OpenGl Course Summer 2020

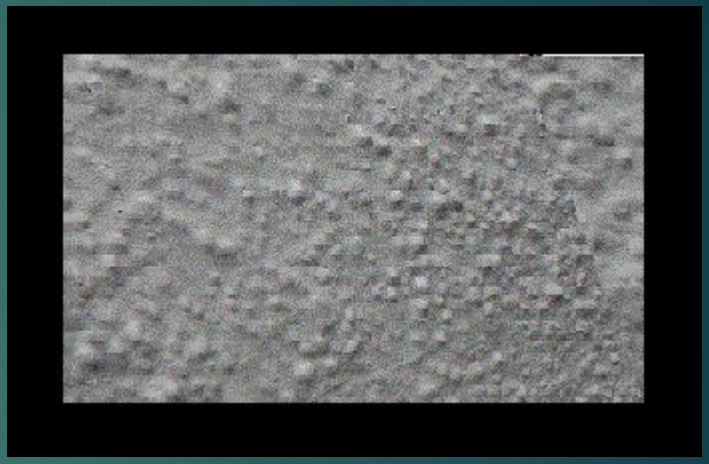
Texture

Texture

- Achieving a high level of realism with nothing but thousands or millions of tiny lit and shaded polygons is a matter of brute force and a lot of hard work. Unfortunately, the more geometry you throw at graphics hardware, the longer it takes to render. A clever technique allows you to use simpler geometry but achieve a higher degree of realism. This technique takes an image, such as a photograph of a real surface or detail, and then applies that image to the surface of a polygon.
- This technique of applying an image to a polygon to supply additional detail is called *texture mapping*

Texture

Before Using Texture

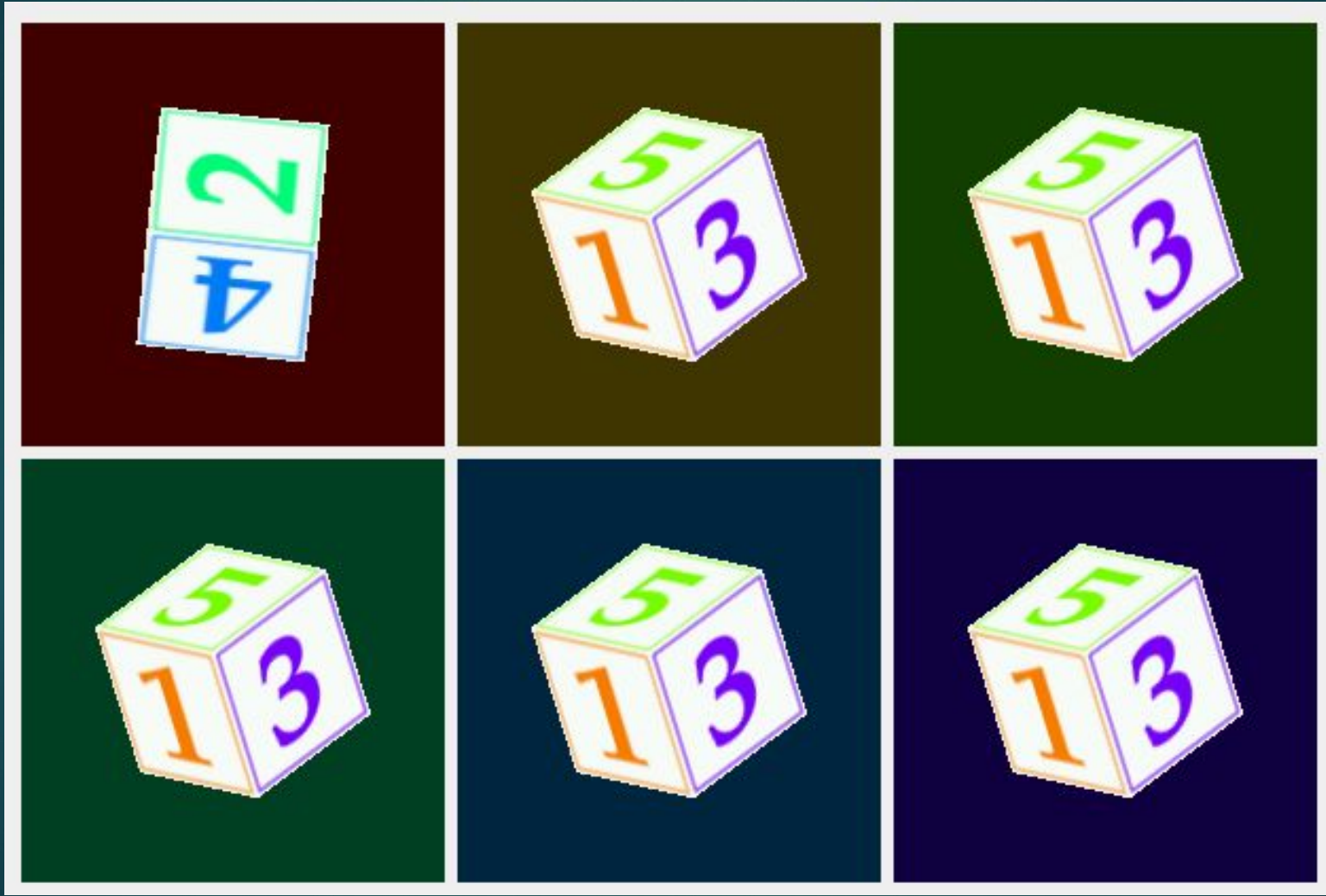


After Using Texture

Texture

Example:

Texture



Texture

- There are various ways to do texture mapping. Some of them use a technique called Bitmapping in OpenGL, the others use a way that load ready images from some resource. We'll deal with the second way in this course.
- The second technique needs some way to load textures. We'll use a texture loader called 'texture.h'
- there is four steps to use texture in our scence.

Texture

Step 1

Texture

- **Step 1:**

Enable the texturing

- Function:

void glEnable(Glenum parameter);

- Parameters:

i. GL_TEXTURE_1D
ii. **GL_TEXTURE_2D**
iii. GL_TEXTURE_3D

Texture

Step 2

Texture

Load Texture

- **Step 2:**

```
Image= LoadTexture("Pic.bmp");
```

- Function:

```
int LoadTexture(char* filename,int alpha=255);
```

- Example:
  - We'll put the image in some integer defined before InitGL function:

```
int Image;
```

  - Inside **InitGL** function we'll load the image, for example:
  - Now the image is stored in the variable Image.

Texture

Step 3

Texture

- **Step 3:**

Bind Texture

- Function:

Now, the following work is inside **DrawGLScene** function.

void glBindTexture(Glenum Target, Gluint Texture);

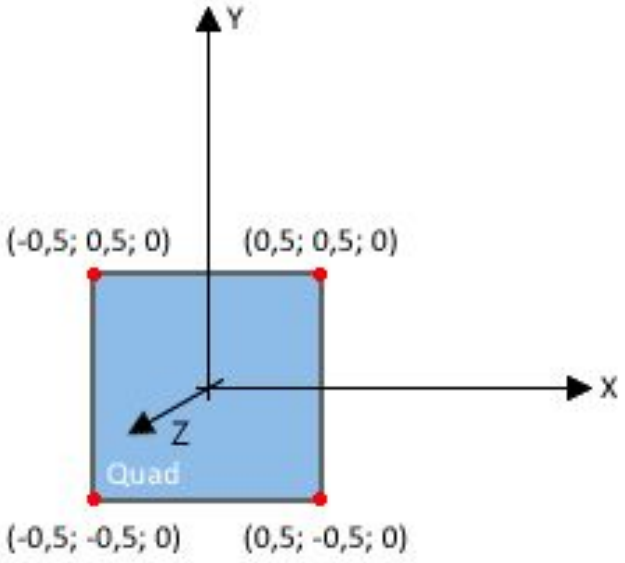
- **Target** :GL_TEXTURE_1D,GL_TEXTURE_2D,GL_TEXTURE_3D
- **Texture:** the int that was returned by the LoadTexture function in step 2 ,of the texture we want to use now ('cause it's not necessary to have just one texture).

Texture

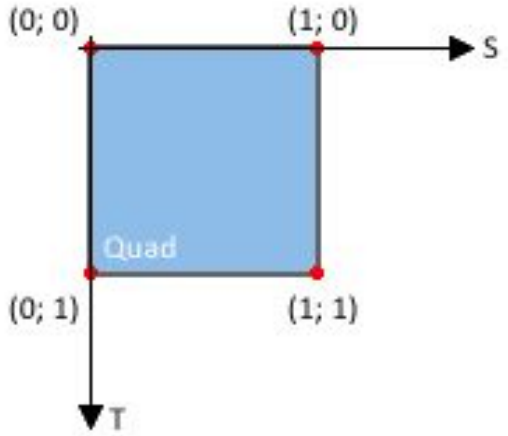
Step 4

Mapping Texture Coordinates

- **Step 4:**



XYZ coordinates:
- Vertices defined in counter clockwise order
- Z grows inwards (meaning, a positive Z is "behind" you)



ST coordinates:
- Also known as UV coordinates
- T is flipped
- Each axis has a range of [0, 1]
- Tiling possible when exceeding the range

Texture

- Now we'll draw the primitive that we'll bind the

texture on it. It's common to bind the texture on a quad.
The function that is used to bind the vertex of the texture is:

glTexCoord2f (s,t);

- The values of x and y are 0 or 1, it's just used to specify the angle that contains the vertex.

Texture

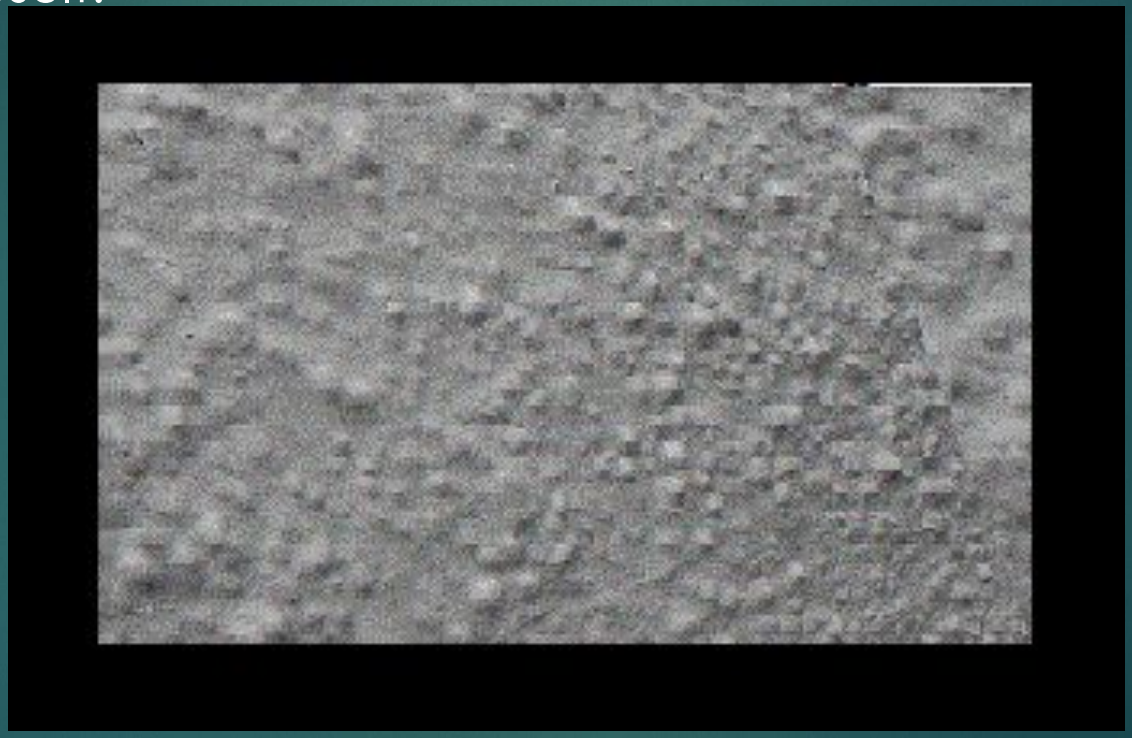
- The code of drawing the quad that contains the image which we loaded:

```
glBegin(GL_QUADS);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(-5.0f,0.0f,0.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(-5.0f,6.0f,0.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(5.0f,6.0f,0.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(5.0f,0.0f,0.0f);
glEnd();
```

- Noti

Texture

- The result:



Texture

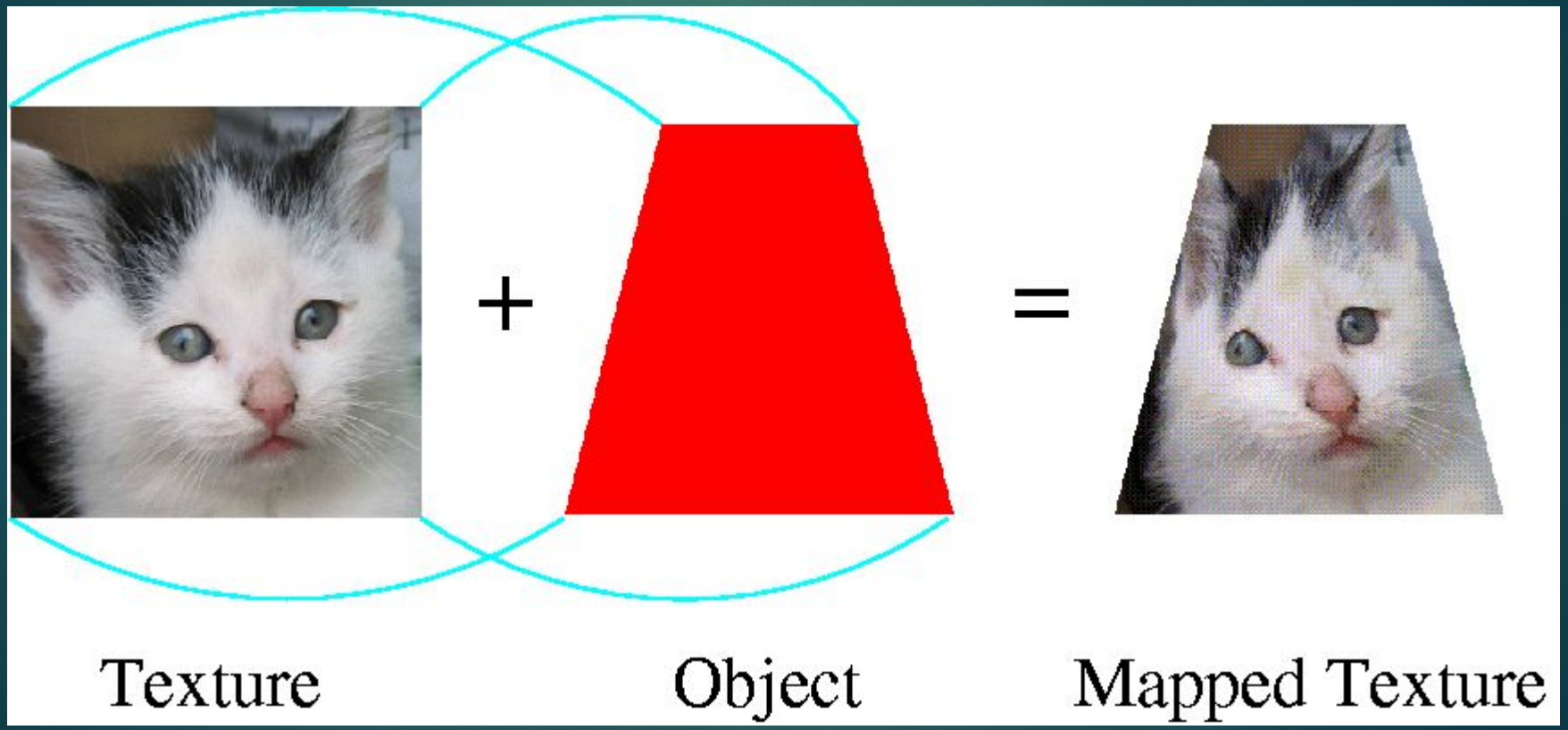
- **Tips:**
  - To draw a quad with a texture, it's advisable to enable texture before drawing it immediately, **then disable the texture** after it immediately. Never forget to disable it. The reason is that it will affect the colors of the objects which is drawn after it.
  - Use images with the extension BMP. Most of loaders use this extension to load images. (Sure, there are loaders that use other extensions)
  - Use images with size of powers of number 2 and their sum. For example, 128*128 , 64*64, 256*128, 384*256, ..etc
    - Notice that 384 is not from the powers of 2, but it's the sum of 128+256 ☺
    - The reason that it's easier for the loader to bind the images with these sizes.

Texture

Irregular quadrilateral polygon

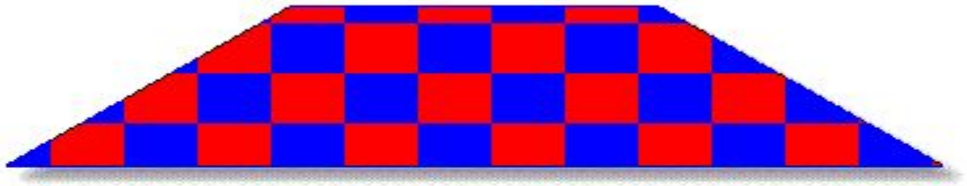
Texture

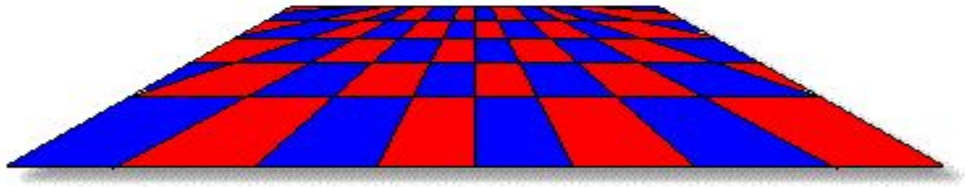
- Irregular quadrilateral polygon



Texture

- Irregular quadrilateral polygon

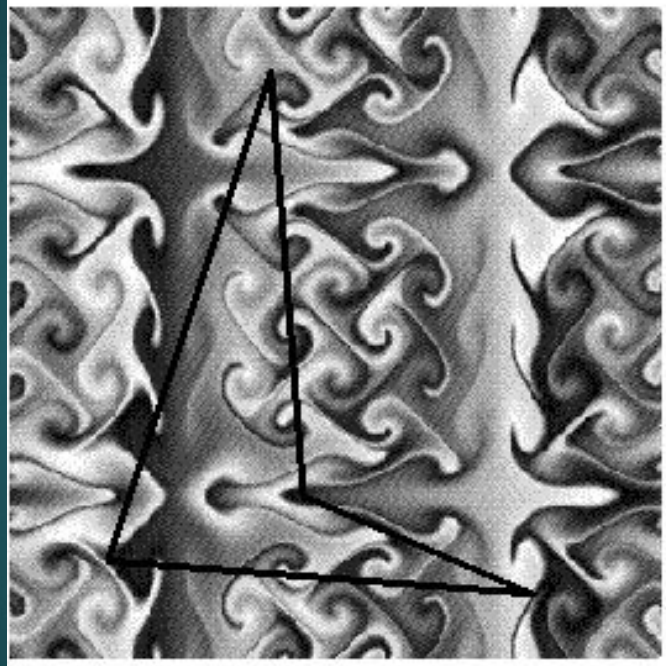


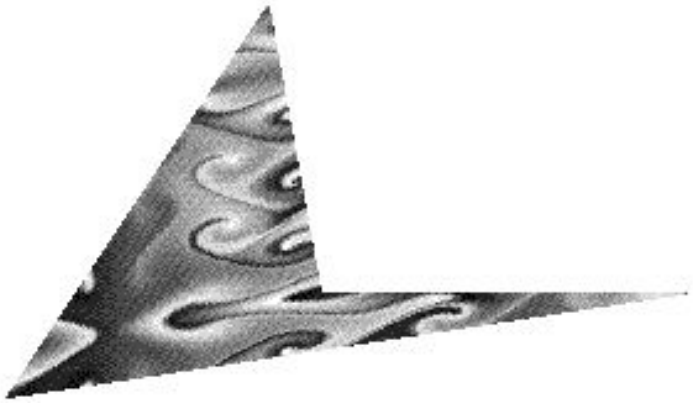


Non quadrilateral polygon

Texture

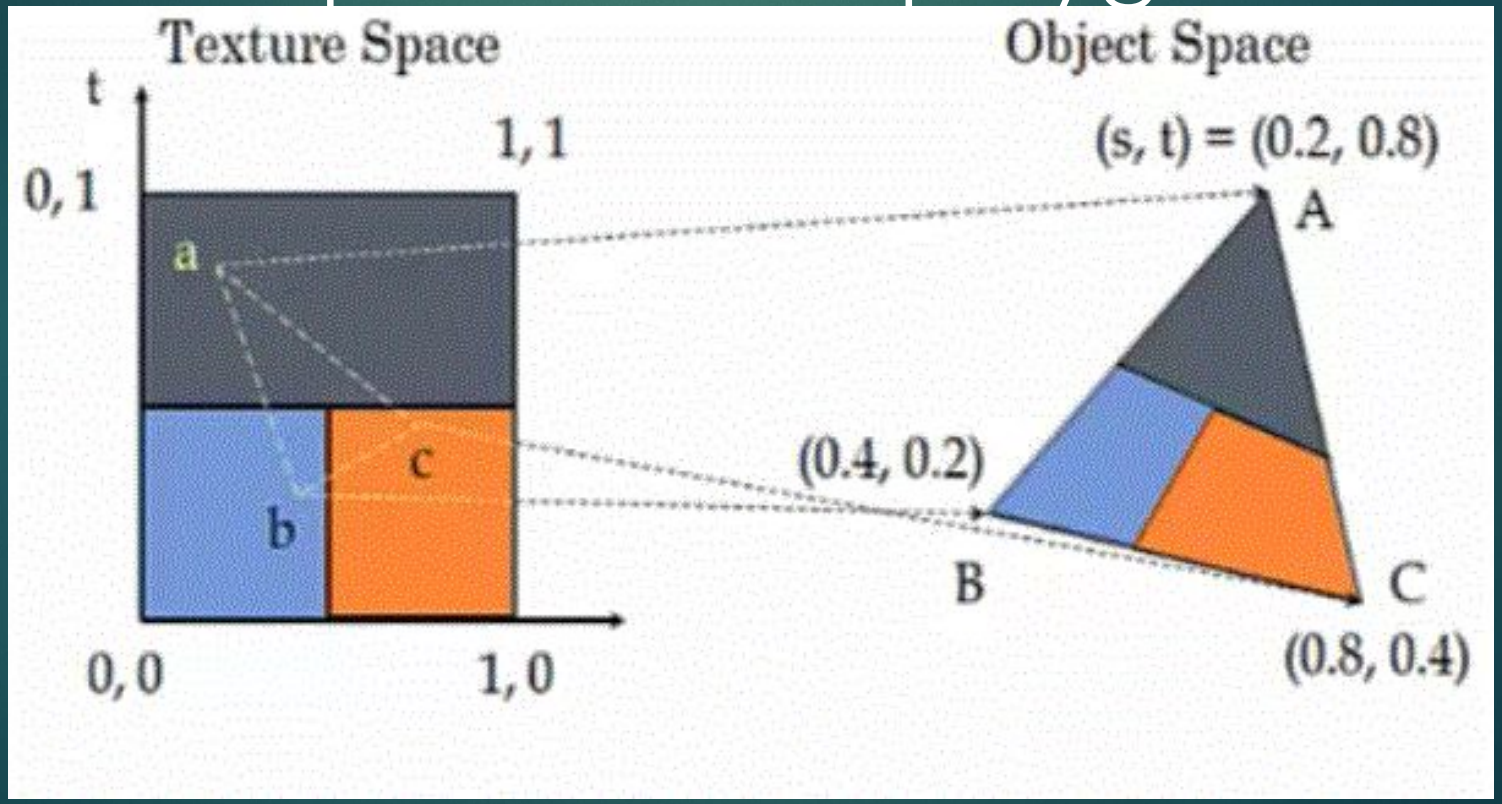
- Non quadrilateral polygon





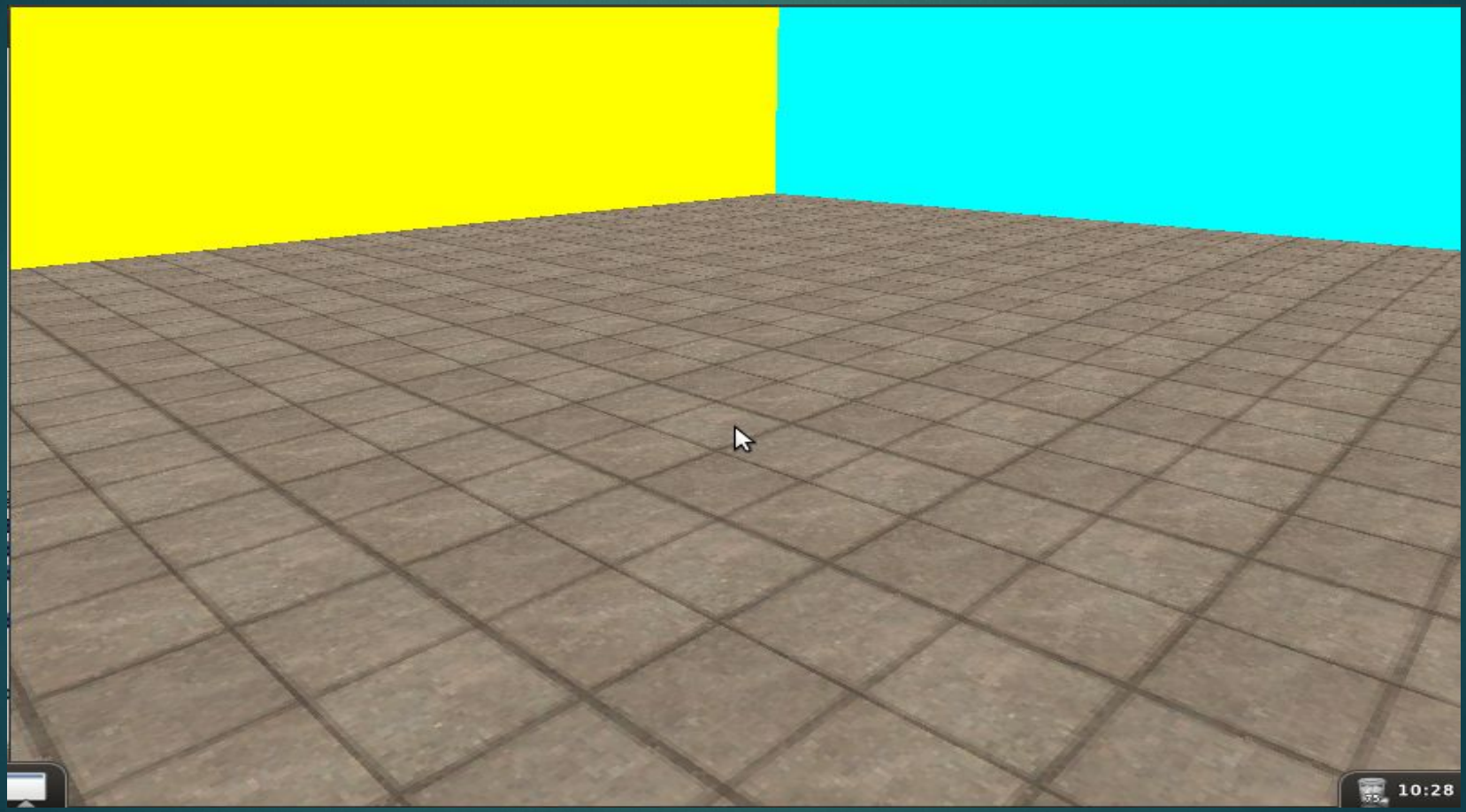
Texture

- Non quadrilateral polygon



Repeating Texture

Texture



Texture

- Repeating Texture

Texture

Repeating Texture

some times we need to repeat texture on the same polygon

- Function call **InitGL** function:

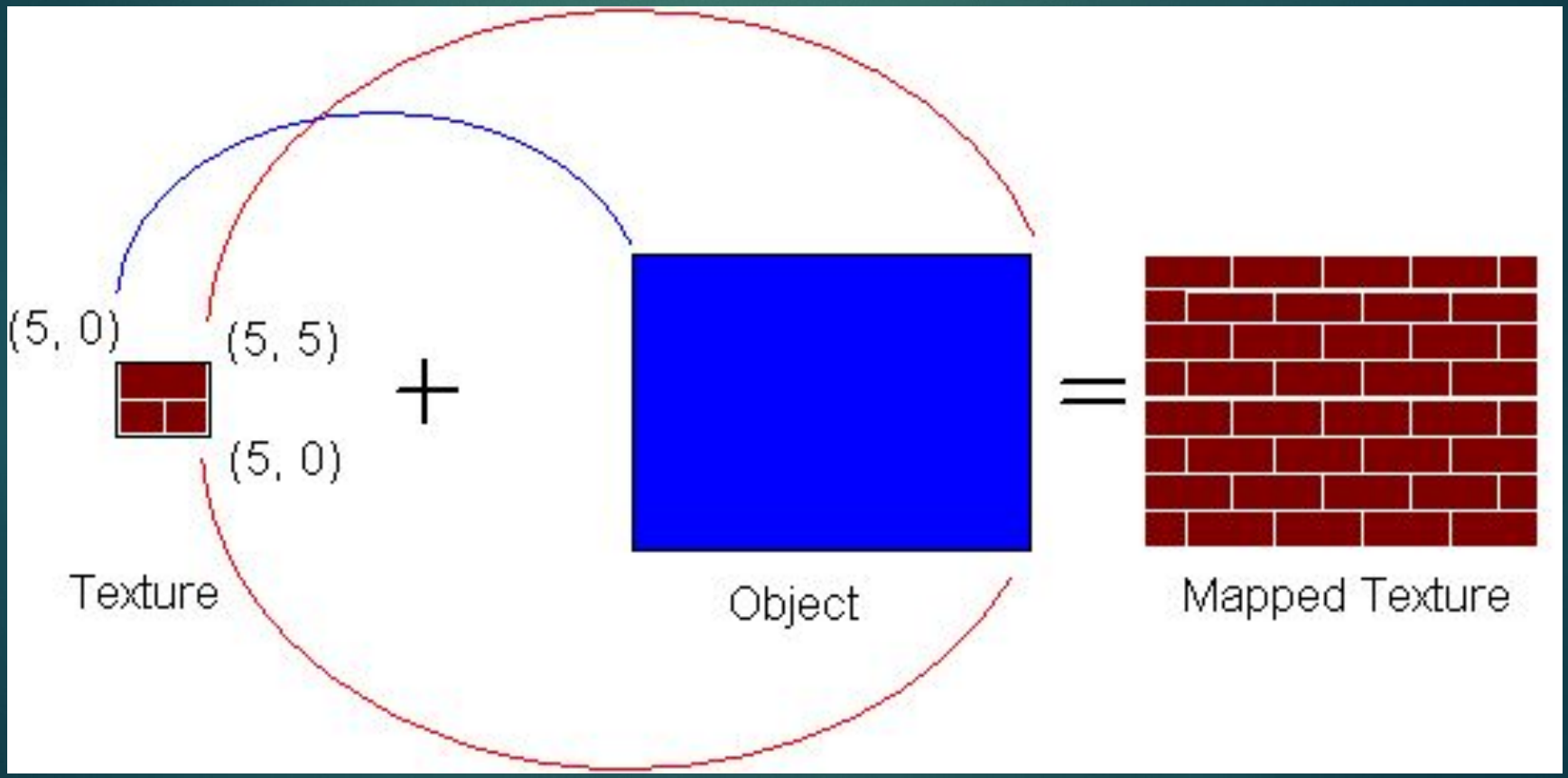
```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
```

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

- GL_TEXTURE_WRAP_S and GL_TEXTURE_WRAP_T mean that we call glTexCoord2f (s,t) with parameters bigger than one, the texture will be repeated by the number mentioned.

Texture

- Repeating Texture



Texture
- Example

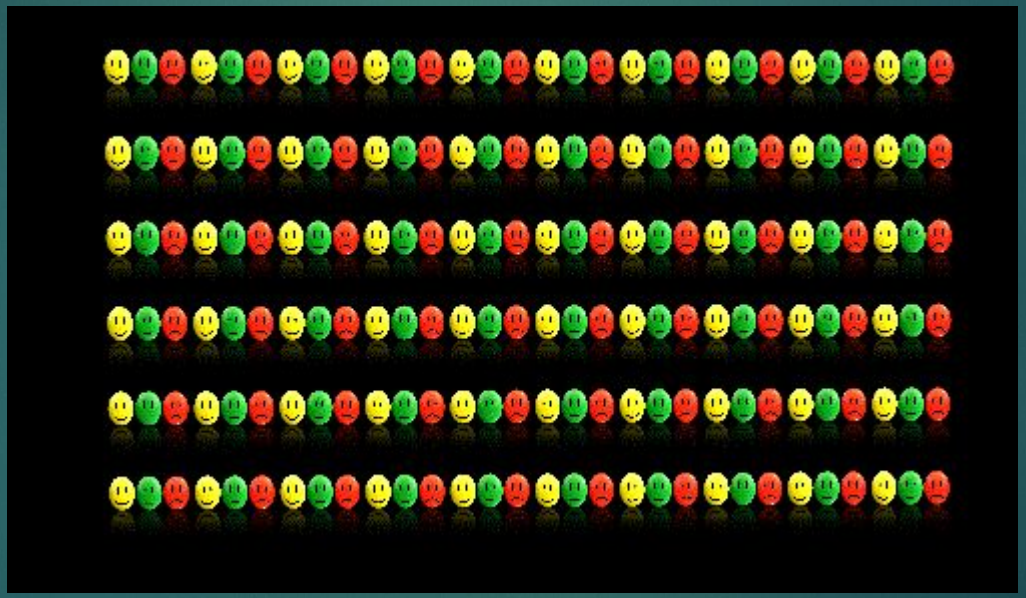
Example

Texture
- Example


```
glBegin(GL_QUADS);
glColor3f(1,1,1);
glNormal3f(0,0,1);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(-5.0f,0.0f,0.0f);
glTexCoord2f(0.0f, 6.0f);
glVertex3f(-5.0f,6.0f,0.0f);
glTexCoord2f(10.0f, 6.0f);
glVertex3f(5.0f,6.0f,0.0f);
glTexCoord2f(10.0f, 0.0f);
glVertex3f(5.0f,0.0f,0.0f);
glEnd();
```

Texture
- Example

- The result:



Texture
- Example

- Example : without using glTexParameteri in InitGL function.

```
glBegin(GL_QUADS);
glColor3f(1,1,1);
glNormal3f(0,0,1);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(-5.0f,0.0f,0.0f);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(-5.0f,6.0f,0.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(5.0f,6.0f,0.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(5.0f,0.0f,0.0f);
glEnd();
```

Texture
- Example

- The result:

Add texture.h

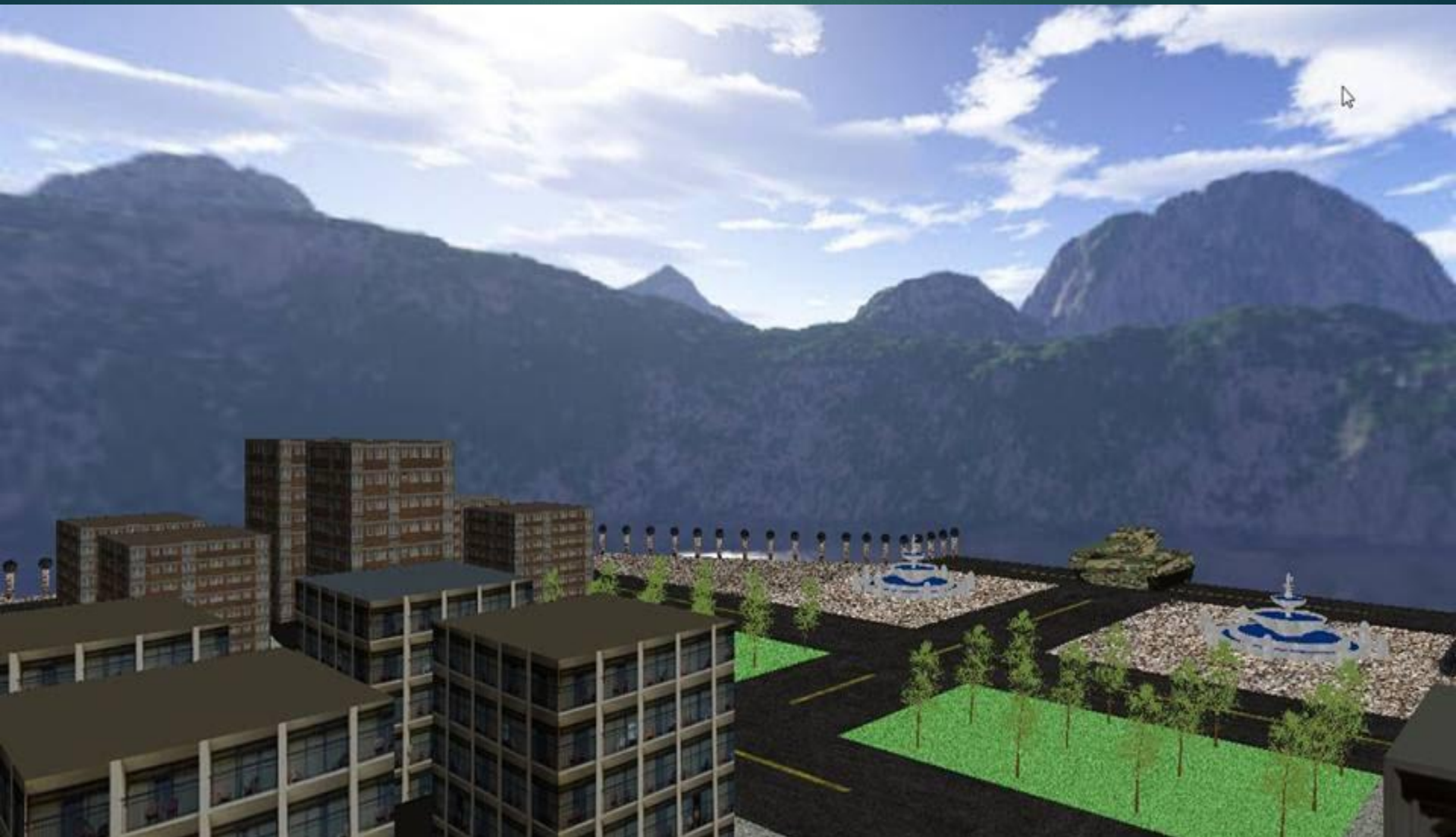
- Project □ property □ c/c++ □ preprocessor □ preprocessor definitions □ _CRT_SECURE_NO_WARNINGS

skybox

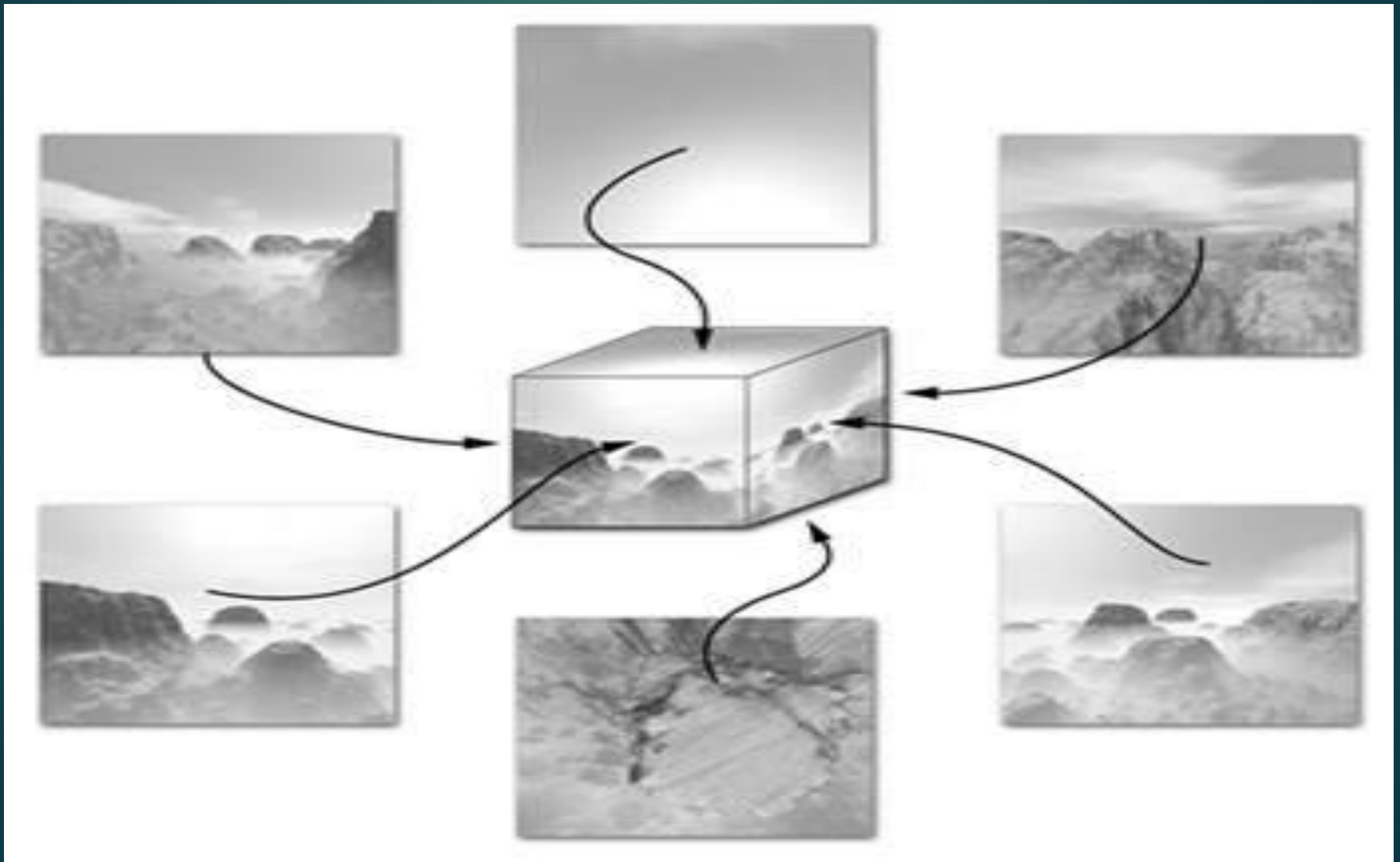
skybox

skybox

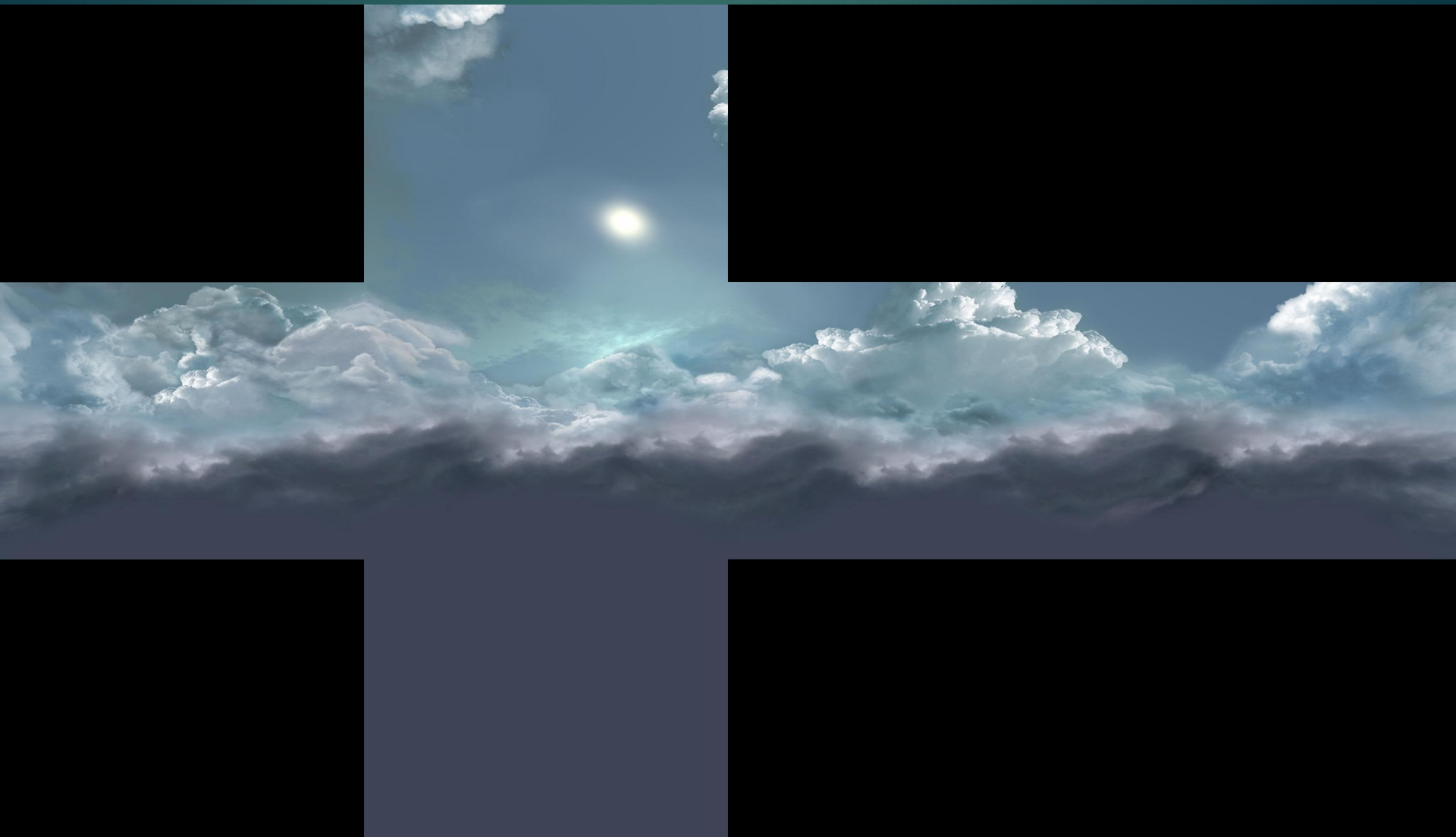
Homework



skybox



skybox



Homework

- Add another library to load texture
- skybox
- Camera controller

skybox