Computer Graphics
Theory and Applications

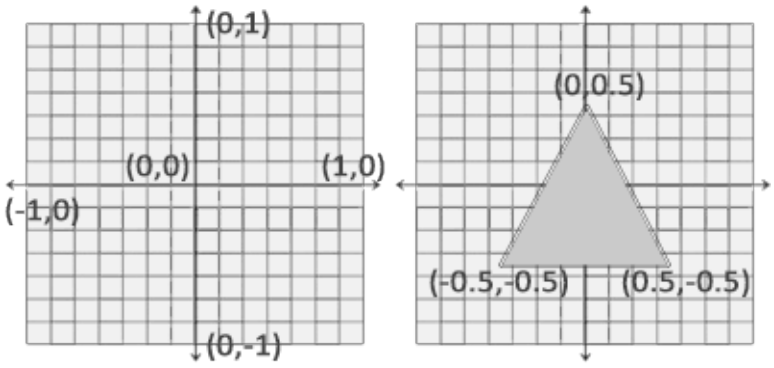
Konstantinos Zouridakis

Contents

1 OpenGL

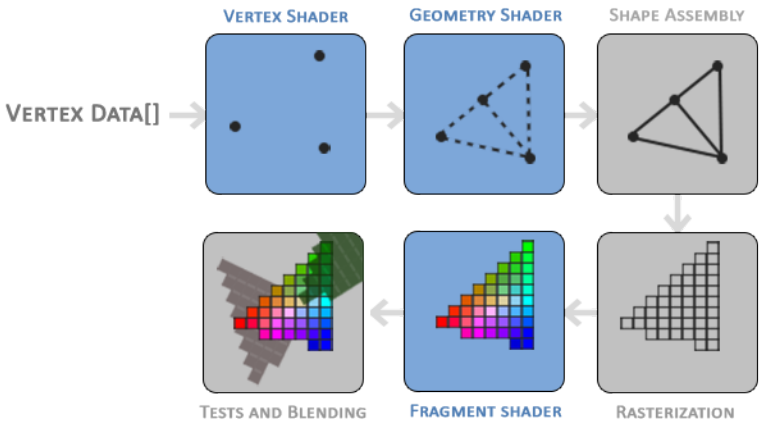
1.1 Normalized Device Coordinates (NDC)

In Normalized Device Coordinates, x, y and $z \in [-1, 1]$. This is the default coordinate system in OpenGL. The following image shows the NDC space.



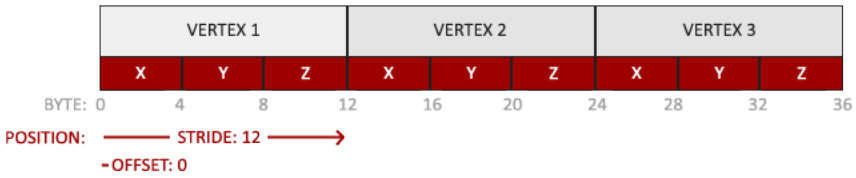
1.2 Graphics Pipeline

The order with which OpenGL proccesses vertex data:



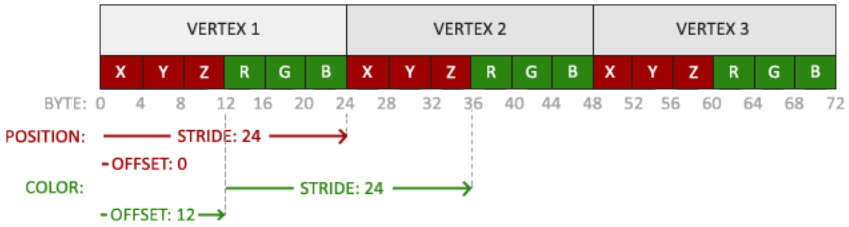
1.3 Vertex Buffer Data

Vertex attributes are stored in the following order:



Vertices are stored in the Vertex Buffer Object (VBO).

We can also specify colors in the vertex buffer data. In the shader file, coordinates will be defined in `location 0` and colors in `location 1`.



In the main file:

```
float vertices[] = {
    // positions // colors
    0.5f, -0.5f, 0.0f, 1.0f, 0.0f, 0.0f, // bottom right
    -0.5f, -0.5f, 0.0f, 0.0f, 1.0f, 0.0f, // bottom left
    0.0f, 0.5f, 0.0f, 0.0f, 0.0f, 1.0f // top
};
```

In the shader file:

```
#version 330 core
layout (location = 0) in vec3 aPos; // the position variable has attribute position
     0
layout (location = 1) in vec3 aColor; // the color variable has attribute position
     1

out vec3 ourColor; // output a color to the fragment shader

void main()
{
    gl_Position = vec4(aPos, 1.0);
    ourColor = aColor; // set ourColor to the input color we got from the vertex
        data
}
```

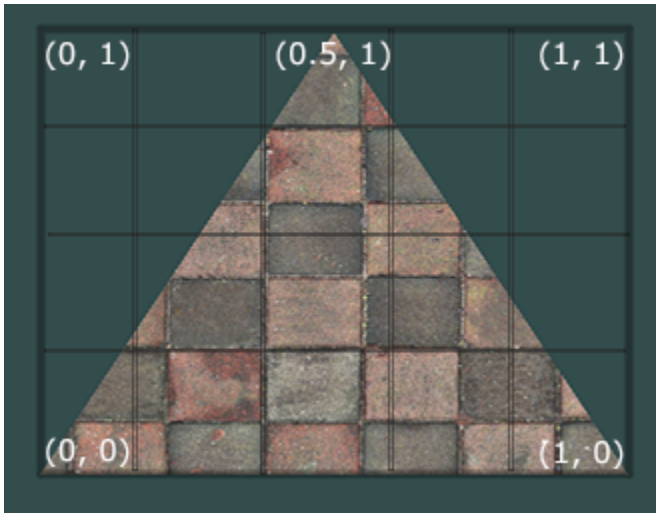
1.4 VBO, VAO, EBO

- **VBO** stores vertex data (e.g., positions, normals, colors) in GPU memory. It allows efficient transfer of vertex data to the GPU, enabling faster rendering. You bind a VBO and then specify the vertex data using functions like glBufferData.
- **VAO** stores the configuration of vertex attributes and their associated buffers. It simplifies the process of switching between different vertex data configurations. When you bind a VAO, it automatically sets up the vertex attribute pointers and buffer bindings.
- **EBO** stores indices that define the order in which vertices should be drawn. It helps in reducing the amount of vertex data needed by reusing vertices. You bind an EBO and specify the indices using functions like glBufferData.

1.5 Shaders

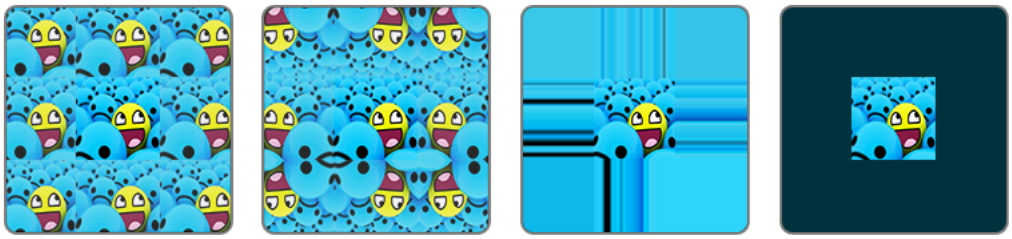
1.6 Textures

Texture coordinates in OpenGL are in range $[0, 1] \times [0, 1]$.



Specifying coordinates outside this range, OpenGL will repeat the texture by default. There are also other options:

1. `GL_REPEAT`: Default.
2. `GL_MIRRORED_REPEAT`: Same as `GL_REPEAT` but mirrors the image with each repeat.
3. `GL_CLAMP_TO_EDGE`: Clamps the coordinates between 0 and 1. The result is that higher coordinates become clamped to the edge, resulting in a stretched edge pattern.
4. `GL_CLAMP_TO_BORDER`: Coordinates outside the range are now given a user-specified border color.



1.7 Hello Window

You can find the whole source code for the Hello Window here.

1.8 Hello Triangle

The whole source code for the Hello Triangle can be found here.