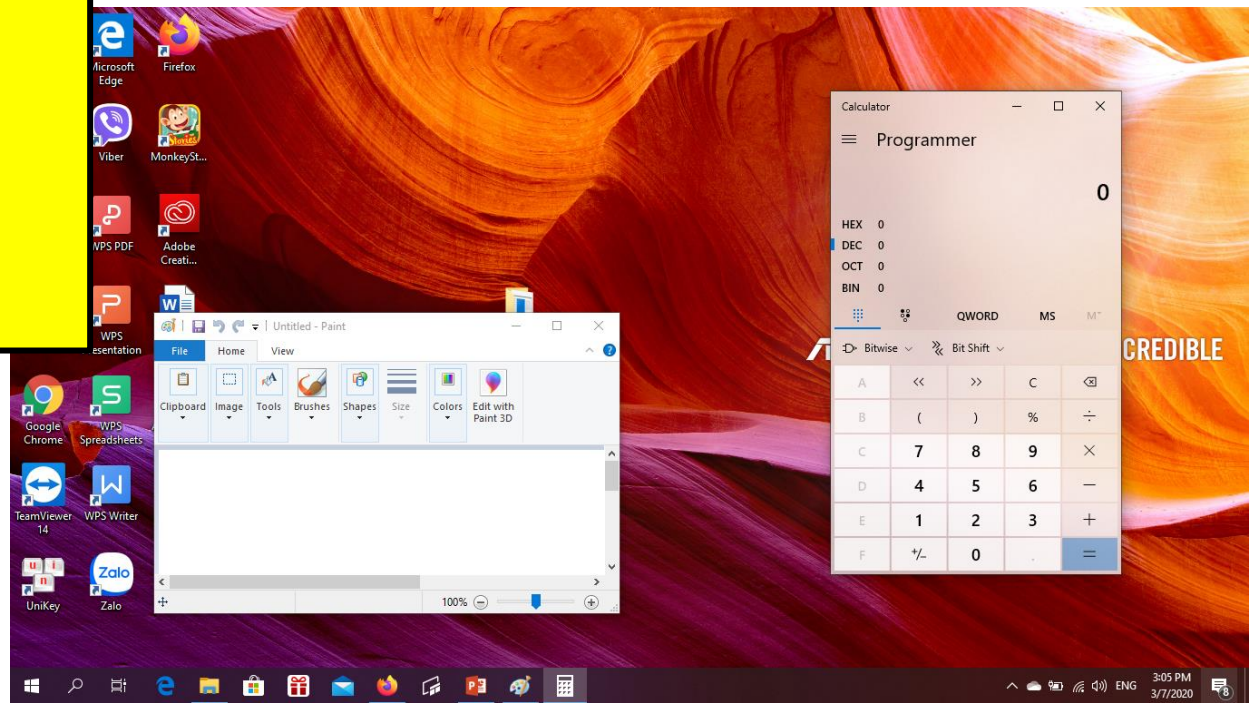
## Event Queue



# Windows-based programming

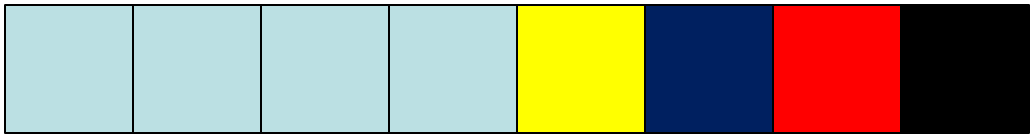
Event Queue

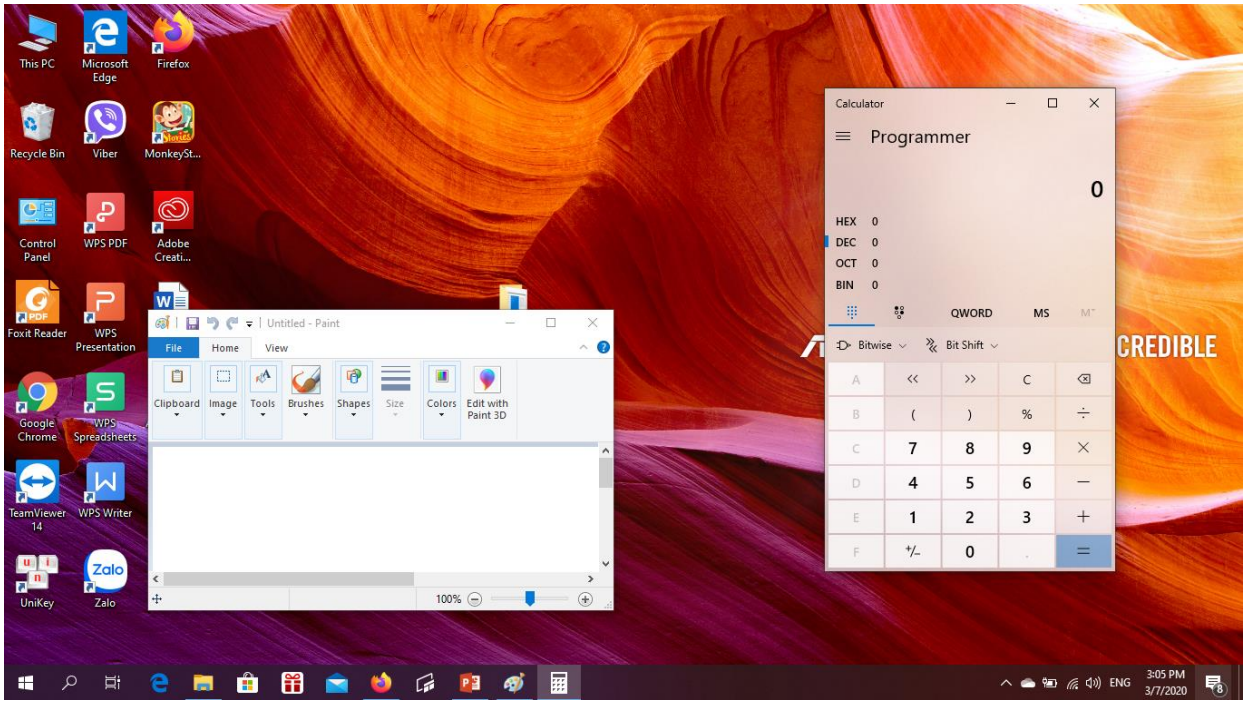
ID:….
Event:…
X: …
Y: …
……



# Windows-based programming

## Event Queue

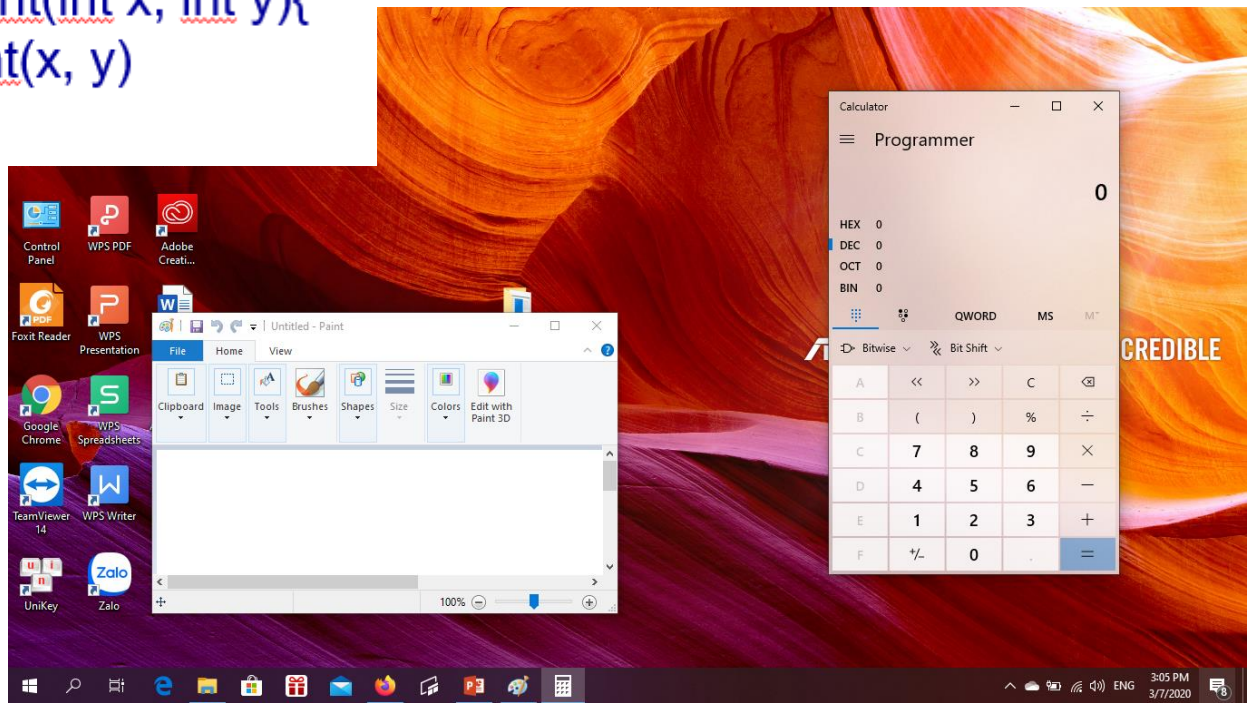




# Windows-based programming

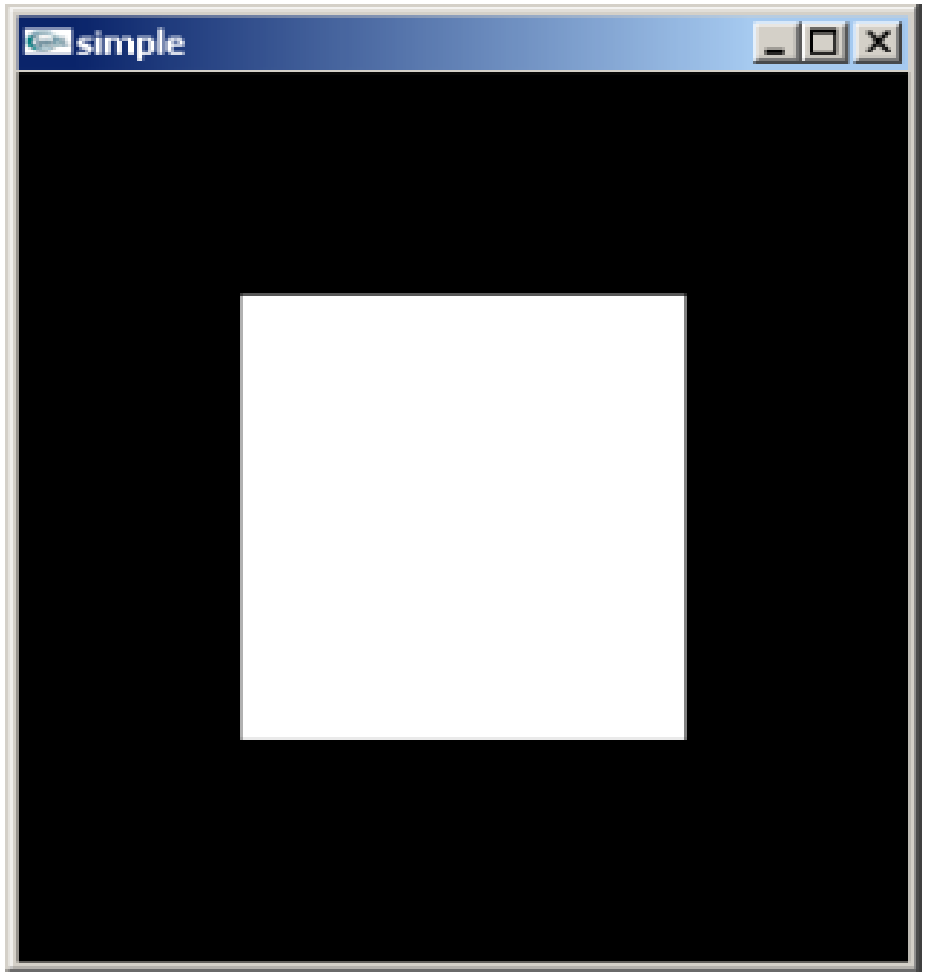
Event Queue

```
void drawPoint(int x, int y){
    DrawPoint(x, y)
}
```



# A simple program

- Generate a square on a solid background



# A simple program

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

# Structure of the program

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

int main(int argc, char** argv)
{
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
 glutInitWindowSize(500,500);
 glutInitWindowPosition(0,0);
 glutCreateWindow("simple");
 glutDisplayFunc(mydisplay);

 init();

 glutMainLoop();
}
```

includes gl.h

define window properties

display callback

set OpenGL state

enter event loop

# Structure of the program

black clear color

opaque window

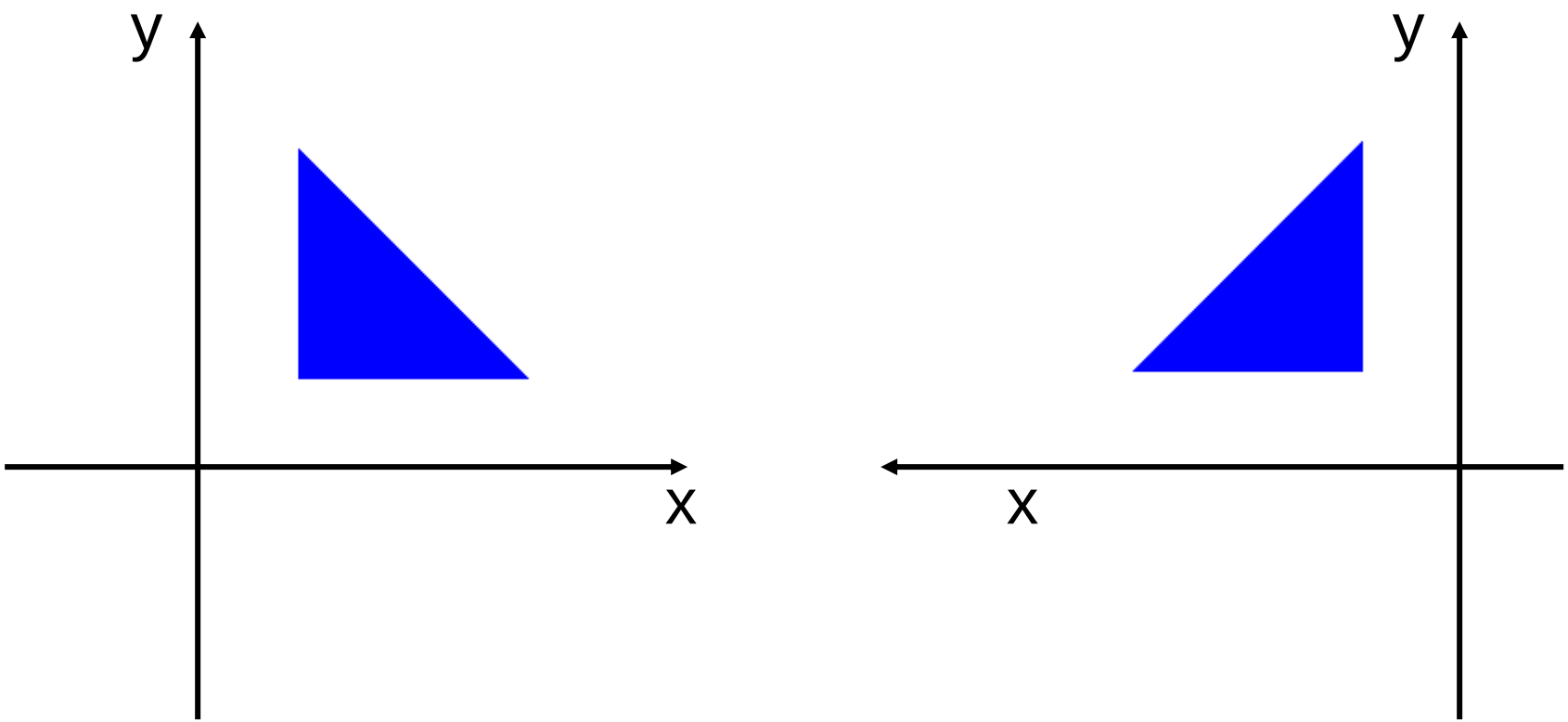
```
void init()
{
 glClearColor (0.0, 0.0, 0.0, 1.0);

 glColor3f(1.0, 1.0, 1.0);

 glMatrixMode (GL_PROJECTION);
 glLoadIdentity ();
 glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
}
```

fill/draw with white

viewing volume

# Viewing

A=(1, 1); B=(3, 1); C=(1, 3)

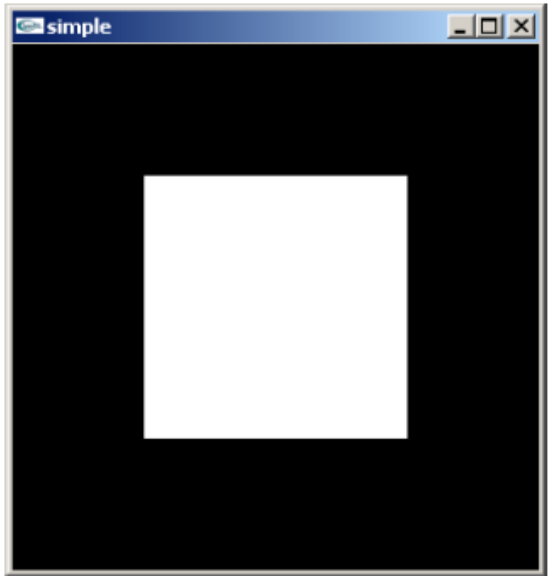


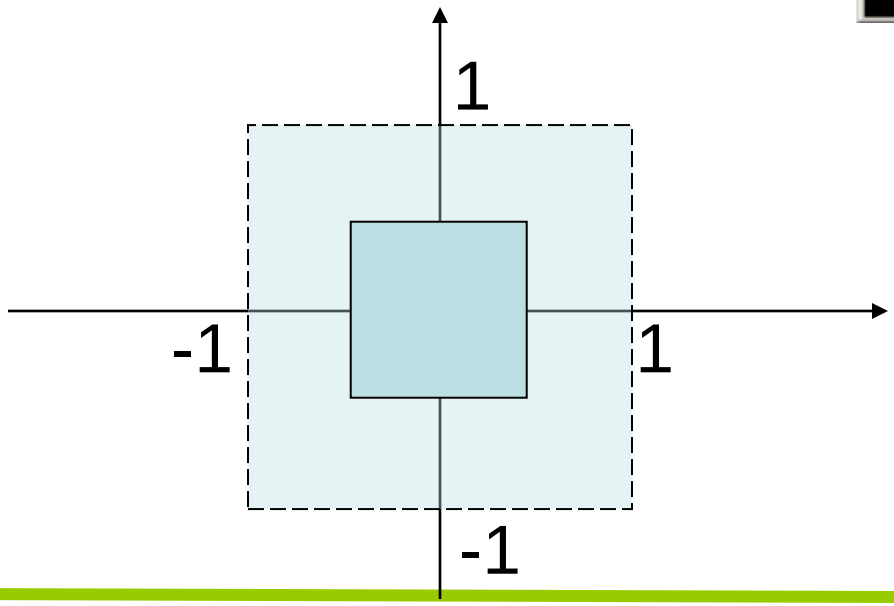
# Viewing

- Default Coordinate System
  - X axis: -1, 1; Y axis: -1, 1
  - Objects (parts of Object) outside will be clipped

# Viewing

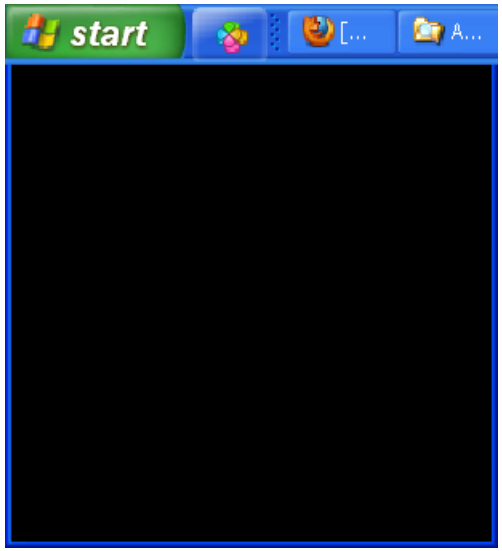
```
glBegin(GL_POLYGON);
        glVertex2f(-0.5, -0.5);
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, -0.5);
glEnd();
```

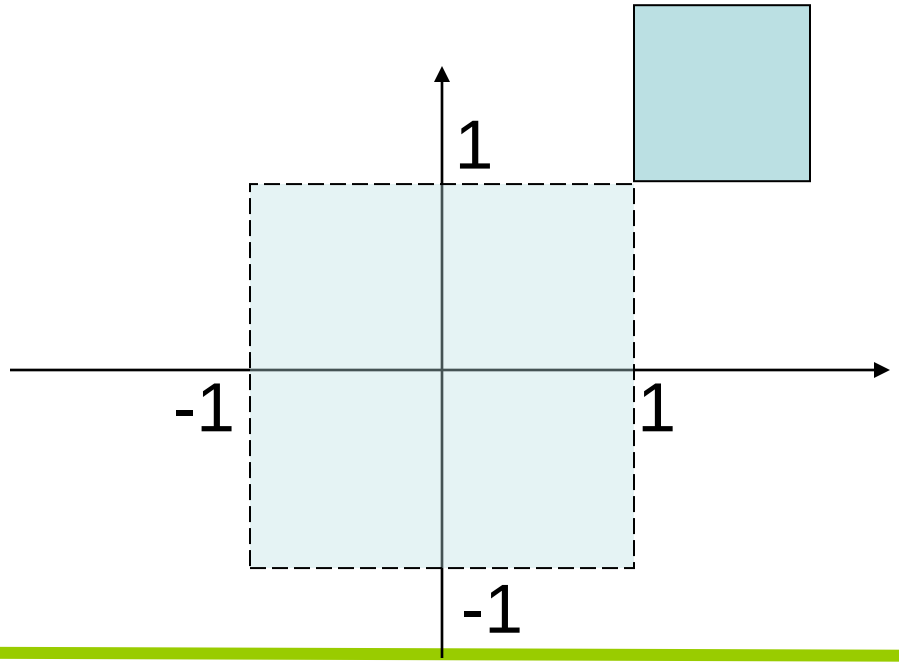


# Viewing

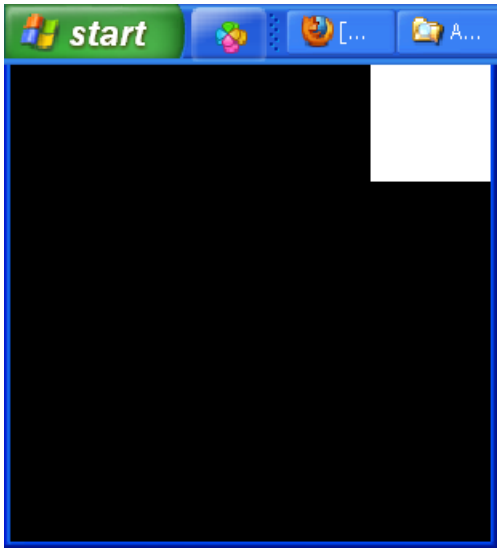
```
glBegin(GL_POLYGON);
    glVertex2f(1.0, 1.0);
    glVertex2f(1.0, 2.0);
    glVertex2f(2.0, 2.0);
    glVertex2f(2.0, 1.0);
glEnd();
```

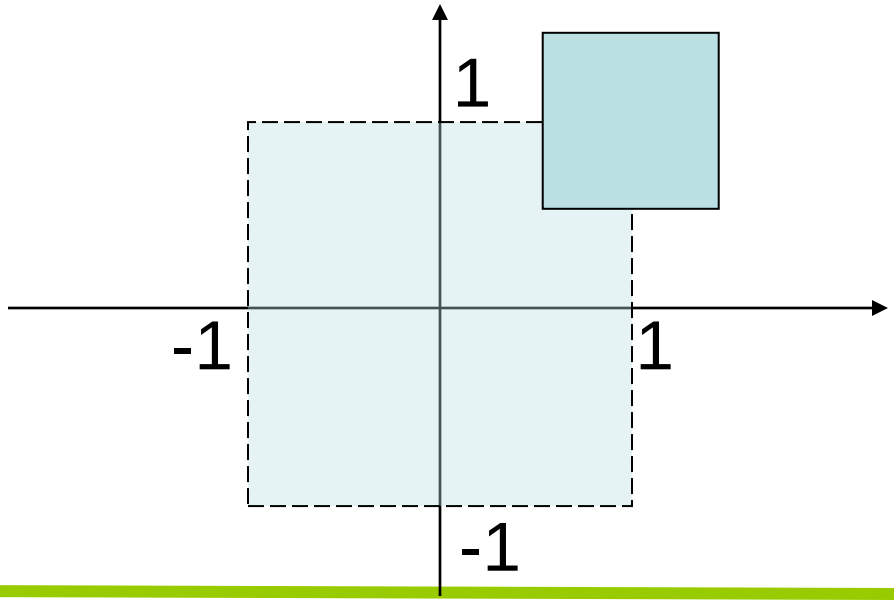


# Viewing

```
glBegin(GL_POLYGON);
    glVertex2f(0.5, 0.5);
    glVertex2f(0.5, 1.5);
    glVertex2f(1.5, 1.5);
    glVertex2f(1.5, 0.5);
glEnd();
```

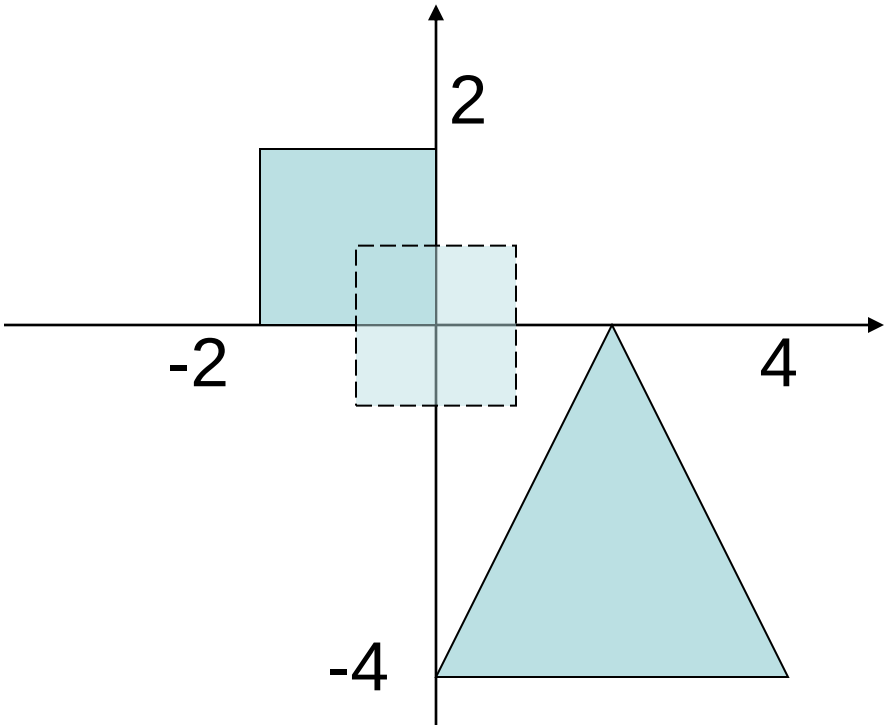


# Viewing

- **glMatrixMode (GL_PROJECTION);**
  **glLoadIdentity();**
  **glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);**
- **glMatrixMode (GL_PROJECTION);**
  **glLoadIdentity();**
  **glOrtho(-1.0, 1.0, -1.0, 1.0)**
- **glOrtho(left, right, bottom, top, near, far)**
- **gluOrtho2D(left, right,bottom,top)**

# Viewing

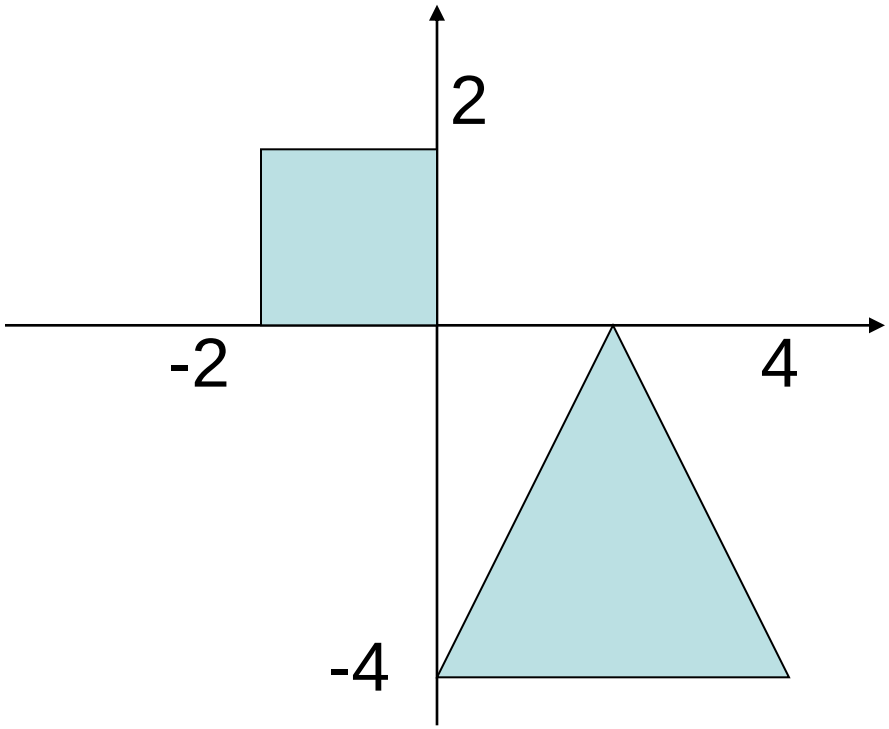


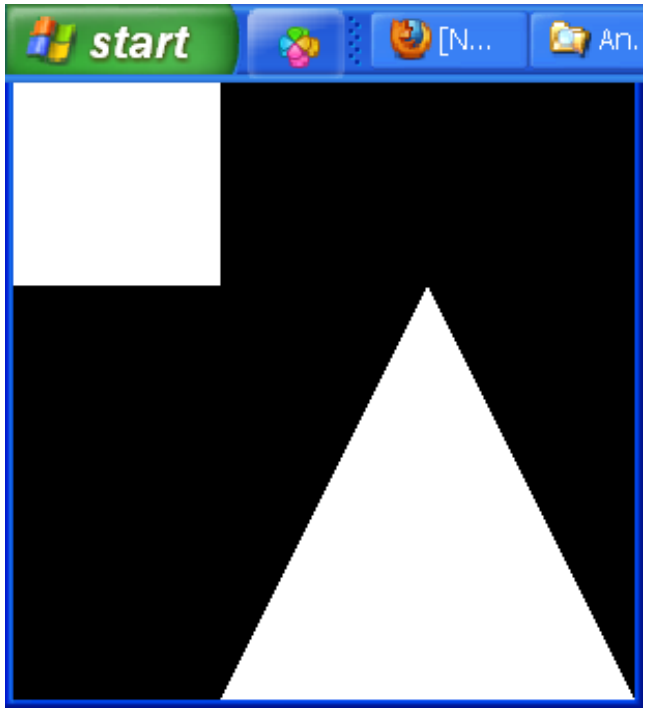
# Viewing

```
glBegin(GL_POLYGON);
    glVertex2f(-2.0, 0.0);
    glVertex2f(-2.0, 2.0);
    glVertex2f(0.0, 2.0);
    glVertex2f(0.0, 0.0);
glEnd();

glBegin(GL_POLYGON);
    glVertex2f( 0.0, -4.0);
    glVertex2f( 2.0, 0.0);
    glVertex2f( 4.0, -4.0);
glEnd();
```

# Viewing

- How to get the picture of triangle and square?



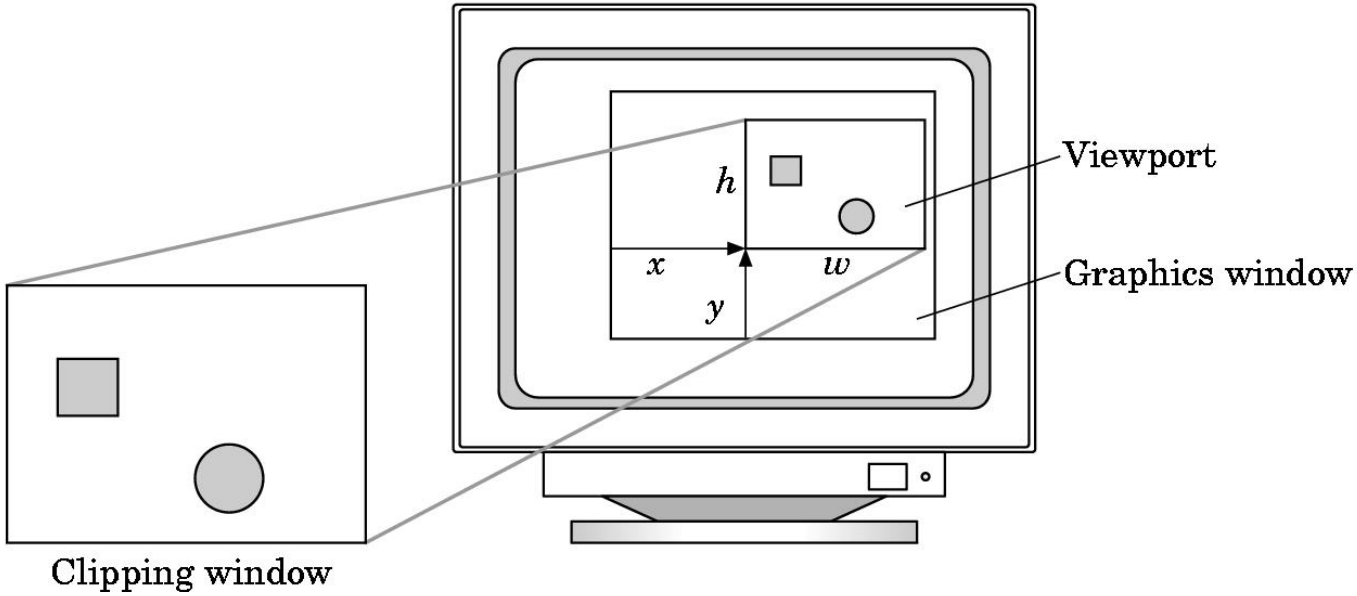



# Viewing

- How to get the picture of triangle and square?
  - glMatrixMode (GL_PROJECTION);
  - glLoadIdentity();
  - gluOrtho2D(-2.0, 4.0, -4.0, 2.0);
- How to get the picture of the square?
- How to get the picture of the triangle?

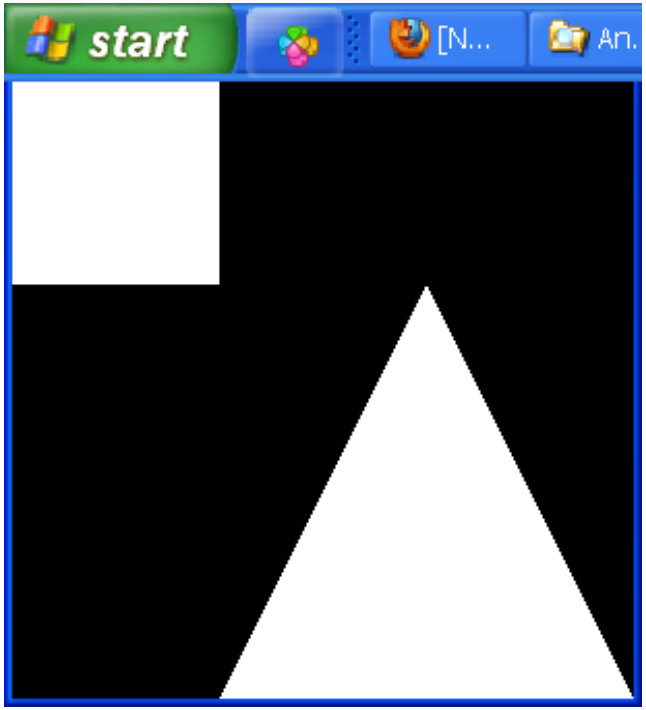
# Viewport

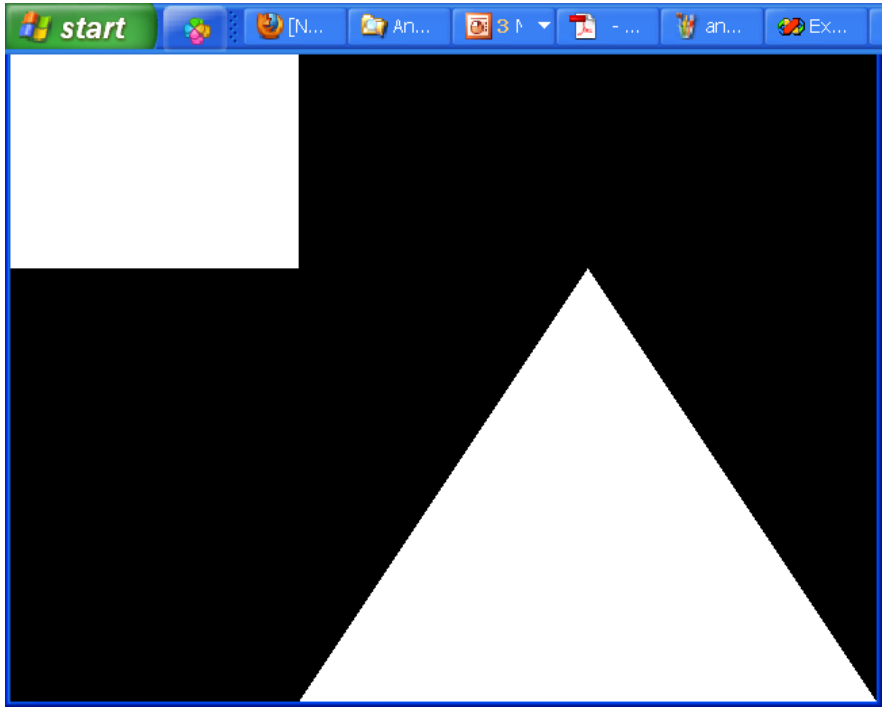
- Do not have use the entire window for the image: **glViewport(x,y,w,h)**
- Values in pixels (screen coordinates)



# Viewport

- Size of the graphics window
  - glutInitWindowSize(cx, cy);

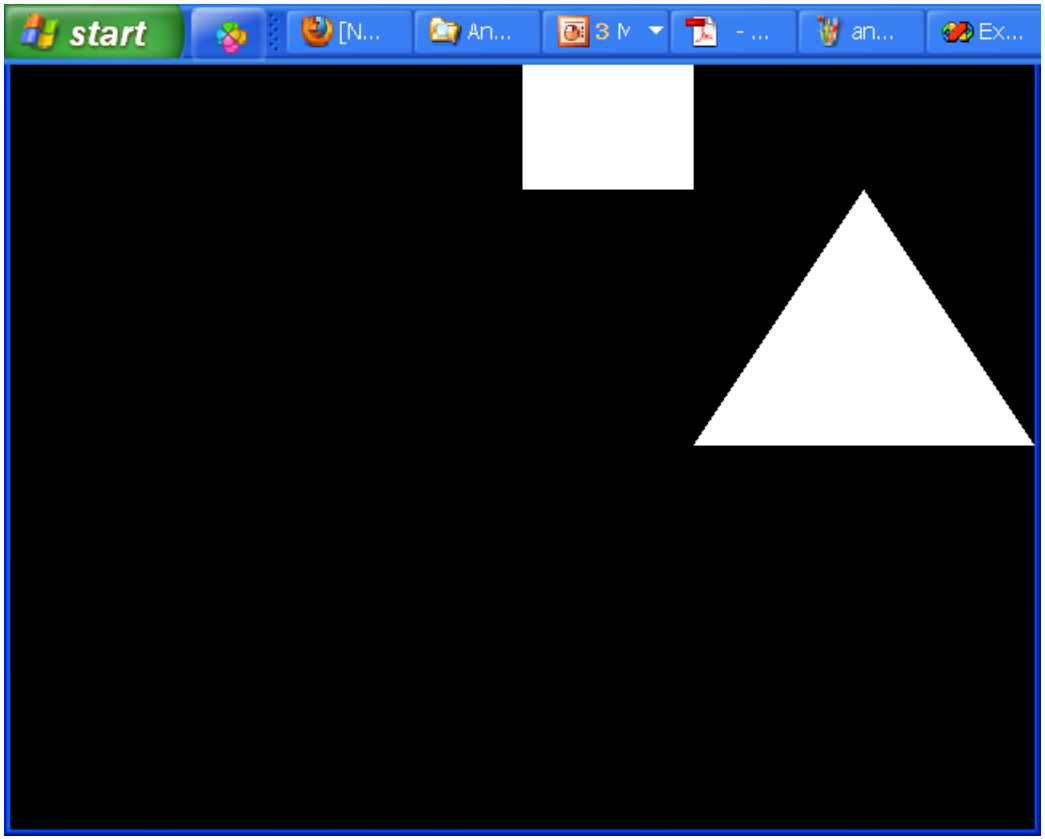



glutInitWindowSize(640, 480);

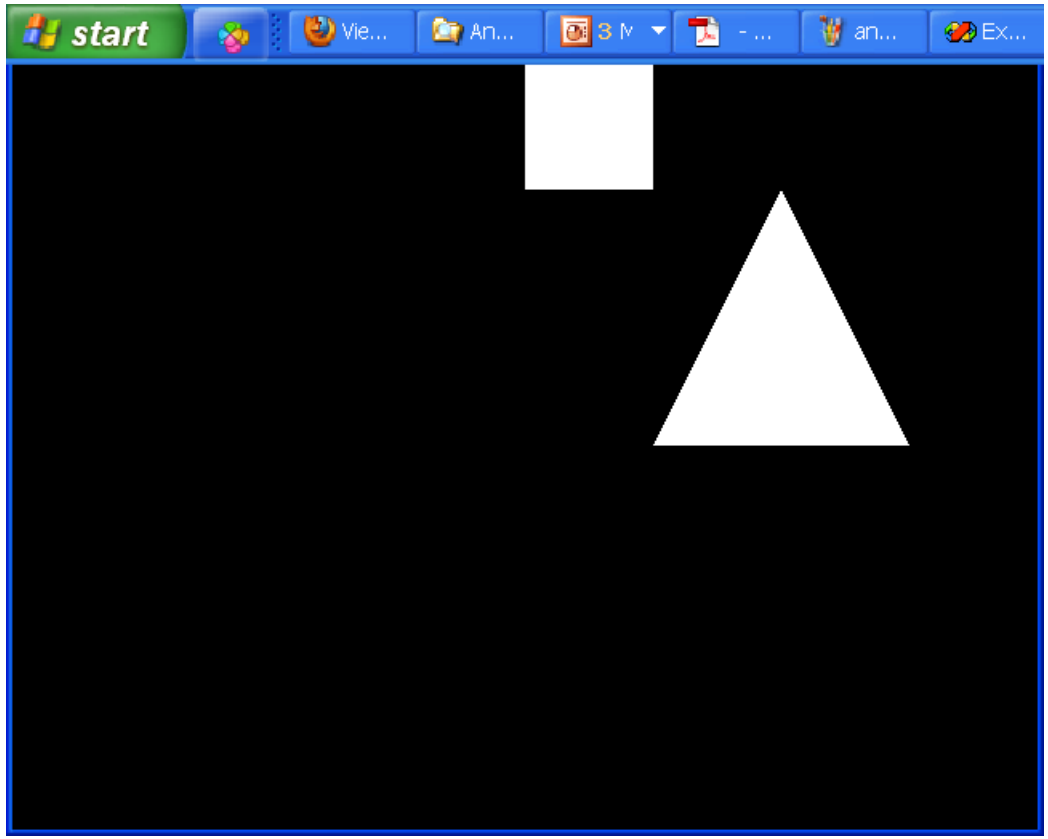
# Viewport

- **glViewport(320, 240, 320, 240)**



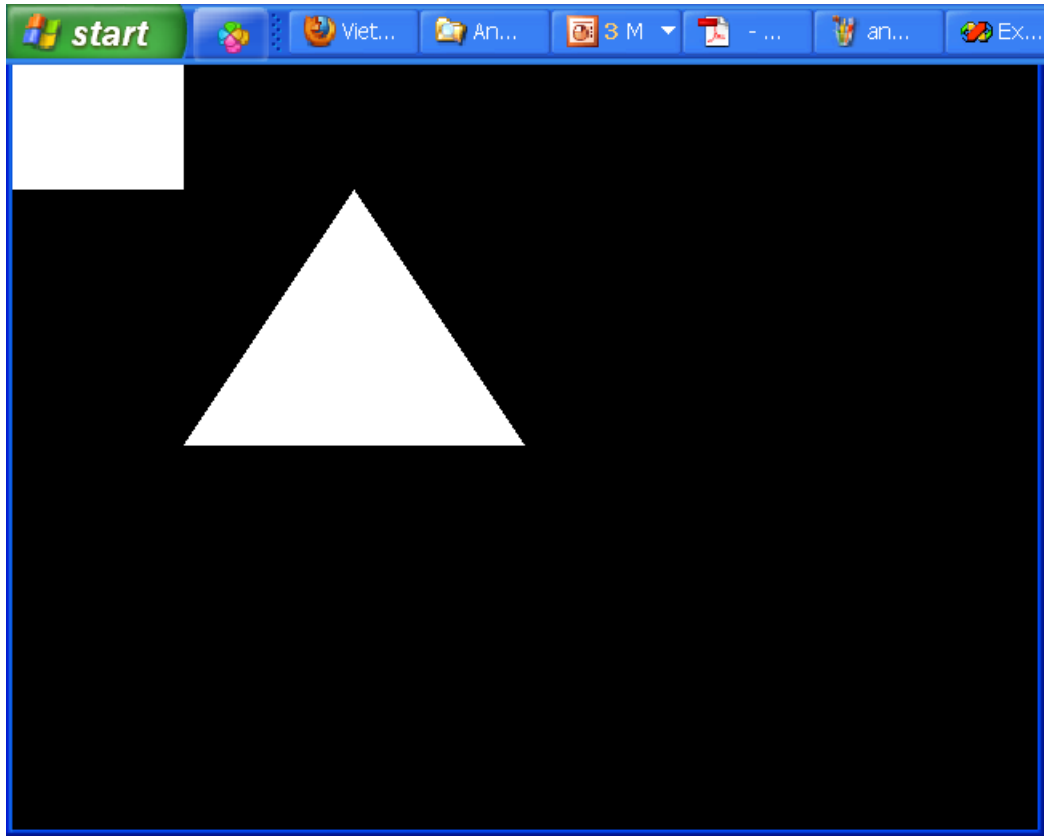
# Viewport

- **glViewport(320, 240, 240, 240)**



# Viewport

- How to draw picture in the second quadrant?

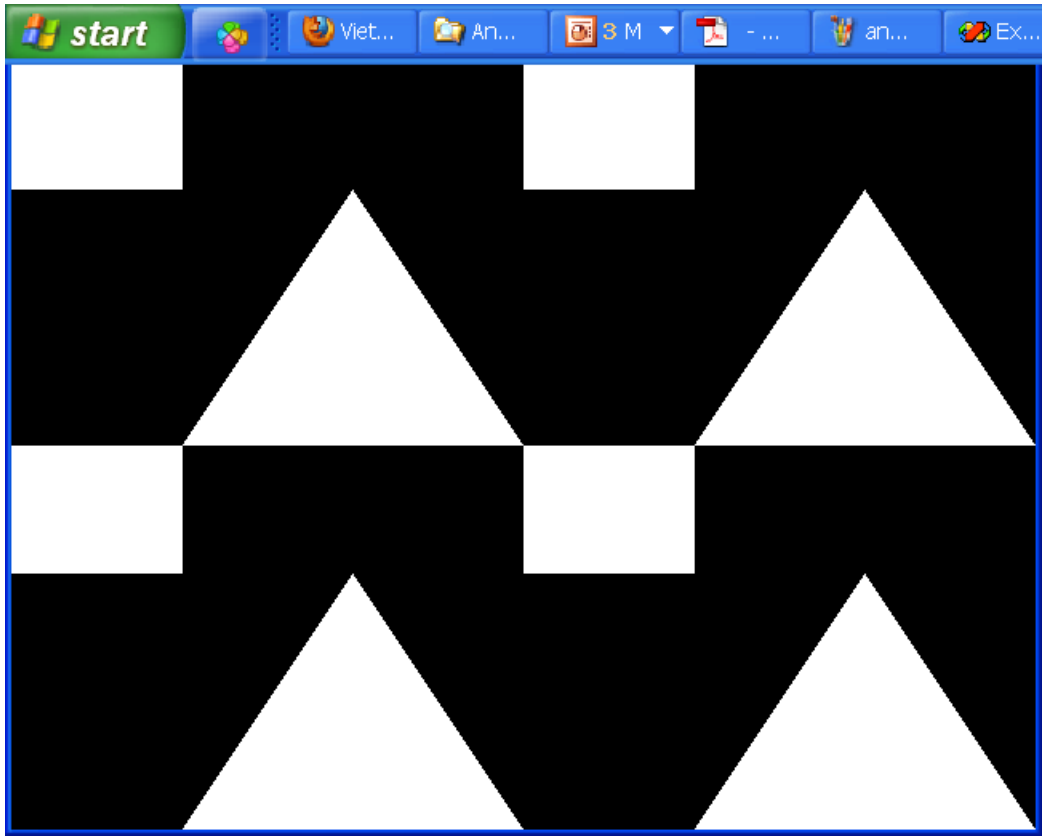


# Viewport

- How to draw picture in the second quadrant?
  - glViewport(0, 240, 320, 240);
- How to draw picture in the third quadrant?
- How to draw picture in the fourth quadrant?
- How to draw picture in all quadrant?

# Viewport

- How to draw picture in all quadrant?



# Viewport

- glViewport(320, 240, 320, 240);

```
glBegin()     //draw square
      ..................
glEnd()
glBegin()     //draw triangle
      ..................
glEnd()
```

- glViewport(0, 240, 320, 240);

  .....

- glViewport(0, 0, 320, 240);

  .....

- glViewport(320, 0, 320, 240);

  .....

# Primitives

- **Objects**
- **Viewer**
- **Light Source(s)**
- **Materials**

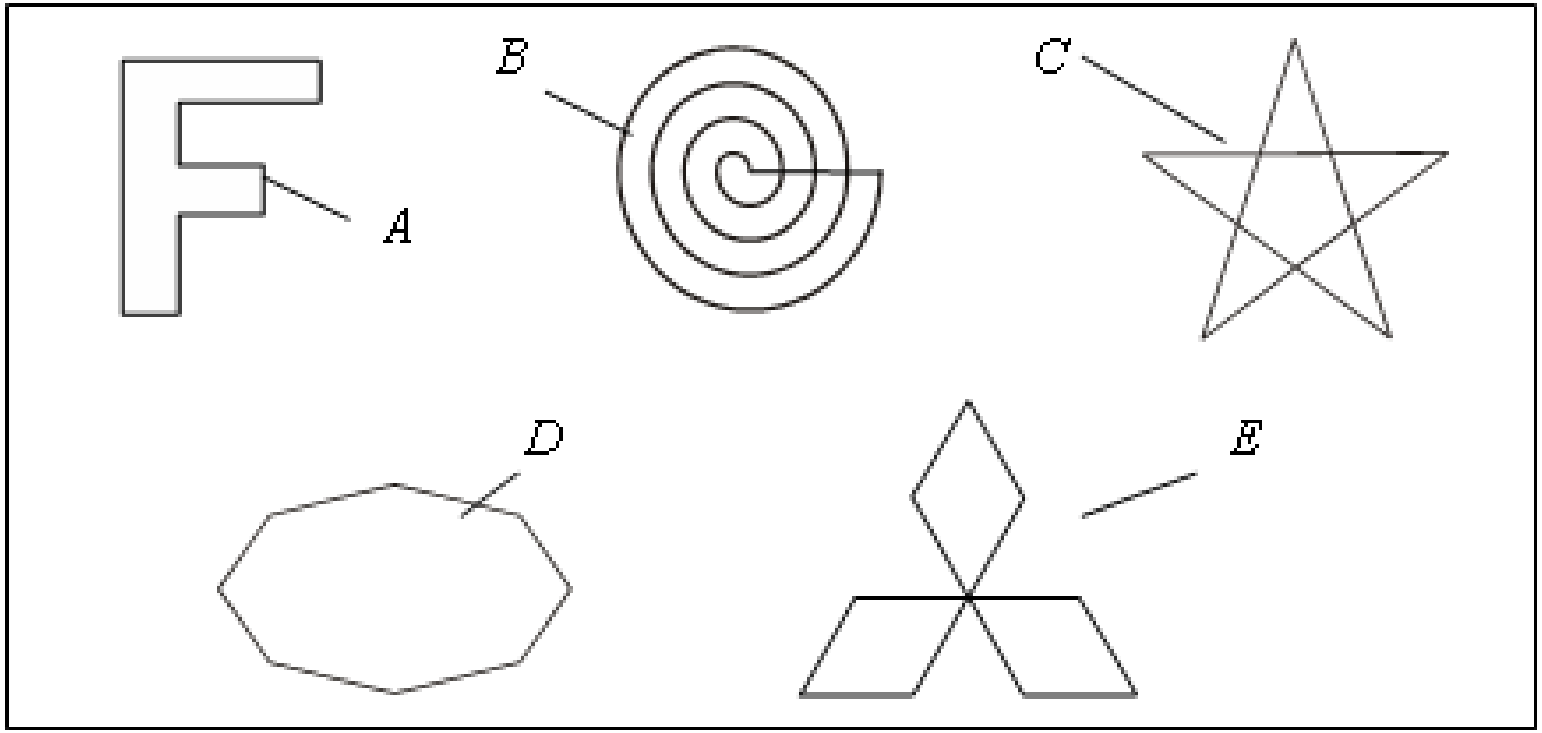
- Polyline
- Filled region
- Text
- Raster image

# Primitives

- Polyline
  - A polyline is a connected sequence of straight lines
  - A polyline can be used to approximated a smooth curve

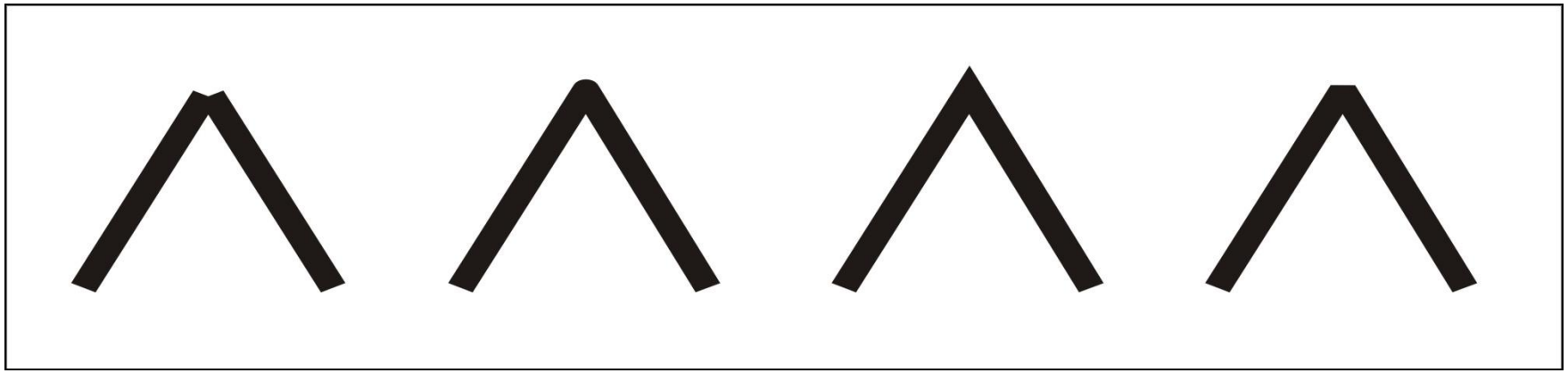
# Primitives

- Polyline
  - Polygon: polyline if the first and the last points are connected by an edge
  - Polygon type: simple, convex



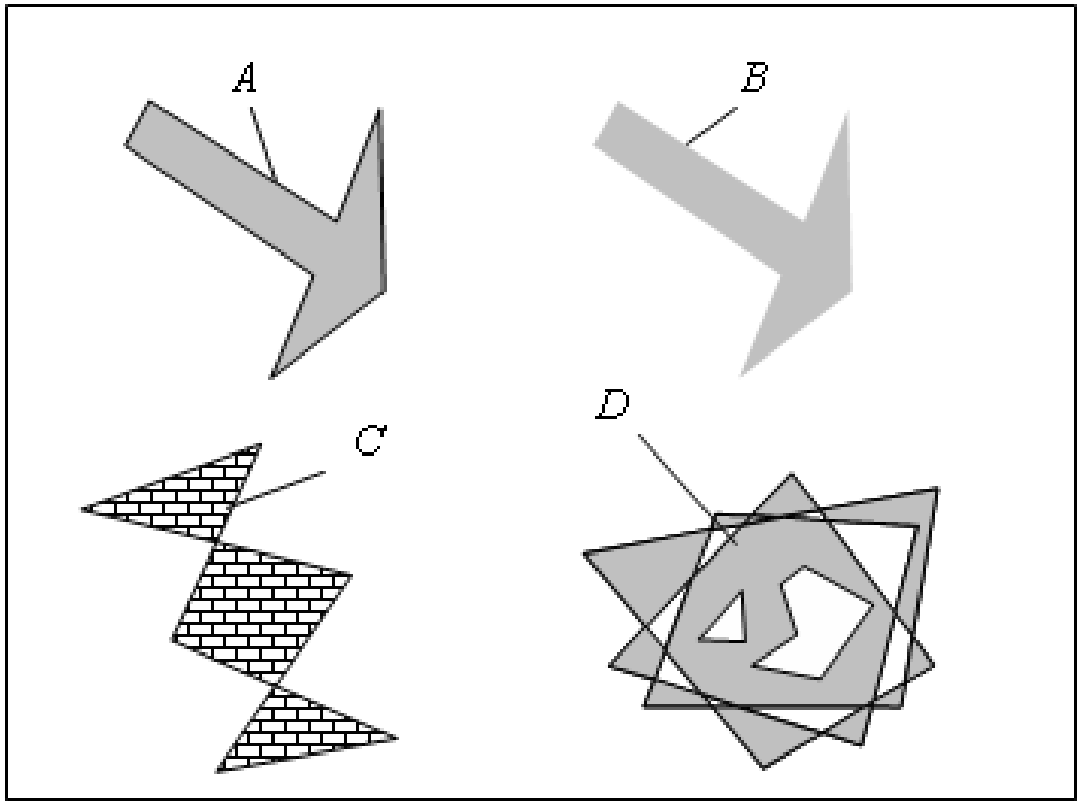
# Primitives

- Polyline
  - Attributes: Color, thickness, type (solid, dash), **join points**



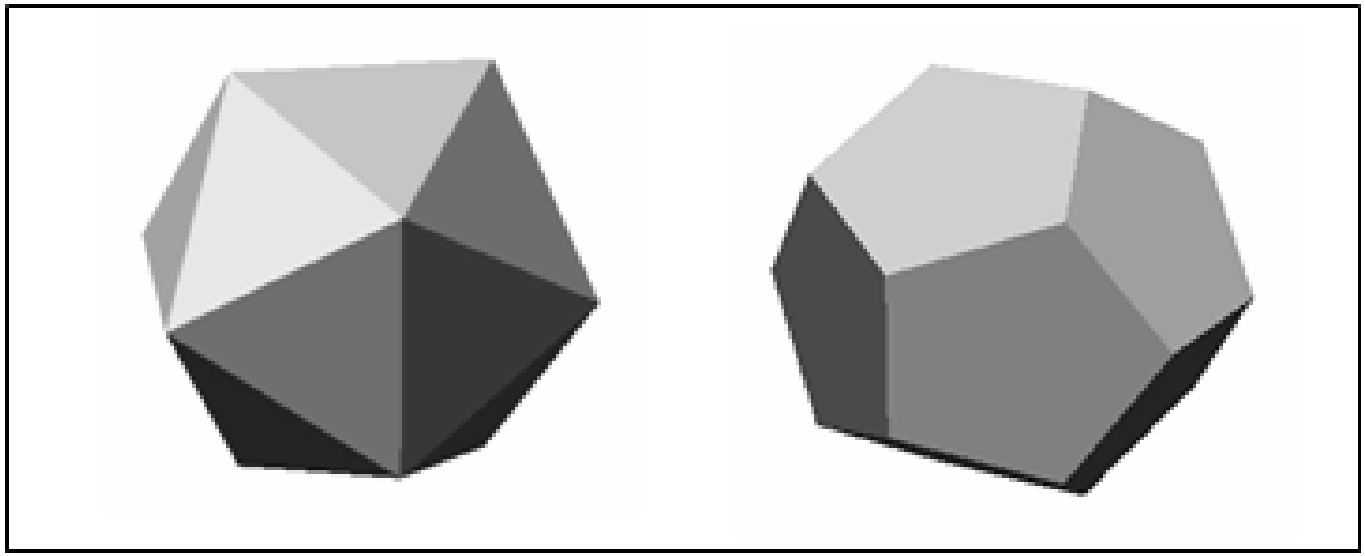
# Primitives

- Filled region
  - Filled region is a shape filled with some color or pattern. The boundary is often a polygon



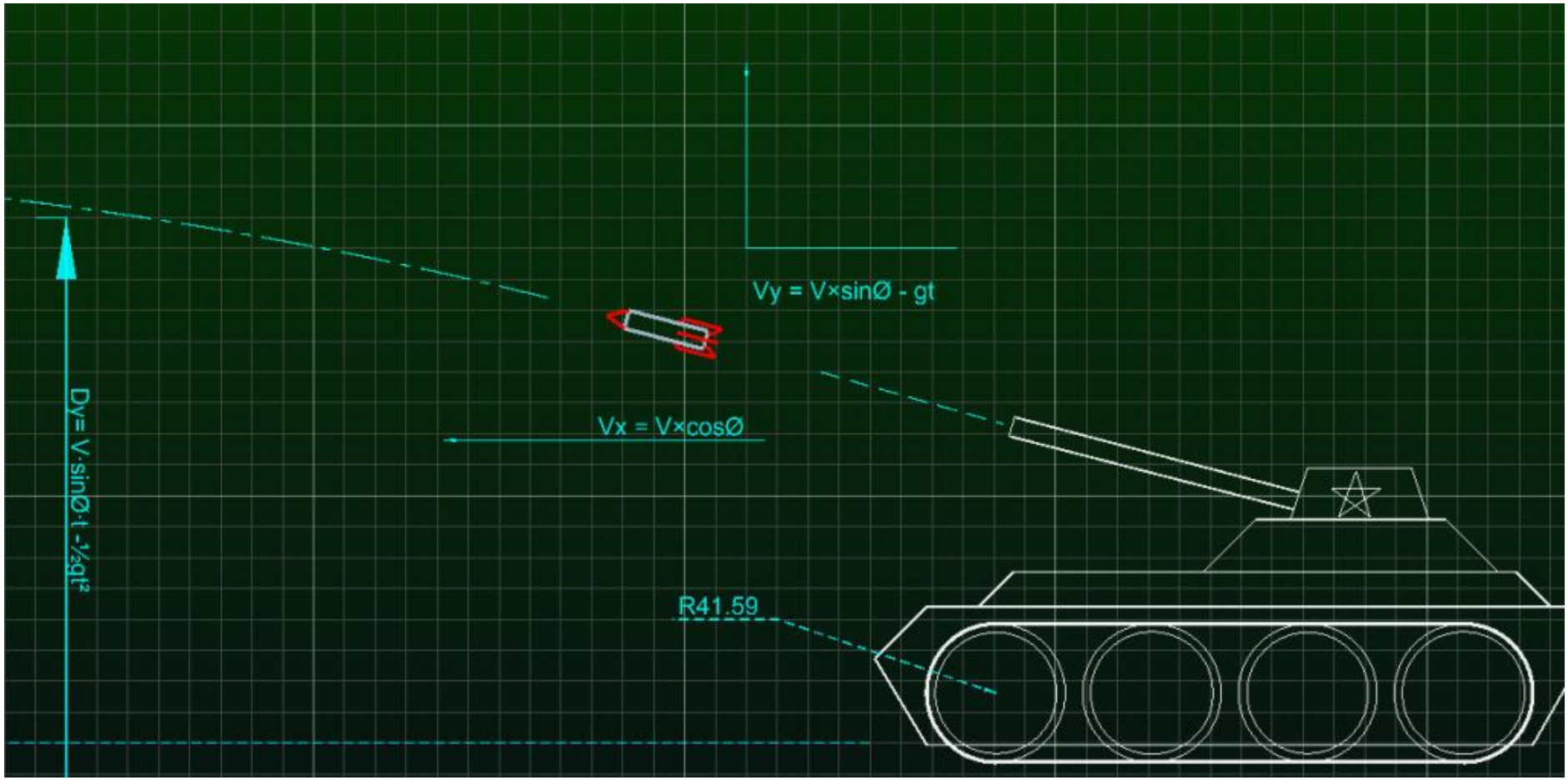
# Primitives

- Use filled region to shade the different faces of a three-dimensional object



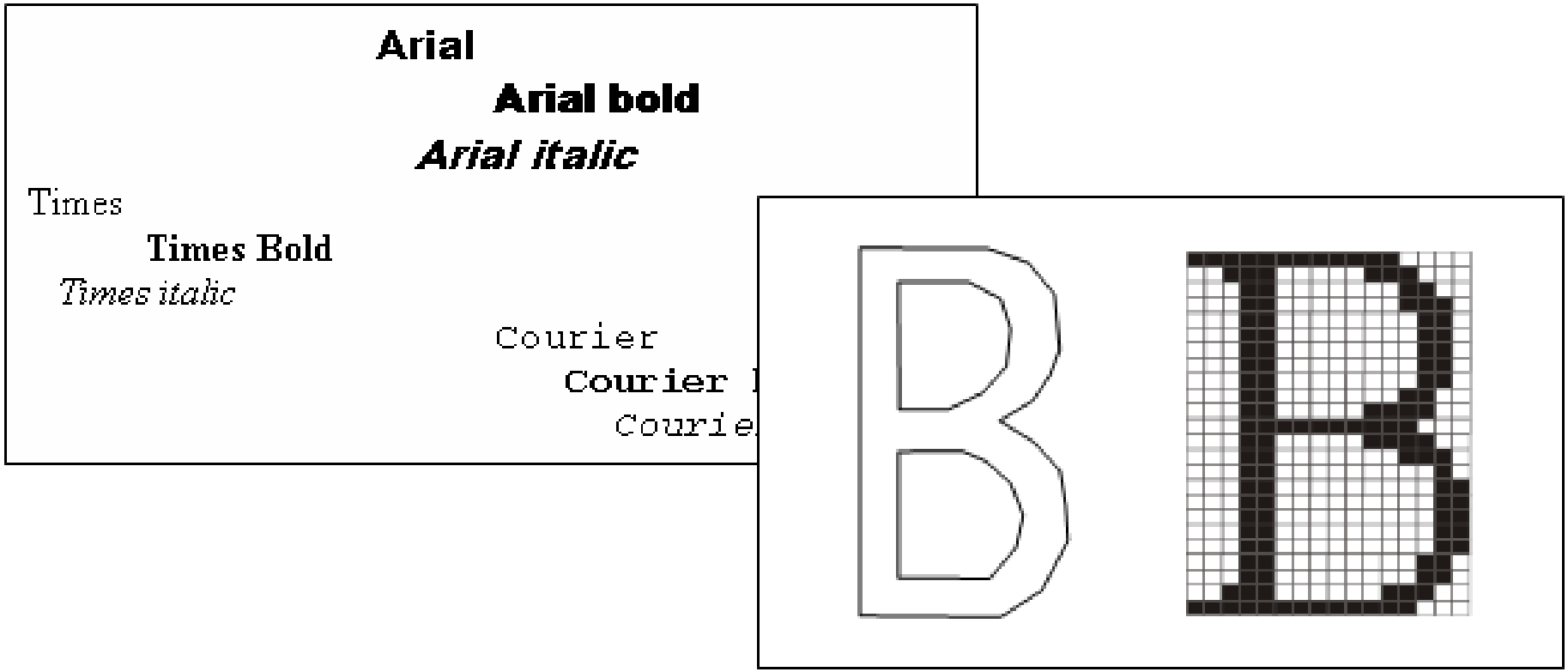
# Primitives

- Text



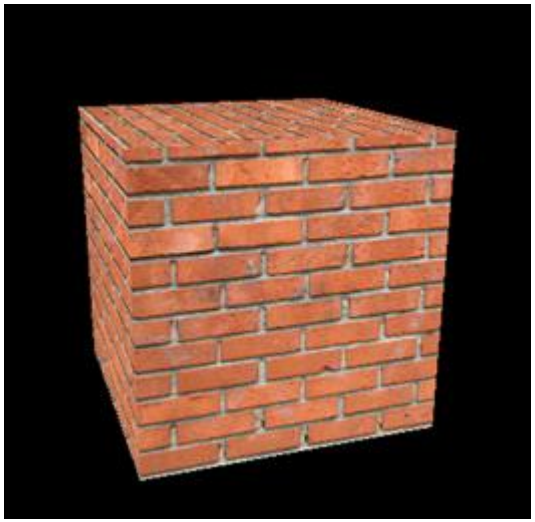
# Primitives

- Text:
  - Attributes: **Font**, color, size, orientation, space



# Primitives

- Texture



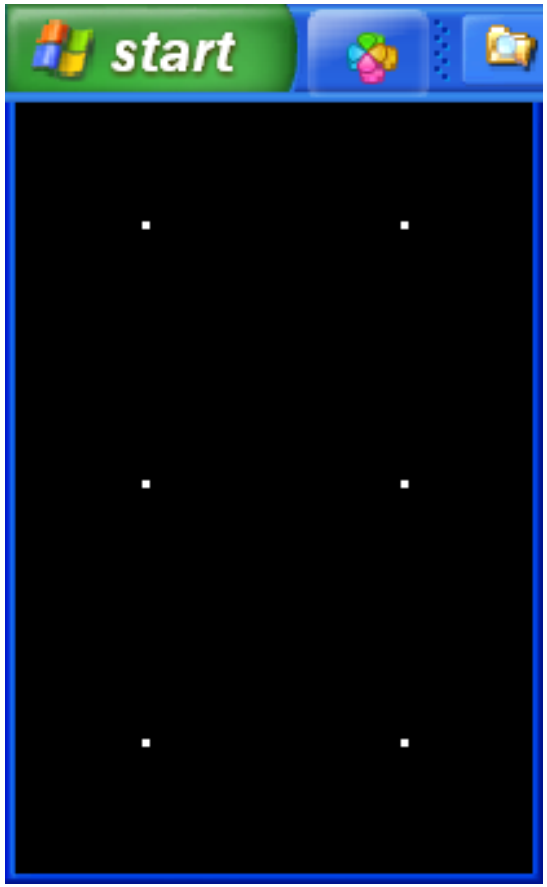
# Draw Object

```
glBegin(parameter)
    glVertex2f(…)     //or glVertex3f(…)
    glVertex2f(…)
    ……................
glEnd()
```

- Parameter
  - GL_POINTS, GL_LINES, GL_TRIANGLES, v.v

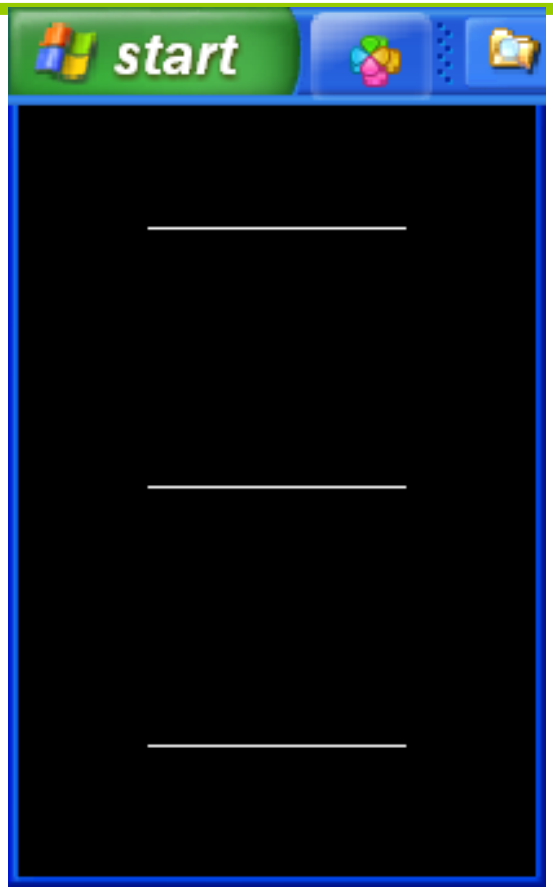
# Draw Object

```
glBegin(GL_POINTS);
    glVertex2f(-0.5, 1.0);
    glVertex2f( 0.5, 1.0);
    glVertex2f(-0.5, 0.0);
    glVertex2f( 0.5, 0.0);
    glVertex2f(-0.5, -1.0);
    glVertex2f( 0.5, -1.0);
glEnd();
```



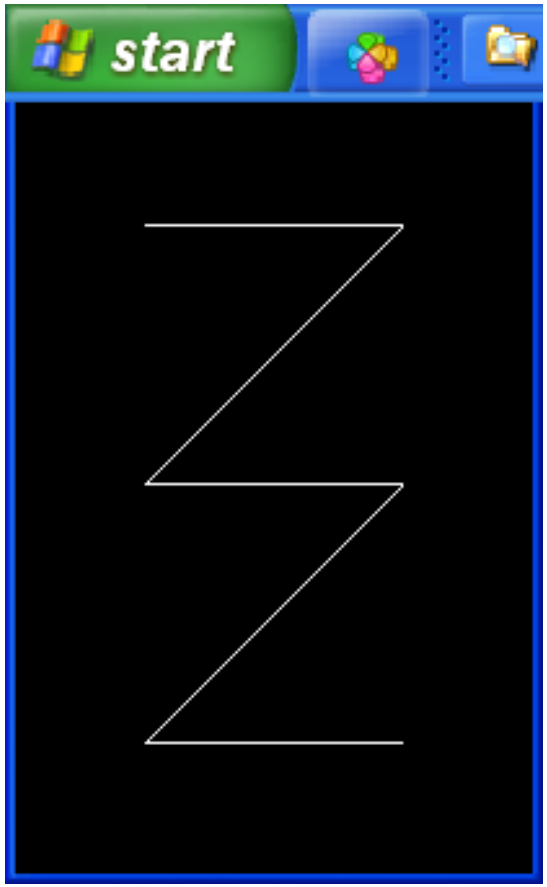
# Draw Object

```
glBegin(GL_LINES);
    glVertex2f(-0.5, 1.0);
    glVertex2f( 0.5, 1.0);
    glVertex2f(-0.5, 0.0);
    glVertex2f( 0.5, 0.0);
    glVertex2f(-0.5, -1.0);
    glVertex2f( 0.5, -1.0);
glEnd();
```



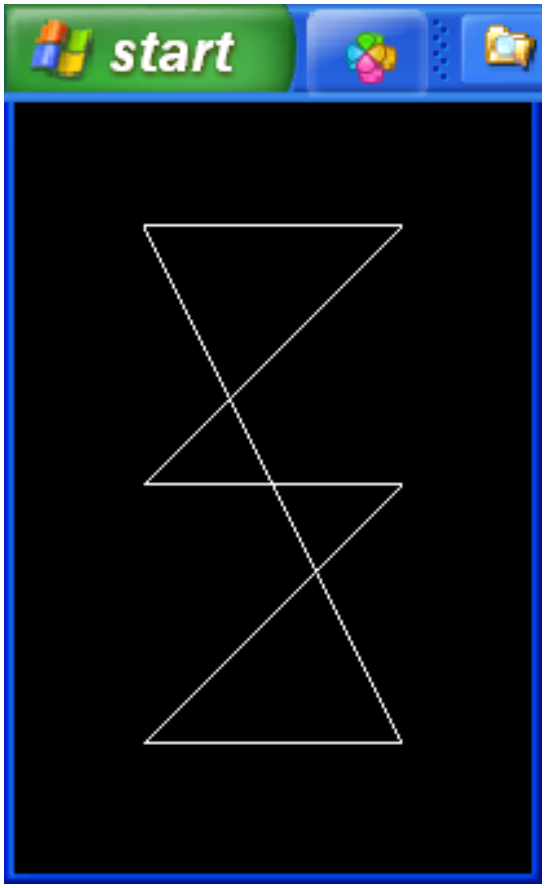
# Draw Object

```
glBegin(GL_LINE_STRIP);
    glVertex2f(-0.5, 1.0);
    glVertex2f( 0.5, 1.0);
    glVertex2f(-0.5, 0.0);
    glVertex2f( 0.5, 0.0);
    glVertex2f(-0.5, -1.0);
    glVertex2f( 0.5, -1.0);
glEnd();
```



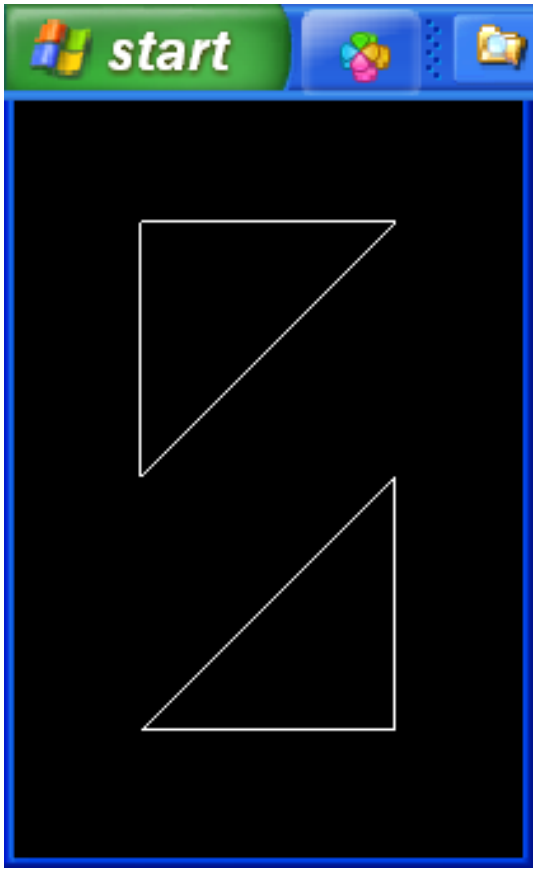
# Draw Object

```
glBegin(GL_LINE_LOOP);
    glVertex2f(-0.5, 1.0);
    glVertex2f( 0.5, 1.0);
    glVertex2f(-0.5, 0.0);
    glVertex2f( 0.5, 0.0);
    glVertex2f(-0.5, -1.0);
    glVertex2f( 0.5, -1.0);
glEnd();
```



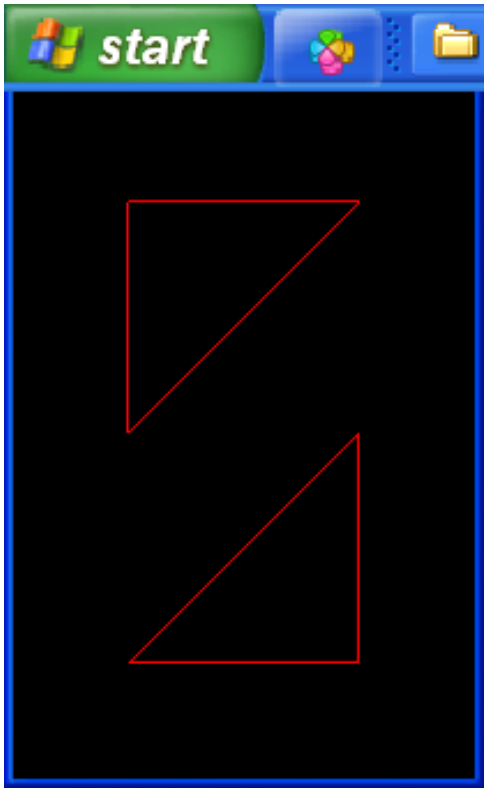
# Draw Object

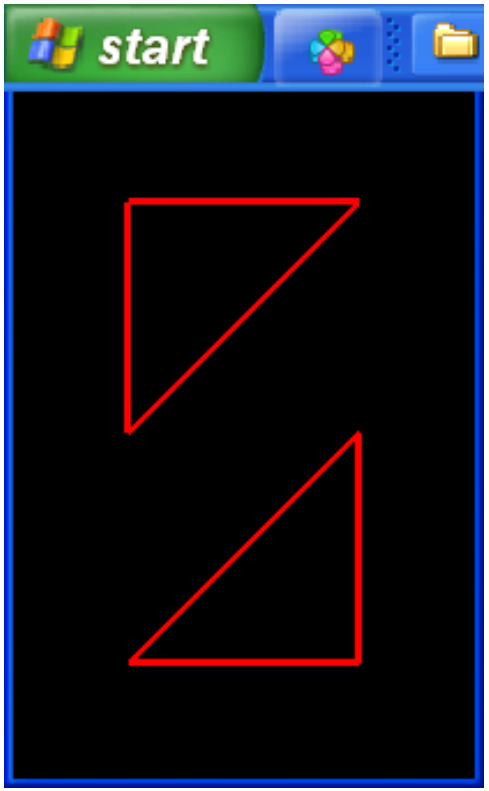
```
glBegin(GL_TRIANGLES);
    glVertex2f(-0.5, 1.0);
    glVertex2f( 0.5, 1.0);
    glVertex2f(-0.5, 0.0);
    glVertex2f( 0.5, 0.0);
    glVertex2f(-0.5, -1.0);
    glVertex2f( 0.5, -1.0);
glEnd();
```



# Draw Object

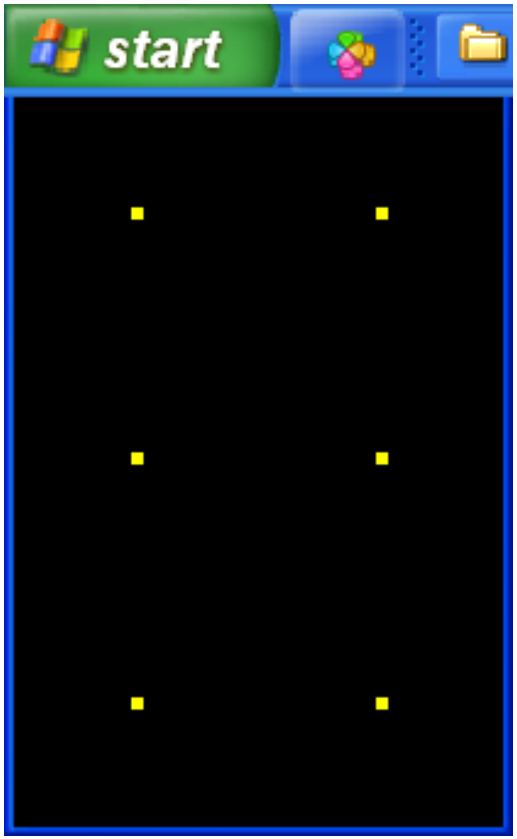
- glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
- glColor3f(1.0, 0.0, 0.0);
- glLineWidth(3.0);



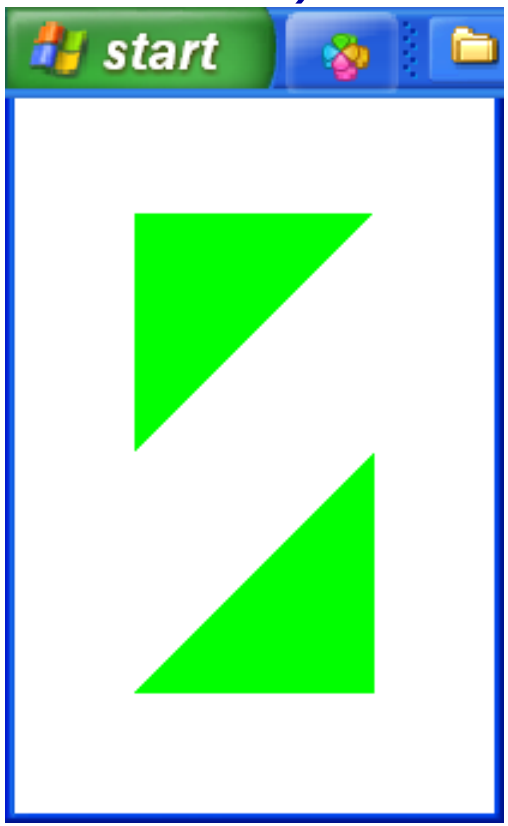
# Draw Object

- glPolygonMode(GL_FRONT_AND_BACK, GL_POINT);
- glColor3f(1.0, 1.0, 0.0);
- glPointSize(5);

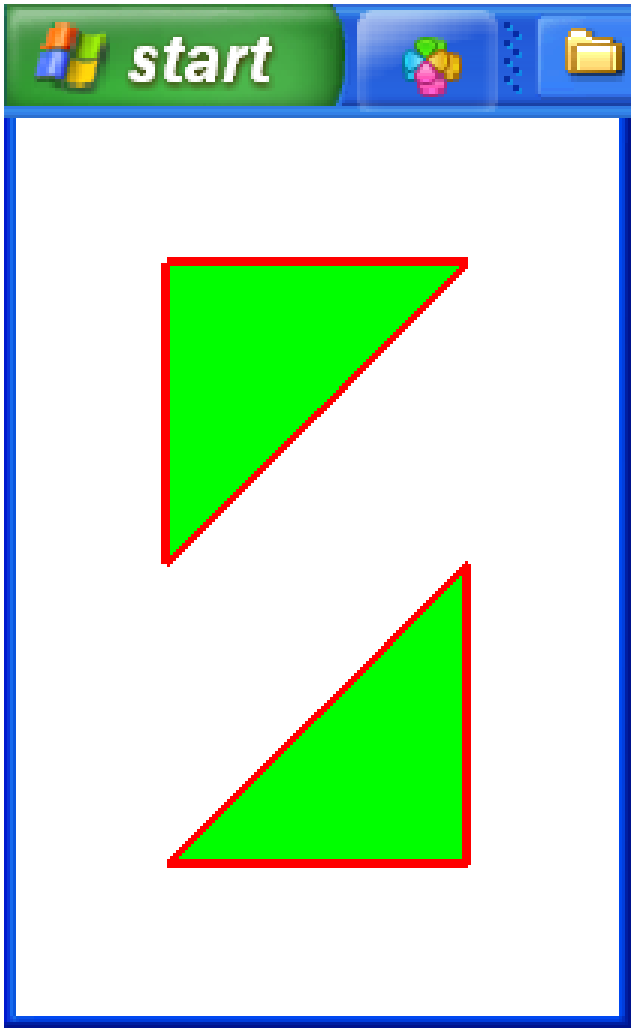


# Draw Object

- glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
- glColor3f(0.0, 1.0, 0.0);
- glClearColor(1.0, 1.0, 1.0, 1.0);



# Draw Object

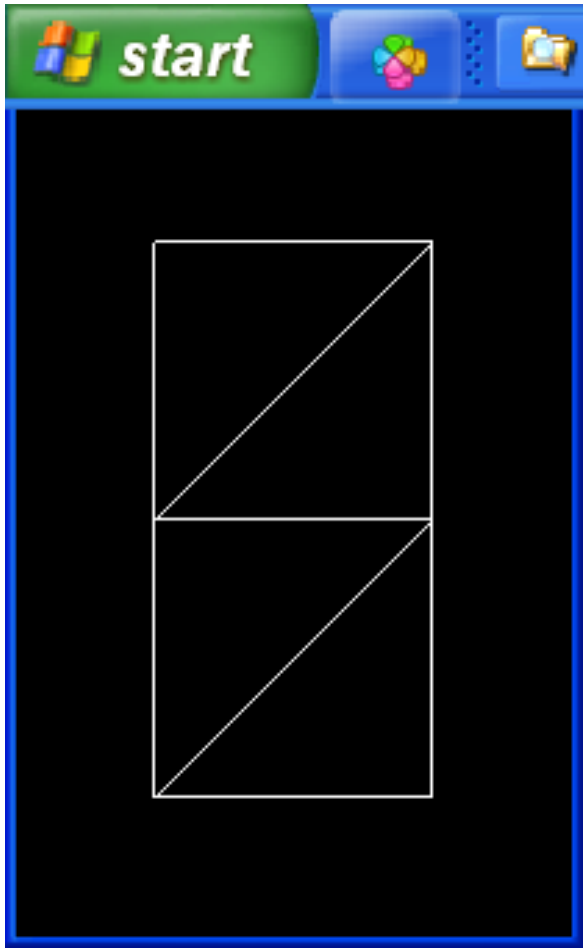


# Draw Object

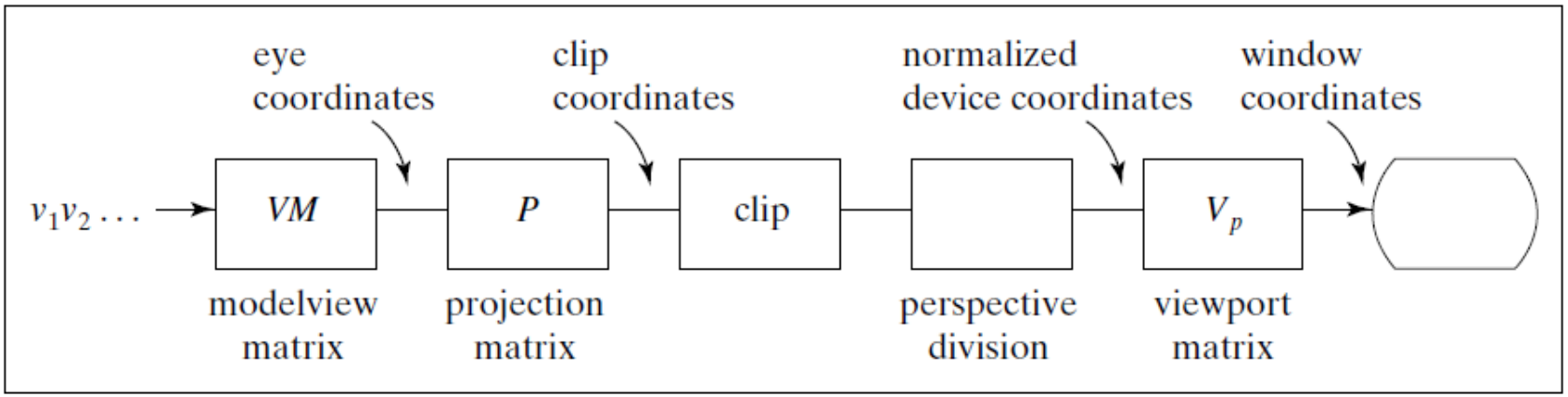
```
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glColor3f(0.0, 1.0, 0.0);
glClearColor(1.0, 1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
    ……................
glEnd();
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
glColor3f(1.0, 0.0, 0.0);
glLineWidth(3);
glBegin(GL_TRIANGLES);
    ……................
glEnd();
```

# Draw Object

```
glBegin(GL_TRIANGLES);
    glVertex2f(-0.5, 1.0);
    glVertex2f( 0.5, 1.0);
    glVertex2f(-0.5, 0.0);
    glVertex2f(-0.5, 0.0);
    glVertex2f( 0.5, 1.0);
    glVertex2f( 0.5, 0.0);
    glVertex2f(-0.5, -1.0);
    glVertex2f(-0.5, 0.0);
    glVertex2f( 0.5, 0.0);
    glVertex2f( 0.5, 0.0);
    glVertex2f(-0.5, -1.0);
    glVertex2f( 0.5, -1.0);
glEnd();
```

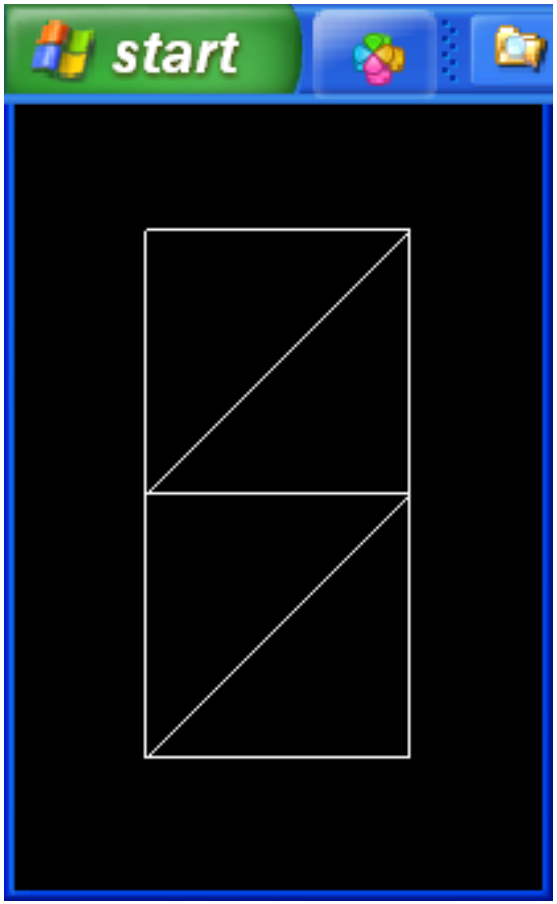


# Draw Object

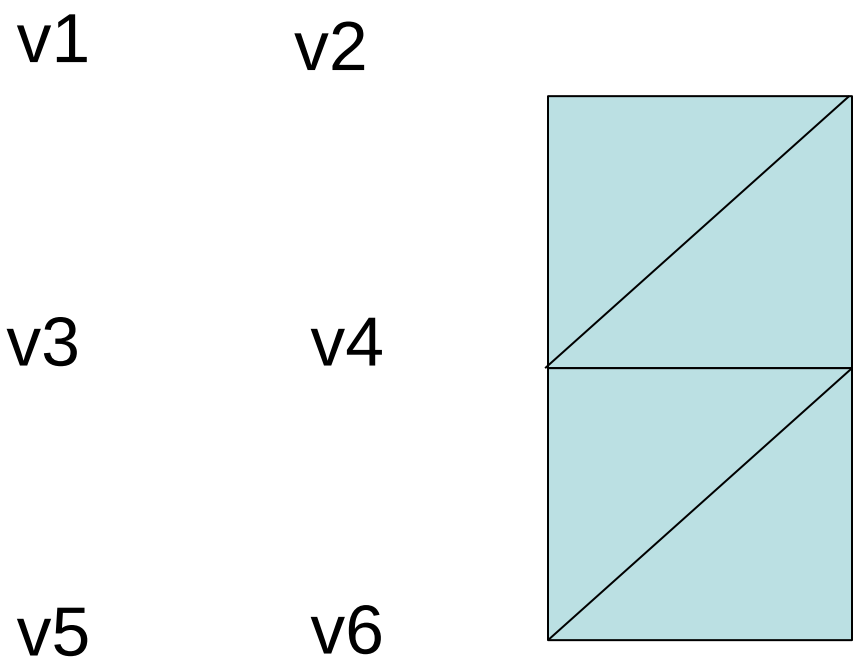

eye coordinates
clip coordinates
normalized device coordinates
window coordinates
$v_1 v_2 \ldots$
VM
P
clip
$V_p$
modelview matrix
projection matrix
perspective division
viewport matrix

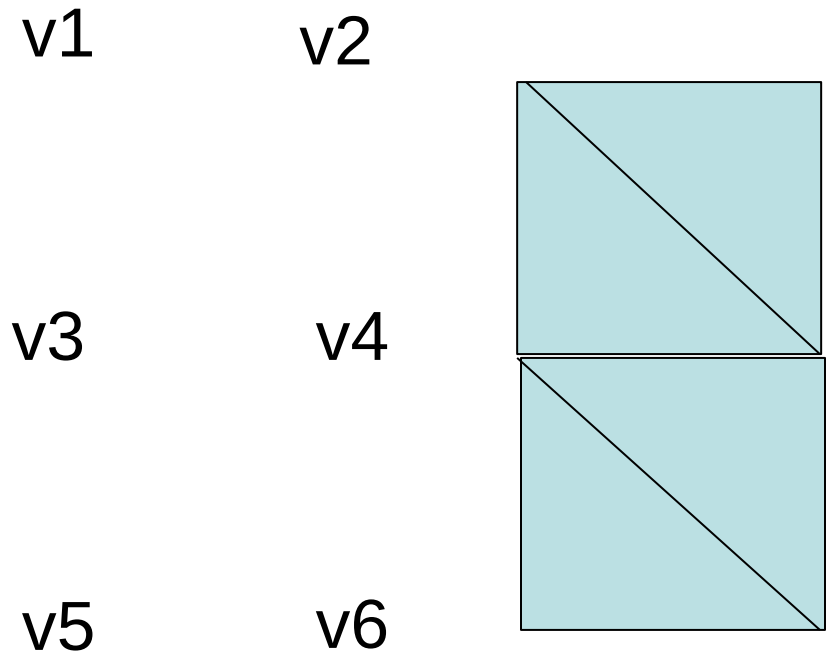
# Draw Object

```
glBegin(GL_TRIANGLE_STRIP);
    glVertex2f(-0.5, 1.0);//v1
    glVertex2f( 0.5, 1.0);//v2
    glVertex2f(-0.5, 0.0);//v3
    glVertex2f( 0.5, 0.0);//v4
    glVertex2f(-0.5, -1.0);//v5
    glVertex2f( 0.5, -1.0);//v6
glEnd();
```



# Draw Object



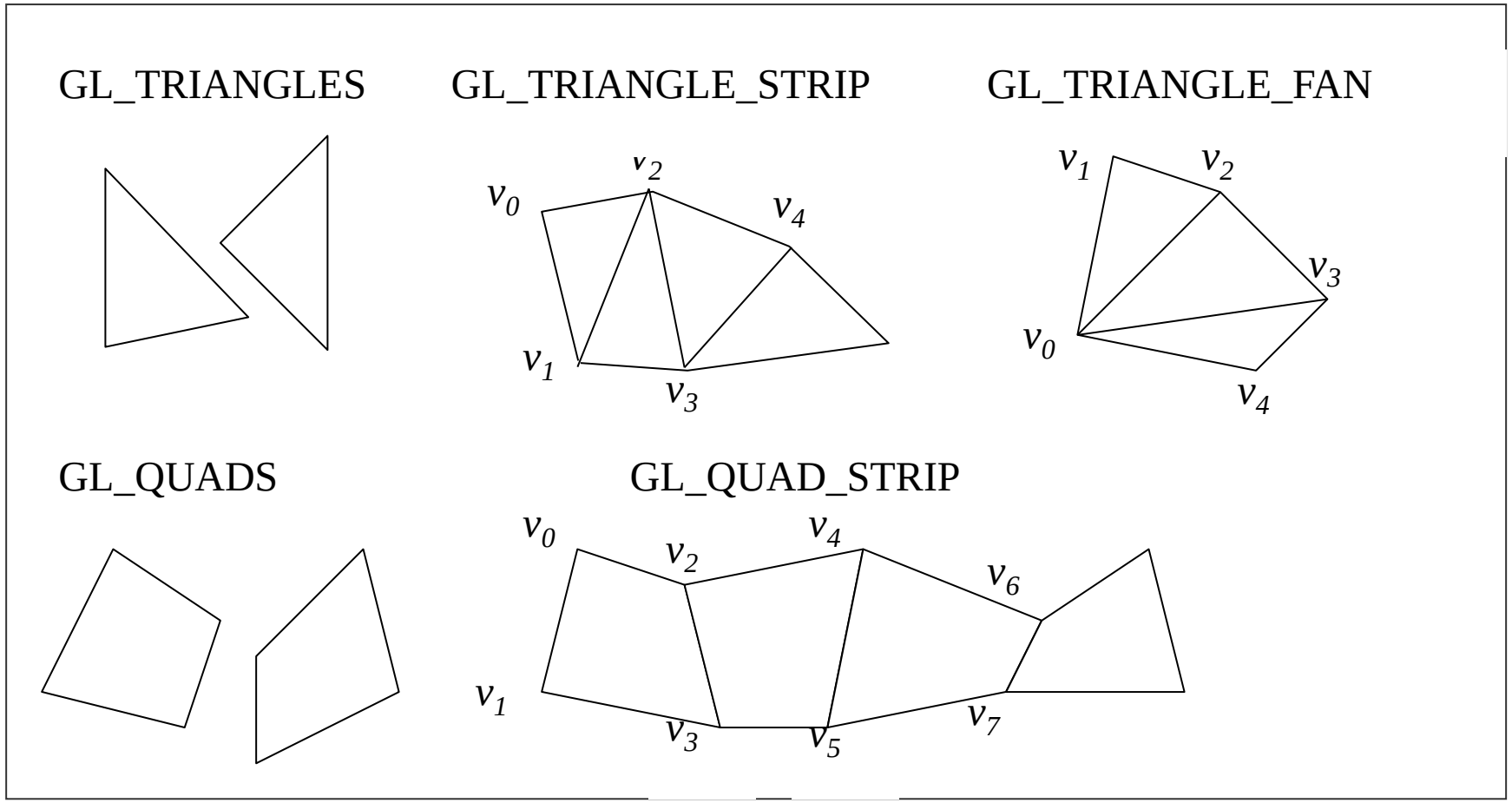




v1, v2, v3, v4, v5, v6

v2, v1, v4, v3, v6, v5

# Draw Object

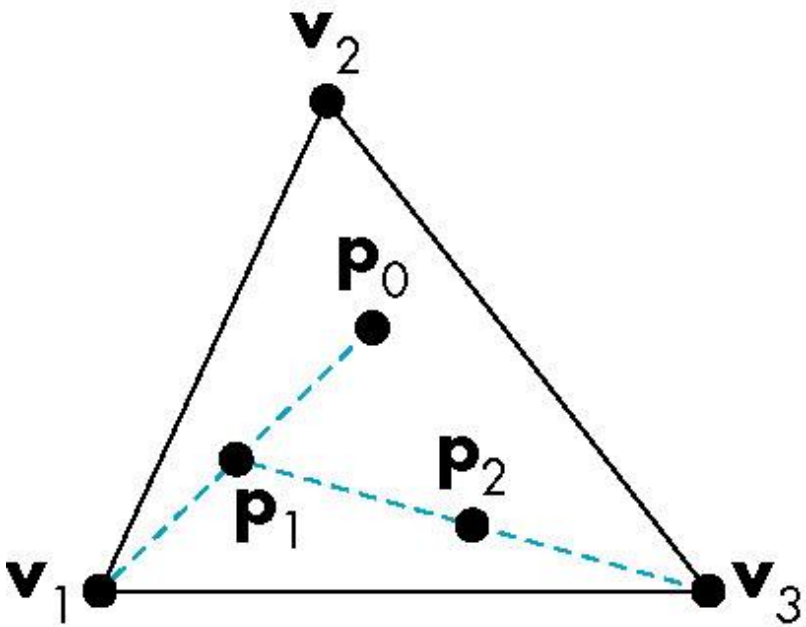
GL_QUADS, GL_QUAD_STRIP, GL_TRIANGLE_FAN
GL_POLYGON

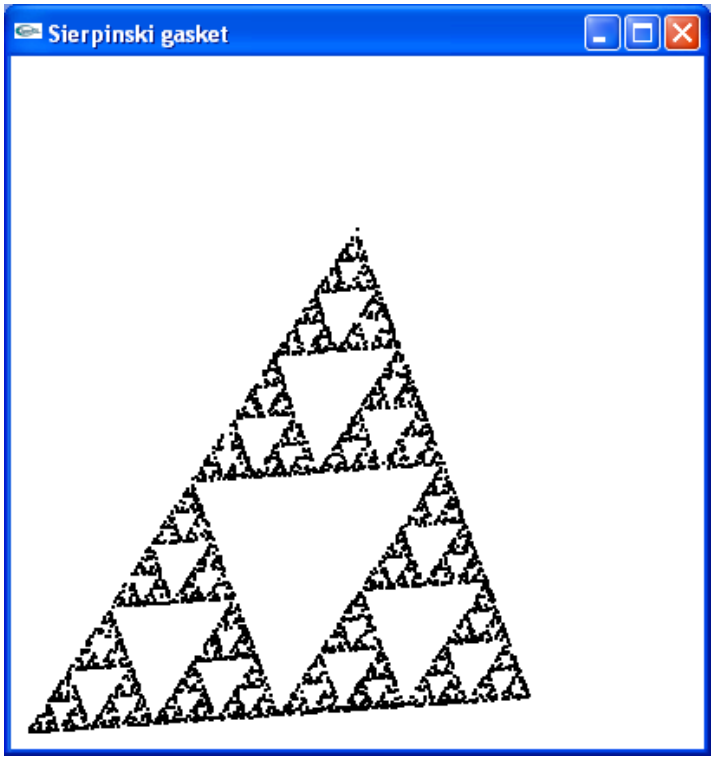


# Draw Object

```
void drawPoint(GLint x, GLint y) {
    glBegin(GL_POINTS);
        glVertex2i(x, y);
    glEnd();
}
void drawLine(GLint x1, GLint y1, GLint x2, GLint y2){
    glBegin(GL_LINES);
        glVertex2i(x1, y1);
        glVertex2i(x2, y2);
    glEnd();
}
```

# The Sierpinski Gasket



# The Sierpinski Gasket

1. Pick an initial point (x, y, z) at random inside the triangle
2. Select one of the three vertices at random
3. Find the location halfway between the initial point and the randomly selected vertex
4. Display this new point by putting some sort of marker, such as a small circle at the corresponding location on the display
5. Replace the point at (x, y, z) with this new point
6. Return to step 2

# The Sierpinski Gasket

```
main()
{
    Initialize_the_system();

    for(some_number_of_points)
    {
        pt = generate_a_point();
        Display_the_point(pt);
    }
}
```

# The Sierpinski Gasket

```
void myinit()
{
    glClearColor(1.0, 1.0, 1.0, 1.0); /* white background */
    glColor3f(1.0, 0.0, 0.0); /* draw in red */

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 50.0, 0.0, 50.0);
    glMatrixMode(GL_MODELVIEW);
}
```

# The Sierpinski Gasket

```
void display( void ){
    GLfloat vertices[3][2]={{0.0,0.0},{25.0,50.0},{50.0,0.0}}; /* A triangle */
    int j, k;
    srand(time(NULL));       /* standard random number generator */
    GLfloat p[2] ={7.5,5.0};  /* An arbitrary initial point inside traingle */

    glClear(GL_COLOR_BUFFER_BIT);  /*clear the window */
    glBegin(GL_POINTS);
     for( k=0; k<5000; k++) {
            j = rand()%3; /* pick a vertex at random */
            p[0] = (p[0]+vertices[j][0])/2.0;
            p[1] = (p[1]+vertices[j][1])/2.0;
            glVertex2fv(p);
     }
     glEnd();
     glFlush(); /* clear buffers */
 }
```

# Further Reading

- "**Interactive Computer Graphics: A Topdown Approach Using OpenGL**", *Edward Angel*
  - Chapter 2: Graphics Programming
- "Đồ họa máy tính trong không gian hai chiều", Trần Giang Sơn
  - Chương 2: Bước đầu tạo hình ảnh
- "Đồ họa máy tính trong không gian ba chiều", Trần Giang Sơn
  - Chương 1: Mô hình hóa đối tượng ba chiều bằng lưới đa giác