CSE 404

COMPUTER GRAPHICS

Mohammad Imrul Jubair

What we do with this 'Computer Graphics'

Simply, Computer Graphics is……

Producing pictures or images using a computer

What we do with this 'Computer Graphics'

Simply, Computer Graphics is……

Producing pictures or images using a computer

What we do with this 'Computer Graphics'

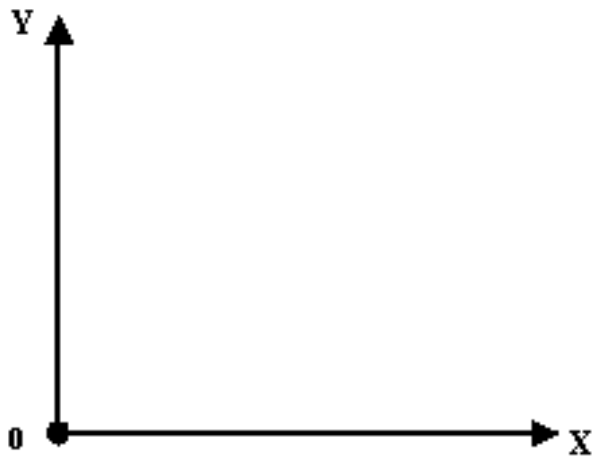
Simply, Computer Graphics is……

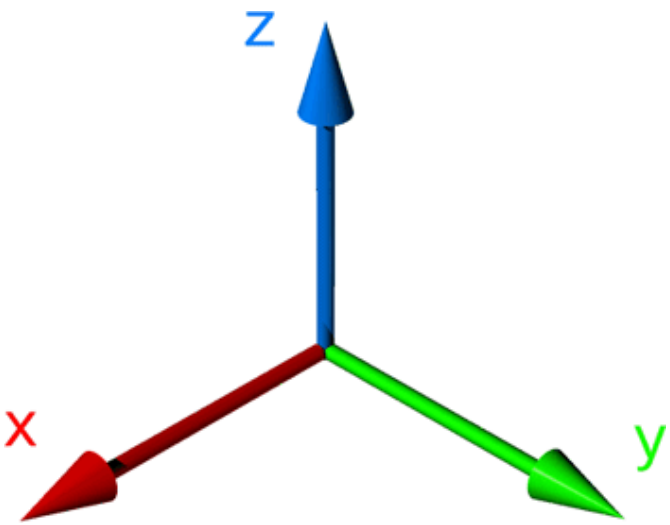
Producing pictures or images using a computer

basic terms

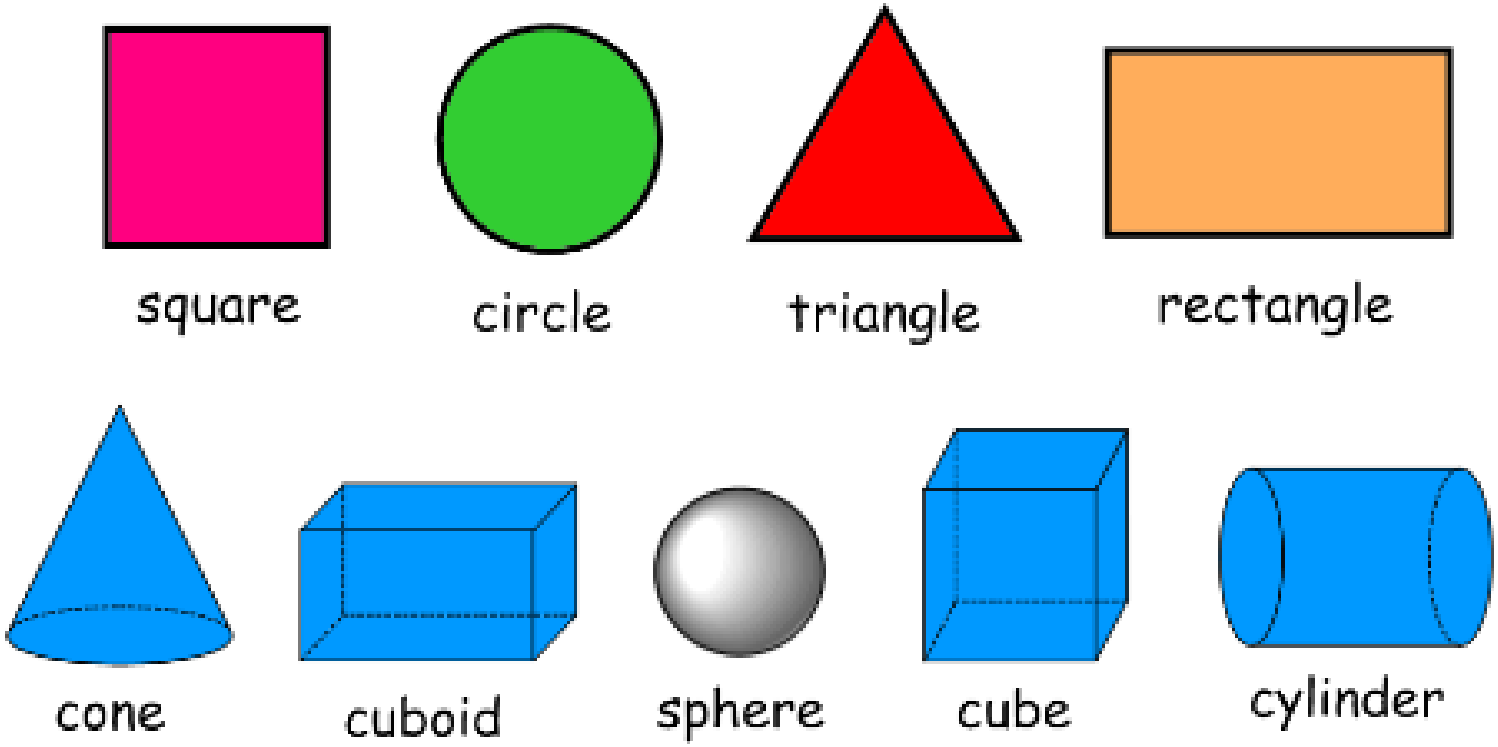
related to computer graphics

2D and 3D





2D and 3D


square
circle
triangle
rectangle
cone
cuboid
sphere
cube
cylinder

2D vs. 3D

2D vs. 3D

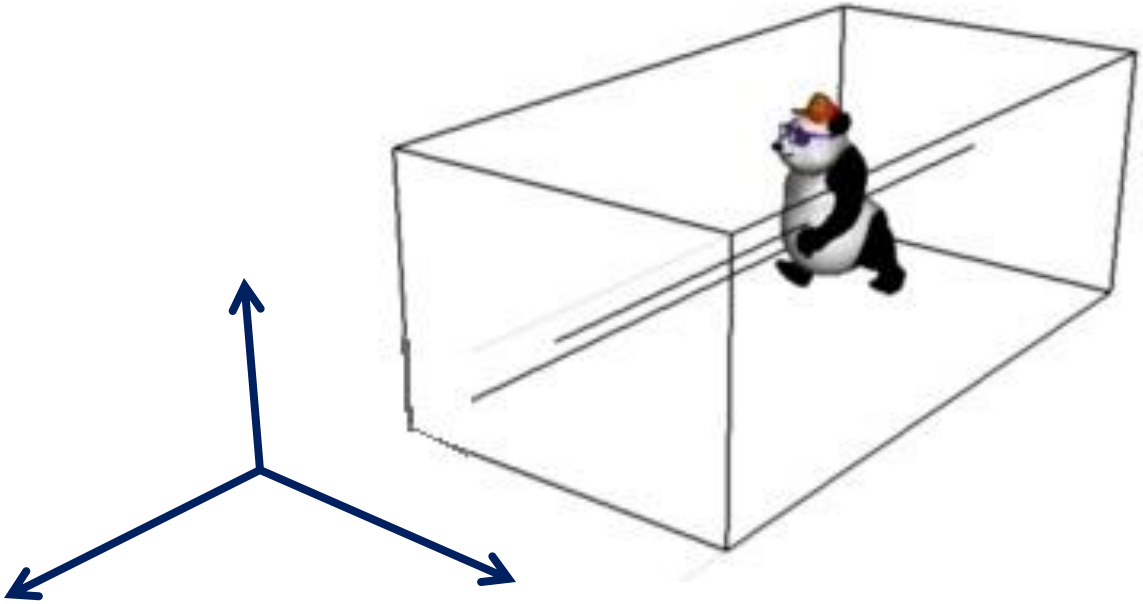
‘Computer Graphics’ Definition

To be more precise,

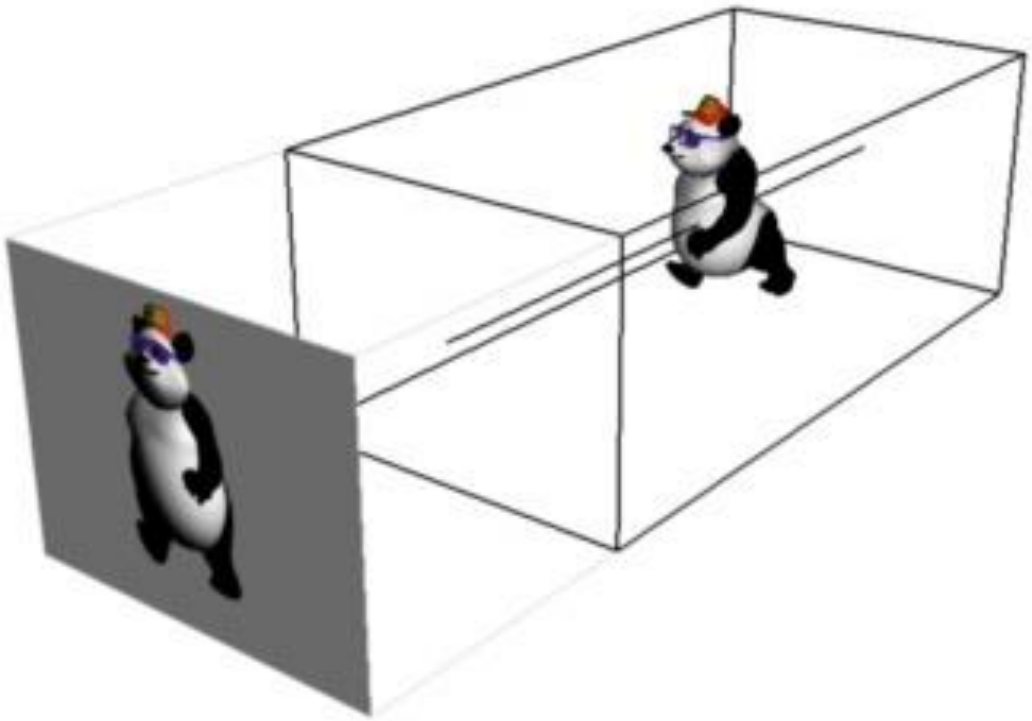
Computer Graphics is……

Creation, Manipulation, and Storage of geometric objects **(modeling)** and their images **(rendering)**

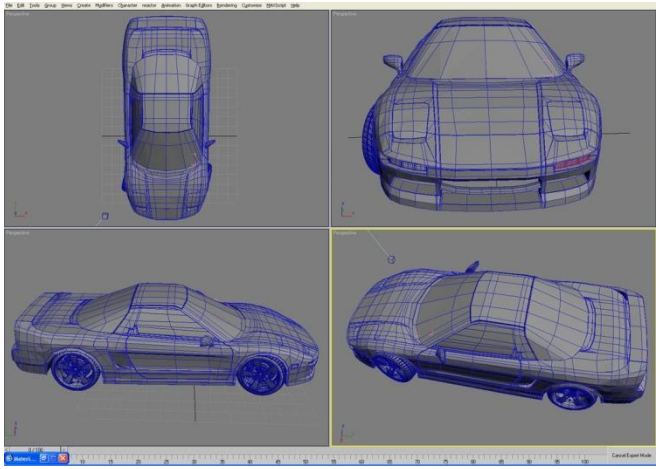
Modeling



Rendering



Modeling and Rendering



openGL

What is openGL ?

- Open Graphics Library

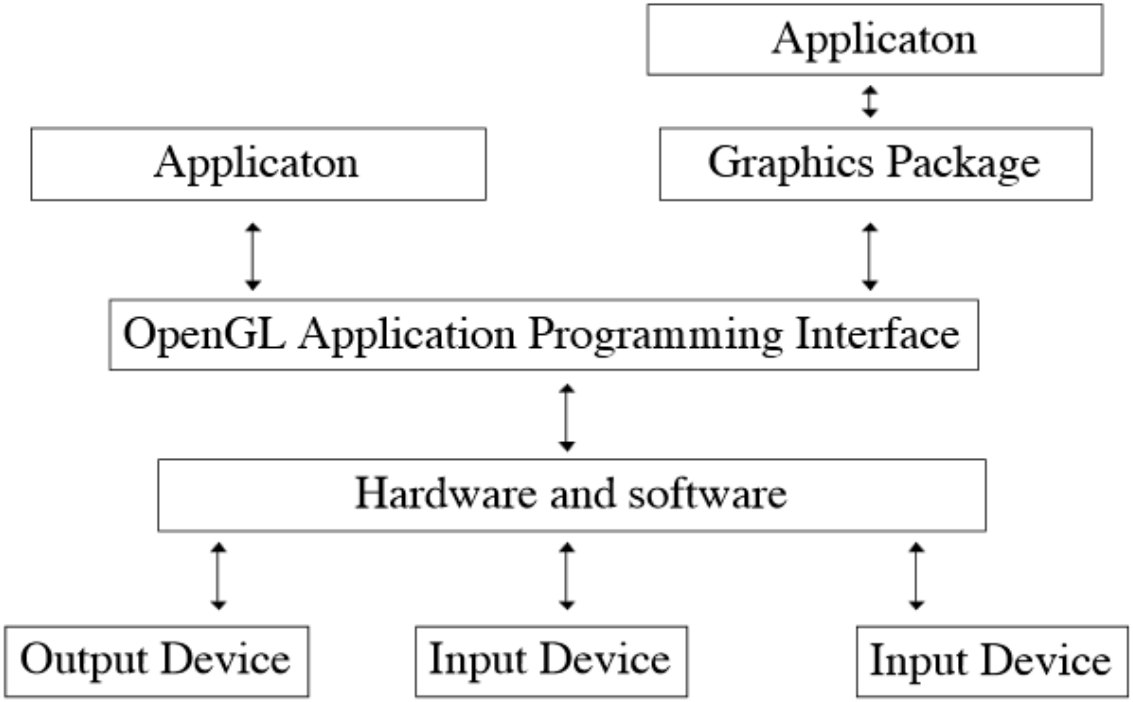
- It's a way to draw stuff in 3D.

- The graphics card is where the 3D computation happens. The purpose of OpenGL is to communicate with the graphics card about your 3D scene.

- So why not talk to the graphics card directly? Each graphics card is a little different. In a sense, they all speak different "languages". To talk to them all, you can either learn all of their languages, or find a "translator" that knows all of their languages and talk to the translator, so that you only have to know one language. OpenGL serves as a "translator" for graphics cards.

What is openGL ?

Programmer's View



OpenGL Utility Toolkit

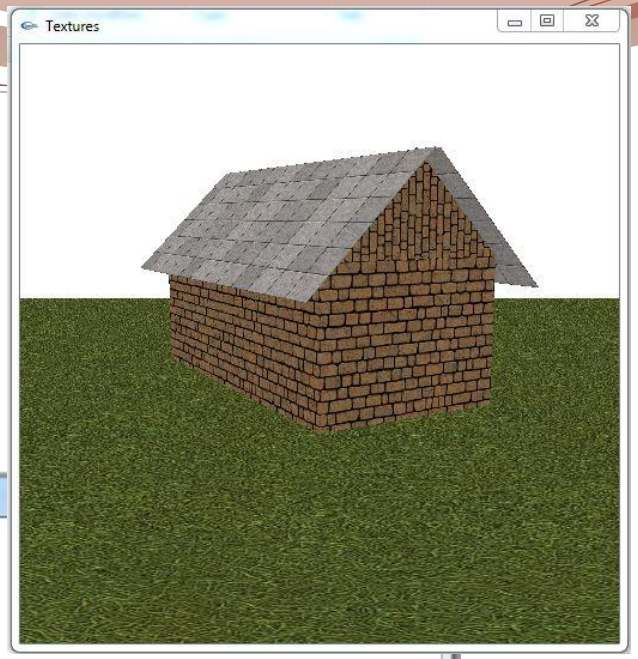
(GLUT)

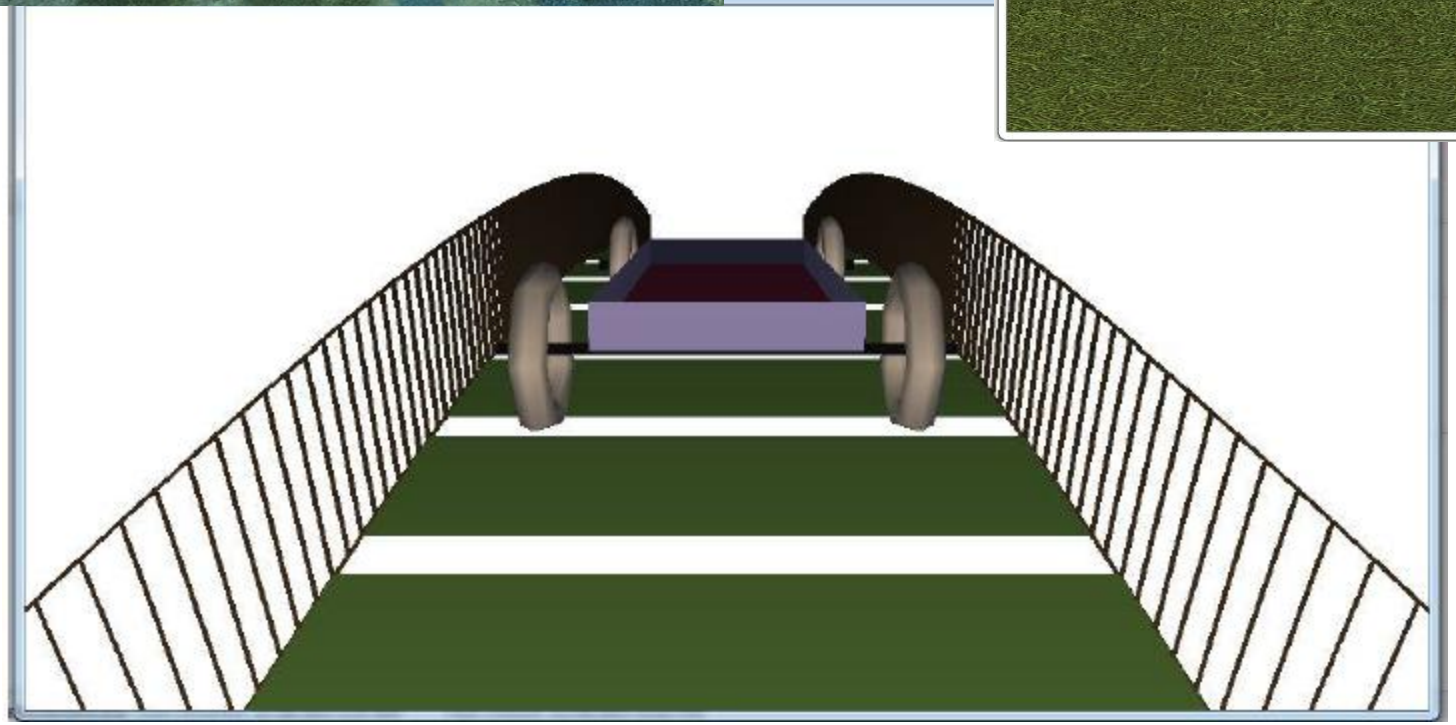
What is GLUT ?

✓ The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs

✓GLUT makes it considerably easier to learn about and explore OpenGL programming.

✓GLUT provides a portable API so you can write a single OpenGL program that works across all PC and workstation OS platforms.


Textures



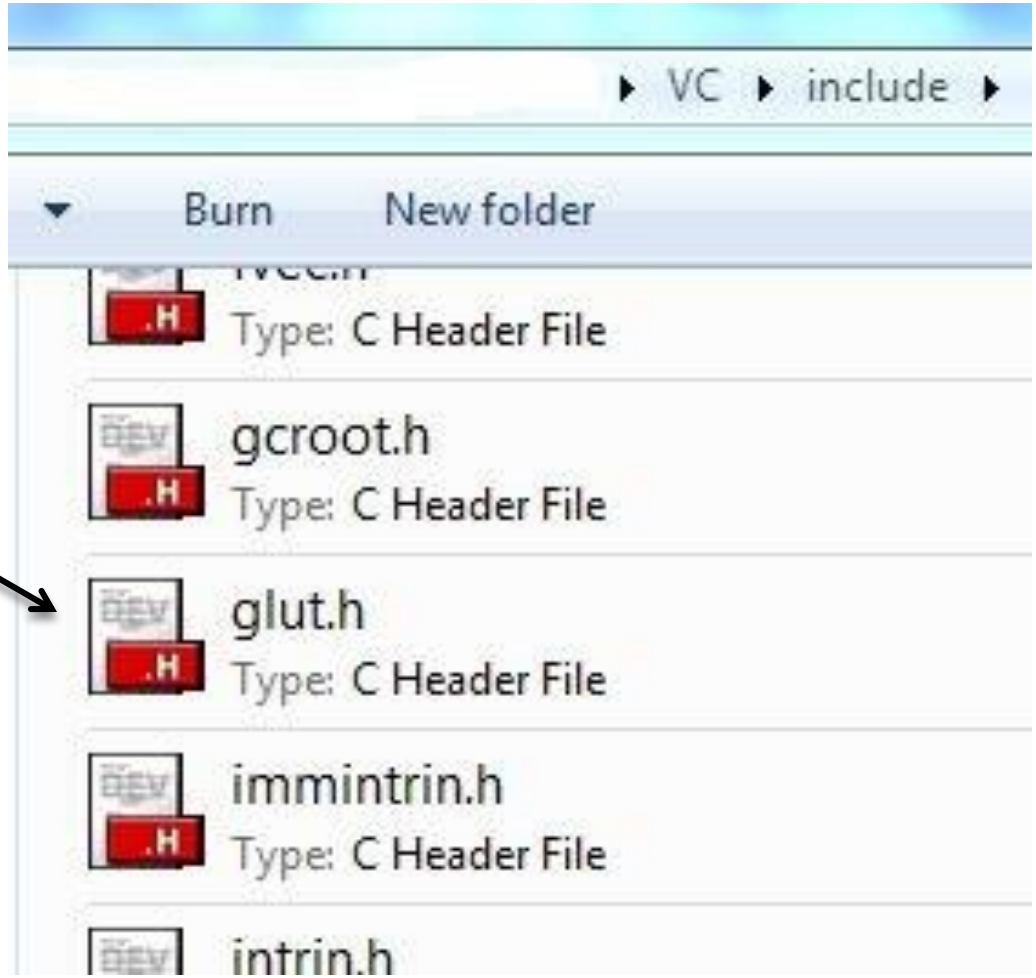
Let’s Start

Lets configure openGL in our PC !

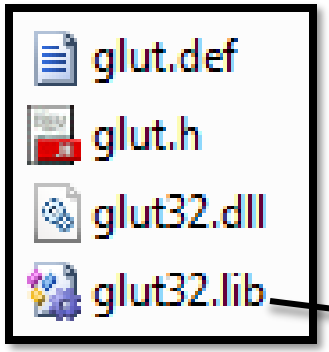
https://sites.google.com/site/imruljubair/teaching

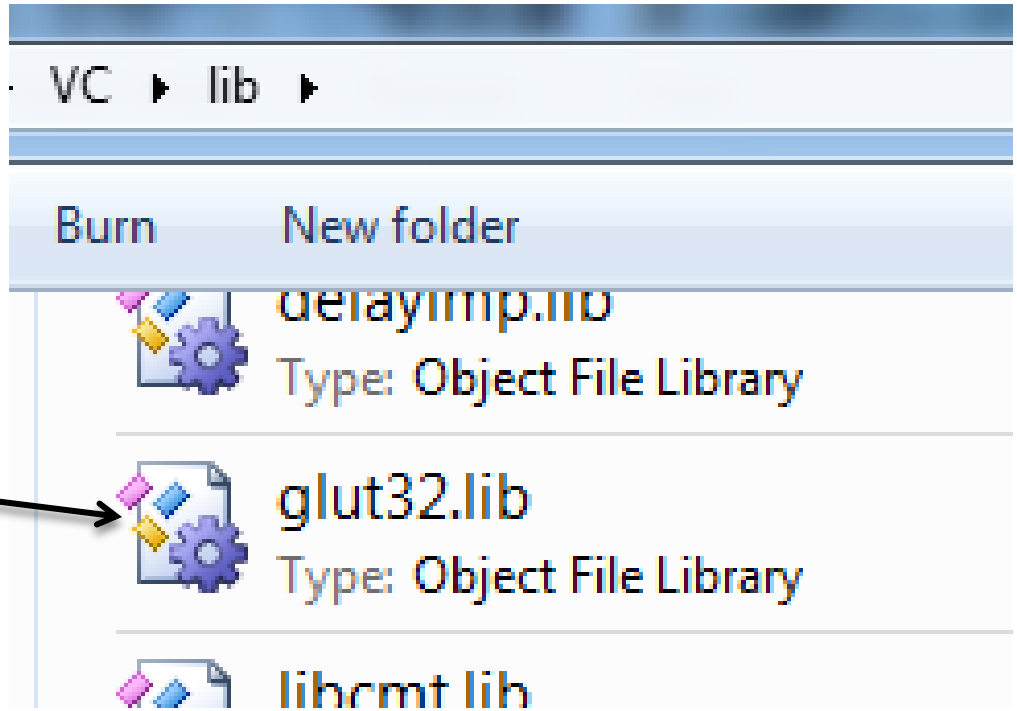
Microsoft Visual Studio

Lets configure openGL in our PC !

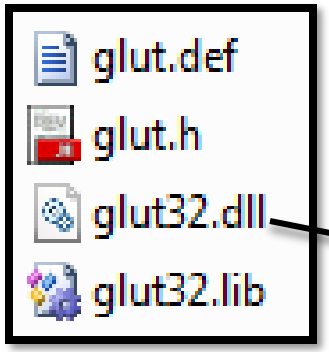


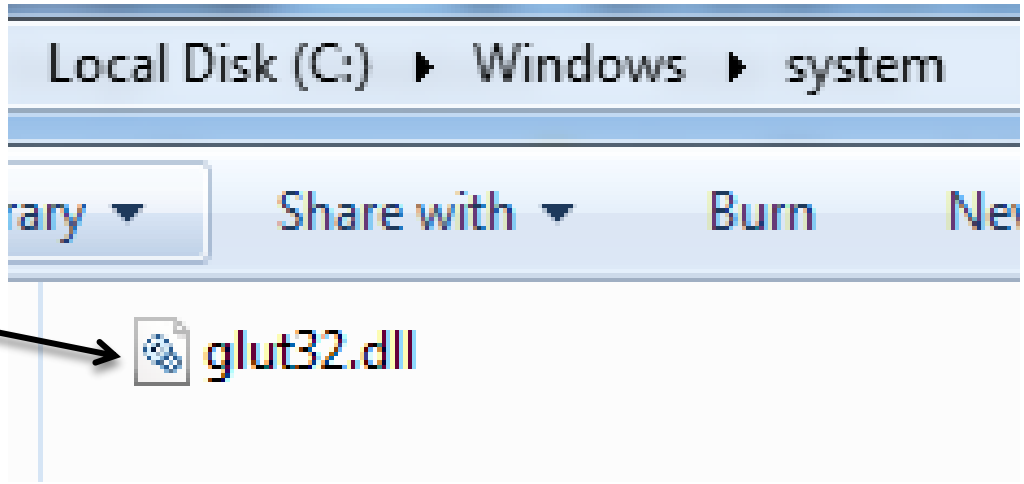
Lets configure openGL in our PC !





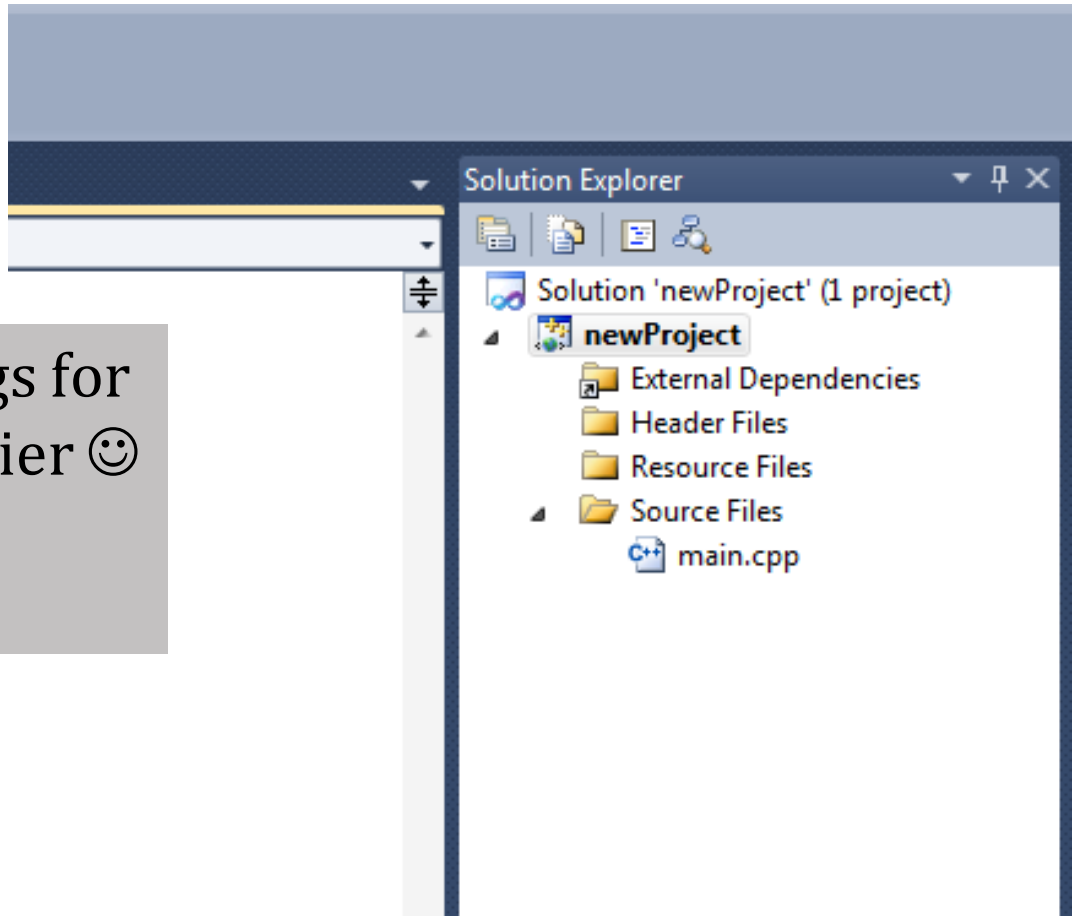
Lets configure openGL in our PC !





How to code?

Just create a new empty project .....



There are some built-in things for you, which makes coding easier ☺ !

Lets know about those.

Lets explain our first code

```
#include <glut.h>

void Draw( ) {
……................................
}

void Initialize( ) {
……..........................................
}

int main(int iArgc, char** cppArgv) {
……..................................
}
```

```
#include <glut.h>

void Draw( ) {
……................................
}

void Initialize( ) {
……..........................................
}

int main(int iArgc, char** cppArgv) {
……..................................
}
```

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

program arguments

```
int main(int iArgc, char** cppArgv)
{
    glutInit(&iArgc, cppArgv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(250, 250);
    glutInitWindowPosition(200, 200);
    glutCreateWindow("CSE_404");
    Initialize();
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0;
}
```

initializes GLUT

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

set up its frame buffer

```
int main(int iArgc, char** cppArgv)
{
        glutInit(&iArgc, cppArgv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(250, 250);
        glutInitWindowPosition(200, 200);
        glutCreateWindow("CSE_404");
        Initialize();
        glutDisplayFunc(Draw);
        glutMainLoop();
        return 0;
}
```

Display Mode	Meaning
GLUT_RGB	Use RGB colors
GLUT_RGBA	Use RGB plus α (for transparency)
GLUT_INDEX	Use colormapped colors (not recommended)
GLUT_DOUBLE	Use double buffering (recommended)
GLUT_SINGLE	Use single buffering (not recommended)
GLUT_DEPTH	Use depth buffer (needed for hidden surface removal)

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

- width and height of the graphics window
- The general form is- glutInitWindowSize(int width, int height).

```
int main(int iArgc, char** cppArgv)
{
        glutInit(&iArgc, cppArgv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(250, 250);
        glutInitWindowPosition(200, 200);
        glutCreateWindow("CSE_404");
        Initialize();
        glutDisplayFunc(Draw);
        glutMainLoop();
        return 0;
}
```

location of the upper left corner of the graphics window

```
int main(int iArgc, char** cppArgv)
{
        glutInit(&iArgc, cppArgv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(250, 250);
        glutInitWindowPosition(200, 200);
        glutCreateWindow("CSE_404");
        Initialize();
        glutDisplayFunc(Draw);
        glutMainLoop();
        return 0;
}
```

creates the graphics window. The general form of the command is – glutCreateWindowchar(*title) where title is a character string.

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

```
void Initialize( ) {
	glClearColor(0.0, 0.0, 0.0, 0.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

```
int main(int iArgc, char** cppArgv)
{
        glutInit(&iArgc, cppArgv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(250, 250);
        glutInitWindowPosition(200, 200);
        glutCreateWindow("CSE_404");
        Initialize();
        glutDisplayFunc(Draw);
        glutMainLoop();
        return 0;
}
```

clears the window, by overwriting it with the background color. glClearColor(Red, Green, Blue, Alpha).

```
void Initialize( ) {
        glClearColor(0.0, 0.0, 0.0, 0.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

```
int main(int iArgc, char** cppArgv)
{
        glutInit(&iArgc, cppArgv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(250, 250);
        glutInitWindowPosition(200, 200);
        glutCreateWindow("CSE_404");
        Initialize();
        glutDisplayFunc(Draw);
        glutMainLoop();
        return 0;
}
```

```
void Initialize( ) {
        glClearColor(0.0,  0.0,  0.0,  0.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

- OpenGL has a number of commands for handling matrices
- glMatrixMode(mode) is used to specify current matrix

```
int main(int iArgc, char** cppArgv)
{
        glutInit(&iArgc, cppArgv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(250, 250);
        glutInitWindowPosition(200, 200);
        glutCreateWindow("CSE_404");
        Initialize();
        glutDisplayFunc(Draw);
        glutMainLoop();
        return 0;
}
```

- GL_MODELVIEW,
- GL_PROJECTION, and
- GL_TEXTURE

```
void Initialize( ) {
        glClearColor(0.0,  0.0,  0.0,  0.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

```
int main(int iArgc, char** cppArgv)
{
        glutInit(&iArgc, cppArgv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(250, 250);
        glutInitWindowPosition(200, 200);
        glutCreateWindow("CSE_404");
        Initialize();
        glutDisplayFunc(Draw);
        glutMainLoop();
        return 0;
}
```

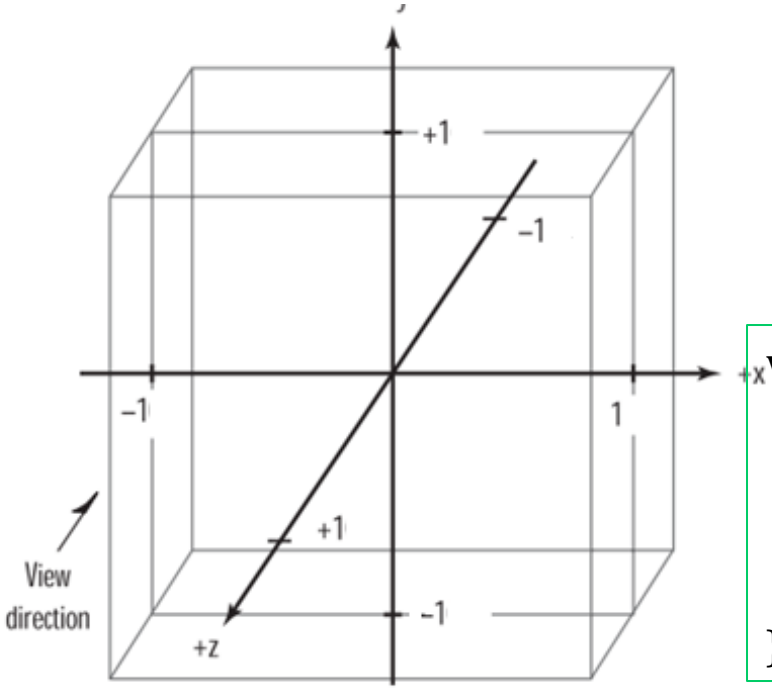
```
void Initialize( ) {
        glClearColor(0.0,  0.0,  0.0,  0.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

initialize to identity

```
int main(int iArgc, char** cppArgv)
{
        glutInit(&iArgc, cppArgv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(250, 250);
        glutInitWindowPosition(200, 200);
        glutCreateWindow("CSE_404");
        Initialize();
        glutDisplayFunc(Draw);
        glutMainLoop();
        return 0;
}
```

```
void Initialize( ) {
        glClearColor(0.0,  0.0,  0.0,  0.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(-1, 1, -1, 1, 1, -1);
}
```

- glOrtho(left, right, bottom, top, near, far);
- To set up viewing cube




```
void Initialize( ) {
        glClearColor(0.0,  0.0,  0.0,  0.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(-1, 1, -1, 1, 1, -1);
}
```

- glOrtho(left, right, bottom, top, near, far);
- To set up viewing cube

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

```
void Initialize( ) {
	glClearColor(0.0,  0.0,  0.0,  0.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-1, 1, -1, 1, 1, -1);
}
```

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

```
void Draw()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0, 1.0, 1.0);
	glBegin(GL_POINTS);
		glVertex3f(0.2, 0.2, 0.0);
	glEnd();
	glFlush();
}
```

```
int main(int iArgc, char** cppArgv)
{
        glutInit(&iArgc, cppArgv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(250, 250);
        glutInitWindowPosition(200, 200);
        glutCreateWindow("CSE_404");
        Initialize();
        glutDisplayFunc(Draw);
        glutMainLoop();
        return 0;
}
```

clear the window

```
void Draw()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0, 1.0, 1.0);
        glBegin(GL_POINTS);
                glVertex3f(0.2, 0.2, 0.0);
        glEnd();
        glFlush();
}
```

```
int main(int iArgc, char** cppArgv)
{
        glutInit(&iArgc, cppArgv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(250, 250);
        glutInitWindowPosition(200, 200);
        glutCreateWindow("CSE_404");
        Initialize();
        glutDisplayFunc(Draw);
        glutMainLoop();
        return 0;
}
```

- Setting color of objects
- glColor3f (R,G,B)

```
void Draw()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0, 1.0, 1.0);
        glBegin(GL_POINTS);
                glVertex3f(0.2, 0.2, 0.0);
        glEnd();
        glFlush();
}
```

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

To draw something

```
void Draw()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0, 1.0, 1.0);
	glBegin(GL_POINTS);
		glVertex3f(0.2, 0.2, 0.0);
	glEnd();
	glFlush();
}
```

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

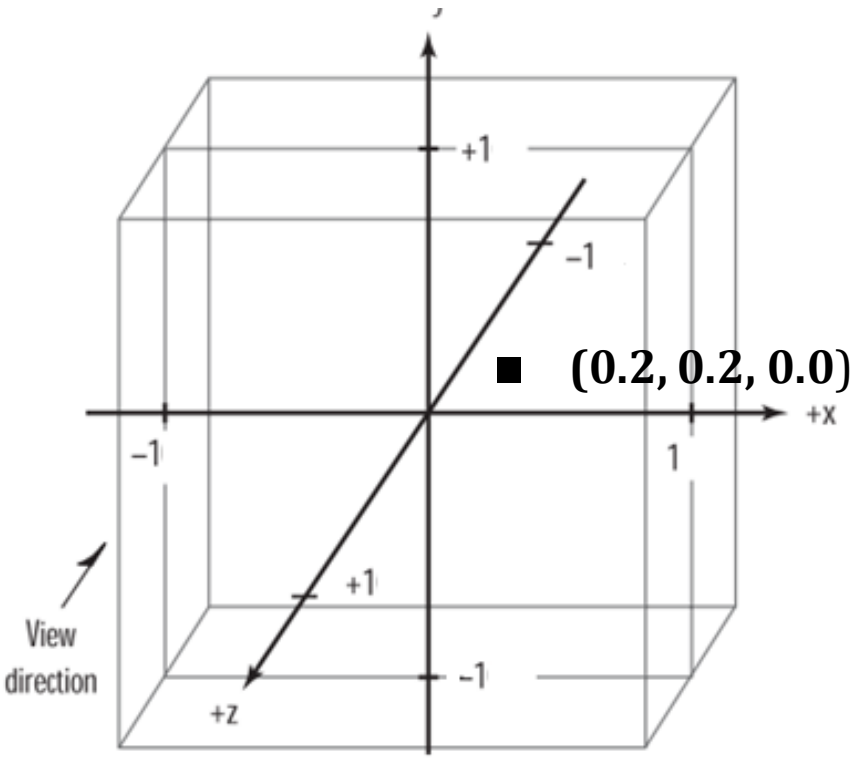
```
void Draw()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0, 1.0, 1.0);
	glBegin(GL_POINTS);
		glVertex3f(0.2, 0.2, 0.0);
	glEnd();
	glFlush();
}
```

We want to draw a Point

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

Coordinate values of that point

```
void Draw()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0, 1.0, 1.0);
	glBegin(GL_POINTS);
	glVertex3f(0.2, 0.2, 0.0);
	glEnd();
	glFlush();
}
```




```
void Draw()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0, 1.0, 1.0);
        glBegin(GL_POINTS);
                glVertex3f(0.2, 0.2, 0.0);
        glEnd();
        glFlush();
}
```

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

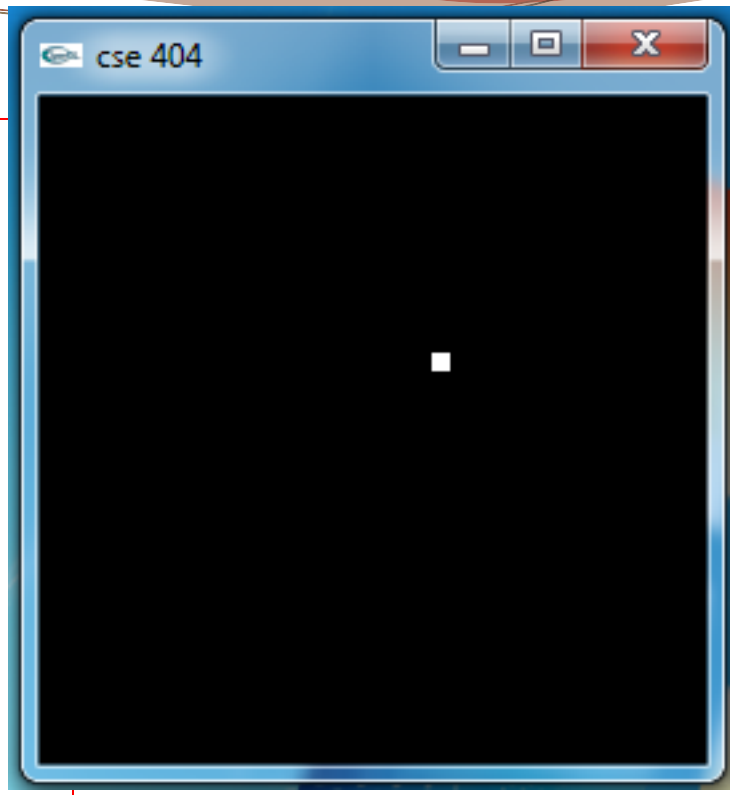
```
void Draw()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0, 1.0, 1.0);
	glBegin(GL_POINTS);
		glVertex3f(0.2, 0.2, 0.0);
	glEnd();
	glFlush();
}
```

to ensure all objects in the scene are drawn before beginning to accept user input.

```
int main(int iArgc, char** cppArgv)
{
	glutInit(&iArgc, cppArgv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(250, 250);
	glutInitWindowPosition(200, 200);
	glutCreateWindow("CSE_404");
	Initialize();
	glutDisplayFunc(Draw);
	glutMainLoop();
	return 0;
}
```

- start it running
- loops within itself, processing events and triggering your callback functions when necessary

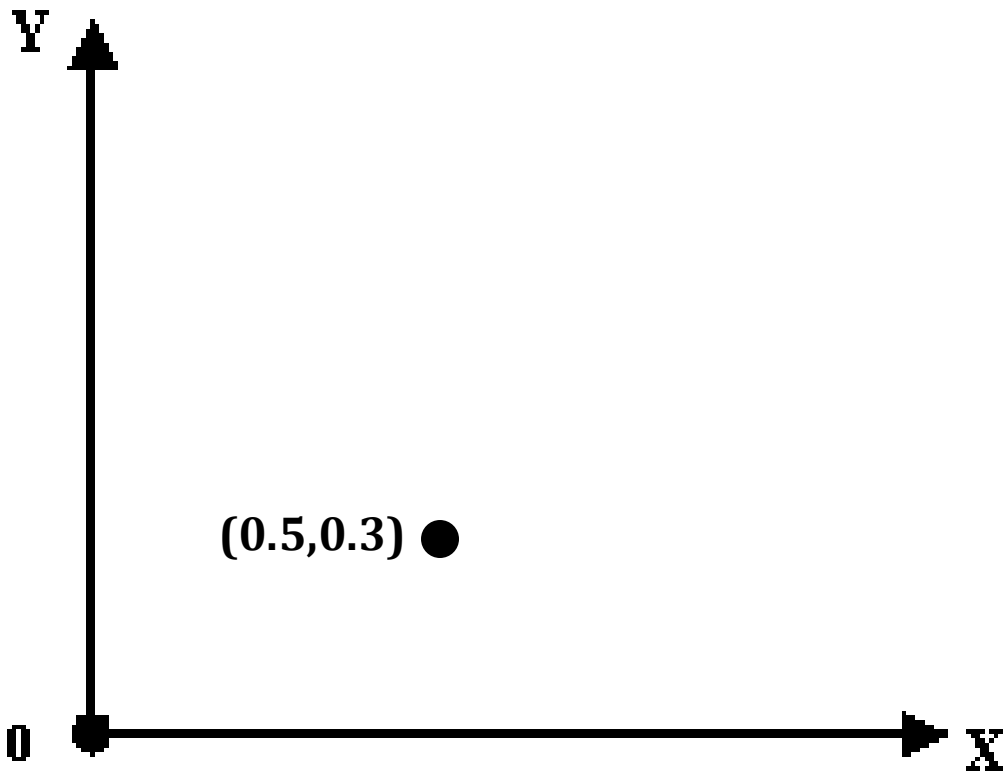
```
#include <glut.h>

void Draw( ) {
……..
}

void Initialize( ) {
……..
}

int main(int iArgc, char** cppArgv) {
……
}
```



transformation

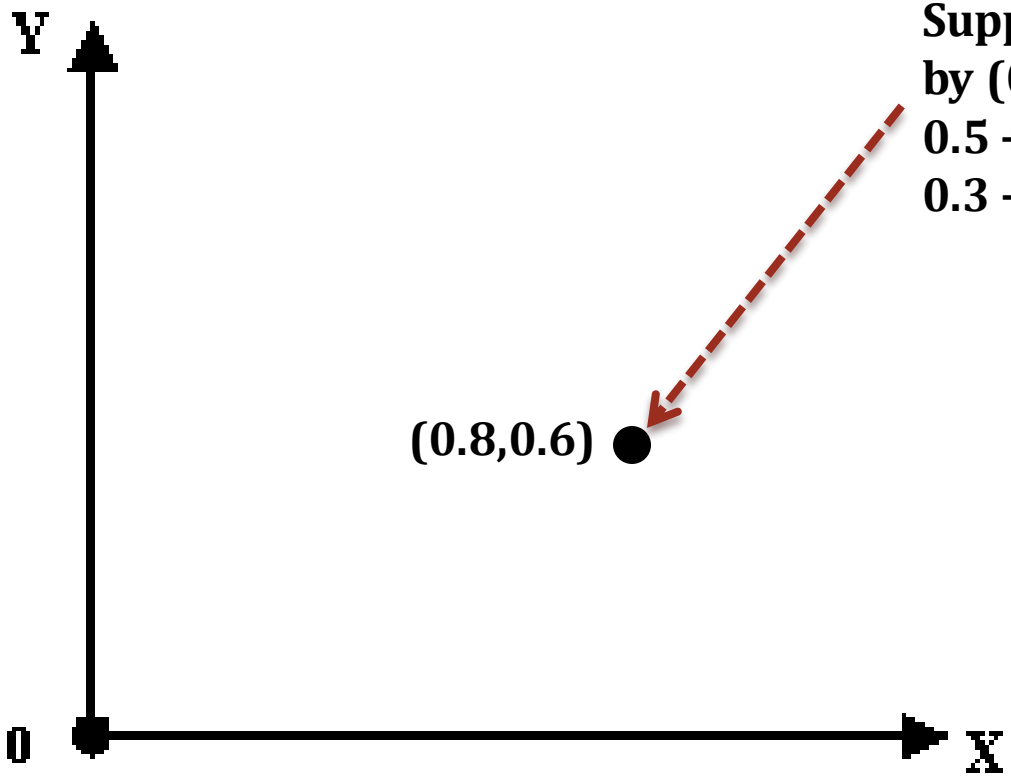
TRANSFORMATION

Translate :



TRANSFORMATION

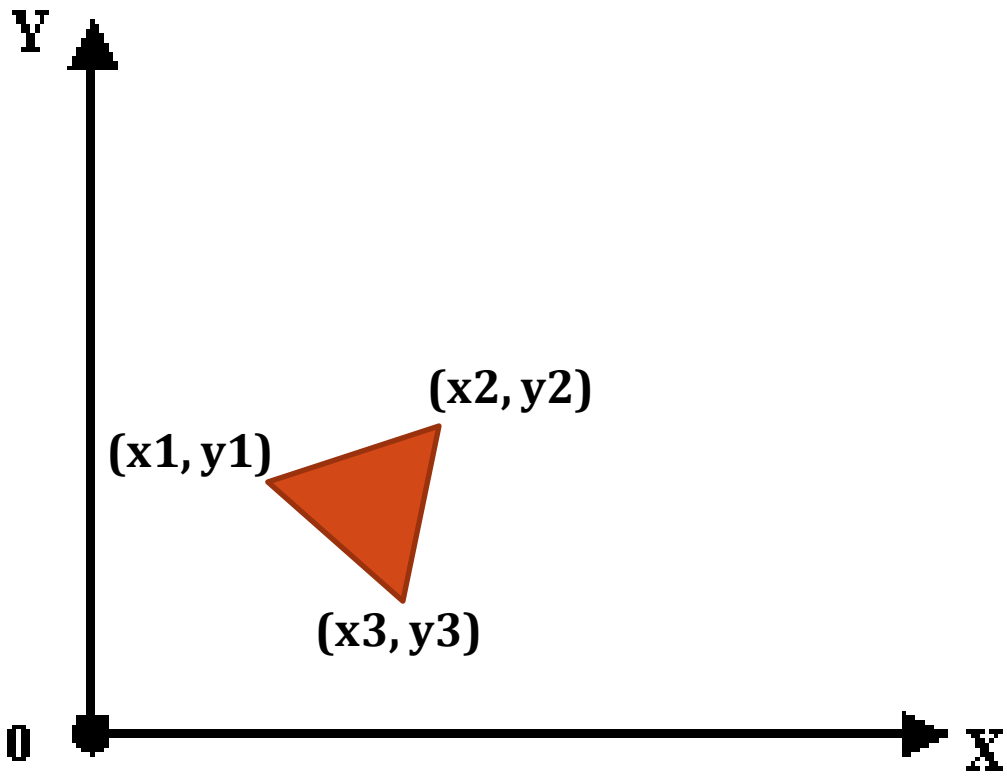
Translate :



Suppose, we've translated it by (0.3, 0.3) –
0.5 + 0.3 = 0.8
0.3 + 0.3 = 0.6

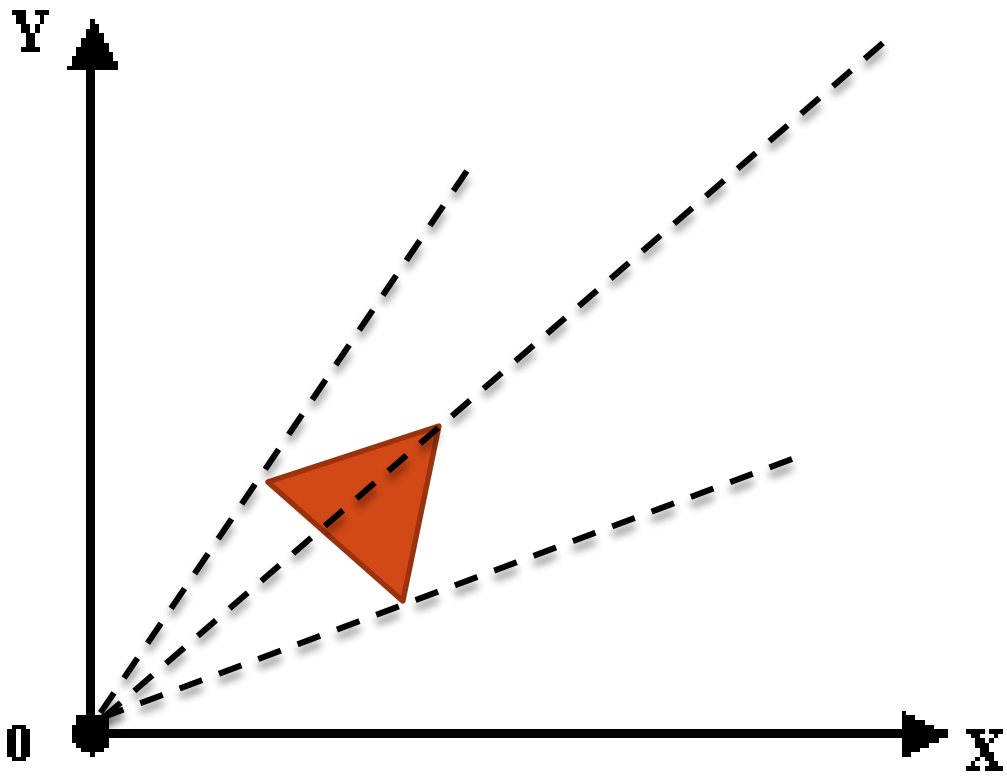
TRANSFORMATION

Scale :



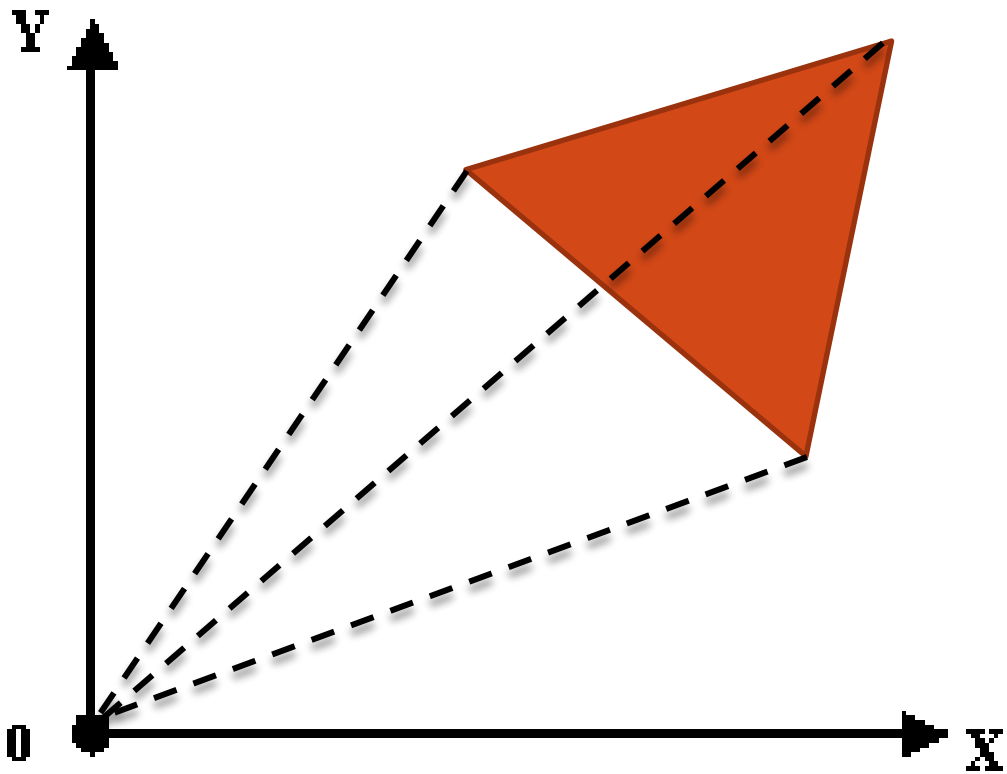
TRANSFORMATION

Scale :



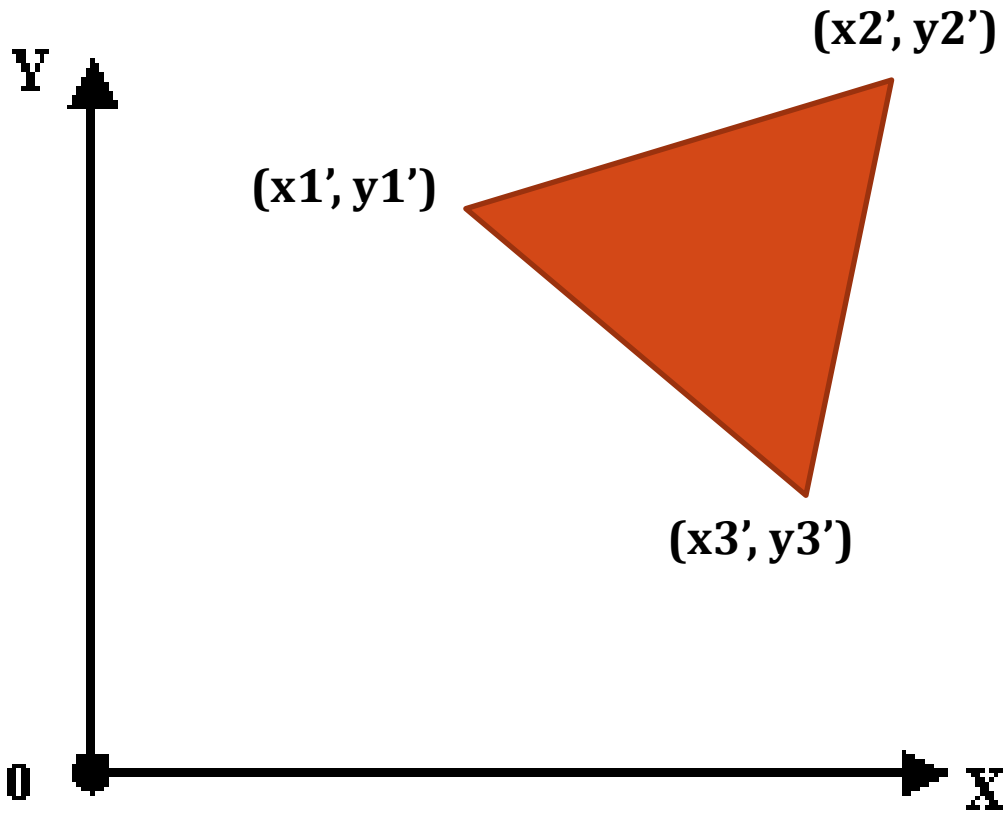
TRANSFORMATION

Scale :



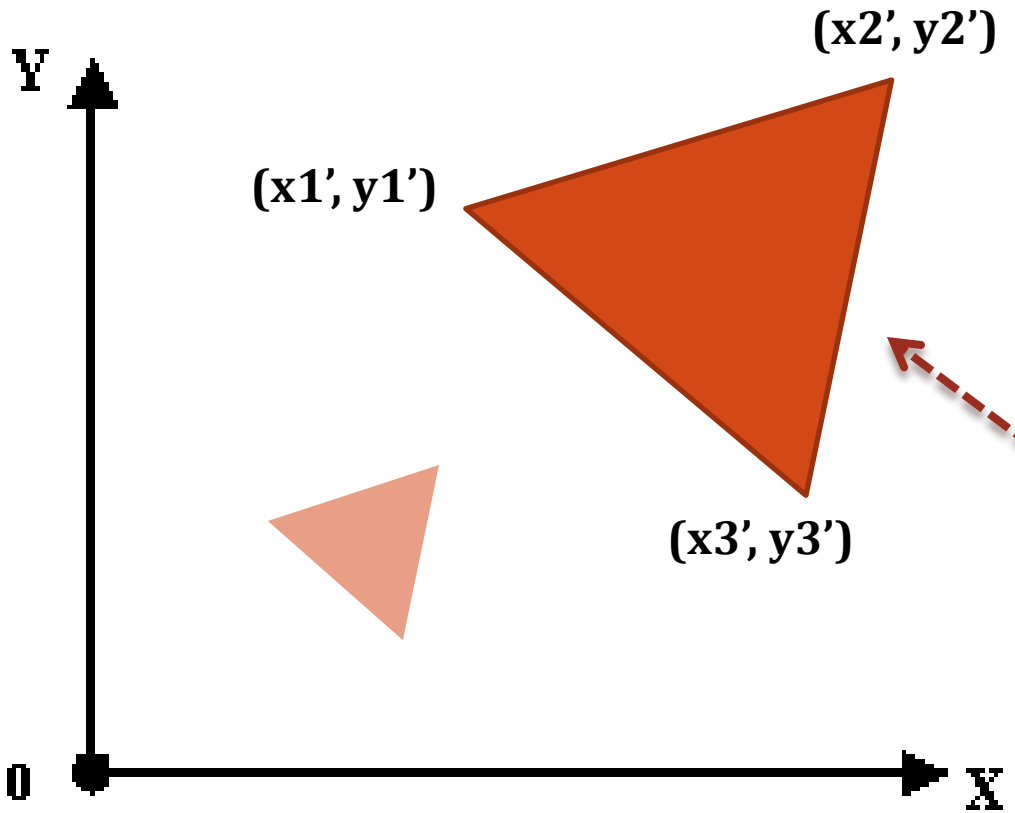
TRANSFORMATION

Scale :



TRANSFORMATION

Scale :



Suppose, we've scaled it by 130% with respect to origin–

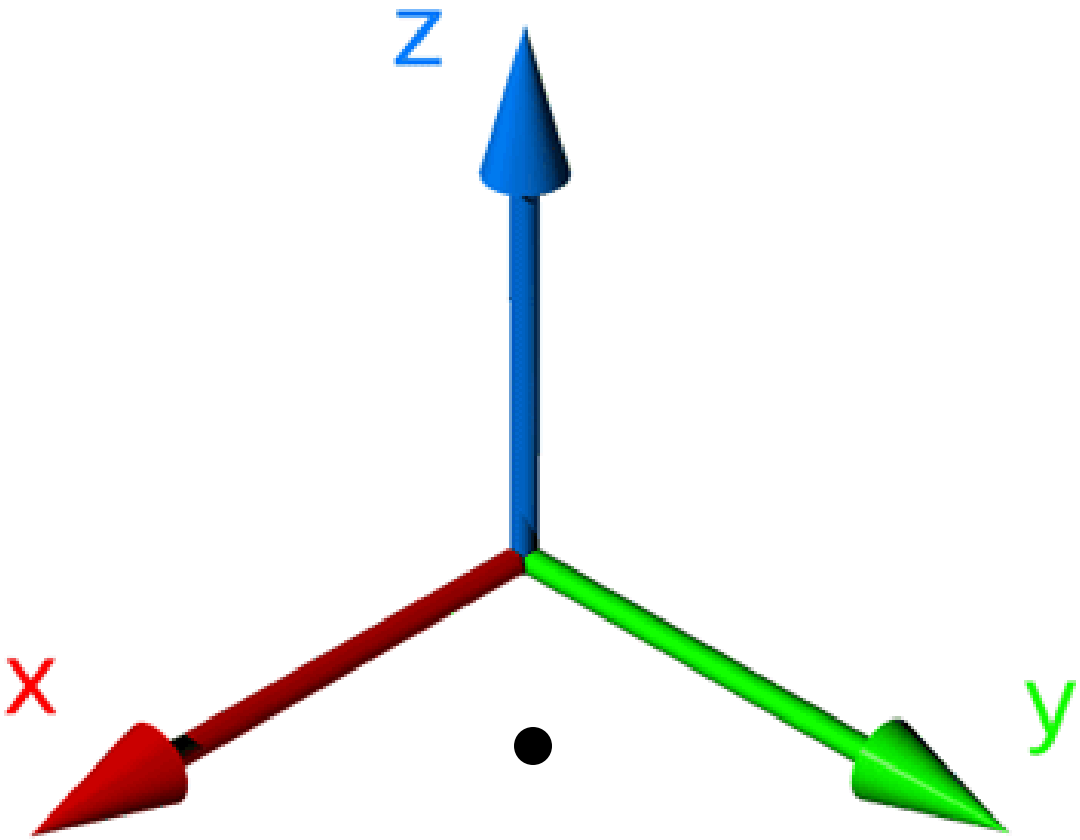
x1 * 1.3 = x1'

y1 * 1.3 = y1'

……..

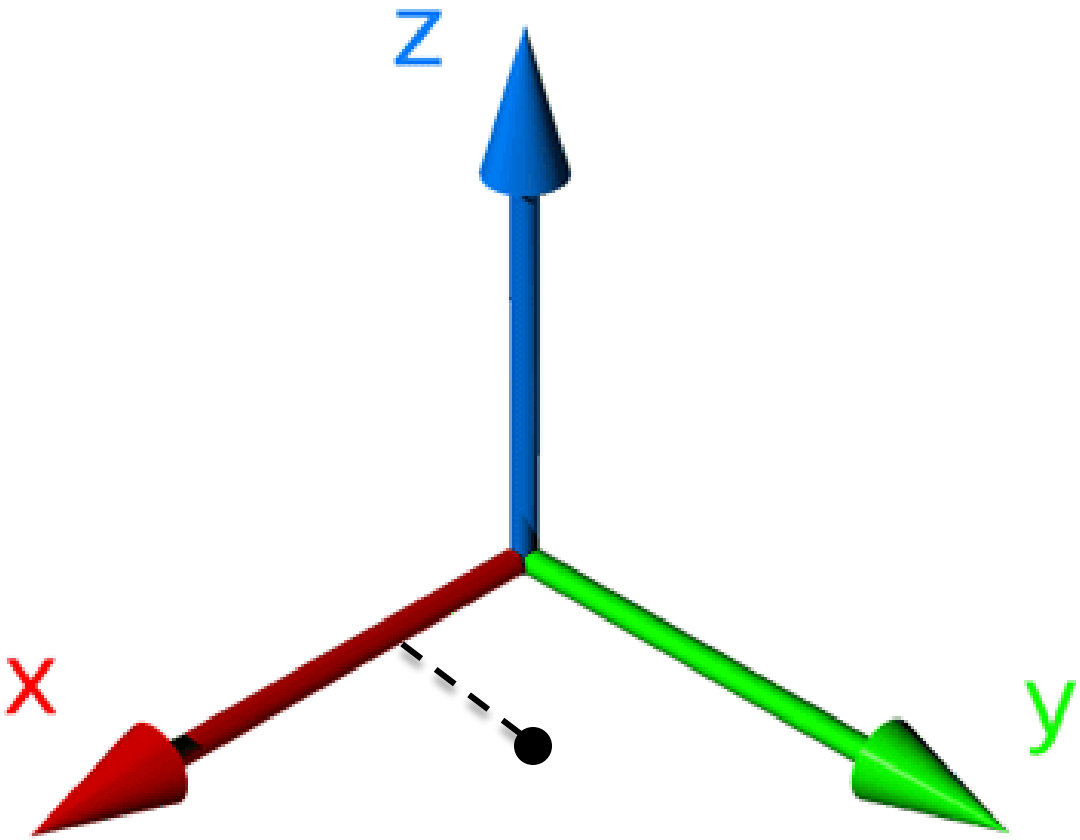
TRANSFORMATION

Rotate :



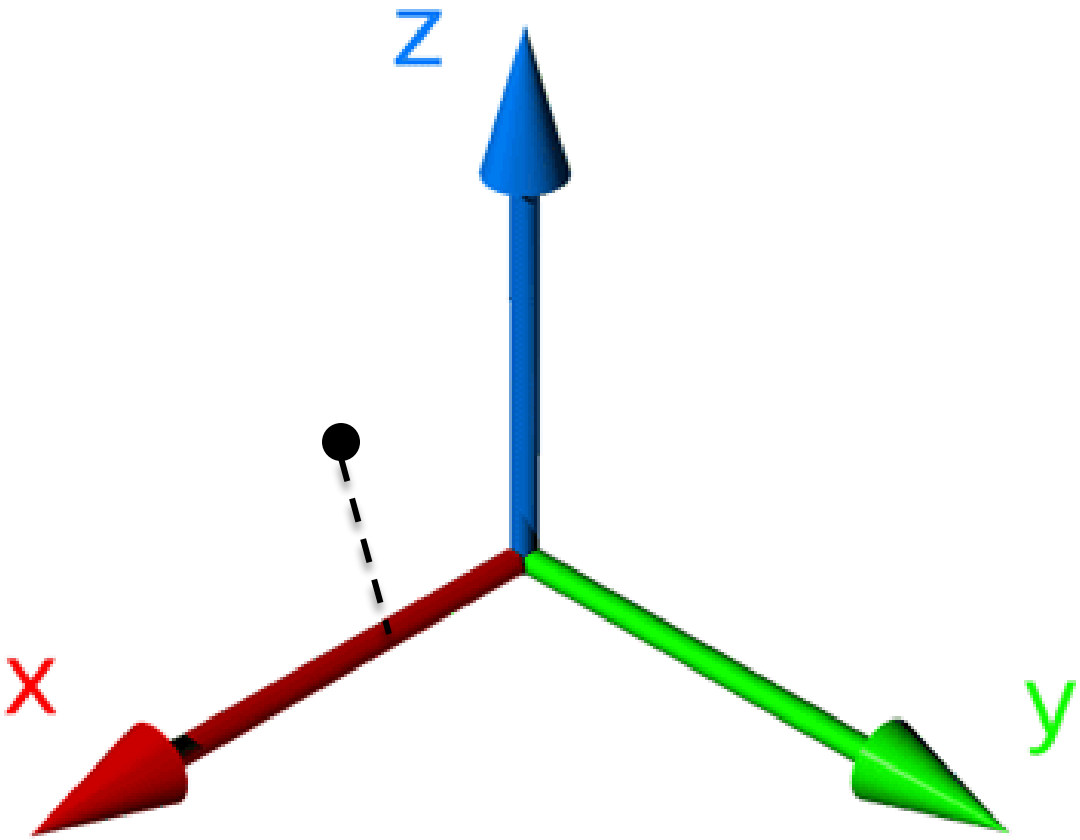
TRANSFORMATION

Rotate :



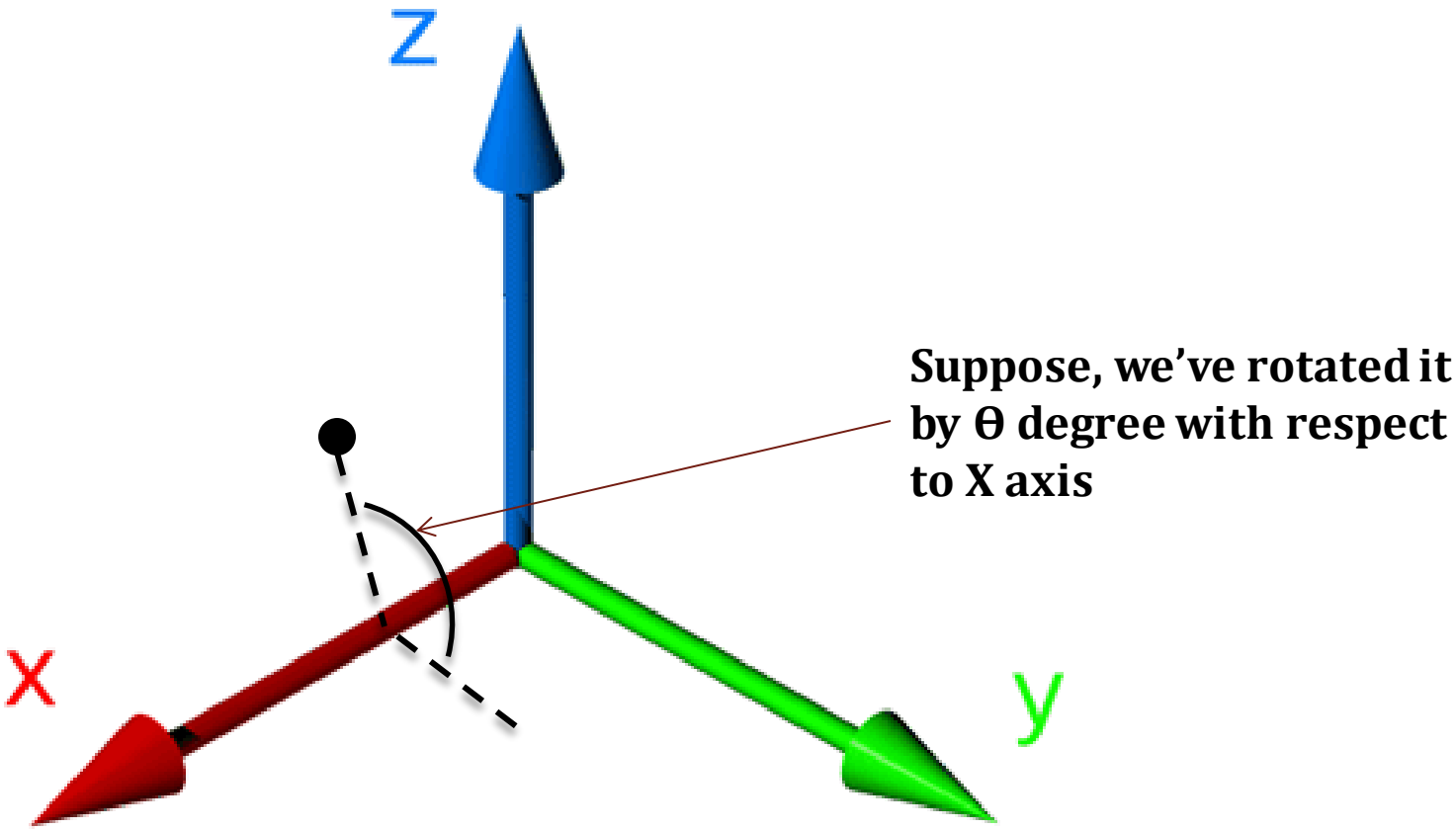
TRANSFORMATION

Rotate :



TRANSFORMATION

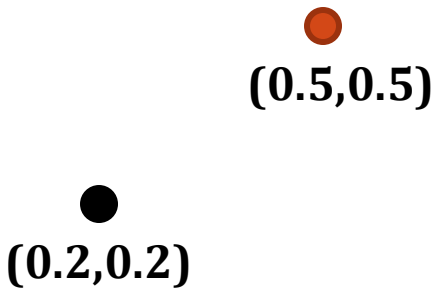
Rotate :



glPushMatrix()
glPopMatrix()

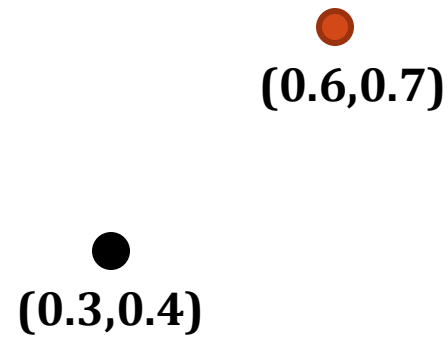
Example - 1

```
void Draw()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glPointSize(16);

        glColor3f(1.0, 1.0, 1.0);


                glBegin(GL_POINTS);
                        glVertex3f(0.2, 0.2, 0.0);
                glEnd();


        glColor3f(1.0, 0.0, 0.0);
                glBegin(GL_POINTS);
                        glVertex3f(0.5, 0.5, 0.0);
                glEnd();
        glFlush();
}
```



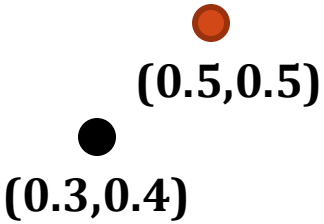
Example - 1

```
void Draw()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glPointSize(16);

        glColor3f(1.0, 1.0, 1.0);

                glTranslatef(0.1,0.2,0.0);
                        glBegin(GL_POINTS);
                                glVertex3f(0.2, 0.2, 0.0);
                        glEnd();


        glColor3f(1.0, 0.0, 0.0);
                        glBegin(GL_POINTS);
                                glVertex3f(0.5, 0.5, 0.0);
                        glEnd();
        glFlush();
}
```



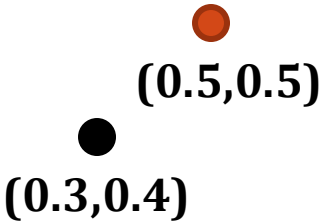
Example - 1

```
void Draw()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(16);

    glColor3f(1.0, 1.0, 1.0);
    glPushMatrix();
        glTranslatef(0.1,0.2,0.0);
            glBegin(GL_POINTS);
                glVertex3f(0.2, 0.2, 0.0);
            glEnd();
    glPopMatrix();

    glColor3f(1.0, 0.0, 0.0);
            glBegin(GL_POINTS);
                glVertex3f(0.5, 0.5, 0.0);
            glEnd();
    glFlush();
}
```



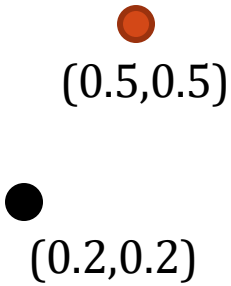
Example - 1

```
void Draw()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glPointSize(16);

    glColor3f(1.0, 1.0, 1.0);
    glPushMatrix();
        glTranslatef(0.1,0.2,0.0);
            glBegin(GL_POINTS);
                glVertex3f(0.2, 0.2, 0.0);
            glEnd();
    glPopMatrix();

    glColor3f(1.0, 0.0, 0.0);
            glBegin(GL_POINTS);
                glVertex3f(0.5, 0.5, 0.0);
            glEnd();
    glFlush();
}
```



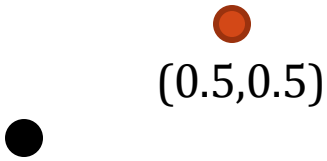
Example - 2

```
void Draw()
{       glClear(GL_COLOR_BUFFER_BIT);
        glPointSize(16);
        glPushMatrix();
        glTranslatef(-0.1,0.2,0.0);

                glColor3f(1.0, 1.0, 1.0);
                        glPushMatrix();
                        glRotatef(45, 0.0, 0.0, 1.0);
                                        glBegin(GL_POINTS);
                                                glVertex3f(0.2, 0.2, 0.0);
                                        glEnd();
                        glPopMatrix();
                                        glBegin(GL_POINTS);
                                                glVertex3f(0.5, 0.5, 0.0);
                                        glEnd();
                        glFlush();

        glPopMatrix();}
```



Example - 2

```
void Draw()
{       glClear(GL_COLOR_BUFFER_BIT);
        glPointSize(16);
        glPushMatrix();
        glTranslatef(-0.1,0.2,0.0);

                glColor3f(1.0, 1.0, 1.0);
                glPushMatrix();
                glRotatef(45, 0.0, 0.0, 1.0);
                        glBegin(GL_POINTS);
                                glVertex3f(0.2, 0.2, 0.0);
                        glEnd();
                glPopMatrix();
                        glBegin(GL_POINTS);
                                glVertex3f(0.5, 0.5, 0.0);
                        glEnd();
                glFlush();

        glPopMatrix();}
```



Example - 2

```
void Draw()
{       glClear(GL_COLOR_BUFFER_BIT);
        glPointSize(16);
        glPushMatrix();
        glTranslatef(-0.1,0.2,0.0);

                glColor3f(1.0, 1.0, 1.0);
                        glPushMatrix();
                        glRotatef(45, 0.0, 0.0, 1.0);
                                        glBegin(GL_POINTS);
                                                glVertex3f(0.2, 0.2, 0.0);
                                        glEnd();
                        glPopMatrix();
                                        glBegin(GL_POINTS);
                                                glVertex3f(0.5, 0.5, 0.0);
                                        glEnd();
                        glFlush();

        glPopMatrix(); }
```