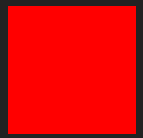
# Color and Textures

# Color in OpenGL

Instead of values 0 - 255 or #00 - #ff,
OpenGL colors have 3 or 4 channels ranging from
0.0 to 1.0 (floating point).

## RGB



```
1.0, 0.0, 0.0
```

```
1.0, 0.5, 0.0
```

## RGBA

```
1.0, 0.0, 0.0, 0.75
```

```
1.0, 0.5, 0.0, 0.2
```

Start with a clear screen each frame.

# glClearColor

Sets the color to use when clearing the screen.

```
glClearColor(float red, float green, float blue, float alpha);
```

```
glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
```

# glClear

Clears the screen using the color last set by glClearColor.

```
void Initialize() {
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
}

void Render() {
    glClear(GL_COLOR_BUFFER_BIT);
}
```

# Experiment!

Need a blue sky? Dark cave? Desert?

You can use clear color!

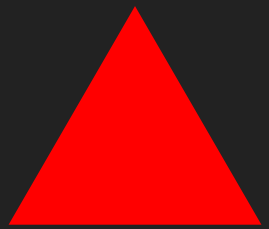

Setting a solid color of an untextured polygon.

# ShaderProgram::SetColor

Sets the color to use when drawing a polygon.

```
ShaderProgram::SetColor(float red, float green, float blue,
                                                  float alpha);
```

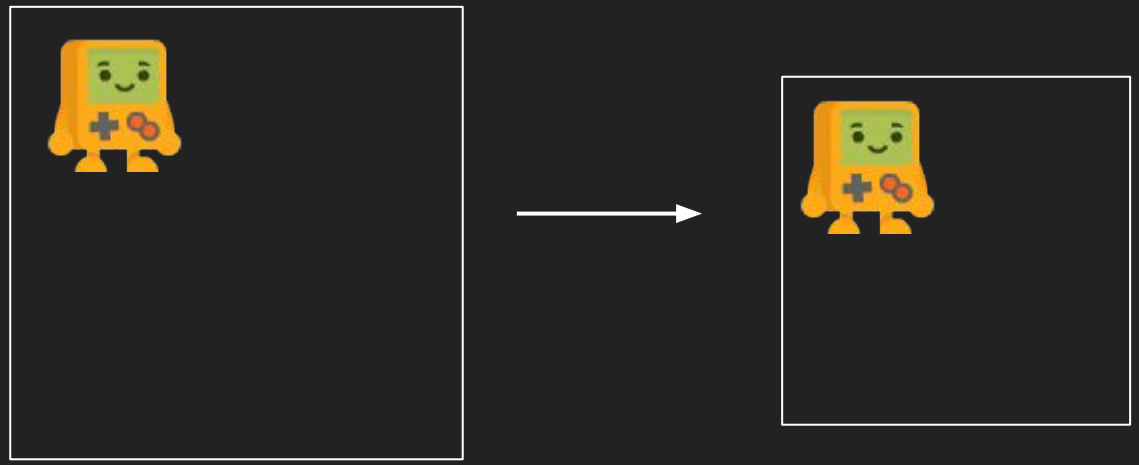
```
program.SetColor(1.0f, 0.0f, 0.0f, 1.0f);
```



# Textures/Images in OpenGL

You do this during your setup, not every frame!

# OpenGL Textures



RAM

Video Card RAM

# Loading an image with STB_image

You must include STB_IMAGE_IMPLEMENTATION in one of the files you are including it from!

```
#define STB_IMAGE_IMPLEMENTATION
#include "stb_image.h"
```

Use stbi_load to load the pixel data from an image file.

```
int w, h, n;
unsigned char* image = stbi_load("pacman.png",
                                 &w, &h, &n, STBI_rgb_alpha);
```

After you are done loading the image data, you must free it.

```
stbi_image_free(image);
```

# Create a texture in OpenGL

```
GLuint textureID;
glGenTextures(1, &textureID);
```

# Binding a texture

```
glBindTexture(GL_TEXTURE_2D, textureID);


// GL_TEXTURE_2D is a “target”
// Next slide will make this make sense...
```

# Setting the texture pixel data

This is what sends the image to the graphics card.

```
// Our images must be RGBA!

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, w, h, 0, GL_RGBA,
                                    GL_UNSIGNED_BYTE, image);
```

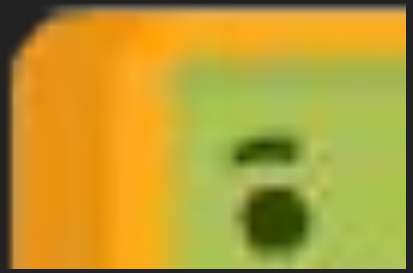
# Texture Filtering

Original

Minification

Magnification

# Texture Filtering



Linear

Good for high resolution textures.

Nearest neighbor

Good for pixel art.

# Texture filtering settings.

```
// Use GL_LINEAR or GL_NEAREST
// MIN = Minifying, MAG = Magnifying

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

# Let’s make that into a function.

```
GLuint LoadTexture(const char* filePath) {
    int w, h, n;
    unsigned char* image = stbi_load(filePath, &w, &h, &n, STBI_rgb_alpha);

    if (image == NULL) {
        std::cout << "Unable to load image. Make sure the path is correct\n";
        assert(false);
    }

    GLuint textureID;
    glGenTextures(1, &textureID);
    glBindTexture(GL_TEXTURE_2D, textureID);
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, w, h, 0, GL_RGBA, GL_UNSIGNED_BYTE, image);

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

    stbi_image_free(image);
    return textureID;
}
```

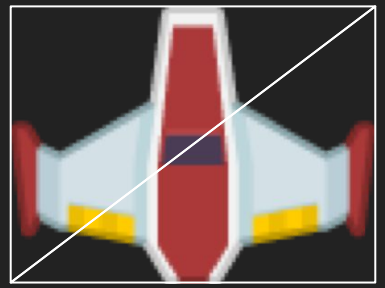
Now that the texture is loaded.
We can apply it to our models
as we draw each frame.

# Texture Coordinates

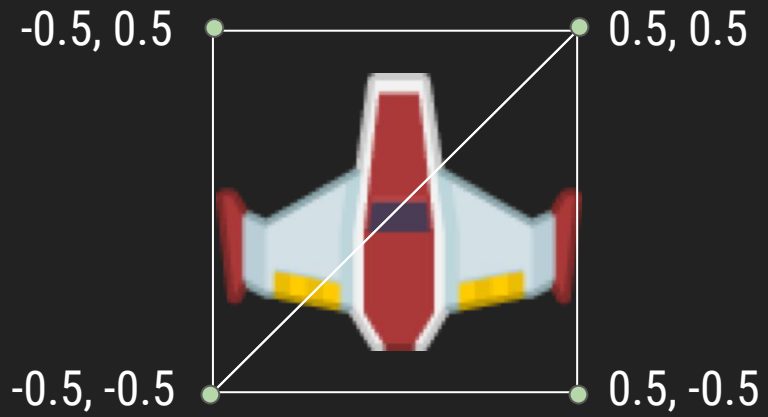

Texture coordinates are referred to as UV coordinates (X, Y and Z were already taken) :)

Notice the range from 0.0 to 1.0 and not by pixels.

# 2D Sprite Made of 2 Triangles



# Need to match vertices to UV coordinates





```
float vertices[]  = {-0.5, -0.5, 0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5, -0.5, 0.5};
float texCoords[] = {0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0};
```

Cool story Prof…
But how do we code that?

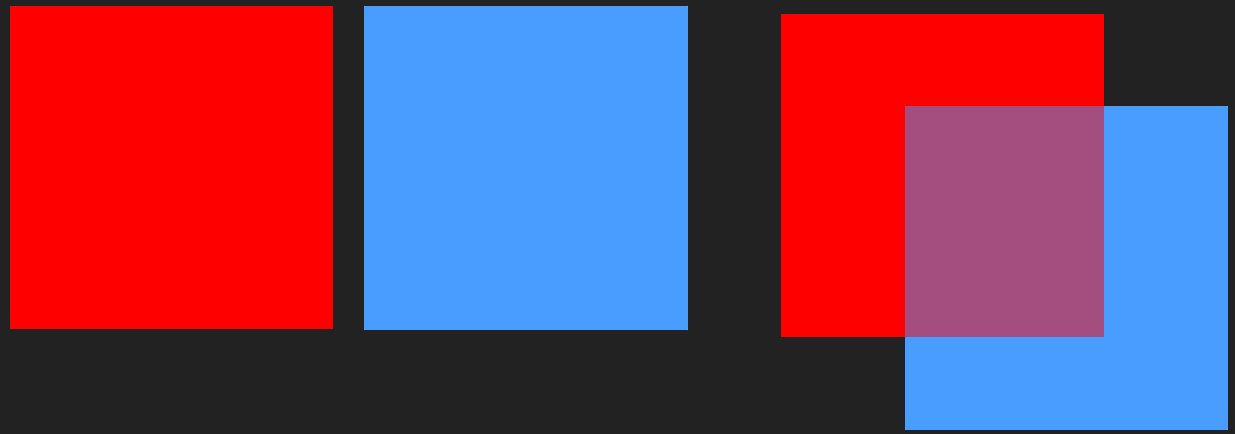
# Initialization

```
GLuint playerTextureID;

void Initialize()
{

    // Load the shaders for handling textures!
    program.Load("shaders/vertex_textured.glsl",
                 "shaders/fragment_textured.glsl");

    // Load our player image
    playerTextureID = LoadTexture("player.png");
}
```

# Rendering

```
void Render() {
    glClear(GL_COLOR_BUFFER_BIT);

    program.SetModelMatrix(modelMatrix);

    float vertices[]  = { -0.5, -0.5, 0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5, -0.5, 0.5 };
    float texCoords[] = { 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0 };

    glBindTexture(GL_TEXTURE_2D, playerTextureID);

    glVertexAttribPointer(program.positionAttribute, 2, GL_FLOAT, false, 0, vertices);
    glEnableVertexAttribArray(program.positionAttribute);

    glVertexAttribPointer(program.texCoordAttribute, 2, GL_FLOAT, false, 0, texCoords);
    glEnableVertexAttribArray(program.texCoordAttribute);

    glDrawArrays(GL_TRIANGLES, 0, 6);

    glDisableVertexAttribArray(program.positionAttribute);
    glDisableVertexAttribArray(program.texCoordAttribute);

    SDL_GL_SwapWindow(displayWindow);
}
```

# Blending



# Blending

(blending is off by default,  we need to enable it so our images are transparent)

```
glEnable(GL_BLEND);

// Good setting for transparency
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

https://learnopengl.com/Advanced-OpenGL/Blending

# If your image does not load…

(In Xcode) You may have to go to “Build Phases” and add your image to the Copy Files area.

(Visual Studio) Use the file explorer to copy images into your project’s folder.

# Project 1:

Create a simple 2D Scene:

- Have at least 1 untextured object.
- Have at least 2 textured objects (with different textures).

You can use any images you want or use the ones in the github repository.

Commit your code to your GitHub repository.

Post the link in the Assignments area.

For example, your link might look like:

https://github.com/tonystark/CS3113/P1/

# In-Class Activity!

Go to https://www.pixilart.com

Draw something!

After 15 minutes, let's code!