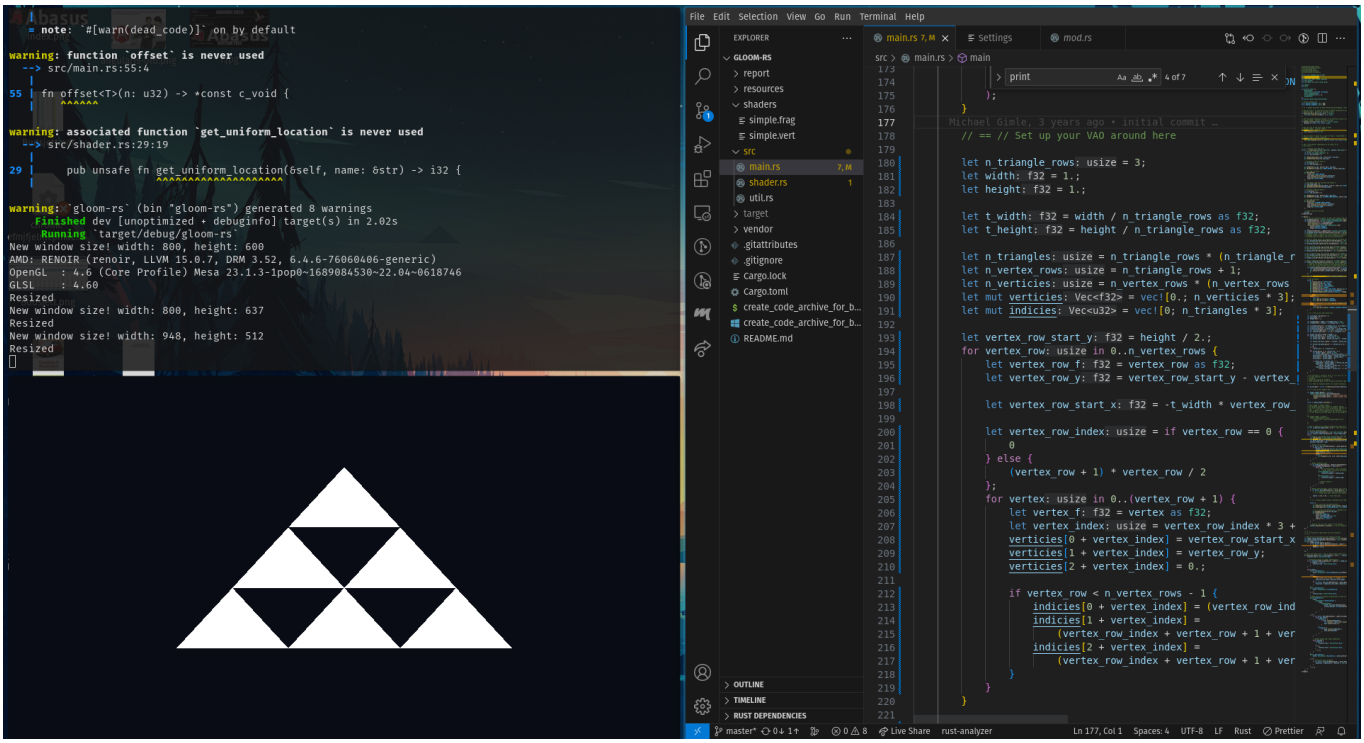
Assignment 1

Peter Johan Flått-Bjørnstad (Individual, Group 32)

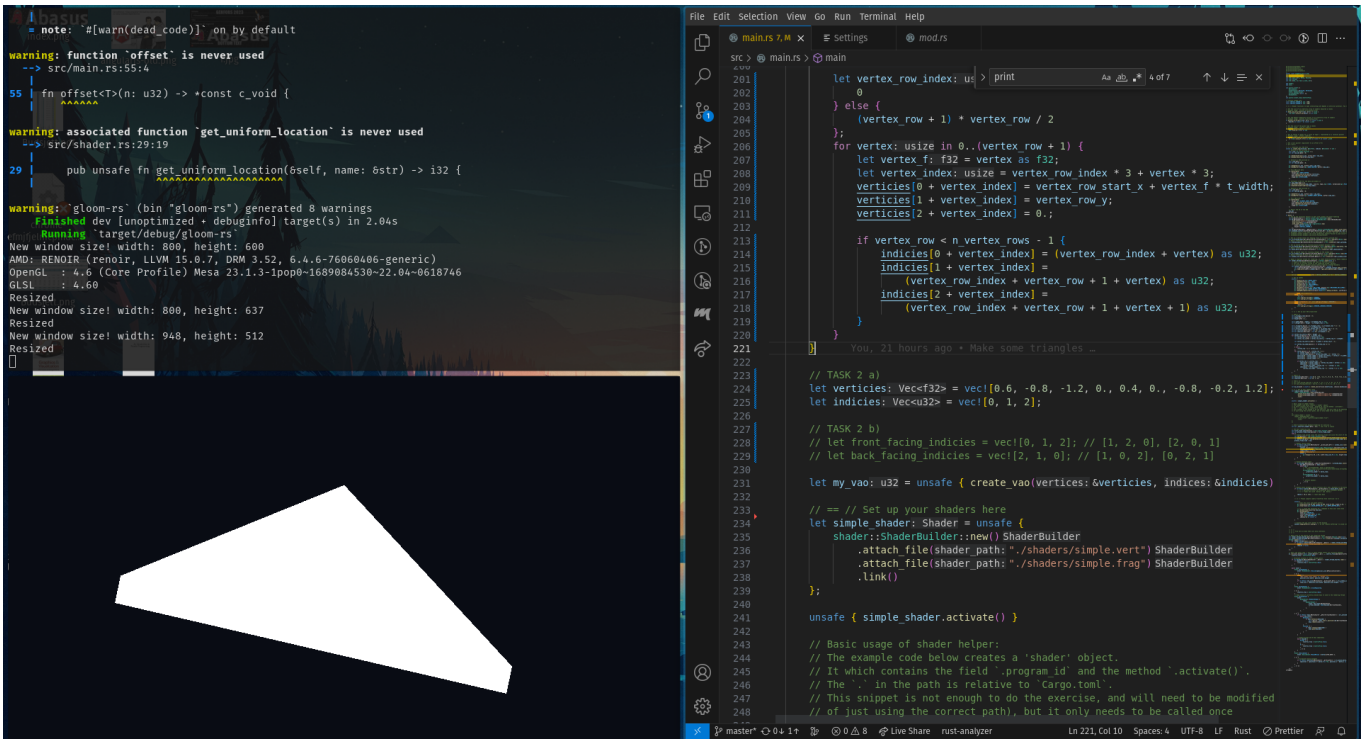
Task 1

c)



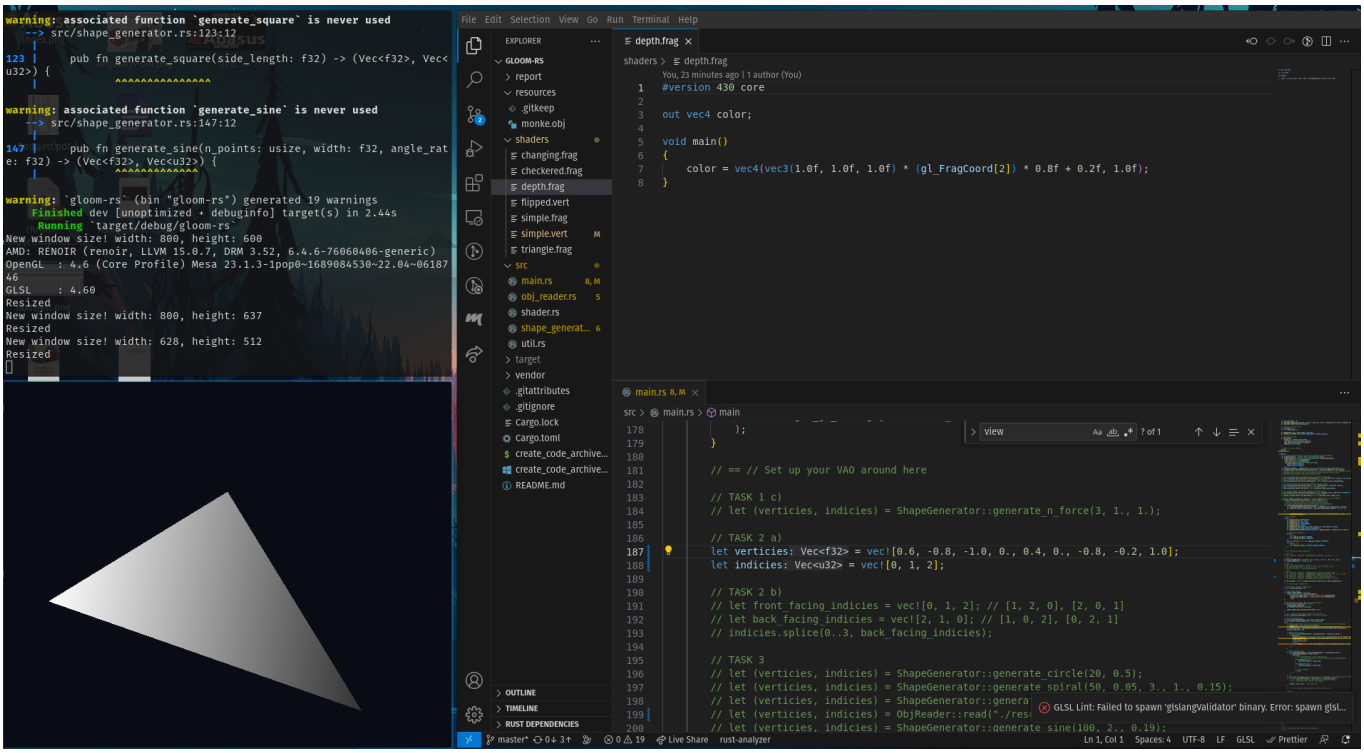
Task 2

a)



Clipping

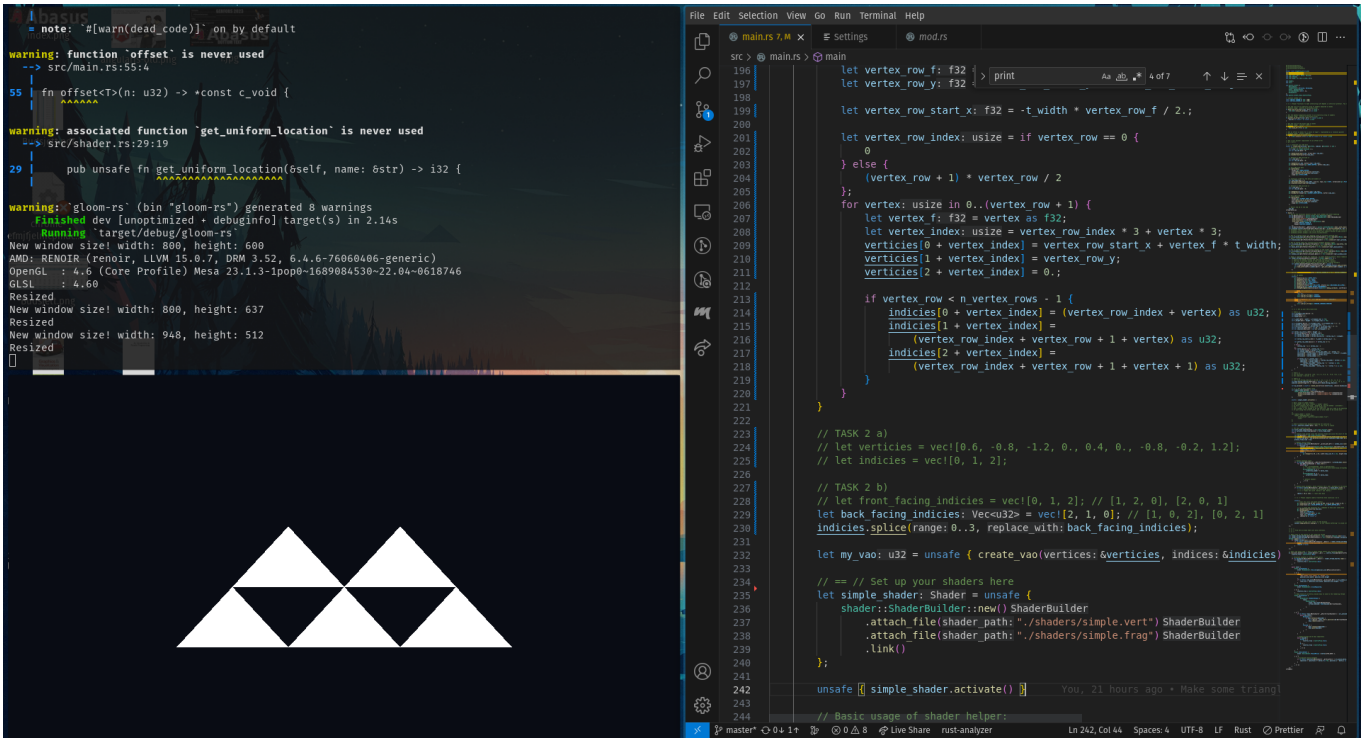
Parts of the object is outside the clipping object, Which by default is orthographic → a centered cube with sides of 2. Note that the only values outside the cube is the z-values, meaning it is clipping the near and far plane. Changing the z-values to 1 and -1 verifies this.



(Also added a depth-info in the fragment shader to more easily see its orientation)

The purpose of the clipping object is to save computing power by not rendering parts of the object which is not visible

b)

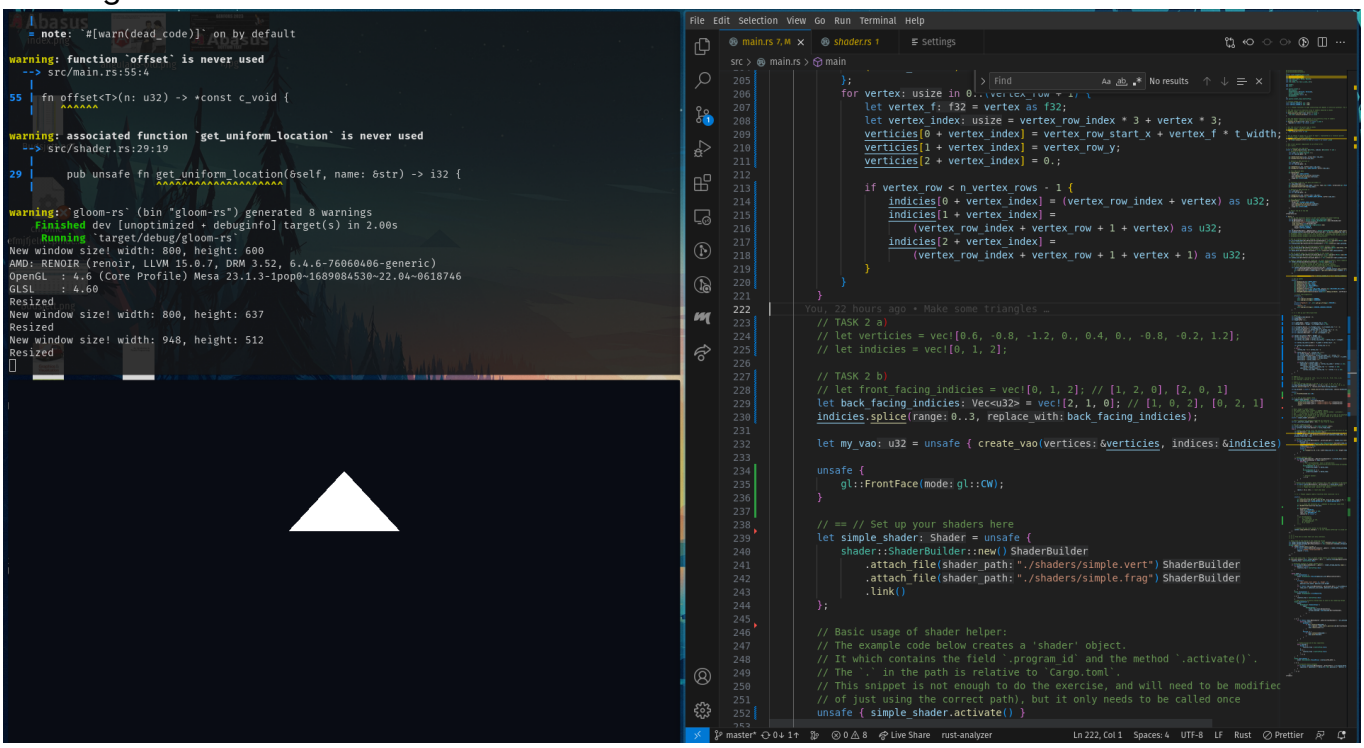


i) The polygon which we changed order of became invisible.

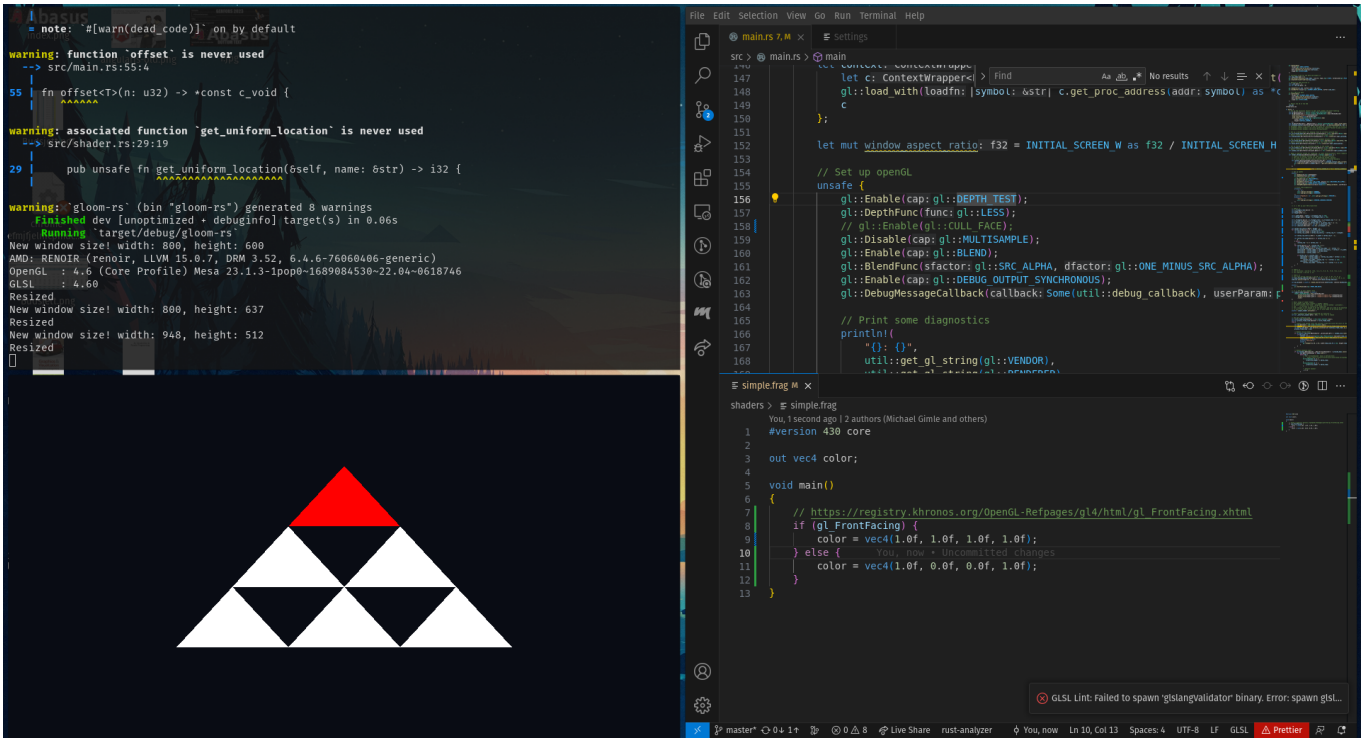
ii) First it is nice to know that indicies 0, 1, 2 maps to the topmost triangle, in the order of top, bottom-left and bottom-right. Notice how the indicies originally was ordered as 0, 1,

2, but changing the order to 2, 1, 0 made it invisible. Other patterns which hides it is 1, 0, 2 and 0, 2, 1, while 1, 2, 0 and 2, 0, 1 made it visible.

iii) All visible patterns traverses the polygon anti-clockwise in respect to the window, while the other patterns travel clockwise. We can easily verify this by changing opengls winding order to clockwise:



We could also disable face-culling and modify the fragment-shader to show a different color on the back-face (winding=CCW):



c)

i) Not strictly necessary in a still-scene, but in a moving scene, the distance objects have to the camera will change. This distance is stored in the depth-buffer and helps determine which fragment overlap each other. Not clearing this buffer would mean all polygon renders will order the pixels in regards to the first render, and we would get incorrect overlapping of primitives.

ii) Multiple renderings of a single pixel will happen whenever there is overlapping geometry, since the fragment shader will run on the geometry even though different geometry is covering it on the given pixel.

iii)

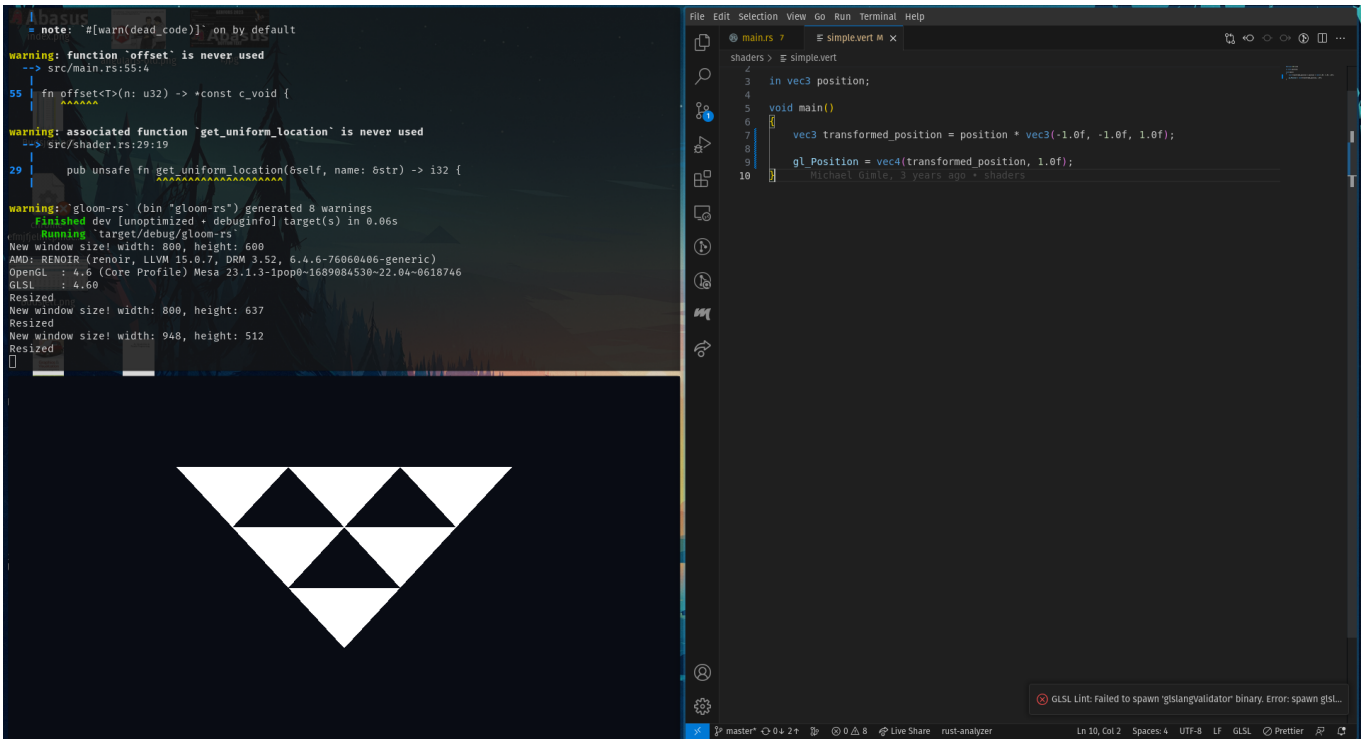
- Vertex shader
  - Maps verticies to desired projection, e.g. perspective projection.
- Fragment shader
  - Map vertices and their connections to colored pixels on the screen, also called rasterization

iv) Relying on the order of vertex buffer means we have to re-specify positions which have more than one connection, quickly increasing the amount of space used on duplicate data. Refering by indicies solves this by not having to repeat position-data, and instead only repeating the index-position of the verticies.

v) If our VAO stores different type of data in each entry (e.g. vertex position and then texture position data), it makes it easy to retrieve only the texture position data. It could therefore be interpreted as some type of offset.

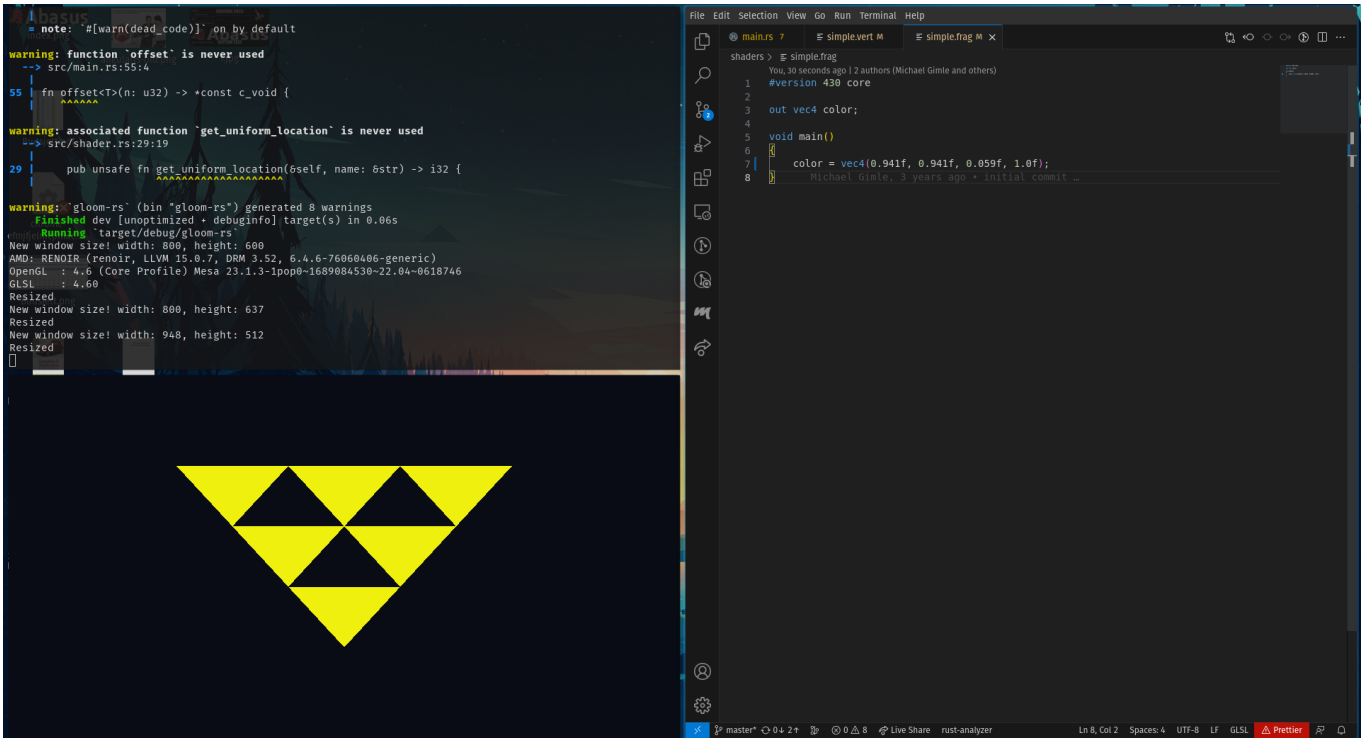
d)

i)
Solved by multiplying position with vector that has negative x and y in the vertex shader.



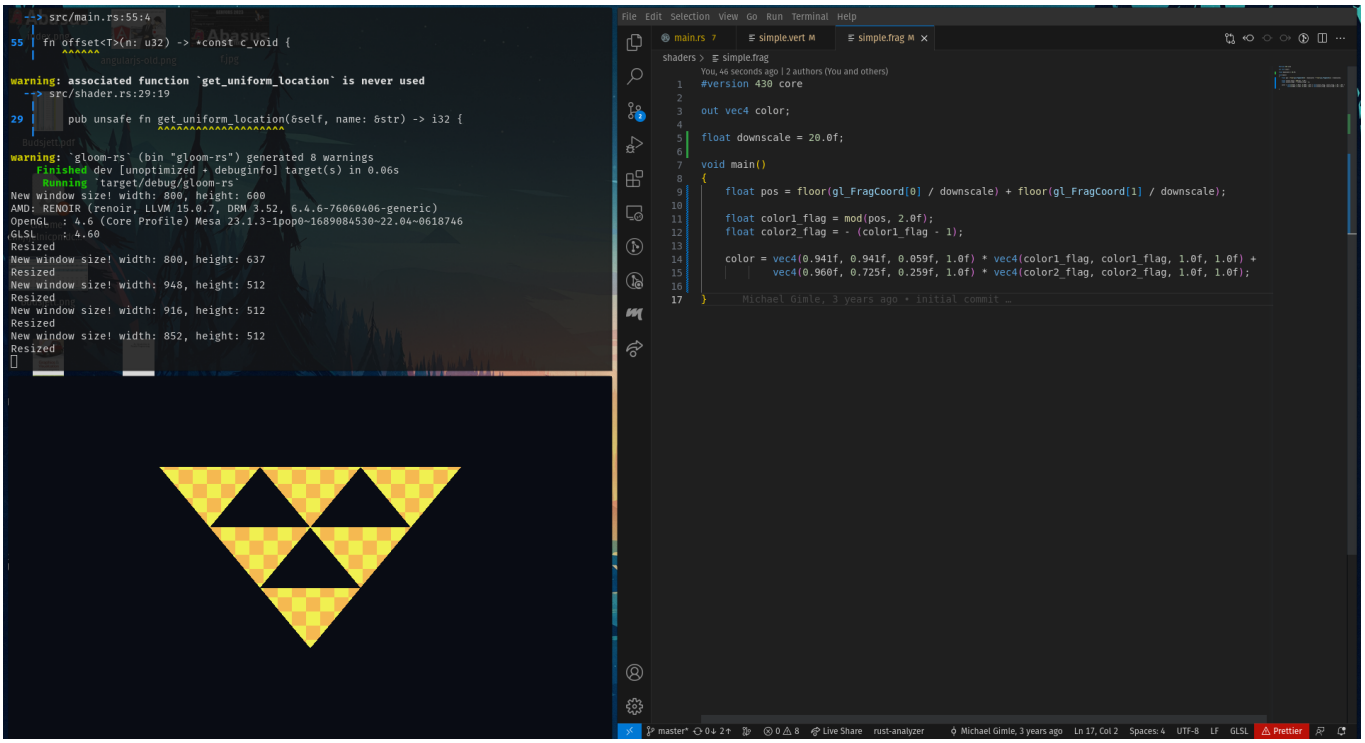
ii)

Solved by modifying the rgb-floats in the fragment shader

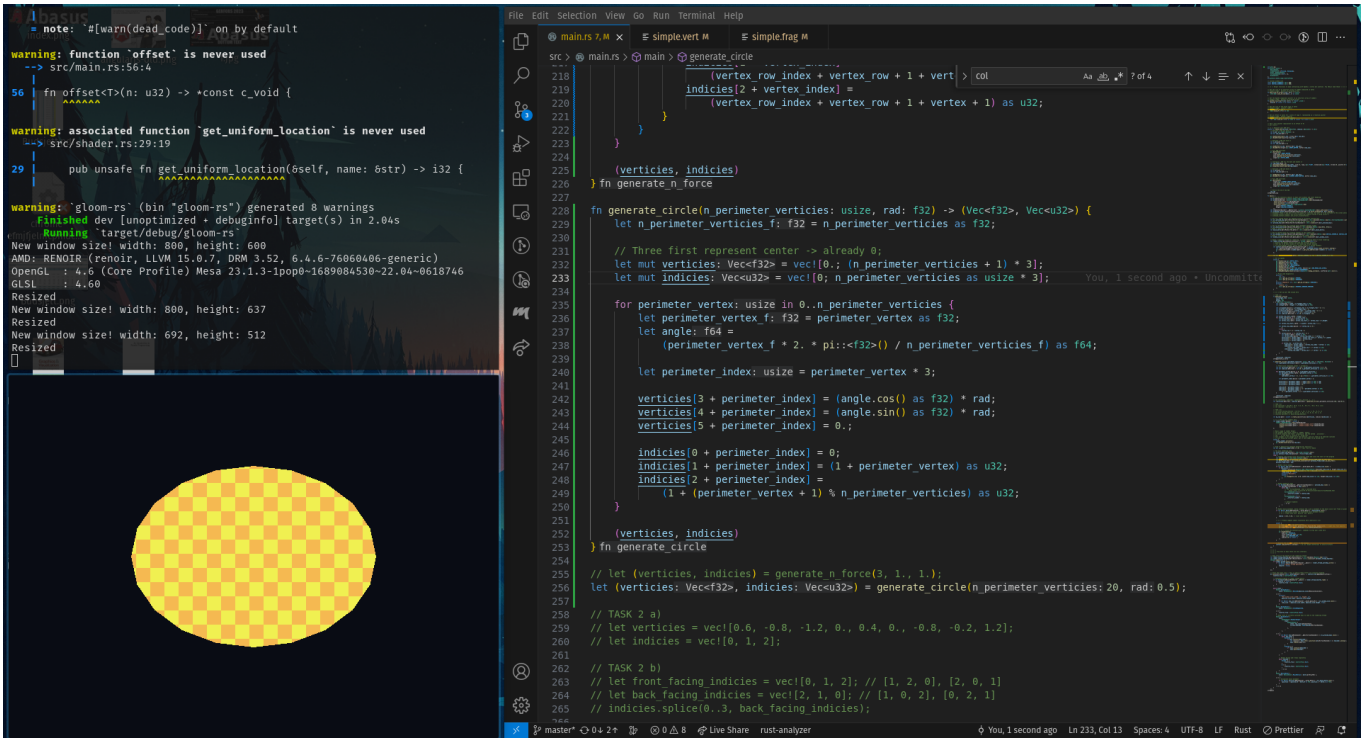


Task 3

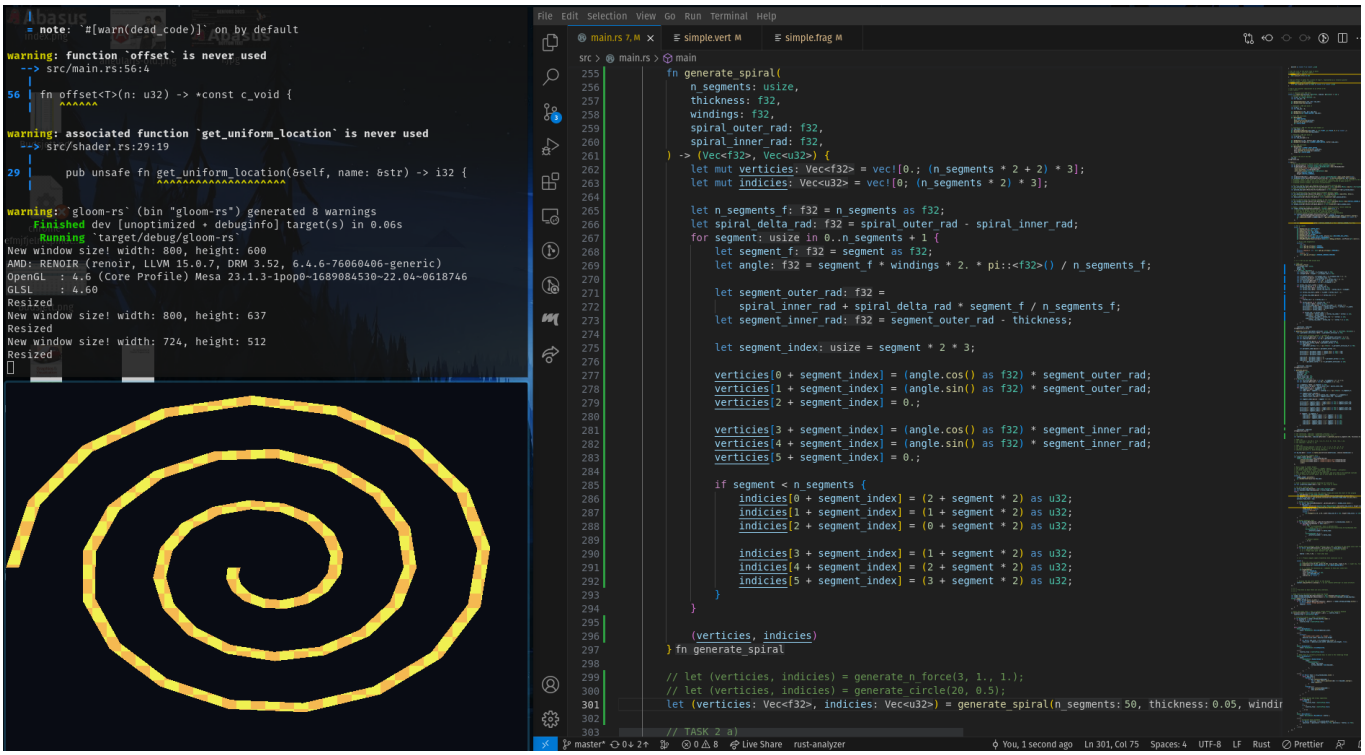
a)


--> src/main.rs:55:4
55 | fn offset<T>(n: u32) -> *const c_void {
warning: associated function `get_uniform_location` is never used
--> src/shader.rs:29:19
29 | pub unsafe fn get_uniform_location(&self, name: &str) -> i32 {
warning: `gloom-rs` (bin "gloom-rs") generated 8 warnings
Finished dev [unoptimized + debuginfo] target(s) in 0.06s
Running `target/debug/gloom-rs`
New window size! width: 800, height: 600
AMD: RENOIR (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-76060406-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0~1689084530~22.04~0618746
GLSL : 4.60
Resized
New window size! width: 800, height: 637
Resized
New window size! width: 948, height: 512
Resized
New window size! width: 916, height: 512
Resized
New window size! width: 852, height: 512
Resized
File Edit Selection View Go Run Terminal Help
main.rs 7
simple.vert M
simple.frag M
shaders > simple.frag
You, 46 seconds ago | 2 authors (You and others)
#version 430 core
out vec4 color;
float downscale = 20.0f;
void main()
{
float pos = floor(gl_FragCoord[0] / downscale) + floor(gl_FragCoord[1] / downscale);
float color1_flag = mod(pos, 2.0f);
float color2_flag = - (color1_flag - 1);
color = vec4(0.941f, 0.941f, 0.059f, 1.0f) * vec4(color1_flag, color1_flag, 1.0f, 1.0f) +
vec4(0.960f, 0.725f, 0.259f, 1.0f) * vec4(color2_flag, color2_flag, 1.0f, 1.0f);
}
Michael Gimle, 3 years ago • initial commit …
master* 0↓ 2↑
0 8
Live Share
rust-analyzer
Michael Gimle, 3 years ago
Ln 17, Col 2
Spaces: 4
UTF-8
LF
GLSL
Prettier

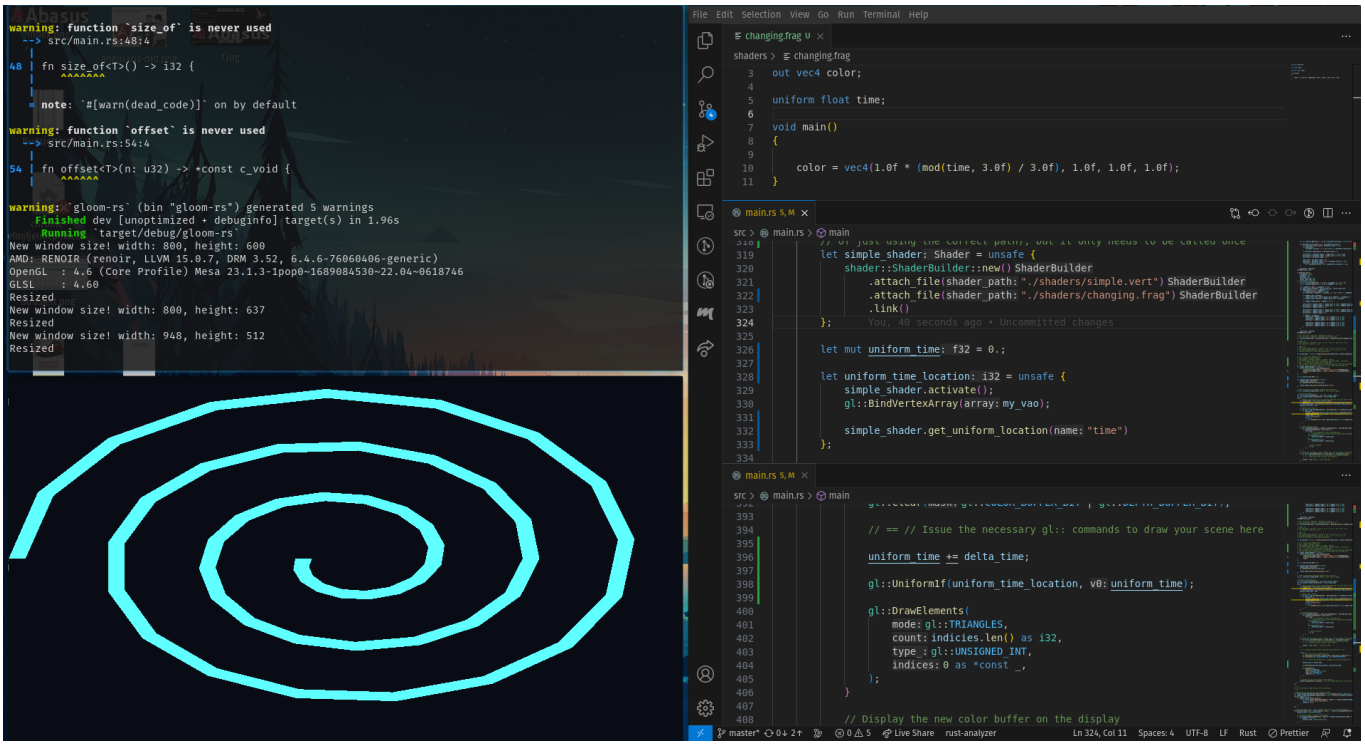
b)


= note: `#[warn(dead_code)]` on by default
warning: function `offset` is never used
--> src/main.rs:56:4
56 | fn offset<T>(n: u32) -> *const c_void {
warning: associated function `get_uniform_location` is never used
--> src/shader.rs:29:19
29 | pub unsafe fn get_uniform_location(&self, name: &str) -> i32 {
warning: `gloom-rs` (bin "gloom-rs") generated 8 warnings
Finished dev [unoptimized + debuginfo] target(s) in 2.04s
Running `target/debug/gloom-rs`
New window size! width: 800, height: 600
AMD: RENOIR (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-76060406-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0~1689084530~22.04~0618746
GLSL : 4.60
Resized
New window size! width: 800, height: 637
Resized
New window size! width: 692, height: 512
Resized
File Edit Selection View Go Run Terminal Help
main.rs 7, M
simple.vert M
simple.frag M
src > main.rs > main > generate_circle
(vertex_row_index + vertex_row + 1 + vert
col
? of 4
indicies[2 + vertex_index] =
(vertex_row_index + vertex_row + 1 + vertex + 1) as u32;
(verticies, indicies)
} fn generate_n_force
fn generate_circle(n_perimeter_verticies: usize, rad: f32) -> (Vec<f32>, Vec<u32>) {
let n_perimeter_verticies_f: f32 = n_perimeter_verticies as f32;
// Three first represent center -> already 0;
let mut verticies: Vec<f32> = vec![0.; (n_perimeter_verticies + 1) * 3];
let mut indicies: Vec<u32> = vec![0; n_perimeter_verticies as usize * 3];
You, 1 second ago • Uncommitt
for perimeter_vertex: usize in 0..n_perimeter_verticies {
let perimeter_vertex_f: f32 = perimeter_vertex as f32;
let angle: f64 =
(perimeter_vertex_f * 2. * pi::<f32>() / n_perimeter_verticies_f) as f64;
let perimeter_index: usize = perimeter_vertex * 3;
verticies[3 + perimeter_index] = (angle.cos() as f32) * rad;
verticies[4 + perimeter_index] = (angle.sin() as f32) * rad;
verticies[5 + perimeter_index] = 0.;
indicies[0 + perimeter_index] = 0;
indicies[1 + perimeter_index] = (1 + perimeter_vertex) as u32;
indicies[2 + perimeter_index] =
(1 + (perimeter_vertex + 1) % n_perimeter_verticies) as u32;
(verticies, indicies)
} fn generate_circle
// let (verticies, indicies) = generate_n_force(3, 1., 1.);
let (verticies: Vec<f32>, indicies: Vec<u32>) = generate_circle(n_perimeter_verticies: 20, rad: 0.5);
// TASK 2 a)
// let verticies = vec![0.6, -0.8, -1.2, 0., 0.4, 0., -0.8, -0.2, 1.2];
// let indicies = vec![0, 1, 2];
// TASK 2 b)
// let front_facing_indicies = vec![0, 1, 2]; // [1, 2, 0], [2, 0, 1]
// let back_facing_indicies = vec![2, 1, 0]; // [1, 0, 2], [0, 2, 1]
// indicies.splice(0..3, back_facing_indicies);
master* 0↓ 2↑
0 8
Live Share
rust-analyzer
You, 1 second ago
Ln 233, Col 13
Spaces: 4
UTF-8
LF
Rust
Prettier

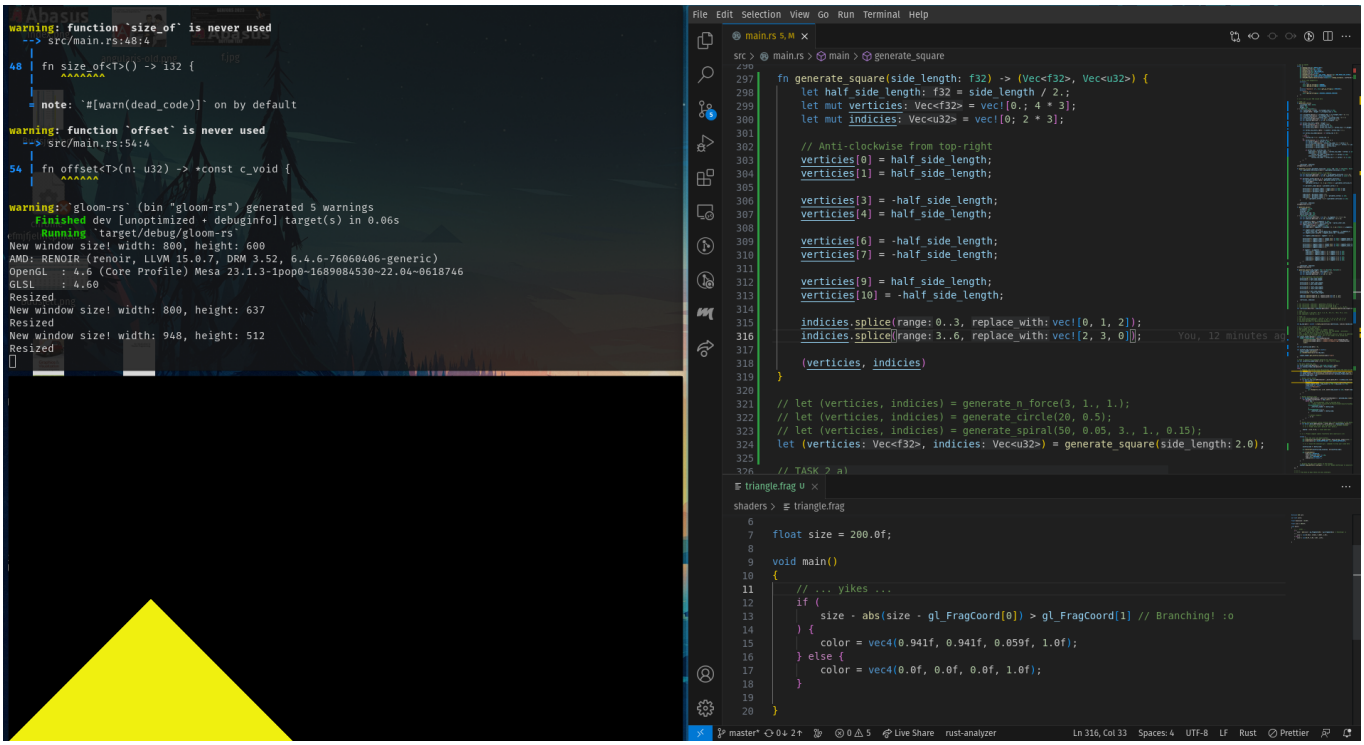
c)


= note: `#[warn(dead_code)]` on by default
warning: function `offset` is never used
--> src/main.rs:56:4
56 | fn offset<T>(n: u32) -> *const c_void {
warning: associated function `get_uniform_location` is never used
--> src/shader.rs:29:19
29 | pub unsafe fn get_uniform_location(&self, name: &str) -> i32 {
warning: `gloom-rs` (bin "gloom-rs") generated 8 warnings
Finished dev [unoptimized + debuginfo] target(s) in 0.06s
Running `target/debug/gloom-rs`
New window size! width: 800, height: 600
AMD: RENOIR (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-76060406-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0~1689084530~22.04~0618746
GLSL : 4.60
Resized
New window size! width: 800, height: 637
Resized
New window size! width: 724, height: 512
Resized
File Edit Selection View Go Run Terminal Help
main.rs 7, M
simple.vert M
simple.frag M
src > main.rs > main
fn generate_spiral(
n_segments: usize,
thickness: f32,
windings: f32,
spiral_outer_rad: f32,
spiral_inner_rad: f32,
) -> (Vec<f32>, Vec<u32>) {
let mut vertices: Vec<f32> = vec![0.; (n_segments * 2 + 2) * 3];
let mut indicies: Vec<u32> = vec![0; (n_segments * 2) * 3];
let n_segments_f: f32 = n_segments as f32;
let spiral_delta_rad: f32 = spiral_outer_rad - spiral_inner_rad;
for segment: usize in 0..n_segments + 1 {
let segment_f: f32 = segment as f32;
let angle: f32 = segment_f * windings * 2. * pi::<f32>() / n_segments_f;
let segment_outer_rad: f32 =
spiral_inner_rad + spiral_delta_rad * segment_f / n_segments_f;
let segment_inner_rad: f32 = segment_outer_rad - thickness;
let segment_index: usize = segment * 2 * 3;
vertices[0 + segment_index] = (angle.cos() as f32) * segment_outer_rad;
vertices[1 + segment_index] = (angle.sin() as f32) * segment_outer_rad;
vertices[2 + segment_index] = 0.;
vertices[3 + segment_index] = (angle.cos() as f32) * segment_inner_rad;
vertices[4 + segment_index] = (angle.sin() as f32) * segment_inner_rad;
vertices[5 + segment_index] = 0.;
if segment < n_segments {
indicies[0 + segment_index] = (2 + segment * 2) as u32;
indicies[1 + segment_index] = (1 + segment * 2) as u32;
indicies[2 + segment_index] = (0 + segment * 2) as u32;
indicies[3 + segment_index] = (1 + segment * 2) as u32;
indicies[4 + segment_index] = (2 + segment * 2) as u32;
indicies[5 + segment_index] = (3 + segment * 2) as u32;
}
}
(vertices, indicies)
} fn generate_spiral
// let (vertices, indicies) = generate_n_force(3, 1., 1.);
// let (vertices, indicies) = generate_circle(20, 0.5);
let (vertices: Vec<f32>, indicies: Vec<u32>) = generate_spiral(n_segments: 50, thickness: 0.05, windi
// TASK 2 a)
master* 0↓ 2↑
0 8
Live Share
rust-analyzer
You, 1 second ago
Ln 301, Col 75
Spaces: 4
UTF-8
LF
Rust
Prettier

d)


warning: function `size_of` is never used
--> src/main.rs:48:4
48 | fn size_of<T>() -> i32 {
= note: `#[warn(dead_code)]` on by default
warning: function `offset` is never used
--> src/main.rs:54:4
54 | fn offset<T>(n: u32) -> *const c_void {
warning: `gloom-rs` (bin "gloom-rs") generated 5 warnings
Finished dev [unoptimized + debuginfo] target(s) in 1.96s
Running `target/debug/gloom-rs`
New window size! width: 800, height: 600
AMD: RENOIR (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-76060406-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0~1689084530~22.04~0618746
GLSL : 4.60
Resized
New window size! width: 800, height: 637
Resized
New window size! width: 948, height: 512
Resized
File Edit Selection View Go Run Terminal Help
changing.frag U
shaders > changing.frag
out vec4 color;
uniform float time;
void main()
{
color = vec4(1.0f * (mod(time, 3.0f) / 3.0f), 1.0f, 1.0f, 1.0f);
}
main.rs 5, M
src > main.rs > main
let simple_shader: Shader = unsafe {
shader::ShaderBuilder::new().ShaderBuilder
.attach_file(shader_path: "./shaders/simple.vert").ShaderBuilder
.attach_file(shader_path: "./shaders/changing.frag").ShaderBuilder
.link()
};
You, 30 seconds ago • Uncommitted changes
let mut uniform_time: f32 = 0.;
let uniform_time_location: i32 = unsafe {
simple_shader.activate();
gl::BindVertexArray(array: my_vao);
simple_shader.get_uniform_location(name: "time")
};
// == // Issue the necessary gl:: commands to draw your scene here
uniform_time += delta_time;
gl::Uniform1f(uniform_time_location, v0: uniform_time);
gl::DrawElements(
mode: gl::TRIANGLES,
count: indicies.len() as i32,
type_: gl::UNSIGNED_INT,
indices: 0 as *const _,
);
// Display the new color buffer on the display
master* 0↓ 2↑
0 5
Live Share
rust-analyzer
Ln 324, Col 11
Spaces: 4
UTF-8
LF
Rust
Prettier

e)


warning: function `size_of` is never used
--> src/main.rs:48:4
48 | fn size_of<T>() -> i32 {
= note: `#[warn(dead_code)]` on by default
warning: function `offset` is never used
--> src/main.rs:54:4
54 | fn offset<T>(n: u32) -> *const c_void {
warning: `gloom-rs` (bin "gloom-rs") generated 5 warnings
Finished dev [unoptimized + debuginfo] target(s) in 0.06s
Running `target/debug/gloom-rs`
New window size! width: 800, height: 600
AMD: RENOIR (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-76060406-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0~1689084530~22.04~0618746
GLSL : 4.60
Resized
New window size! width: 800, height: 637
Resized
New window size! width: 948, height: 512
Resized
File Edit Selection View Go Run Terminal Help
main.rs 5, M
fn generate_square(side_length: f32) -> (Vec<f32>, Vec<u32>) {
let half_side_length: f32 = side_length / 2.;
let mut verticies: Vec<f32> = vec![0.; 4 * 3];
let mut indicies: Vec<u32> = vec![0; 2 * 3];
// Anti-clockwise from top-right
verticies[0] = half_side_length;
verticies[1] = half_side_length;
verticies[3] = -half_side_length;
verticies[4] = half_side_length;
verticies[6] = -half_side_length;
verticies[7] = -half_side_length;
verticies[9] = half_side_length;
verticies[10] = -half_side_length;
indicies.splice(range: 0..3, replace_with: vec![0, 1, 2]);
indicies.splice(range: 3..6, replace_with: vec![2, 3, 0]);
(verticies, indicies)
}
// let (verticies, indicies) = generate_n_force(3, 1., 1.);
// let (verticies, indicies) = generate_circle(20, 0.5);
// let (verticies, indicies) = generate_spiral(50, 0.05, 3., 1., 0.15);
let (verticies: Vec<f32>, indicies: Vec<u32>) = generate_square(side_length: 2.0);
// TASK 2 a)
triangle.frag U
shaders > triangle.frag
float size = 200.0f;
void main()
{
// ... yikes ...
if (
size - abs(size - gl_FragCoord[0]) > gl_FragCoord[1] // Branching! :o
) {
color = vec4(0.941f, 0.941f, 0.059f, 1.0f);
} else {
color = vec4(0.0f, 0.0f, 0.0f, 1.0f);
}
}
master* 0↓ 2↑
0 5
Live Share
rust-analyzer
Ln 316, Col 33
Spaces: 4
UTF-8
LF
Rust
Prettier

f)

depth.frag U

```
out vec4 color;

void main()
{
    color = vec4(vec3(1.0f, 1.0f, 1.0f) * (gl_FragCoord[2]) * 0.8f + 0.2f, 1.0f);
}
```

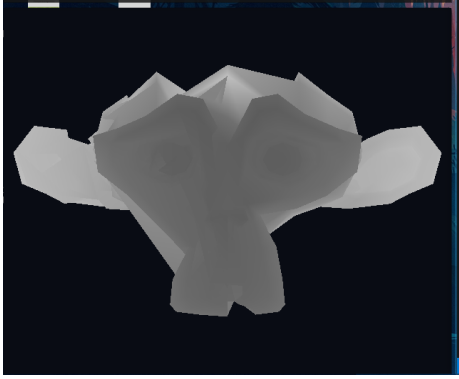
obj_reader.rs U

```
pub fn compile_obj(self, obj_src: &str) -> (Vec<f32>, Vec<u32>) {
    let mut vertices: Vec<f32> = vec![];
    let mut indicies: Vec<u32> = vec![];

    for line: &str in obj_src.lines() {
        let lexemes: Vec<&str> = line.split_whitespace().collect();

        match ObjEntry::from_code(&lexemes[0]) {
            Ok(ObjEntry::Face) => indicies.append(
                &mut lexemes[1..] [&str]
                    .iter() Iter<'_, &str>
                    .map(|&lex: &str| lex.parse().unwrap()) impl Iterator<Item = u32>
                    .collect(),
            ),
            Ok(ObjEntry::Vertex) => vertices.append(
                &mut lexemes[1..] [&str]
                    .iter() Iter<'_, &str>
                    .map(|&lex: &str| lex.parse().unwrap()) impl Iterator<Item = f32>
                    .collect(),
            ),
            Ok(ObjEntry::Comment) => (),
            Err(s: String) => {
                eprintln!("Unknown entry: {}", s)
            }
        }
    }

    (verticies, indicies)
} fn compile_obj
} impl ObjReader
```

obj_reader.rs U src/obj_reader.rs/ { } impl ObjReader/ compile_obj

```
use std::path::Path;

1 implementation
pub struct ObjReader {}

1 implementation
pub enum ObjEntry {
    Vertex,
    Face,
    Comment,
}

impl ObjEntry {
    fn from_code(code: &str) -> Result<ObjEntry, String> {
        match code {
            "v" => Ok(ObjEntry::Vertex),
            "f" => Ok(ObjEntry::Face),
            "#" => Ok(ObjEntry::Comment),
            e: &str => Err(e.to_string()),
        }
    }
}

impl ObjReader {
    pub fn new() -> ObjReader {
        ObjReader {}
    }

    pub fn read(self, obj_path: &str) -> (Vec<f32>, Vec<u32>) {
        let path: &Path = Path::new(obj_path);
        if let Some(extension: &OsStr) = path.extension() {
            match extension.to_str().expect(msg: "Failed to read extension.") {
                "obj" => Ok(()),
                e: &str => Err(e.to_string()),
            } Result<(), String>
            .expect(msg: "Failed to parse extension.");

            let obj_src: String = std::fs::read_to_string(path) Result<String, Error>
                .expect(msg: &format!("Failed to read shader source. {}", obj_path));

            self.compile_obj(&obj_src)
        } else {
            panic!("Failed to read extension of file with path: {}", obj_path);
        }
    }
```

```
  = note: `#[warn(dead_code)]` on by default

warning: function `offset` is never used
  --> src/main.rs:55:4
   |
55 | fn offset<T>(n: u32) -> *const c_void {
   |    ^^^^^^

warning: `gloom-rs` (bin "gloom-rs") generated 5 warnings
    Finished dev [unoptimized + debuginfo] target(s) in 0.07s
     Running `target/debug/gloom-rs`
New window size! width: 800, height: 600
AMD: RENOIR (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-76060406-generic)
OpenGL  : 4.6 (Core Profile) Mesa 23.1.3-1pop0~1689084530~22.04~06187
46
GLSL    : 4.60
Unknown entry: o
Unknown entry: s
Resized
New window size! width: 800, height: 637
Resized
New window size! width: 692, height: 512
Resized
New window size! width: 628, height: 512
Resized
^C
```



File Edit Selection View Go Run Terminal Help

depth.frag U

shaders > depth.frag

```
out vec4 color;

void main()
{
    color = vec4(vec3(1.0f, 1.0f, 1.0f) * (gl_FragCoord[2]) * 0.8f + 0.2f, 1.0f);
```

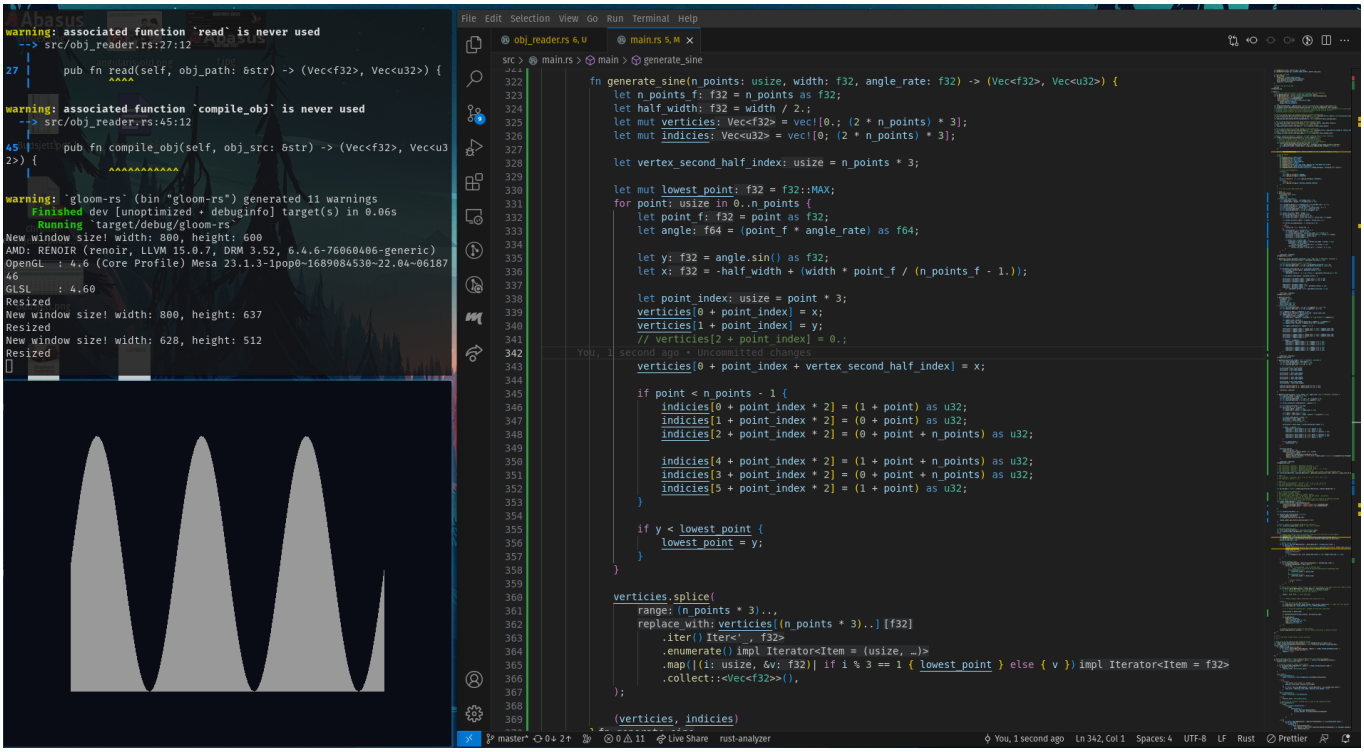
obj_reader.rs U | main.rs 5, M

src > main.rs > ...

```
// let (verticies, indicies) = generate_n_force(3, 1., 1.);
// let (verticies, indicies) = generate_circle(20, 0.5);
// let (verticies, indicies) = generate_spiral(50, 0.05, 3., 1., 0.15);
// let (verticies, indicies) = generate_square(2.0);
let (verticies: Vec<f32>, indicies: Vec<u32>) = obj_reader::ObjReader::new().read(obj_path: "./resources/monke.obj");

// TASK 2 a)
// let verticies = vec![0.6, -0.8, -1.2, 0., 0.4, 0., -0.8, -0.2, 1.2];
// let indicies = vec![0, 1, 2];

// TASK 2 b)
// let front_facing_indicies = vec![0, 1, 2]; // [1, 2, 0], [2, 0, 1]
// let back_facing_indicies = vec![2, 1, 0]; // [1, 0, 2], [0, 2, 1]
// indicies.splice(0..3, back_facing_indicies);

let my_vao: u32 = unsafe { create_vao(vertices: &verticies, indices: &indicies) };

// == // Set up your shaders here
// Basic usage of shader helper:
// The example code below creates a 'shader' object.
// It which contains the field `.program_id` and the method `.activate()`.
// The `.` in the path is relative to `Cargo.toml`.
// This snippet is not enough to do the exercise, and will need to be modified (outside
// of just using the correct path), but it only needs to be called once
let simple_shader: Shader = unsafe {
    shader::ShaderBuilder::new() ShaderBuilder
        .attach_file(shader_path: "./shaders/simple.vert") ShaderBuilder
        .attach_file(shader_path: "./shaders/depth.frag") ShaderBuilder
        .link()
};

let mut uniform_time: f32 = 0.;

let uniform_time_location: i32 = unsafe {
    simple_shader.activate();
    gl::BindVertexArray(array: my_vao);

    simple_shader.get_uniform_location(name: "time")
};
```

master* 0↓ 2↑ | 0 5 | Live Share | rust-analyzer | Peder Bergebakken Sundt, 23 months ago | Ln 1, Col 1 | Spaces: 4 | UTF-8 | LF | Rust | Prettier

g)


warning: associated function `read` is never used
--> src/obj_reader.rs:27:12
pub fn read(self, obj_path: &str) -> (Vec<f32>, Vec<u32>) {
warning: associated function `compile_obj` is never used
--> src/obj_reader.rs:45:12
pub fn compile_obj(self, obj_src: &str) -> (Vec<f32>, Vec<u32>) {
warning: `gloom-rs` (bin "gloom-rs") generated 11 warnings
Finished dev [unoptimized + debuginfo] target(s) in 0.06s
Running `target/debug/gloom-rs`
New window size! width: 800, height: 600
AMD: RENOIR (renoir, LLVM 15.0.7, DRM 3.52, 6.4.6-76060406-generic)
OpenGL : 4.6 (Core Profile) Mesa 23.1.3-1pop0~1689084530~22.04~06187
46
GLSL : 4.60
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
obj_reader.rs 6, U
main.rs 5, M
src > main.rs > main > generate_sine
fn generate_sine(n_points: usize, width: f32, angle_rate: f32) -> (Vec<f32>, Vec<u32>) {
let n_points_f: f32 = n_points as f32;
let half_width: f32 = width / 2.;
let mut verticies: Vec<f32> = vec![0.; (2 * n_points) * 3];
let mut indicies: Vec<u32> = vec![0; (2 * n_points) * 3];
let vertex_second_half_index: usize = n_points * 3;
let mut lowest_point: f32 = f32::MAX;
for point: usize in 0..n_points {
let point_f: f32 = point as f32;
let angle: f64 = (point_f * angle_rate) as f64;
let y: f32 = angle.sin() as f32;
let x: f32 = -half_width + (width * point_f / (n_points_f - 1.));
let point_index: usize = point * 3;
verticies[0 + point_index] = x;
verticies[1 + point_index] = y;
// verticies[2 + point_index] = 0.;
You, 1 second ago • Uncommitted changes
verticies[0 + point_index + vertex_second_half_index] = x;
if point < n_points - 1 {
indicies[0 + point_index * 2] = (1 + point) as u32;
indicies[1 + point_index * 2] = (0 + point) as u32;
indicies[2 + point_index * 2] = (0 + point + n_points) as u32;
indicies[4 + point_index * 2] = (1 + point + n_points) as u32;
indicies[3 + point_index * 2] = (0 + point + n_points) as u32;
indicies[5 + point_index * 2] = (1 + point) as u32;
}
if y < lowest_point {
lowest_point = y;
}
}
verticies.splice(
range: (n_points * 3)..,
replace_with: verticies[(n_points * 3)..][f32]
.iter() Iter<'_, f32>
.enumerate() impl Iterator<Item = (usize, …)>
.map(|(i: usize, &v: f32)| if i % 3 == 1 { lowest_point } else { v }) impl Iterator<Item = f32>
.collect::<Vec<f32>>(),
);
(verticies, indicies)
master* 0↓ 2↑
0 11
Live Share
rust-analyzer
You, 1 second ago
Ln 342, Col 1
Spaces: 4
UTF-8
LF
Rust
Prettier