

Assignment 2

Peter Johan Flått-Bjørnstad (Individual, Group 32)

Task 1

b)

The screenshot shows a development environment with three main windows:

- Terminal:** Shows the output of the command "cargo run". It includes several warnings from the compiler, such as "warning: associated function 'overlapping_triangles' is never used" and "warning: 'gloom-rs' (bin "gloom-rs") generated 17 warnings". The terminal also displays OpenGL version information and window size changes.
- Code Editor:** Displays two files: `simple.vert` and `main.rs`.
 - `simple.vert` contains GLSL vertex shader code. It defines a vertex attribute `vap_index` and a fragment color output. The `main()` function sets the fragment color to the vertex color and specifies a constant pointer for the position.
 - `main.rs` contains Rust code for initializing OpenGL state. It creates a VAO, binds a vertex buffer, generates a color buffer object (CBO), fills it with data, and configures a vertex attribute array (VAP) for vertex colors.
- Rendering Window:** Titled "Gloom-rs", it shows a rendered scene consisting of three overlapping triangles forming a larger triangle. The triangles are colored with a gradient: top-left is green, top-right is red, and bottom is blue.

Task 2

a)



```
123 | pub fn generate_square(side_length: f32) -> (Vec<f32>, Vec<u32>) {  
147 |     pub fn generate_sine(n_points: usize, width: f32, angle_rate: f32) -> (Vec<f32>, Vec<u32>) {  
  
warning: associated function `generate_sine` is never used  
--> src/shape_generator.rs:147:12  
147 |     pub fn generate_sine(n_points: usize, width: f32, angle_rate: f32) -> (Vec<f32>, Vec<u32>) {  
147 |           
147 |             pub fn generate_sine() generated 18 warnings  
147 |             Finished dev [unoptimized + debuginfo] target(s) in 2.68s  
147 |             Running "target/debug/glomm-rs"  
147 |             New window size! width: 800, height: 600  
147 |             AMD: AMD Radeon Graphics (rénoir, LLVM 15.0.7, DRM 3.52, 6.4.6-760604  
147 |             OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0~1689084530~22.04~06187  
147 |             GLSL   : 4.60  
147 |             0.5  
147 |             Resized  
147 |             New window size! width: 800, height: 637  
147 |             Resized  
147 |             New window size! width: 628, height: 512  
147 |             Resized  
147 |           
147 |     }  
147 | }  
  
src > @ shape_generators.s M x  
197 | pub fn overlapping_triangles(width: f32, height: f32, overlap: f32) -> (Vec<f32>, Vec<u32>) {  
198 |     let mut vertices: Vec<f32> = vec![0.; 3 * 3 * 3];  
199 |     let indicies: Vec<u32> = (0..9).collect();  
200 |  
201 |     let t_width: f32 = width / 2.;  
202 |     let t_height: f32 = height / 2.;  
203 |  
204 |     // Top left (Futherest)  
205 |     let futherest_z: f32 = 0.2;  
206 |     vertices[0] = -t_width / 2. + overlap / 2.;  
207 |     vertices[1] = t_height;  
208 |     vertices[2] = futherest_z;  
209 |     vertices[3] = -t_width + overlap / 2.;  
210 |     vertices[4] = 0.;  
211 |     vertices[5] = futherest_z;  
212 |     vertices[6] = 0. + overlap / 2.;  
213 |     vertices[7] = 0.;  
214 |     vertices[8] = futherest_z;  
215 |  
216 |     // Top right (Middle)  
217 |     let middle_z: f32 = 0.1;  
218 |     vertices[9] = t_width / 2. - overlap / 2.;  
219 |     vertices[10] = t_height;  
219 |  
219 | simple.vert M x  
src > @ mains.rs M x  
240 | let (vertices: Vec<f32>, indicies: Vec<u32>) = ShapeGenerator::overlapping_triangles(width: 1., height: 1., overlap: 0.5);  
241 | let mut colors: Vec<f32> = vec![0.; 9 * 4];  
242 | let alpha: f32 = 0.5;  
243 | for n: usize in 0..3 {  
244 |     // Create red, green and blue triangle  
245 |     let color: [f32; 4] = [  
246 |         if n == 0 { 1. } else { 0. },  
247 |         if n == 1 { 1. } else { 0. },  
248 |         if n == 2 { 1. } else { 0. },  
249 |         alpha,  
250 |     ];  
251 |     // Each triangle consist of 3 verticies, hence three splices  
252 |     colors.splice(range: (n * 12)..(n * 12 + 4), replace_with: color);  
253 |     colors.splice(range: (n * 12 + 4)..(n * 12 + 8), replace_with: color);  
254 |     colors.splice(range: (n * 12 + 8)..(n * 12 + 12), replace_with: color);  
255 | }  
256 |  
257 | println!("{} ", colors[colors.len() - 1]);  
258 |  
259 | let my_vao: u32 = unsafe { create_vao(vertices: &vertices, indicies: &indicies, &colors) };  
260 |  
261 | // == // Set up your shaders here
```

b)

i)

Picture in Task 2 a has blue closest, green in the middle, and red far behind.

Picture below has blue closest, then red and finally green.



```
123 | pub fn generate_square(side_length: f32) -> (Vec<f32>, Vec<u32>) {  
147 |     pub fn generate_sine(n_points: usize, width: f32, angle_rate: f32) -> (Vec<f32>, Vec<u32>) {  
  
warning: associated function `generate_sine` is never used  
--> src/shape_generator.rs:147:12  
147 |     pub fn generate_sine(n_points: usize, width: f32, angle_rate: f32) -> (Vec<f32>, Vec<u32>) {  
147 |           
147 |             pub fn generate_sine() generated 18 warnings  
147 |             Finished dev [unoptimized + debuginfo] target(s) in 2.68s  
147 |             Running "target/debug/glomm-rs"  
147 |             New window size! width: 800, height: 600  
147 |             AMD: AMD Radeon Graphics (rénoir, LLVM 15.0.7, DRM 3.52, 6.4.6-760604  
147 |             OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0~1689084530~22.04~06187  
147 |             GLSL   : 4.60  
147 |             0.5  
147 |             Resized  
147 |             New window size! width: 800, height: 637  
147 |             Resized  
147 |             New window size! width: 628, height: 512  
147 |             Resized  
147 |           
147 |     }  
147 | }  
  
src > @ shape_generators.s M x  
197 | pub fn overlapping_triangles(width: f32, height: f32, overlap: f32) -> (Vec<f32>, Vec<u32>) {  
198 |     let mut vertices: Vec<f32> = vec![0.; 3 * 3 * 3];  
199 |     let indicies: Vec<u32> = (0..9).collect();  
200 |  
201 |     let t_width: f32 = width / 2.;  
202 |     let t_height: f32 = height / 2.;  
203 |  
204 |     // Top left (Futherest)  
205 |     let futherest_z: f32 = 0.2;  
206 |     vertices[0] = -t_width / 2. + overlap / 2.;  
207 |     vertices[1] = t_height;  
208 |     vertices[2] = futherest_z;  
209 |     vertices[3] = -t_width + overlap / 2.;  
210 |     vertices[4] = 0.;  
211 |     vertices[5] = futherest_z;  
212 |     vertices[6] = 0. + overlap / 2.;  
213 |     vertices[7] = 0.;  
214 |     vertices[8] = futherest_z;  
215 |  
216 |     // Top right (Middle)  
217 |     let middle_z: f32 = 0.1;  
218 |     vertices[9] = t_width / 2. - overlap / 2.;  
219 |     vertices[10] = t_height;  
219 |  
219 | simple.vert M x  
src > @ mains.rs M x  
240 | let (vertices: Vec<f32>, indicies: Vec<u32>) = ShapeGenerator::overlapping_triangles(width: 1., height: 1., overlap: 0.5);  
241 | let mut colors: Vec<f32> = vec![0.; 9 * 4];  
242 | let alpha: f32 = 0.5;  
243 | for n: usize in 0..3 {  
244 |     // Create red, green and blue triangle  
245 |     let color: [f32; 4] = [  
246 |         if n == 1 { 1. } else { 0. },  
247 |         if n == 0 { 1. } else { 0. },  
248 |         if n == 2 { 1. } else { 0. },  
249 |         alpha,  
250 |     ];  
251 |     // Each triangle consist of 3 verticies, hence three splices  
252 |     colors.splice(range: (n * 12)..(n * 12 + 4), replace_with: color);  
253 |     colors.splice(range: (n * 12 + 4)..(n * 12 + 8), replace_with: color);  
254 |     colors.splice(range: (n * 12 + 8)..(n * 12 + 12), replace_with: color);  
255 | }  
256 |  
257 | println!("{} ", colors[colors.len() - 1]);  
258 |  
259 | let my_vao: u32 = unsafe { create_vao(vertices: &vertices, indicies: &indicies, &colors) };  
260 |  
261 | // == // Set up your shaders here
```

Notice how it has a more red-ish hue than the picture above. We can make it even more

red by moving it to the front:



The screenshot shows the Alasus IDE interface with several code files open in tabs. The main window displays a 3D rendering of three overlapping triangles. The green triangle is at the top, red is middle, and blue is bottom. The code files include:

- `shape_generators.rs`: Contains a function `generate_square` and an implementation of `ShapeGenerator`.
- `main.rs`: Contains the main application logic.
- `simple.vert`: The vertex shader source code.

```

123 pub fn generate_square(side_length: f32) -> (Vec<f32>, Vec<u32>) {
    ...
}
warning: associated function 'generate_sine' is never used
--> src/shape_generator.rs:147:12
147 pub fn generate_sine(n_points: usize, width: f32, angle_rate: f32) -> (Vec<f32>, Vec<u32>)
...
warning: 'gloom-rs' (bin "gloom-rs") generated 18 warnings
Finished dev [unoptimized + debuginfo] target(s) in 2.45s
Running `target/debug/gloom-rs`
New window size! width: 800, height: 600
AMD: AMD Radeon Graphics (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-760604
06-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0-1689084530-22.04-06187
46
GLSL   : 4.60
0.5
Resized
New window size! width: 800, height: 637
Resized
New window size! width: 628, height: 512
Resized

```

```

197 pub fn overlapping_triangles(width: f32, height: f32, overlap: f32) -> (Vec<f32>, Vec<u32>) {
    let mut vertices: Vec<f32> = vec![0.; 3 * 3 * 3];
    let indices: Vec<u32> = (0..9).collect();
    ...
    let t_width: f32 = width / 2.;
    let t_height: f32 = height / 2.;

    // Top left (Futherest)
    let furthest_z: f32 = 0.0;
    vertices[0] = -t_width / 2. + overlap / 2.0;
    vertices[1] = t_height;
    vertices[2] = furthest_z;
    vertices[3] = -t_width + overlap / 2.0;
    vertices[4] = 0.0;
    vertices[5] = furthest_z;
    vertices[6] = 0.0 + overlap / 2.0;
    vertices[7] = 0.0;
    vertices[8] = furthest_z;

    // Top right (Middle)
    let middle_z: f32 = 0.1;
    vertices[9] = t_width / 2. - overlap / 2.0;
    ...
    let indices10 = t_height;
}

240 let (vertices, indices) = ShapeGenerator::overlapping_triangles(width: 1., height: 1., overlap: 0.5);
241 let mut colors: Vec<f32> = vec![0.; 9 * 4];
242 let alpha: f32 = 0.5;
243 for n: usize in 0..3 {
    ...
    let color: [f32; 4] = [
        if n == 2 { 1. } else { 0. },
        if n == 0 { 1. } else { 0. },
        if n == 1 { 1. } else { 0. },
        alpha,
    ];
    ...
    colors.splice(range: (n * 12)..(n * 12 + 4), replace_with: color);
    colors.splice(range: (n * 12 + 4)..(n * 12 + 8), replace_with: color);
    colors.splice(range: (n * 12 + 8)..(n * 12 + 12), replace_with: color);
}
257
258 println!("{}", colors[colors.len() - 1]);
259
260 let my_vao: u32 = unsafe { create_vao(vertices: &vertices, indices: &indices, &colors) };
261

```

Note that the computer has to decide some order to apply the colors to each pixel. The way this is solved is by applying the color farthermost from the camera, then the middle, and finally closest. Since alpha is 50%, the closest color will contribute 50% to the final color, while the middle will contribute 50% to the colors behind it and so on, making the closest color most prominent.

ii)

Picture below is back to same situation as Task 2 a, but red and blue have swapped places in the z-buffer:



The screenshot shows the Alasus IDE interface with several code files open in tabs. The main window displays a 3D rendering of three overlapping triangles. The blue triangle is at the top, green is middle, and red is bottom. The code files include:

- `shape_generators.rs`: Contains a function `generate_square` and an implementation of `ShapeGenerator`.
- `main.rs`: Contains the main application logic.
- `simple.vert`: The vertex shader source code.

```

123 pub fn generate_square(side_length: f32) -> (Vec<f32>, Vec<u32>) {
    ...
}
warning: associated function 'generate_sine' is never used
--> src/shape_generator.rs:147:12
147 pub fn generate_sine(n_points: usize, width: f32, angle_rate: f32) -> (Vec<f32>, Vec<u32>)
...
warning: 'gloom-rs' (bin "gloom-rs") generated 18 warnings
Finished dev [unoptimized + debuginfo] target(s) in 2.14s
Running `target/debug/gloom-rs`
New window size! width: 800, height: 600
AMD: AMD Radeon Graphics (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-760604
06-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0-1689084530-22.04-06187
46
GLSL   : 4.60
0.5
Resized
New window size! width: 800, height: 637
Resized
New window size! width: 628, height: 512
Resized

```

```

204 let t_width: f32 = width / 2.0;
205 let t_height: f32 = height / 2.0;

206 // Top left (Futherest)
207 let furthest_z: f32 = 0.0;
208 vertices[0] = -t_width / 2.0 + overlap / 2.0;
209 vertices[1] = t_height;
210 vertices[2] = furthest_z;
211 vertices[3] = -t_width + overlap / 2.0;
212 vertices[4] = 0.0;
213 vertices[5] = furthest_z;
214 vertices[6] = 0.0 + overlap / 2.0;
215 vertices[7] = 0.0;
216 vertices[8] = furthest_z;

217 // Top right (Middle)
218 let middle_z: f32 = 0.1;
219 vertices[9] = t_width / 2.0 - overlap / 2.0;
220 ...
221 vertices[10] = t_height;
222 vertices[11] = middle_z;
223 vertices[12] = 0.0 - overlap / 2.0;
224 vertices[13] = 0.0;
225 vertices[14] = middle_z;

230 // TASK 2.3 b)
231 let (vertices, indices) = ShapeGenerator::generate_n_force(2, 1., 1.0);
232 let mut rng = rand::thread_rng();
233 let mut colors = vec![1.0];
234 for _ in 0..vertices.len() {
235     colors = vec![colors, vec![rng.gen(), rng.gen(), rng.gen(), 1.0].concat()];
236 }
237
238 // TASK 2.2 a) b)

239 let (vertices, indices) = ShapeGenerator::overlapping_triangles(width: 1., height: 1., overlap: 0.5);
240 let mut colors: Vec<f32> = vec![0.; 9 * 4];
241 let alpha: f32 = 0.5;
242 for n: usize in 0..3 {
    ...
    let color: [f32; 4] = [
        if n == 0 { 1. } else { 0. },
        if n == 1 { 1. } else { 0. },
        if n == 2 { 1. } else { 0. },
        alpha,
    ];
    ...
    colors.splice(range: (n * 12)..(n * 12 + 4), replace_with: color);
    colors.splice(range: (n * 12 + 4)..(n * 12 + 8), replace_with: color);
    colors.splice(range: (n * 12 + 8)..(n * 12 + 12), replace_with: color);
}
257
258 println!("{}", colors[colors.len() - 1]);
259
260 let my_vao: u32 = unsafe { create_vao(vertices: &vertices, indices: &indices, &colors) };
261

```

Where did the other colors go?

Here it's worth noting that the order of indices has not been updated. This means the top-left triangle is still the first entry, which implicitly makes it the first triangle drawn. The computer will then use depth culling to save computing the triangles below, creating no transparency-effect :(

Task 3

b)

Image below shows how i went forward displaying each modification.

The screenshot shows a development environment with two main panes. The left pane is a terminal window displaying the following output:

```
src/shape_generator.rs:123:12
123     pub fn generate_square(side_length: f32) -> (Vec<f32>, Vec<
u32>) {
    ^~~~~~
warning: associated function `generate_sine` is never used
--> src/shape_generator.rs:147:12
147     pub fn generate_sine(n_points: usize, width: f32, angle_rate: f32) -> (Vec<f32>, Vec<u32>)
    ^~~~~~
warning: `gloom-rs` (bin "gloom-rs") generated 18 warnings
Finished dev [unoptimized + debuginfo] target(s) in 0.05s
Running "target/debug/gloom-rs"
AMD: AMD Radeon Graphics (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-768664
06-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-pp0~1689084530~22.04~06187
46
GLSL  : 4.60
0.5
New window size! width: 800, height: 600
Resized
New window size! width: 800, height: 637
New window size! width: 628, height: 512
Resized
```

The right pane is a code editor for a GLSL shader named `simplevert.glsl`. The code is as follows:

```
#version 430 core
in layout(location=0) vec3 position;
in layout(location=1) vec4 vertex_color;
out vec4 fragment_color;
mat4x4 matrix = mat4(1);
uniform float time;
void main()
{
    matrix[0][0] = mod(time, 3.0f) / 3.0f; // a
    matrix[1][0] = mod(time, 3.0f) / 3.0f; // b
    matrix[2][0] = mod(time, 3.0f) / 3.0f; // c
    matrix[0][1] = mod(time, 3.0f) / 3.0f; // d
    matrix[1][1] = mod(time, 3.0f) / 3.0f; // e
    matrix[2][1] = mod(time, 3.0f) / 3.0f; // f
    fragment_color = vertex_color;
    gl_Position = matrix * vec4(position, 1.0f);
}
```

A preview window at the bottom shows a 3D scene with three colored triangles (red, green, blue) against a dark background. A status bar at the bottom right indicates a GLSL Lint error: "GLSL Lint: Failed to spawn glslangvalidator binary. Error: spawn glslangvalidator failed".

- a: Scaling along x-axis

File Edit Selection View Go Run Terminal Help

simple.vert M x @ main.rs *

shaders > simple.vert

```

1 #version 430 core
2
3 in layout(location=0) vec3 position;
4 in layout(location=1) vec4 vertex_color;
5 out vec4 fragment_color;
6
7 mat4x4 matrix = mat4(1);
8 uniform float time;
9
10 void main()
11 {
12     // matrix[0][0] = mod(time, 3.0f) / 3.0f; // a
13     matrix[1][0] = mod(time, 3.0f) / 3.0f; // b
14     // matrix[2][0] = mod(time, 3.0f) / 3.0f; // c
15     // matrix[0][1] = mod(time, 3.0f) / 3.0f; // d
16     // matrix[1][1] = mod(time, 3.0f) / 3.0f; // e
17     // matrix[3][1] = mod(time, 3.0f) / 3.0f; // f
18     fragment_color = vertex_color;
19     gl_Position = matrix * vec4(position, 1.0f);
20 }

```

You, 1 second ago | 2 authors (You and others)

Running "target/debug/gloom-rs"

New window size! width: 800, height: 600
AMD: AMD Radeon Graphics (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-766664
06-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pp08-1689084530-22.04-06187
46
GLSL : 4.60
0.5
Resized Assignment 1.mft
New window size! width: 800, height: 637
New window size! width: 628, height: 512
Resized

GLSL Lint: Failed to spawn "glslangvalidator" binary. Error: spawn glsl...

- b: Shearing x with respect to y-axis

File Edit Selection View Go Run Terminal Help

simple.vert M x @ main.rs *

shaders > simple.vert

```

1 #version 430 core
2
3 in layout(location=0) vec3 position;
4 in layout(location=1) vec4 vertex_color;
5 out vec4 fragment_color;
6
7 mat4x4 matrix = mat4(1);
8 uniform float time;
9
10 void main()
11 {
12     // matrix[0][0] = mod(time, 3.0f) / 3.0f; // a
13     // matrix[1][0] = mod(time, 3.0f) / 3.0f; // b
14     matrix[3][0] = mod(time, 3.0f) / 3.0f; // c
15     // matrix[2][0] = mod(time, 3.0f) / 3.0f; // d
16     // matrix[0][1] = mod(time, 3.0f) / 3.0f; // e
17     // matrix[1][1] = mod(time, 3.0f) / 3.0f; // f
18     fragment_color = vertex_color;
19     gl_Position = matrix * vec4(position, 1.0f);
20 }

```

You, 1 second ago | 2 authors (You and others)

Running "target/debug/gloom-rs"

New window size! width: 800, height: 600
AMD: AMD Radeon Graphics (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-766664
06-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pp08-1689084530-22.04-06187
46
GLSL : 4.60
0.5
Resized Assignment 1.mft
New window size! width: 800, height: 637
New window size! width: 628, height: 512
Resized

GLSL Lint: Failed to spawn "glslangvalidator" binary. Error: spawn glsl...

- c: Translation along x-axis

File Edit Selection View Go Run Terminal Help

simple.vert M x @ mains.rs

shaders > E simplevert

```

1 #version 430 core
2
3 in layout(location=0) vec3 position;
4 in layout(location=1) vec4 vertex_color;
5 out vec4 fragment_color;
6
7 mat4x4 matrix = mat4(1);
8 uniform float time;
9
10 void main()
11 {
12     // matrix[0][0] = mod(time, 3.0f) / 3.0f; // a
13     // matrix[1][0] = mod(time, 3.0f) / 3.0f; // b
14     // matrix[3][0] = mod(time, 3.0f) / 3.0f; // c
15     matrix[0][1] = mod(time, 3.0f) / 3.0f; // d
16     // matrix[1][1] = mod(time, 3.0f) / 3.0f; // e
17     // matrix[3][1] = mod(time, 3.0f) / 3.0f; // f
18     fragment_color = vertex_color;
19     gl_Position = matrix * vec4(position, 1.0f);
20 }

```

You, 1 second ago | 2 authors (You and others)

GLSL Lint: Failed to spawn 'glslangvalidator' binary. Error: spawn glsl...

- d: Shearing y with respect to x-axis

File Edit Selection View Go Run Terminal Help

simple.vert M x @ mains.rs

shaders > E simplevert

```

1 #version 430 core
2
3 in layout(location=0) vec3 position;
4 in layout(location=1) vec4 vertex_color;
5 out vec4 fragment_color;
6
7 mat4x4 matrix = mat4(1);
8 uniform float time;
9
10 void main()
11 {
12     // matrix[0][0] = mod(time, 3.0f) / 3.0f; // a
13     // matrix[1][0] = mod(time, 3.0f) / 3.0f; // b
14     // matrix[3][0] = mod(time, 3.0f) / 3.0f; // c
15     matrix[0][1] = mod(time, 3.0f) / 3.0f; // d
16     matrix[1][1] = mod(time, 3.0f) / 3.0f; // e
17     // matrix[3][1] = mod(time, 3.0f) / 3.0f; // f
18     fragment_color = vertex_color;
19     gl_Position = matrix * vec4(position, 1.0f);
20 }

```

You, 1 second ago | 2 authors (You and others)

GLSL Lint: Failed to spawn 'glslangvalidator' binary. Error: spawn glsl...

- e: Scaling along y-axis

The screenshot shows a development environment with the following components:

- Terminal:** Displays build logs with warnings and information about the OpenGL context and GLSL version.
- Code Editor:** Shows a GLSL vertex shader named `simple.vert`. The code defines a matrix transformation over time and outputs vertex color.
- 3D Preview:** A window showing a 3D scene with a red, green, and blue triangle.
- Status Bar:** Shows a message about failing to spawn the glslangvalidator binary.

```

123 pub fn generate_square(side_length: f32) -> (Vec<f32>, Vec<u32>) {
    ...
    warning: associated function `generate_sine` is never used
    ...
147 pub fn generate_sine(n_points: usize, width: f32, angle_rate: f32) -> (Vec<f32>, Vec<u32>)
    ...

warning: 'gloom-rs' (bin "gloom-rs") generated 18 warnings
Finished dev [unoptimized + debuginfo] target(s) in 0.06s
Running `target/debug/gloom-rs`

New window size! width: 800, height: 600
AMD: AMD Radeon Graphics (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-760664
06-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pp08-1689084530-22.04-06187
46
GLSL : 4.60
0.0
Resized
New window size! width: 800, height: 637
Resized
New window size! width: 628, height: 512
Resized

```

```

File Edit Selection View Go Run Terminal Help
File simple.vert main.rs
shaders > E simple.vert
You, 1 second ago | 2 authors (You and others)
#version 430 core
in layout(location=0) vec3 position;
in layout(location=1) vec4 vertex_color;
out vec4 fragment_color;
mat4x4 matrix = mat4(1);
uniform float time;
void main()
{
    // matrix[0][0] = mod(time, 3.0f) / 3.0f; // a
    // matrix[1][0] = mod(time, 3.0f) / 3.0f; // b
    // matrix[3][0] = mod(time, 3.0f) / 3.0f; // c
    // matrix[0][1] = mod(time, 3.0f) / 3.0f; // d
    // matrix[1][1] = mod(time, 3.0f) / 3.0f; // e
    matrix[3][1] = mod(time, 3.0f) / 3.0f; // f
    fragment_color = vertex_color;
    gl_Position = matrix * vec4(position, 1.0f);
}

```

- f: Translation along y-axis

c)

A rotation requires modification of both x and y, but since we always changed a single value of the identity matrix, it would only modify a single coordinate axis, making a rotation impossible.

Task 4

c)

(Fancy) Camera and keyboard movement implemented :)

- Uses the recommended keys for movement

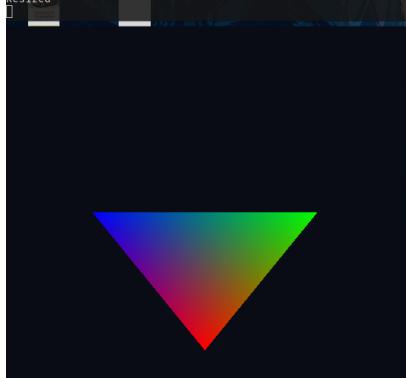
Task 5

a) Implemented ;)

b)

Created a triangle where the blue and green edge is faaaaaar away from the camera:

Without perspective: `noperspective`



The screenshot shows a development environment with several tabs open. On the left, the terminal window displays the command `cargo run` and its output, which includes warnings about overlapping triangles and the generation of a binary named "gloom-rs". The main area shows the code for the vertex and fragment shaders. The vertex shader uses the `noperspective` keyword, and the fragment shader simply outputs the vertex color. The resulting triangle is a flat, multi-colored shape (blue, green, red) with no depth gradient.

```

147 pub fn generate_sine_n_points: usize, width: f32, angle_rate: f32) -> (Vec<f32>, Vec<u32>) {
    ...
}
warning: associated function `overlapping_triangles` is never used
--> src/shape_generator.rs:197:12
197 pub fn overlapping_triangles(width: f32, height: f32, overlap: f32) -> (Vec<f32>, Vec<u32>) {
    ...
}

warning: 'gloom-rs` (bin "gloom-rs") generated 17 warnings
Finished dev [unoptimized + debuginfo] target(s) in 0.06s
Running target/debug/gloom-rs
AMD: AMD Radeon Graphics (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-76060406-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0-1689084530-22.0
4-0618746
GLSL : 4.60
Resized
New window size! width: 800, height: 600
Resized
New window size! width: 564, height: 512
Resized

```

```

simple.vert M x
shaders > simple.vert
You, 3 minutes ago | 2 authors (You and others)
1 #version 430 core
2
3 in layout(location=0) vec3 position;
4 in layout(location=1) vec4 vertex_color;
5 out noperspective vec4 fragment_color;
6
7 uniform mat4x4 matrix;
8 uniform float time;
9
10 void main()
11 {
12     fragment_color = vertex_color;
13     gl_Position = matrix * vec4(position, 1.0f);
14 }

simple.frag M x
shaders > simple.frag
You, 3 seconds ago | 2 authors (Michael Gimle and others)
1 #version 430 core
2
3 in noperspective vec4 fragment_color;
4 out vec4 color;
5
6 void main()
7 {
8     color = fragment_color;
9 }

shape_generators.h M x
src > @ shape_generators.rs () impl ShapeGenerator > flat_thing
145     fn overlapping_triangles()
146         pub fn flat_thing() -> (Vec<f32>, Vec<u32>) []
147             let mut vertices: Vec<f32> = vec![0., 3 * 3];
148             let indicies: Vec<u32> = (0..3).collect();
149
150             vertices[0] = 0.;
151             vertices[1] = -0.5;
152             vertices[2] = 1.;
153
154             vertices[3] = 20.;
155             vertices[4] = -0.5;
156             vertices[5] = 50.; You, 2 minutes ago + Uncommitted changes
157
158             vertices[6] = -20.;
159             vertices[7] = -0.5;
160             vertices[8] = 50.;

161             (vertices, indicies)
162         } impl ShapeGenerator
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264 } impl ShapeGenerator
265

```

With perspective: `smooth`



The screenshot shows a development environment with several tabs open. The code is identical to the previous one, but the vertex shader uses the `smooth` keyword instead of `noperspective`. The resulting triangle has a smooth color gradient from blue at the top to red at the bottom, indicating it is facing upwards. A warning message in the terminal indicates that the GLSL Lint failed to spawn 'glslangvalidator' binary.

```

147 pub fn generate_sine_n_points: usize, width: f32, angle_rate: f32) -> (Vec<f32>, Vec<u32>) {
    ...
}
warning: associated function `overlapping_triangles` is never used
--> src/shape_generator.rs:197:12
197 pub fn overlapping_triangles(width: f32, height: f32, overlap: f32) -> (Vec<f32>, Vec<u32>) {
    ...
}

warning: 'gloom-rs` (bin "gloom-rs") generated 17 warnings
Finished dev [unoptimized + debuginfo] target(s) in 0.06s
Running target/debug/gloom-rs
AMD: AMD Radeon Graphics (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-76060406-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0-1689084530-22.0
4-0618746
GLSL : 4.60
Resized
New window size! width: 800, height: 600
Resized
New window size! width: 564, height: 512
Resized

```

```

simple.vert M x
shaders > simple.vert
You, 1 second ago | 2 authors (You and others)
1 #version 430 core
2
3 in layout(location=0) vec3 position;
4 in layout(location=1) vec4 vertex_color;
5 out smooth vec4 fragment_color;
6
7 uniform mat4x4 matrix;
8 uniform float time;
9
10 void main()
11 {
12     fragment_color = vertex_color;
13     gl_Position = matrix * vec4(position, 1.0f);
14 }

simple.frag M x
shaders > simple.frag
You, 2 seconds ago | 2 authors (Michael Gimle and others)
1 #version 430 core
2
3 in smooth vec4 fragment_color;
4 out vec4 color;
5
6 void main()
7 {
8     color = fragment_color;
9 }

shape_generators.h M x
src > @ shape_generators.rs () impl ShapeGenerator > flat_thing
145     fn overlapping_triangles()
146         pub fn flat_thing() -> (Vec<f32>, Vec<u32>) []
147             let mut vertices: Vec<f32> = vec![0., 3 * 3];
148             let indicies: Vec<u32> = (0..3).collect();
149
150             vertices[0] = 0.;
151             vertices[1] = -0.5;
152             vertices[2] = 1.;

153             vertices[3] = 20.;
154             vertices[4] = -0.5;
155             vertices[5] = 50.;

156             vertices[6] = -20.;
157             vertices[7] = -0.5;
158             vertices[8] = 50.;

159             (vertices, indicies)
160         } impl ShapeGenerator
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264 } impl ShapeGenerator
265

```

Notice the dominating red color. when `noperspective` is used, the color between the edges are interpolated without considering how close the vertex is to the camera. This creates the illusion that the triangle is facing the camera. The picture below uses `smooth` which takes this into consideration, creating the illusion that the large triangle is facing upwards, and therefore a more correct color interpolation.