Computer Graphics | CSE 404

Texture Mapping

Mohammad Imrul Jubair

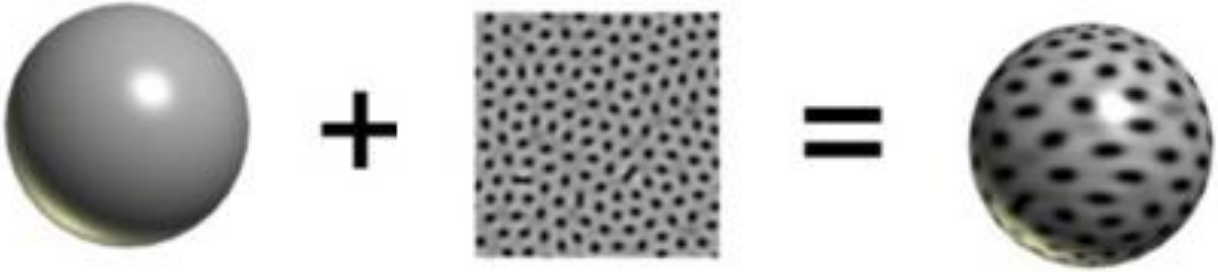
What is Texture ?

Texture:
The feel, appearance, or consistency of a surface or a substance.

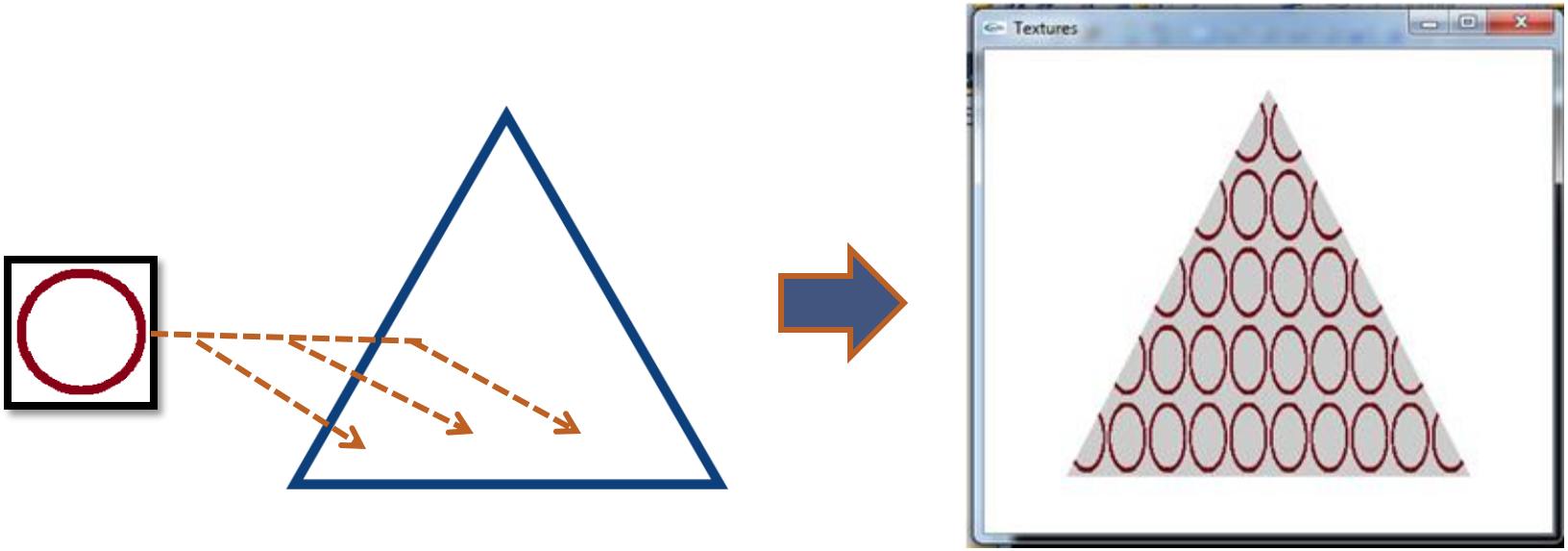
…… The natural world is rich in texture: the surface of any visible object is textured at certain scale

What is Texture Mapping in OpenGL?

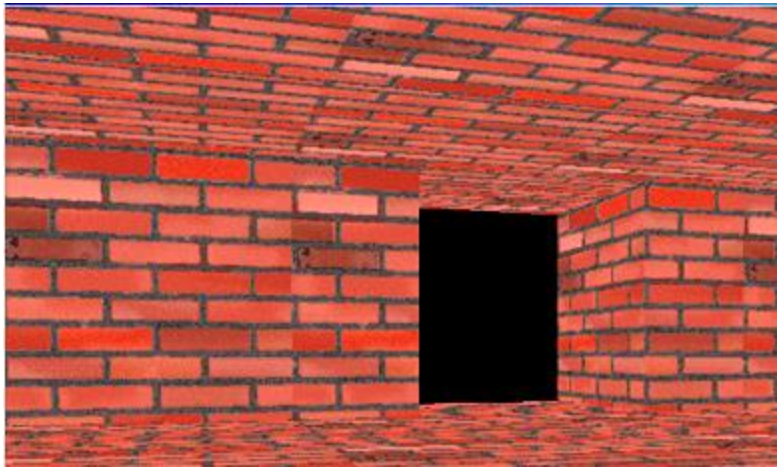
Texture Mapping → Texture Wrapping

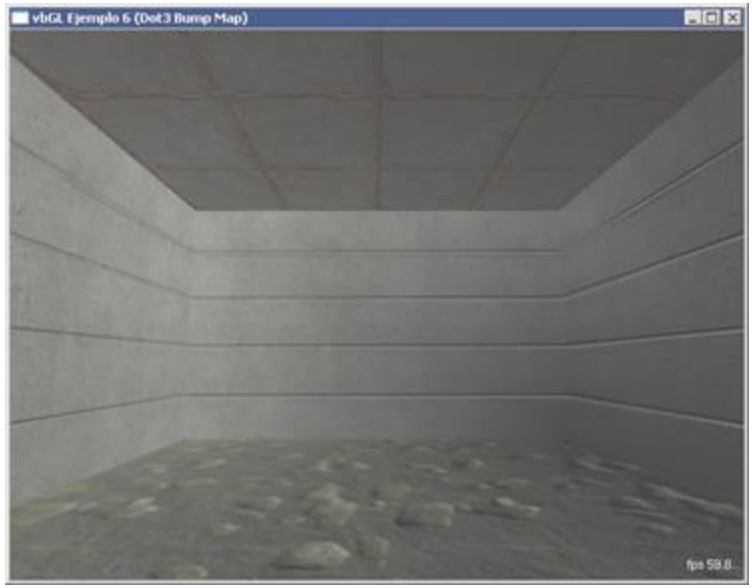


An object + **An image** *(jpg, bmp, etc.)* = **Texture Mapped**

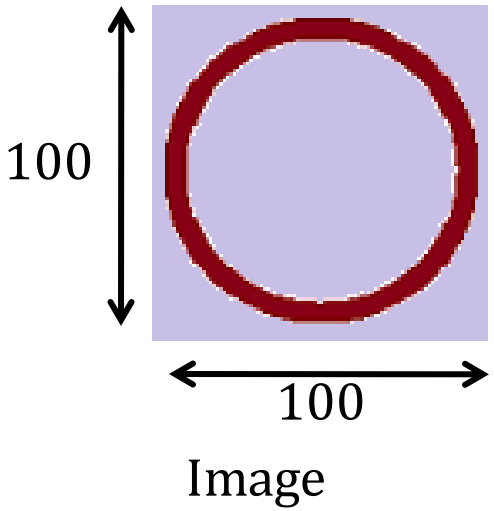


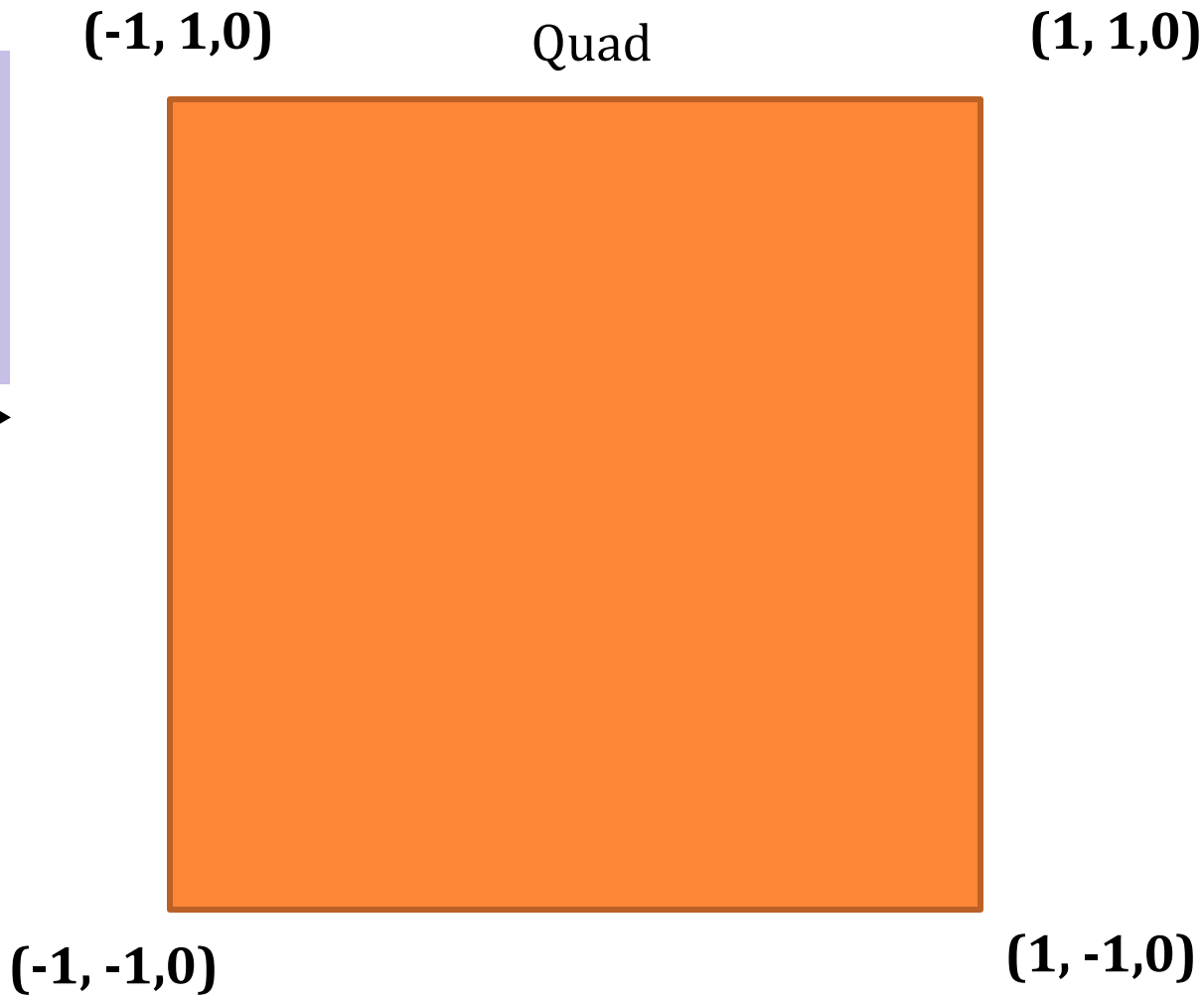
What is Texture Mapping in OpenGL?



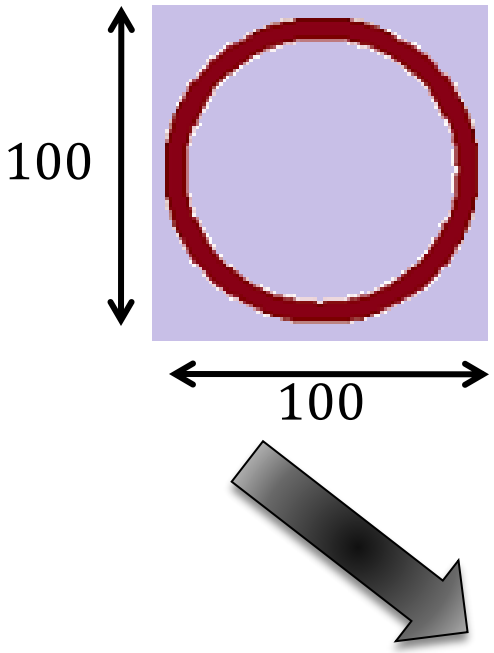


Example

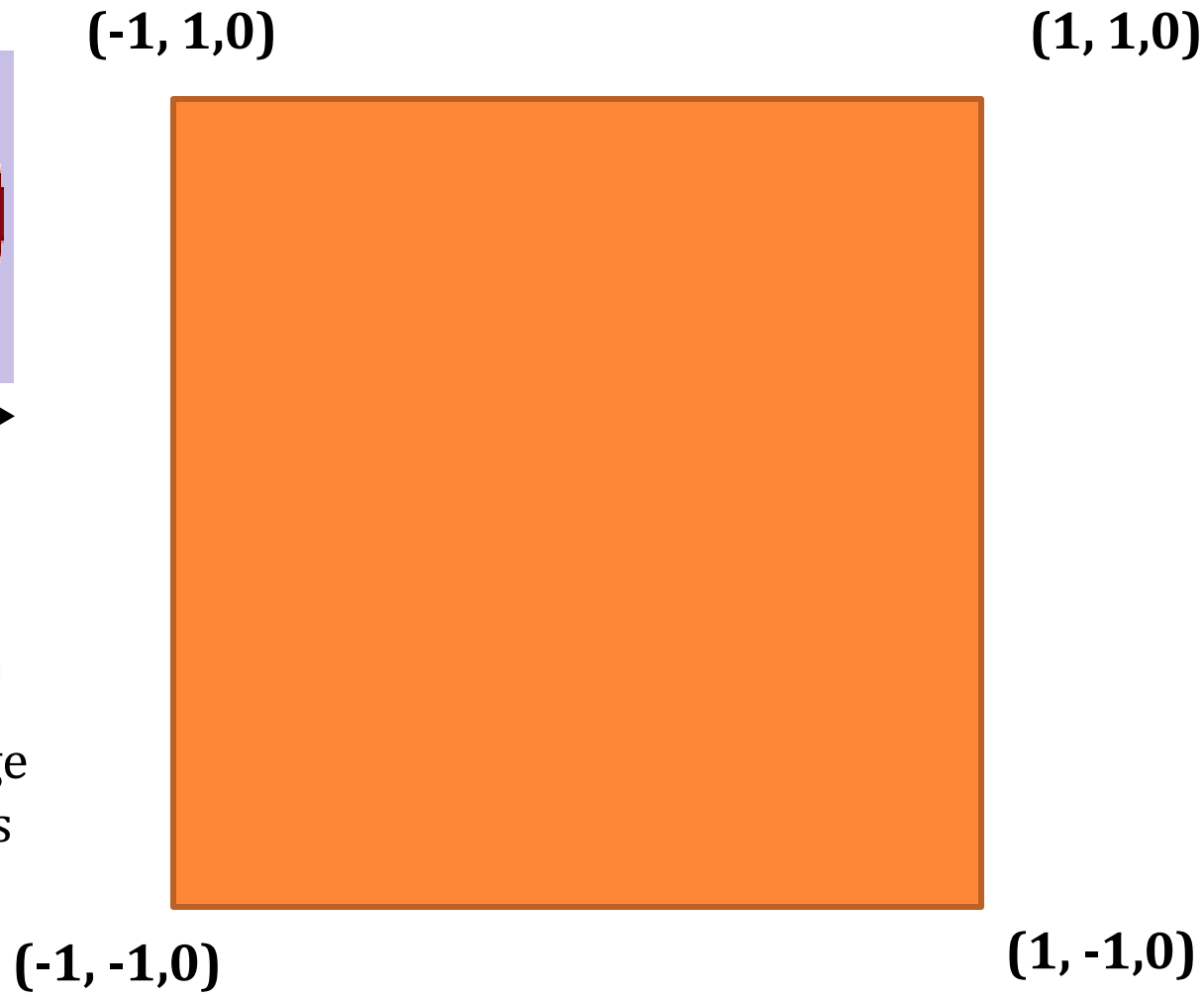




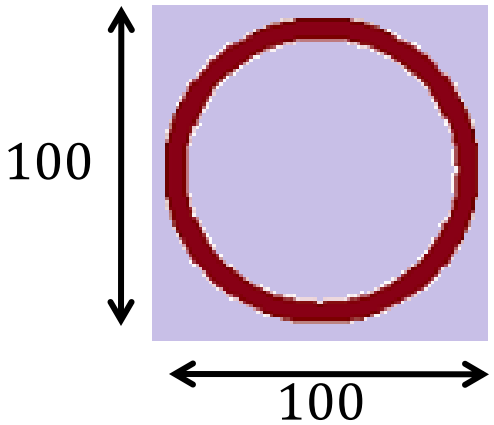
Example

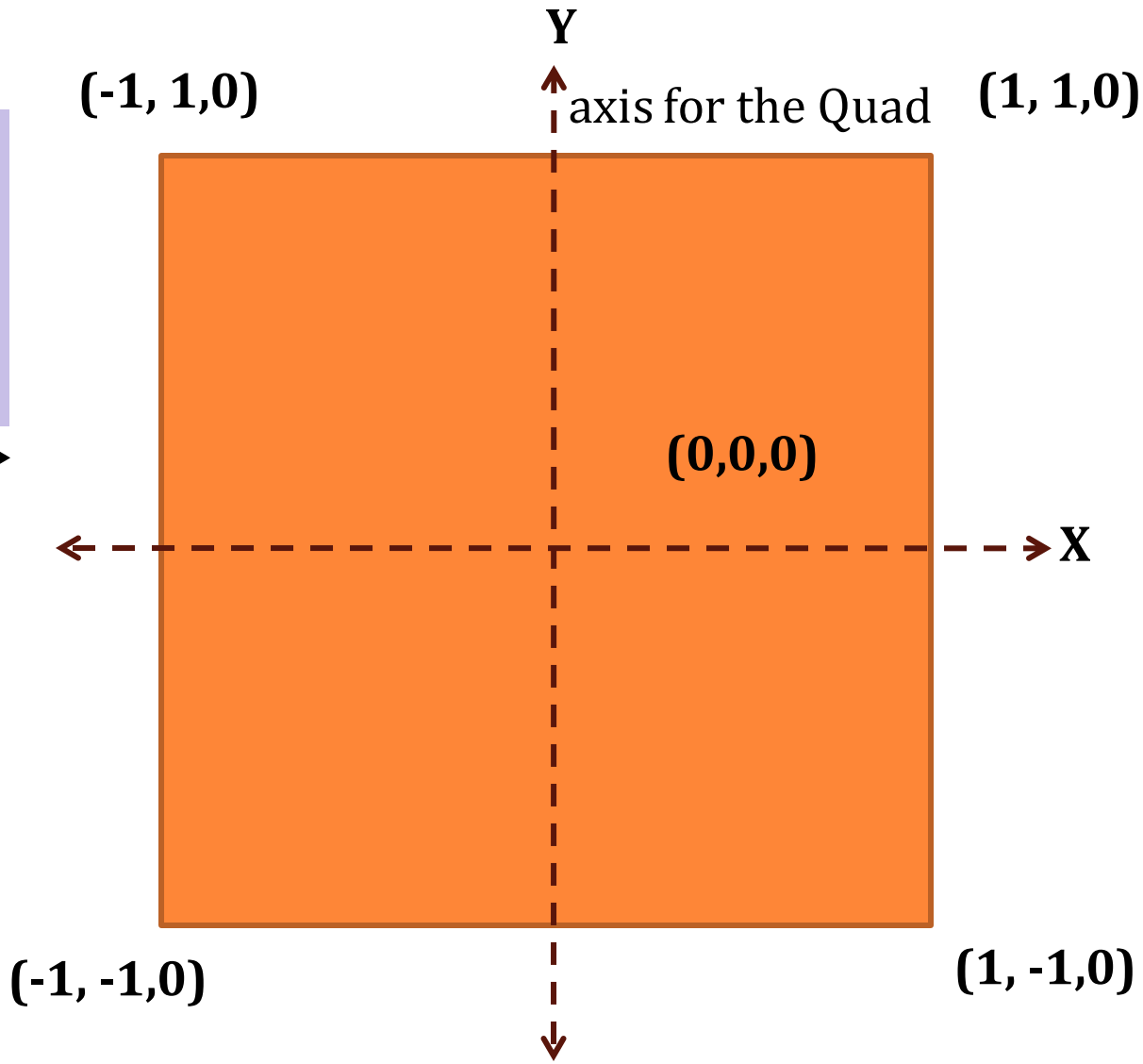


we can put this image block as tiles on this Quad one by one

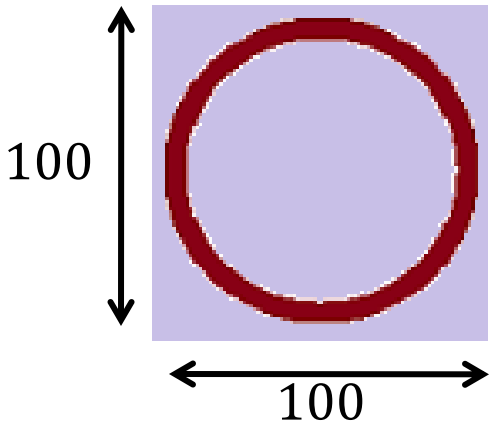


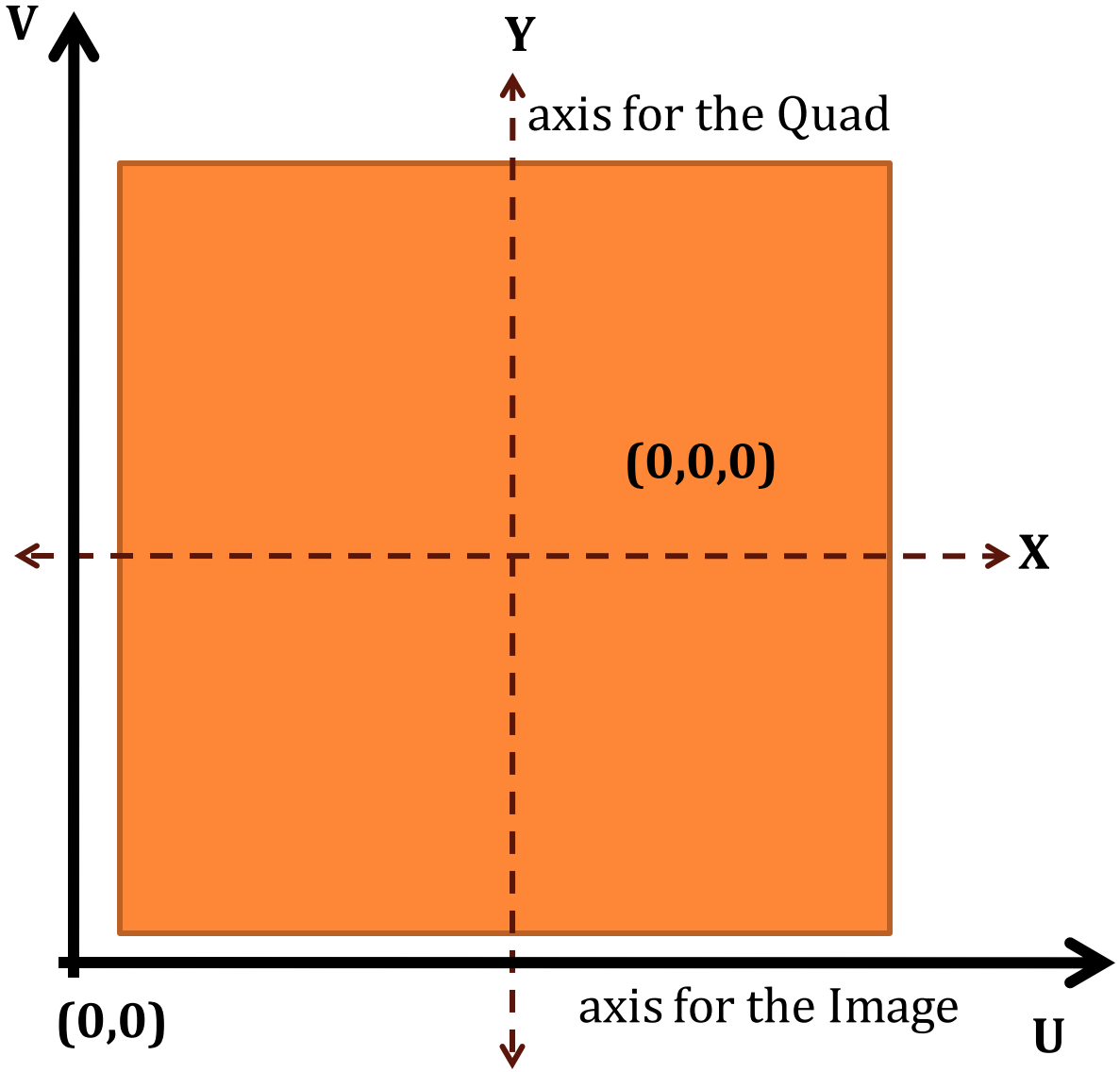
Example



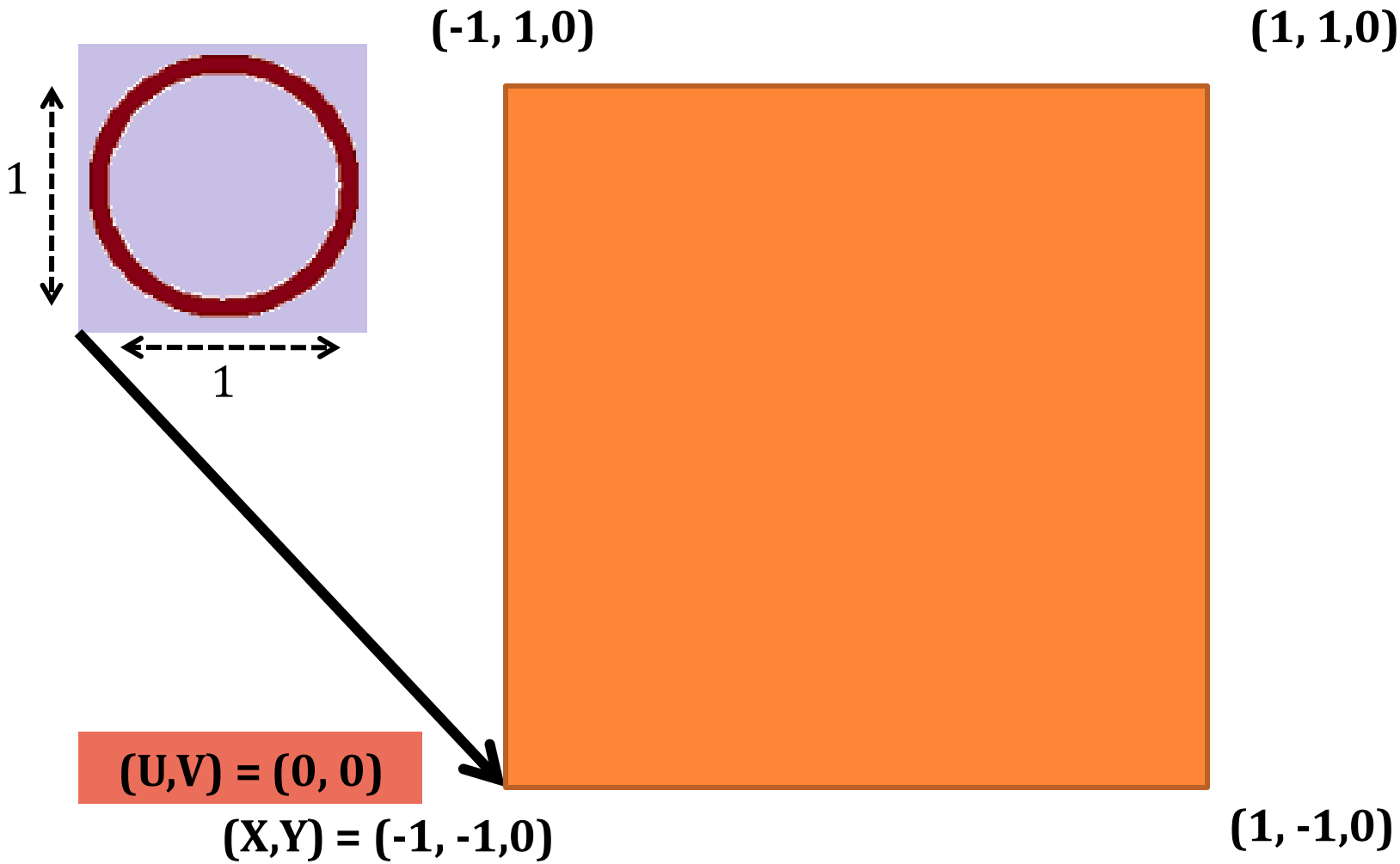


Example

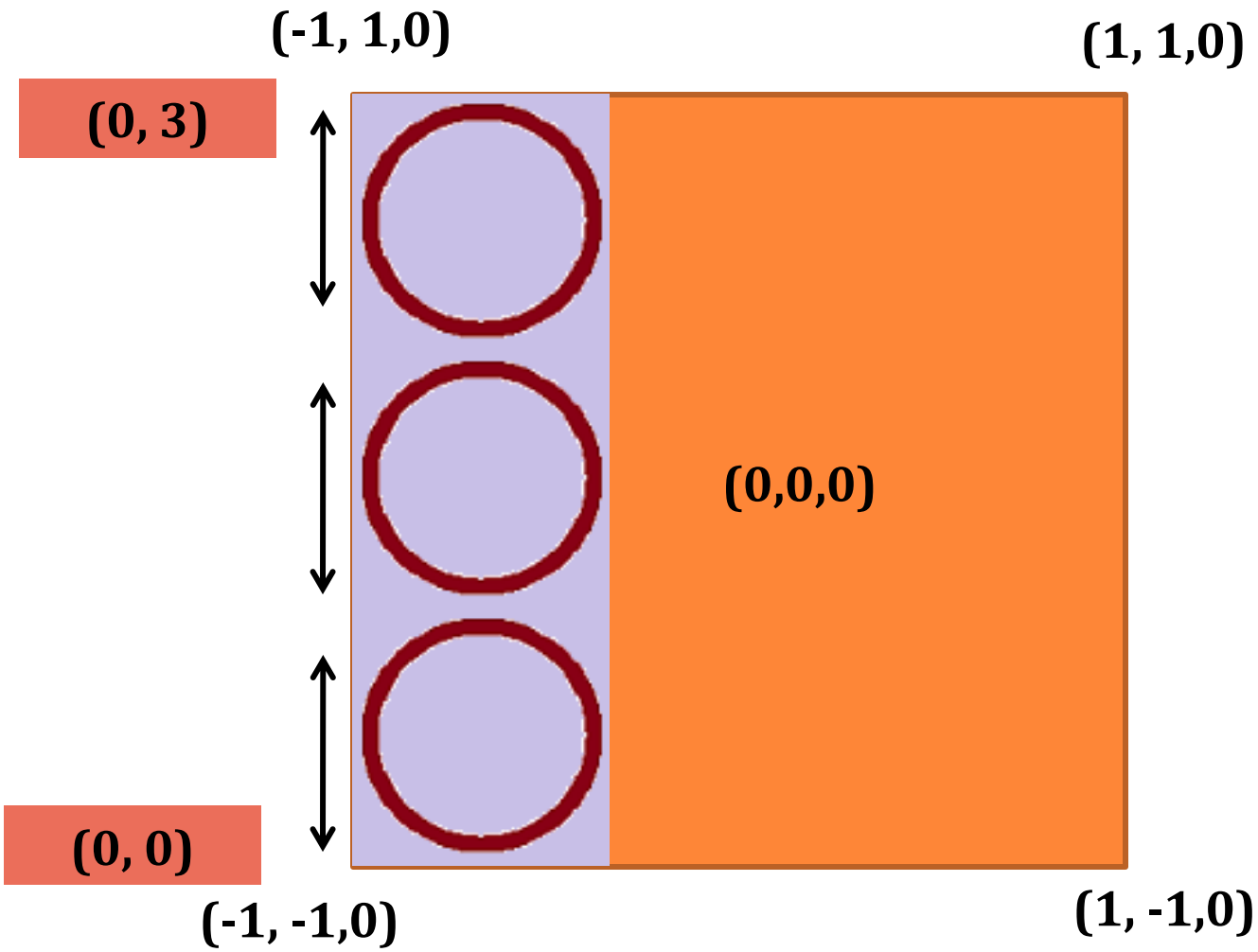




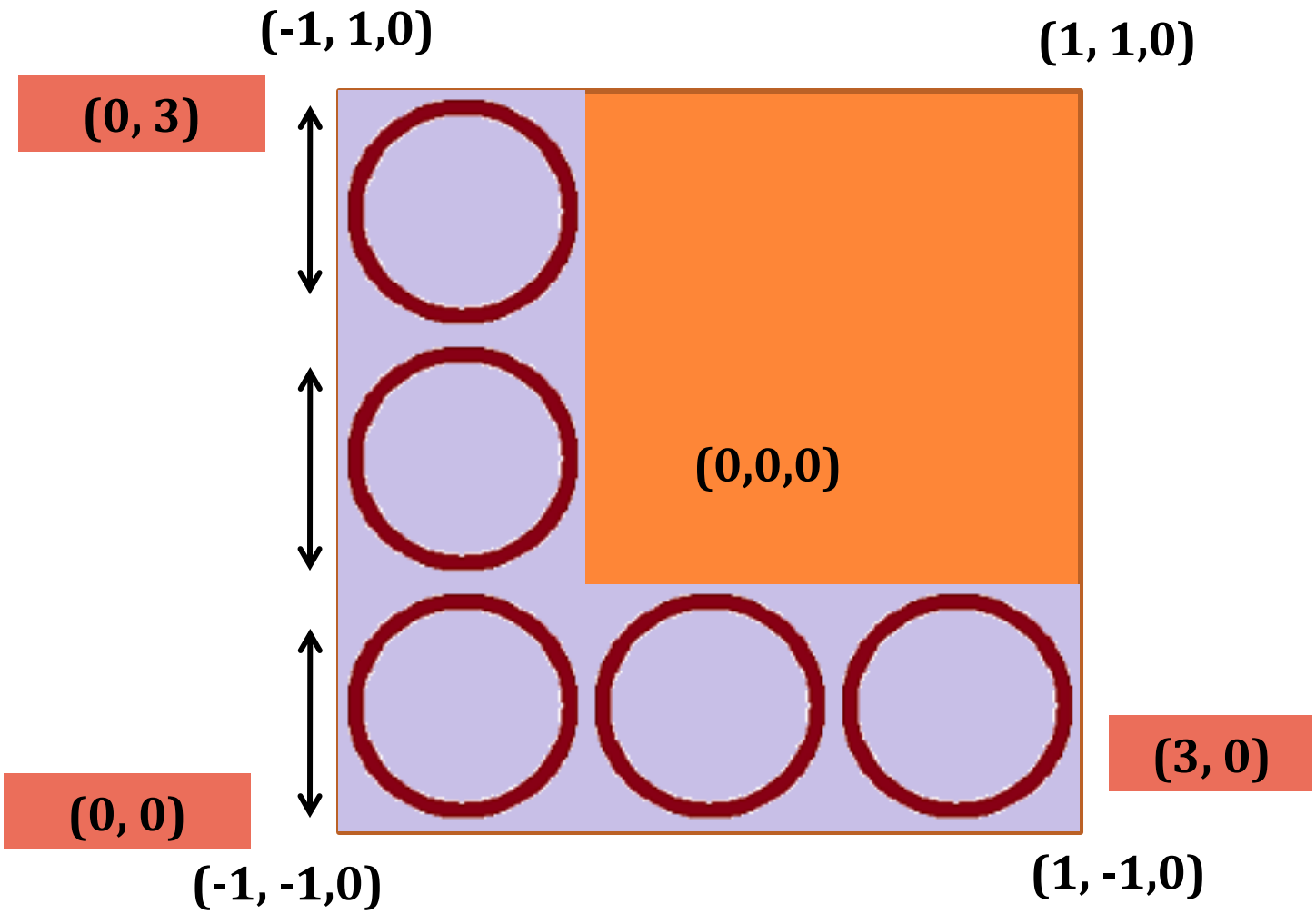
Example



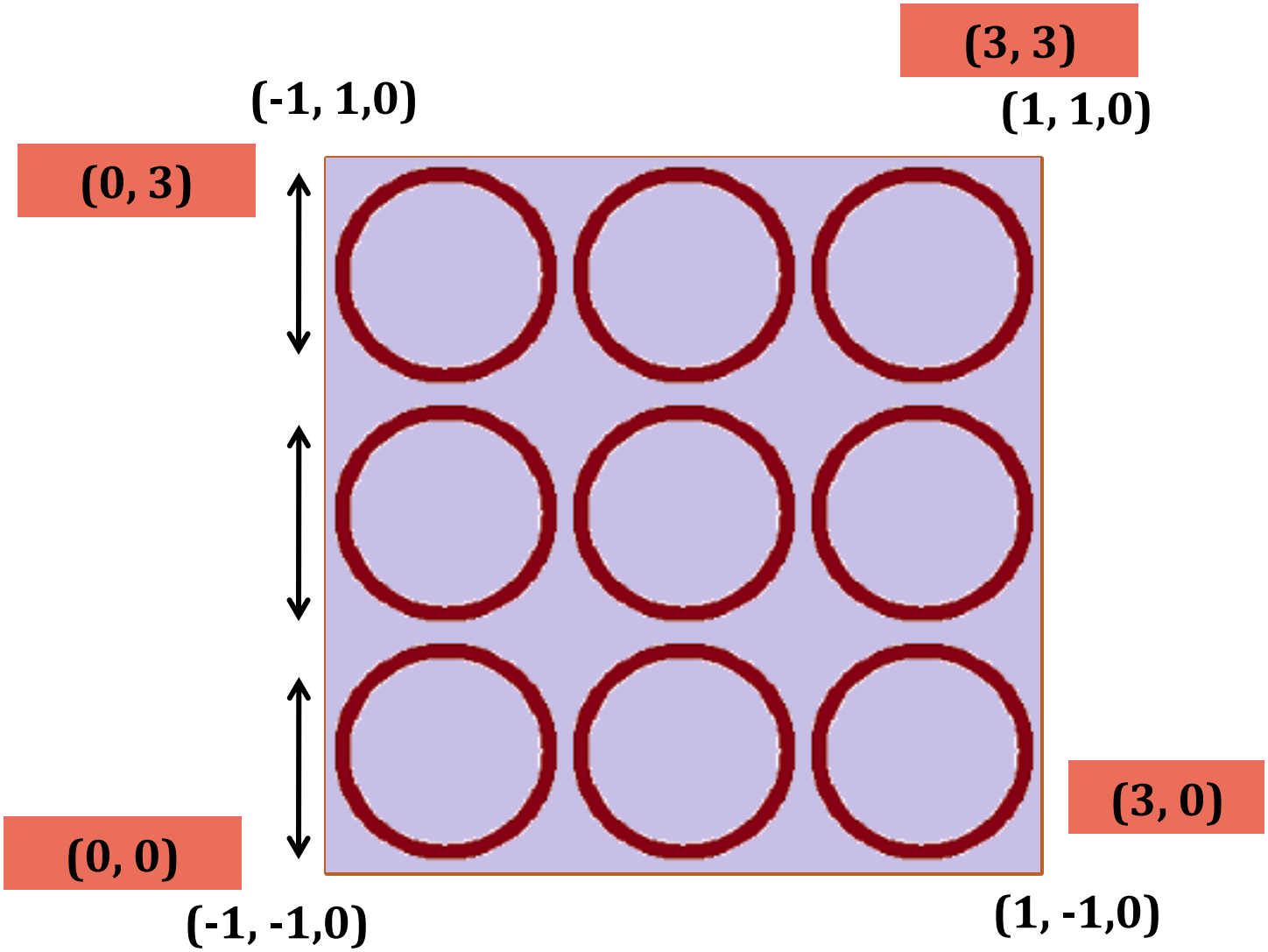
Example



What is Texture Mapping in OpenGL?



What is Texture Mapping in OpenGL?

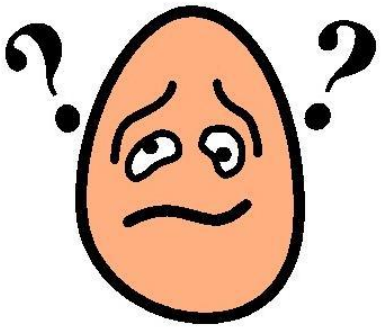


Steps In Texture Mapping.

Step 1 : a) Loading Image
b) Image → OpenGL Texture *(To make the image ready for wrapping an object)*

Step 2: Mapping that ready texture on the object *(Wrapping)*

CODE ! ☹



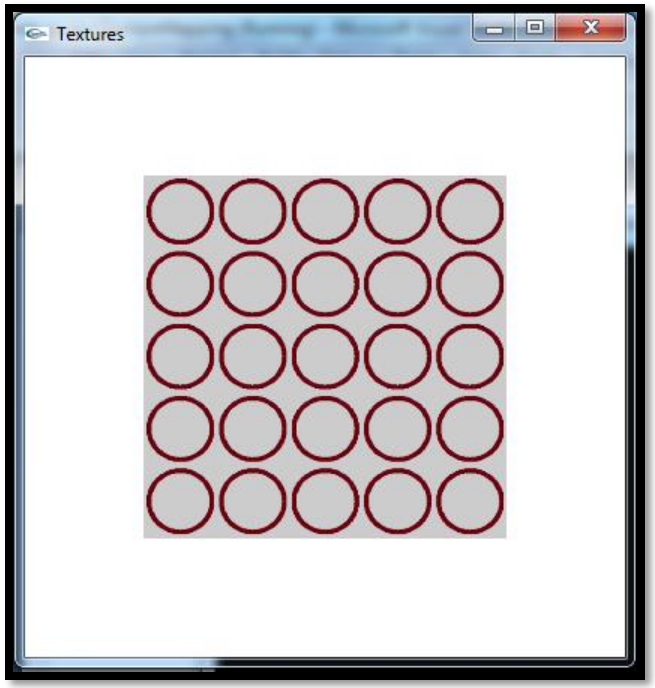
CODES ! CODES!! CODES !!!
Exhausted?

Why So Serious ??? ☺

We don't have to
know **how** it works;
all we have to know
is **what** it does

Lets Know - What It Does !

```
void drawScene() { ... }
GLuint loadTexture(Image* image) { ... }
void initialize() { ... }
void lightSetting() { ... }
int main(int argc, char** argv) { ... }
```



Step 1 : Image → OpenGL Texture

```
void initialize() {

    glClearColor(1.0, 1.0, 1.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    gluPerspective(45.0, 1.00, 1.0, 200.0);
    Image* image = loadBMP("F:\\texture.bmp");
    _textureId = loadTexture(image);
    delete image;
}
```

texture.bmp

Step 1 : Image → OpenGL Texture

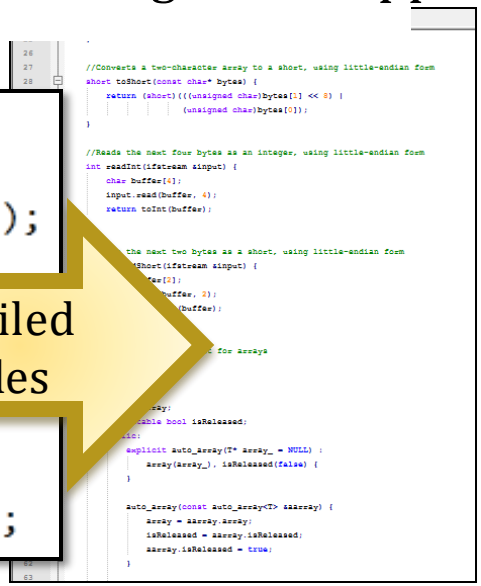
```
void initialize() {

    glClearColor(1.0, 1.0, 1.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    gluPerspective(45.0, 1.00, 1.0, 200.0);
    Image* image = loadBMP("F:\\texture.bmp");
    _textureId = loadTexture(image);
    delete image;
}
```

imageloader.h

```
class Image {
    public:
        Image(char* ps, int w, int h);
        ~Image();
        char* pixels;
        int width;
        int height;
};
Image* loadBMP(const char* filename);
```

Detailed Codes

imageloader.cpp



Step 1 : Image → OpenGL Texture

```
#include <iostream>
#include <stdlib.h>
#include <glut.h>
#include "imageloader.h"
```

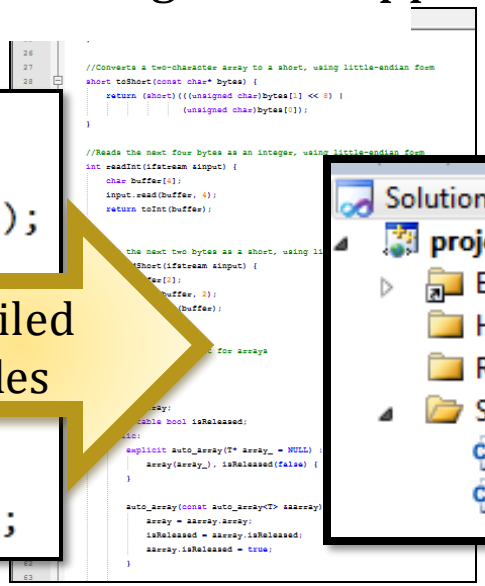
```
void initialize() {

    glClearColor(1.0, 1.0, 1.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    gluPerspective(45.0, 1.00, 1.0, 200.0);
    Image* image = loadBMP("F:\\texture.bmp");
    _textureId = loadTexture(image);
    delete image;
}
```

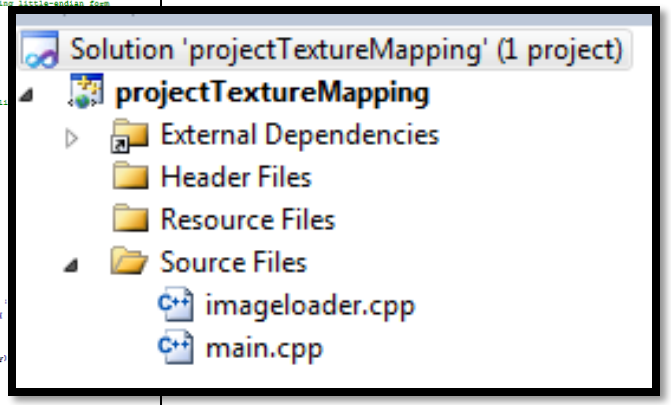
imageloader.h

```
class Image {
    public:
        Image(char* ps, int w, int h);
        ~Image();
        char* pixels;
        int width;
        int height;
};
Image* loadBMP(const char* filename);
```

Detailed Codes

imageloader.cpp





Step 1 : Image → OpenGL Texture

```
GLuint _textureId;
void initialize() {

    glClearColor(1.0, 1.0, 1.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    gluPerspective(45.0, 1.00, 1.0, 200.0);
    Image* image = loadBMP("F:\\texture.bmp");
    _textureId = loadTexture(image);
    delete image;
}
```

```
void drawScene() { ... }
GLuint loadTexture(Image* image) { ... }
void initialize() { ... }
void lightSetting() { ... }
int main(int argc, char** argv) { ... }
```

Step 1 : Image → OpenGL Texture

```
void initialize() {

    glClearColor(1.0, 1.0, 1.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    gluPerspective(45.0, 1.00, 1.0, 200.0);
    Image* image = loadBMP("F:\\texture.bmp");  1
    _textureId = loadTexture(image);  2
    delete image;  3
}
```

Now our Image is reading for wrapping an object!

What we have done here is actually -

1. We load the image
2. load the texture into OpenGL
3. then delete the Image object, since we don't need it any more.

Step 2 : Mapping the Texture on a Plane

```
+void drawScene() { ... }
+GLuint loadTexture(Image* image) { ... }
+void initialize() { ... }
+void lightSetting() { ... }
+int main(int argc, char** argv) { ... }
```

Step 2 : Mapping the Texture on a Plane

```
void drawScene()
{
…….
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, _textureId);
……



}
```

Step 2 : Mapping the Texture on a Plane

```
void drawScene()
{
…….
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, _textureId);
……



}
```

Telling the computer that we're going to use Texture Mapping

Step 2 : Mapping the Texture on a Plane

```
void drawScene()
{
…….
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, _textureId);
……


}
```

Which Texture is going to be used

Step 2 : Mapping the Texture on a Plane

```
void drawScene()
{
.......
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, _textureId);
......


}
```

Which Texture is going to be used

```
void initialize() {

    glClearColor(1.0, 1.0, 1.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    gluPerspective(45.0, 1.00, 1.0, 200.0);
    Image* image = loadBMP("F:\\texture.bmp");
    _textureId = loadTexture(image);
    delete image;
}
```

Step 2 : Mapping the Texture on a Plane

```
void drawScene()
{
…….
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, _textureId);
……

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);


}
```

Setting up the type of mapping (will be explained later……)

Step 2 : Mapping the Texture on a Plane

```
void
{
.......
glEna
glBin                                   textureId);
......

glTex                                   , GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTex                                   , GL_TEXTURE_MAG_FILTER, GL_LINEAR);

........
glBegi  L_QUADS);

glEnd();
......

}
```

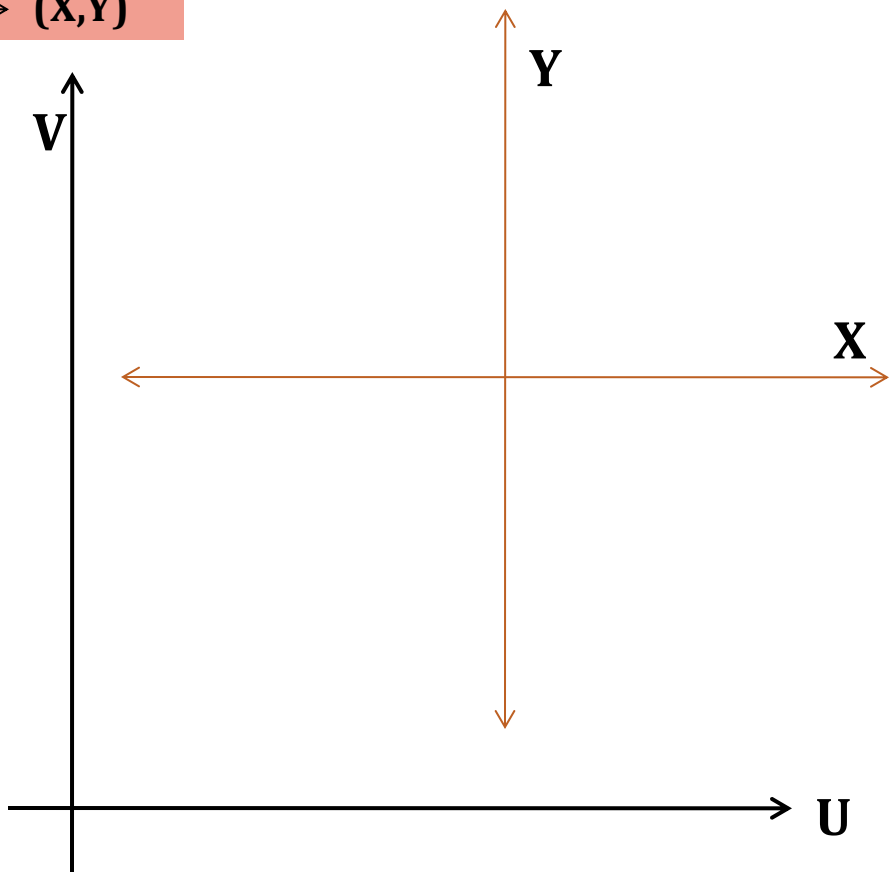
```
glTexCoord2f(0.0, 0.0);
glVertex3f(-1.0, -1.0, 0.0);

glTexCoord2f(0.0, 5.0);
glVertex3f(-1.0, 1.0, 0.0);

glTexCoord2f(5.0, 5.0);
glVertex3f(1.0, 1.0, 0.0);

glTexCoord2f(5.0, 0.0);
glVertex3f(1.0, -1.0, 0.0);
```

Step 2 : Mapping the Texture on a Plane

```
glTexCoord2f(0.0, 0.0);
glVertex3f(-1.0, -1.0, 0.0);

glTexCoord2f(0.0, 5.0);
glVertex3f(-1.0, 1.0, 0.0);

glTexCoord2f(5.0, 5.0);
glVertex3f(1.0, 1.0, 0.0);

glTexCoord2f(5.0, 0.0);
glVertex3f(1.0, -1.0, 0.0);
```

(U, V)

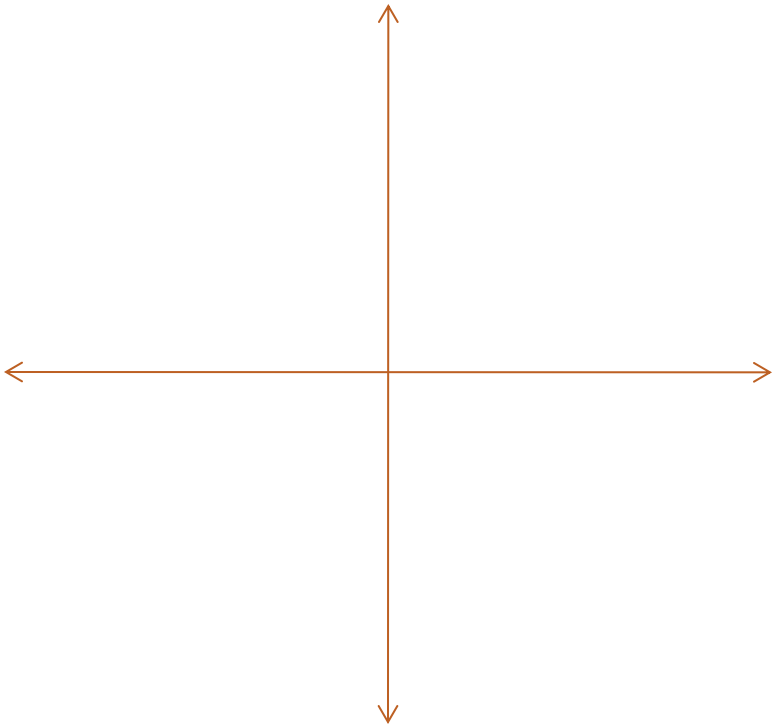
(X,Y)



Step 2 : Mapping the Texture on a Plane

→

```
glTexCoord2f(0.0, 0.0);
glVertex3f(-1.0, -1.0, 0.0);

glTexCoord2f(0.0, 5.0);
glVertex3f(-1.0, 1.0, 0.0);

glTexCoord2f(5.0, 5.0);
glVertex3f(1.0, 1.0, 0.0);

glTexCoord2f(5.0, 0.0);
glVertex3f(1.0, -1.0, 0.0);
```



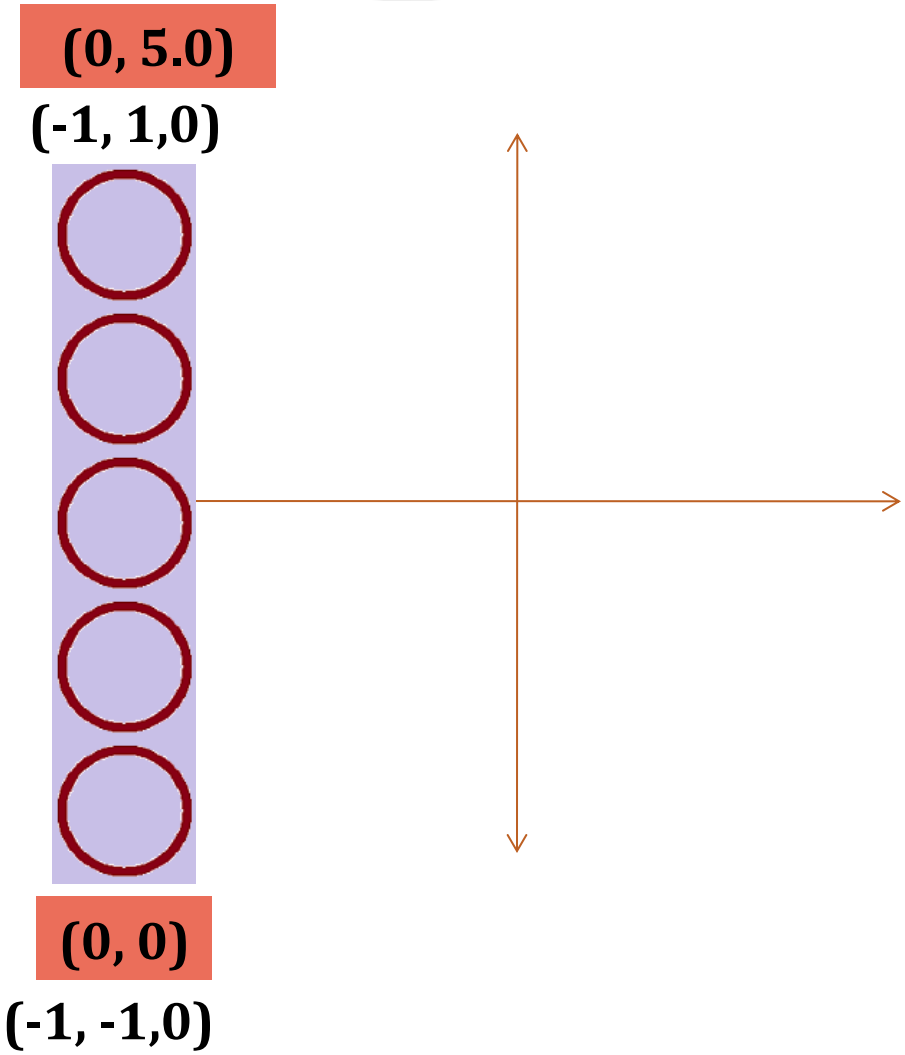
(U,V) :(0, 0)

(X,Y) : (-1, -1,0)

Step 2 : Mapping the Texture on a Plane

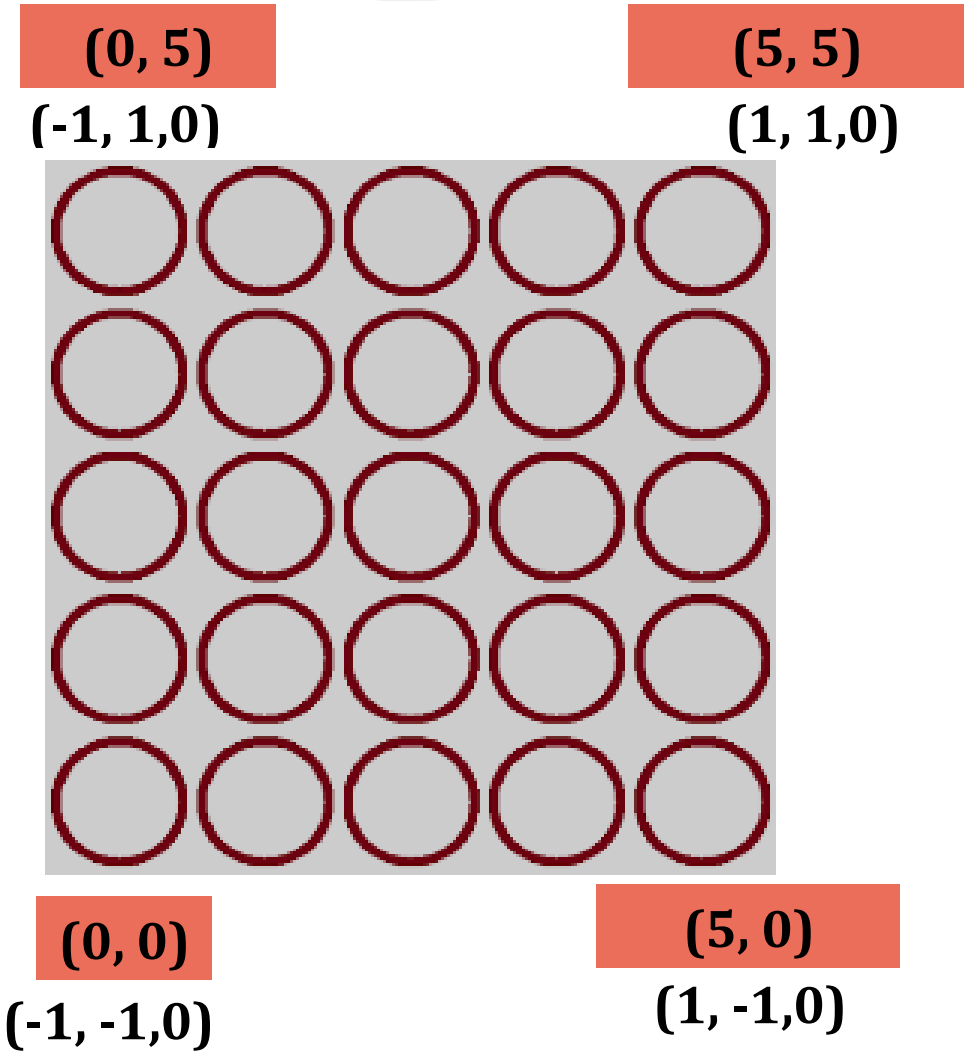
→
```
glTexCoord2f(0.0, 0.0);
glVertex3f(-1.0, -1.0, 0.0);
```

→
```
glTexCoord2f(0.0, 5.0);
glVertex3f(-1.0, 1.0, 0.0);

glTexCoord2f(5.0, 5.0);
glVertex3f(1.0, 1.0, 0.0);

glTexCoord2f(5.0, 0.0);
glVertex3f(1.0, -1.0, 0.0);
```



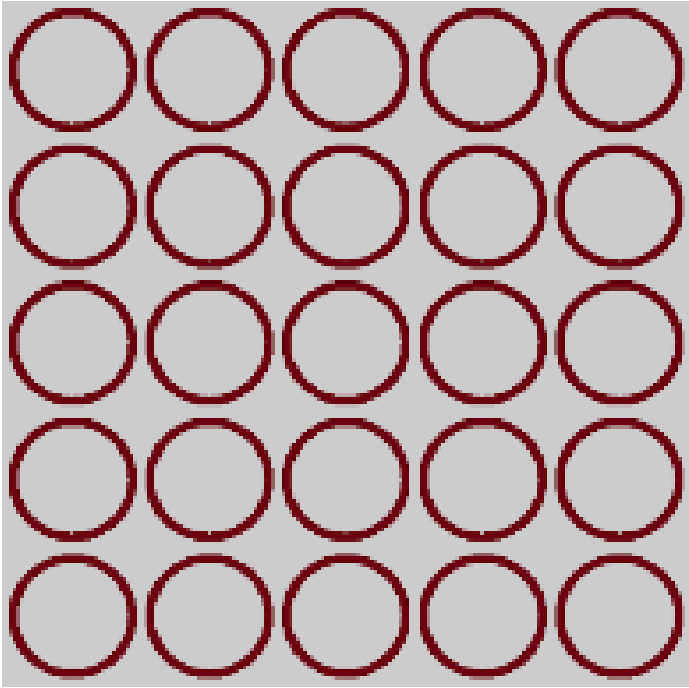
Step 2 : Mapping the Texture on a Plane

```
glTexCoord2f(0.0, 0.0);
glVertex3f(-1.0, -1.0, 0.0);

glTexCoord2f(0.0, 5.0);
glVertex3f(-1.0, 1.0, 0.0);

glTexCoord2f(5.0, 5.0);
glVertex3f(1.0, 1.0, 0.0);

glTexCoord2f(5.0, 0.0);
glVertex3f(1.0, -1.0, 0.0);
```



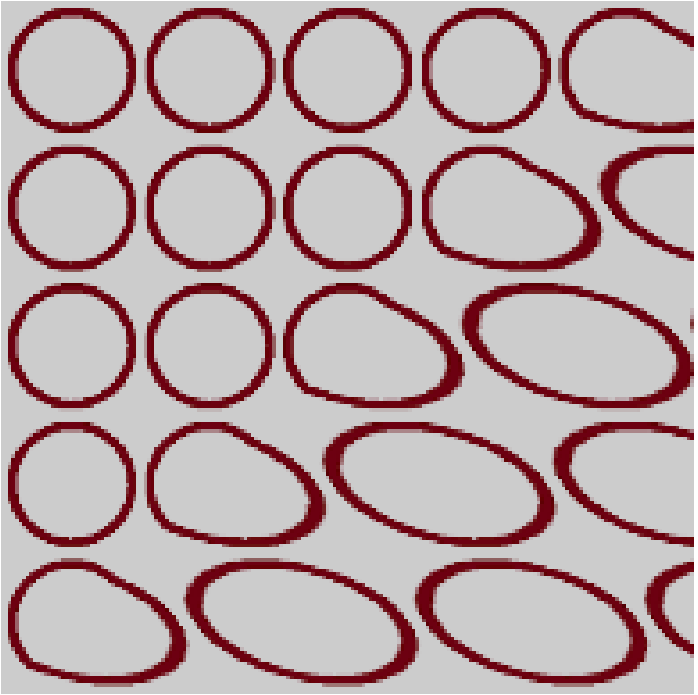
……Calculate your mapping first

```
glBegin(GL_QUADS);

    glTexCoord2f(0.0, 0.0);
    glVertex3f(-1.0, -1.0, 0.0);

    glTexCoord2f(0.0, 5.0);
    glVertex3f(-1.0, 1.0, 0.0);

    glTexCoord2f(5.0, 5.0);
    glVertex3f(1.0, 1.0, 0.0);

    glTexCoord2f(5.0, 0.0);
    glVertex3f(1.0, -1.0, 0.0);

glEnd();
```



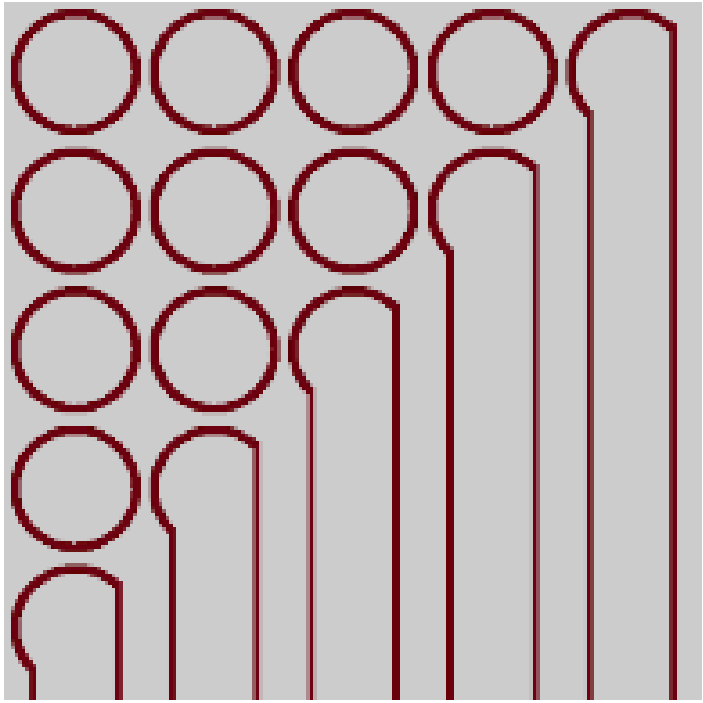
……Calculate your mapping first

```
glBegin(GL_QUADS);

    glTexCoord2f(0.0, 0.0);
    glVertex3f(-1.0, -1.0, 0.0);

    glTexCoord2f(0.0, 5.0);
    glVertex3f(-1.0, 1.0, 0.0);

    glTexCoord2f(5.0, 5.0);
    glVertex3f(1.0, 1.0, 0.0);

    glTexCoord2f(3.0, 0.0);
    glVertex3f(1.0, -1.0, 0.0);

glEnd();
```



……Calculate your mapping first

```
glBegin(GL_QUADS);

    glTexCoord2f(0.0, 0.0);
    glVertex3f(-1.0, -1.0, 0.0);

    glTexCoord2f(0.0, 5.0);
    glVertex3f(-1.0, 1.0, 0.0);

    glTexCoord2f(5.0, 5.0);
    glVertex3f(1.0, 1.0, 0.0);

    //glTexCoord2f(3.0, 0.0);
    glVertex3f(1.0, -1.0, 0.0);

glEnd();
```

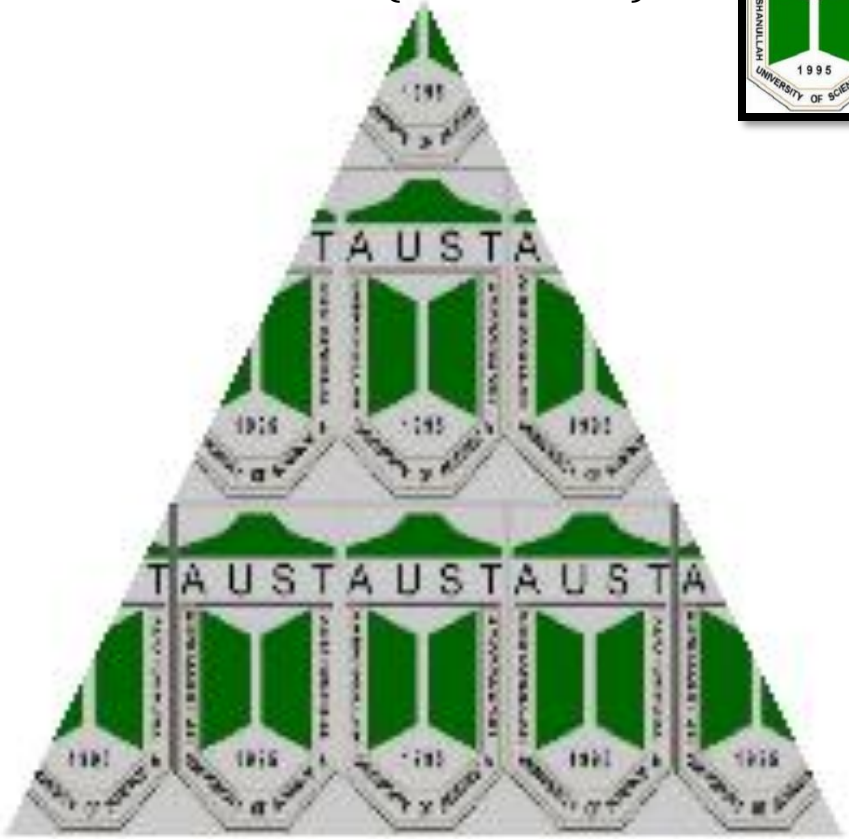


.....More

```
glBegin(GL_TRIANGLES);

    glTexCoord2f(0.0, 0.0);
    glVertex3f(-1.0, -1.0, 0.0);

    glTexCoord2f(2.5, 2.5);
    glVertex3f(0.0, 1.0, 0.0);

    glTexCoord2f(5.0, 0.0);
    glVertex3f(1.0, -1.0, 0.0);

glEnd();
```

U,V = (2.5, 2.5)

X,Y = (0.0, 1.0, 0.0)



U,V = (0.0, 0.0)

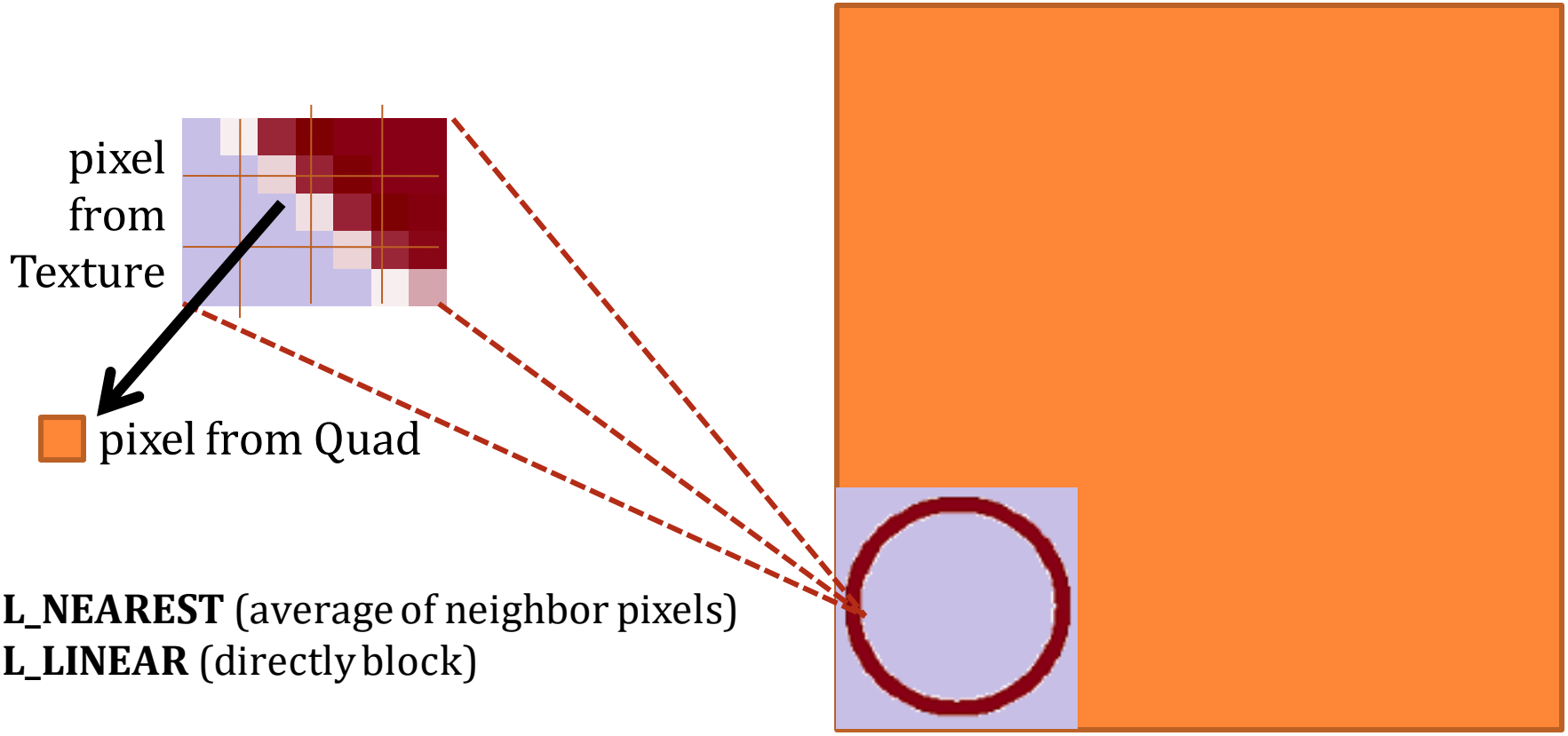
X,Y = (-1.0, -1.0, 0.0)

U,V = (5.0, 0.0)

X,Y = (1.0, -1.0, 0.0)

And At Last

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, _____);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, _____);
```



- **GL_NEAREST** (average of neighbor pixels)
- **GL_LINEAR** (directly block)

Key Press Handling

Alphanumeric Keys

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

Alphanumeric Keys

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

Function that receive keyboard input

Alphanumeric Keys

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

Function where the action against a key press is defined

Alphanumeric Keys

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

```
void handleKeypress(unsigned char key, int x, int y) {
switch (key) {
                case 'a':
                _angle = _angle + 45.0;
                glutPostRedisplay();
                } }
```

Function where the action against a key press is defined

Alphanumeric Keys

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

```
void handleKeypress(unsigned char key, int x, int y) {
switch (key) {
                case 'a':
                _angle = _angle + 45.0;
                glutPostRedisplay();
                } }
```

Which key has been pressed

(x, y) Coordinates of cursor while the key is being pressed

Alphanumeric Keys

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

```
float _angle = 0.0;
```

```
void handleKeypress(unsigned char key, int x, int y) {
switch (key) {
        case 'a':
        _angle = _angle + 45.0;
        glutPostRedisplay();
        } }
```

Alphanumeric Keys

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

```
float _angle = 0.0;
```

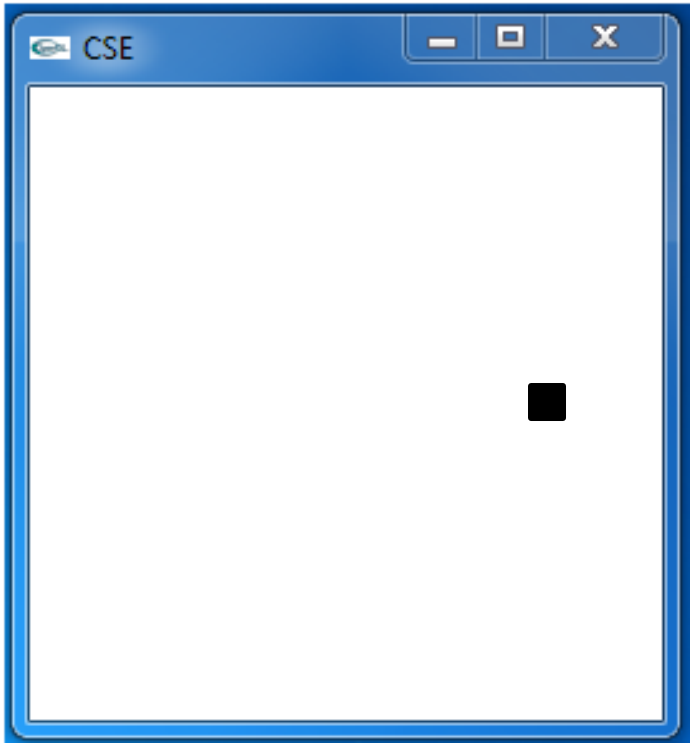
```
void handleKeypress(unsigned char key, int x, int y) {
switch (key) {
        case 'a':
        _angle = _angle + 45.0;
        glutPostRedisplay();
        } }
```

```
void Draw() {
    ..................
        glRotatef(_angle, 0.0, 0.0, 1.0);
        glBegin(GL_POINTS);
            glVertex3f(0.5, 0.0, 0.0);
        glEnd();
    ................
}
```

Alphanumeric Keys

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

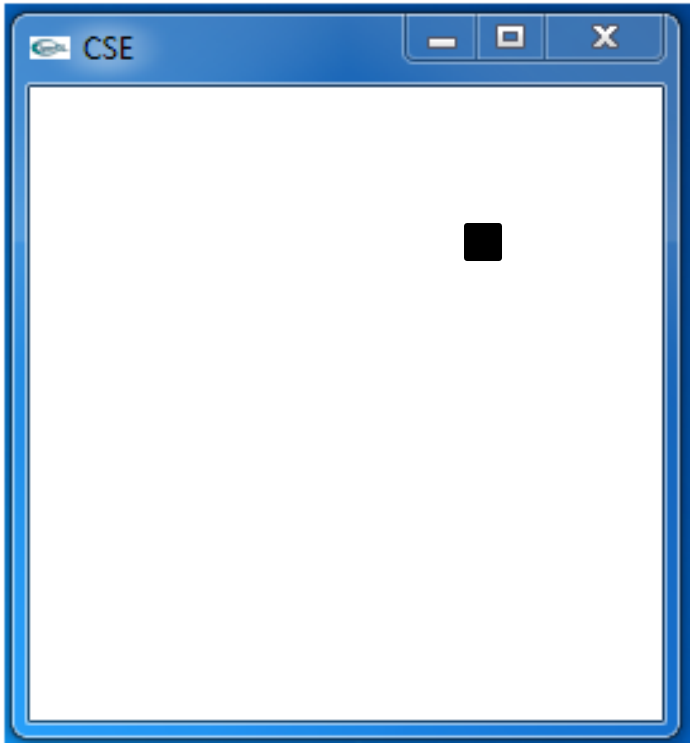
float **_angle** = 0.0;



Alphanumeric Keys

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

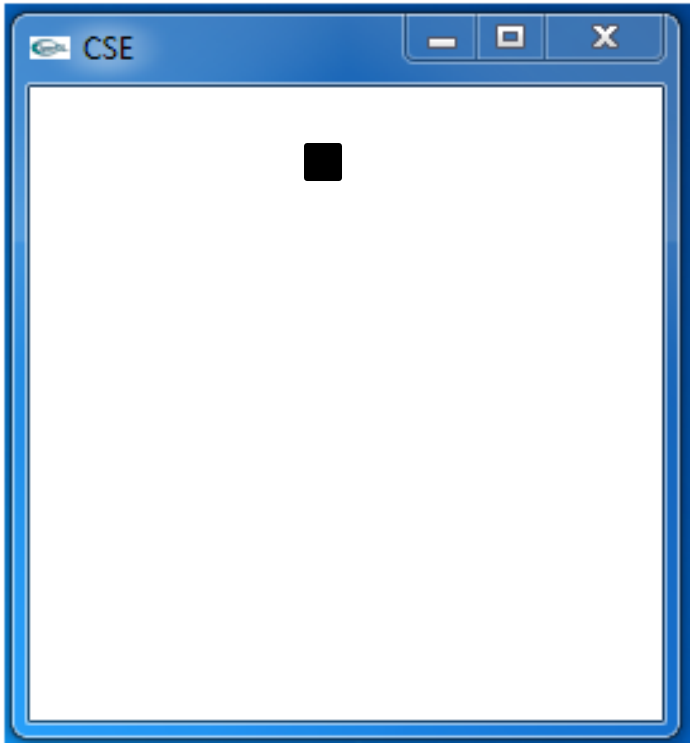
float **_angle** = 45.0;



Alphanumeric Keys

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

float **_angle** = 90.0;



Special Keys

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

Function that receive special keyboard input (e.g. left arrow, right arrow, F2 button etc)

Special Keys

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

Function where the action against a special key-press is defined

Special Keys

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

```
void handleKeypress(int key, int x, int y) {
switch (key) {
                case GLUT_KEY_RIGHT:
                ......................
                } }
```

Special Keys

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

```
void handleKeypress(int key, int x, int y) {
switch (key) {
                case GLUT_KEY_RIGHT:
                ......................
                } }
```

Which key has been pressed (data type is 'int')

Special Keys

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

```
void handleKeypress(int key, int x, int y) {
switch (key) {
                case GLUT_KEY_RIGHT:
                ....................
                } }
```

Constant for Right Arrow Key

Special Keys

For Other Special Keys :

GLUT_KEY_F1 F1 function key.
GLUT_KEY_F2 F2 function key.
GLUT_KEY_F3 F3 function key.
GLUT_KEY_F4 F4 function key.
GLUT_KEY_F5 F5 function key.
GLUT_KEY_F6 F6 function key.
GLUT_KEY_F7 F7 function key.
GLUT_KEY_F8 F8 function key.
GLUT_KEY_F9 F9 function key.
GLUT_KEY_F10 F10 function key.
GLUT_KEY_F11 F11 function key.
GLUT_KEY_F12 F12 function key.
GLUT_KEY_LEFT Left directional key.
GLUT_KEY_UP Up directional key.
GLUT_KEY_RIGHT Right directional key.
GLUT_KEY_DOWN Down directional key.
GLUT_KEY_PAGE_UP Page up directional key.
GLUT_KEY_PAGE_DOWN Page down directional key.
GLUT_KEY_HOME Home directional key.
GLUT_KEY_END End directional key.
GLUT_KEY_INSERT Inset directional key.

```
void handleKeypress(int key, int x, int y) {
switch (key) {
        case GLUT_KEY_RIGHT:
        ....................
        } }
```

Mouse Event Handling

Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
      glutMouseFunc(handleMouseClick);
      ................
}
```

Function that receive mouse input

Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```

Function where the action against a mouse event is defined

Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```

```
void handleMouseClick(int button, int state, int x, int y)
{
if (button == GLUT_LEFT_BUTTON)
        {
         if (state ==  GLUT_DOWN)
                printf("clicked at (%d, %d)\n", x, y);
        }
}
```

Function where the action against a mouse event is defined

Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```

```
void handleMouseClick(int button, int state, int x, int y)
{
if (button == GLUT_LEFT_BUTTON)
        {
         if (state ==  GLUT_DOWN)
                printf("clicked at (%d, %d)\n", x, y);
        }
}
```

On which Button the event is being occurred (Left, Right or Middle)

Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```

```
void handleMouseClick(int button, int state, int x, int y)
{
if (button == GLUT_LEFT_BUTTON)
        {
         if (state ==  GLUT_DOWN)
                printf("clicked at (%d, %d)\n", x, y);
        }
}
```

What is the condition of the event (Down/ Up)

Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```

```
void handleMouseClick(int button, int state, int x, int y)
{
if (button == GLUT_LEFT_BUTTON)
        {
         if (state ==  GLUT_DOWN)
                printf("clicked at (%d, %d)\n", x, y);
        }
}
```

What is the coordinate of the cursor on the ***window*** while the event is being occurred

Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```

```
void handleMouseClick(int button, int state, int x, int y)
{
if (button == GLUT_LEFT_BUTTON)
        {
         if (state ==  GLUT_DOWN)
                printf("clicked at (%d, %d)\n", x, y);
        }
}
```

Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```

```
void handleMouseClick(int button, int state, int x, int y)
{
if (button == GLUT_LEFT_BUTTON)
        {
         if (state ==  GLUT_DOWN)
            printf("clicked at (%d, %d)\n", x, y);
```

GLUT_LEFT_BUTTON
GLUT_MIDDLE_BUTTON
GLUT_RIGHT_BUTTON

Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```

```
void handleMouseClick(int button, int state, int x, int y)
{
if (button == GLUT_LEFT_BUTTON)
        {
         if (state == GLUT_DOWN)
                printf("clicked at (%d, %d)\n", x, y);
        }
}
```

Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```

```
void handleMouseClick(int button, int state, int x, int y)
{
if (button == GLUT_LEFT_BUTTON)
        {
         if (state == GLUT_DOWN)
                printf("clicked at (%d, %d)\n", x, y);
        }
}
```

GLUT_DOWN
GLUT_UP

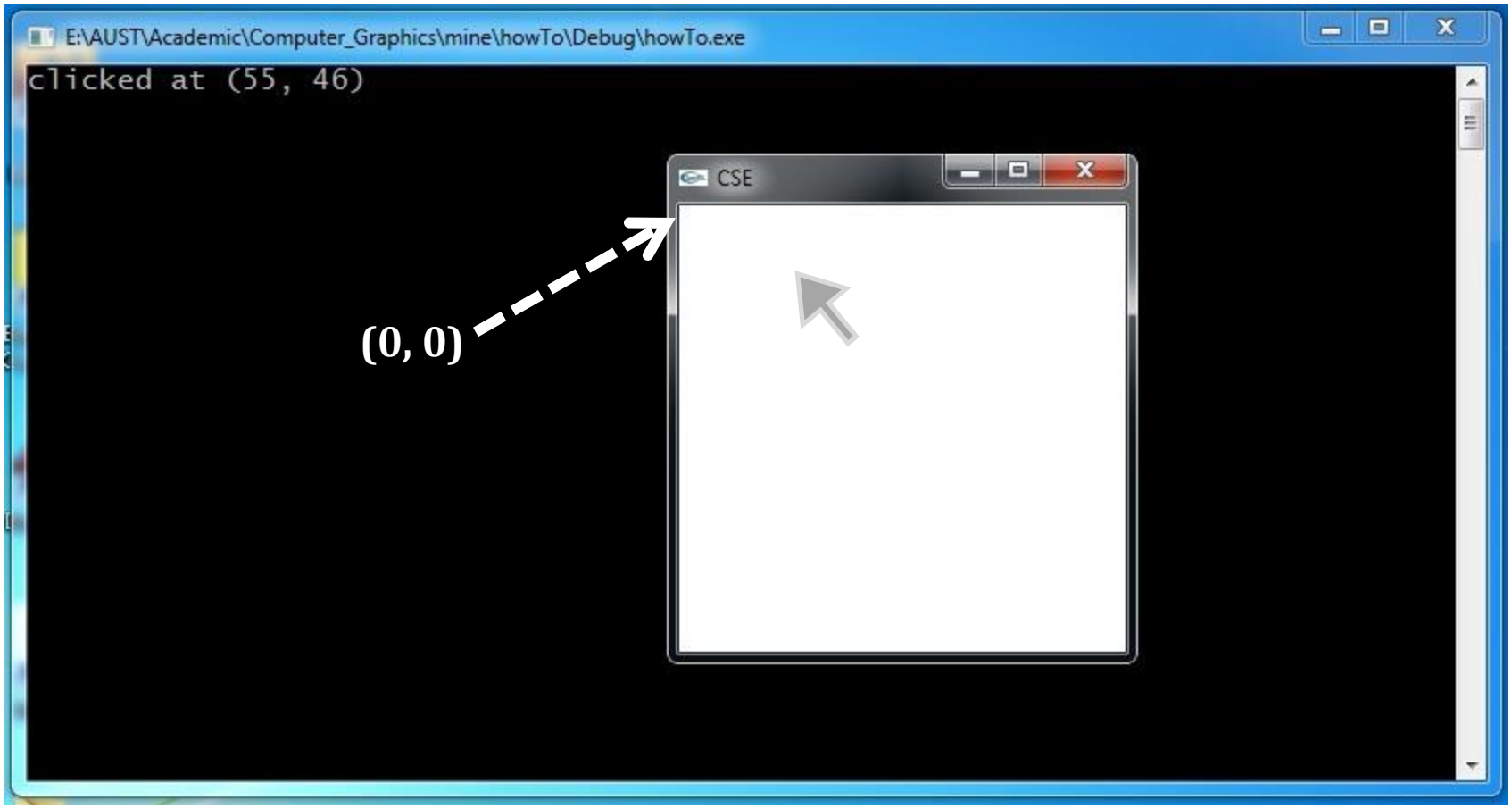
Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```

```
void handleMouseClick(int button, int state, int x, int y)
{
if (button == GLUT_LEFT_BUTTON)
        {
         if (state ==  GLUT_DOWN)
                printf("clicked at (%d, %d)\n", x, y);
        }
}
```

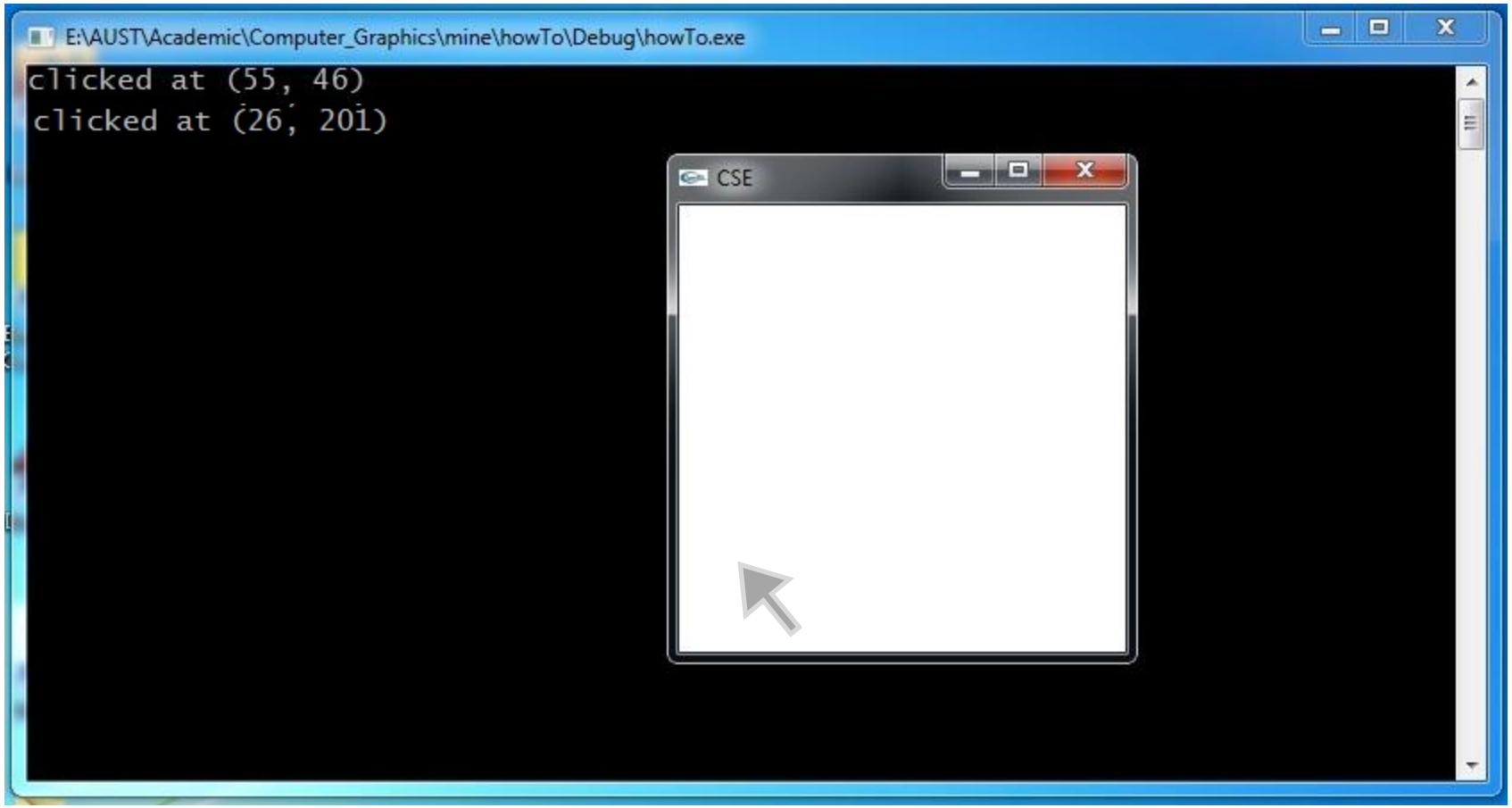
Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```



Mouse Events

```
int main(int iArgc, char** cppArgv)
{
        ...............
        glutMouseFunc(handleMouseClick);
        ................
}
```



THANK YOU