MEWO

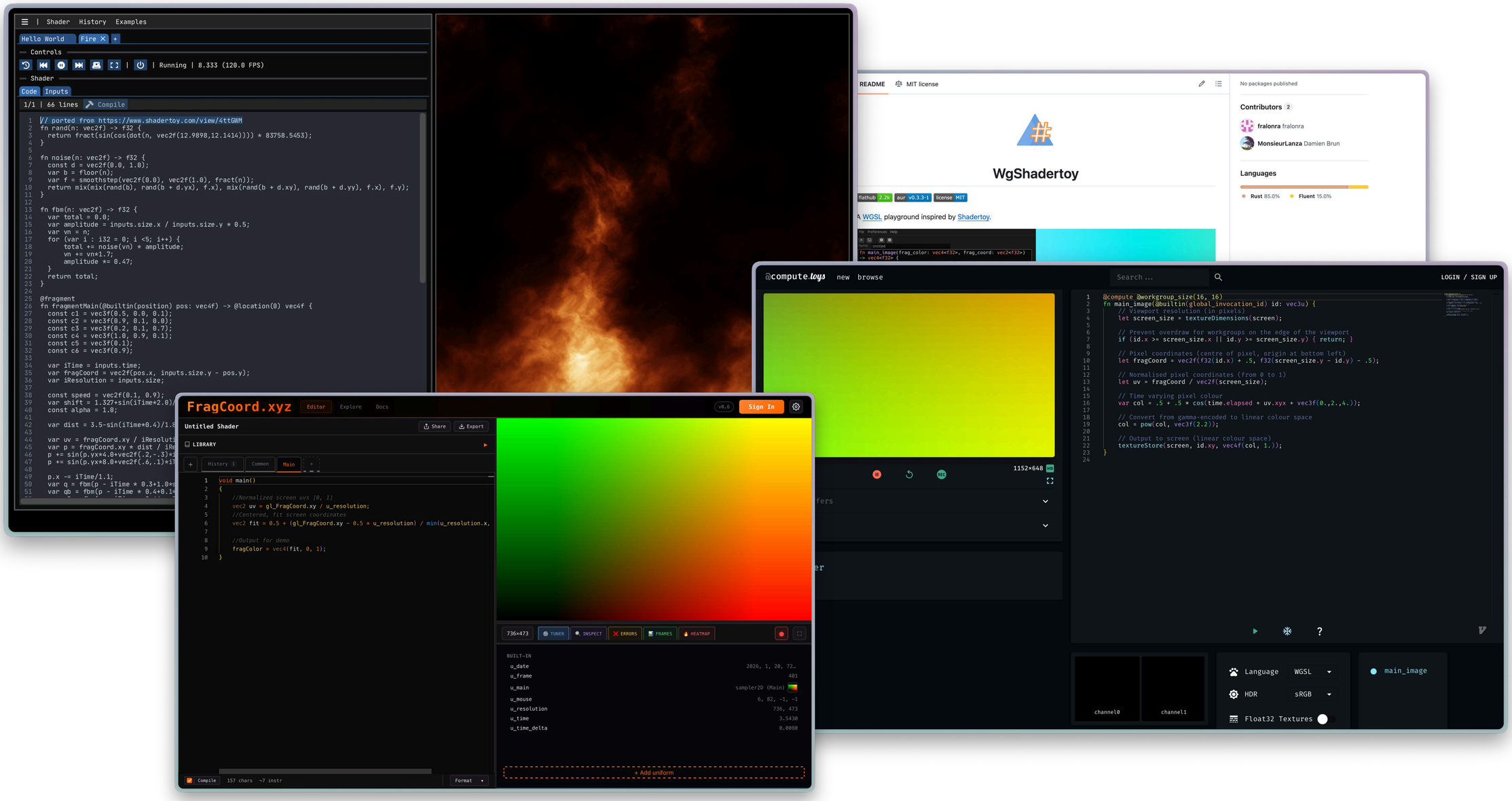
alpha

DMD Senior Project

Charles Wang

*not the final logo

*this presentation was made in 15 minutes

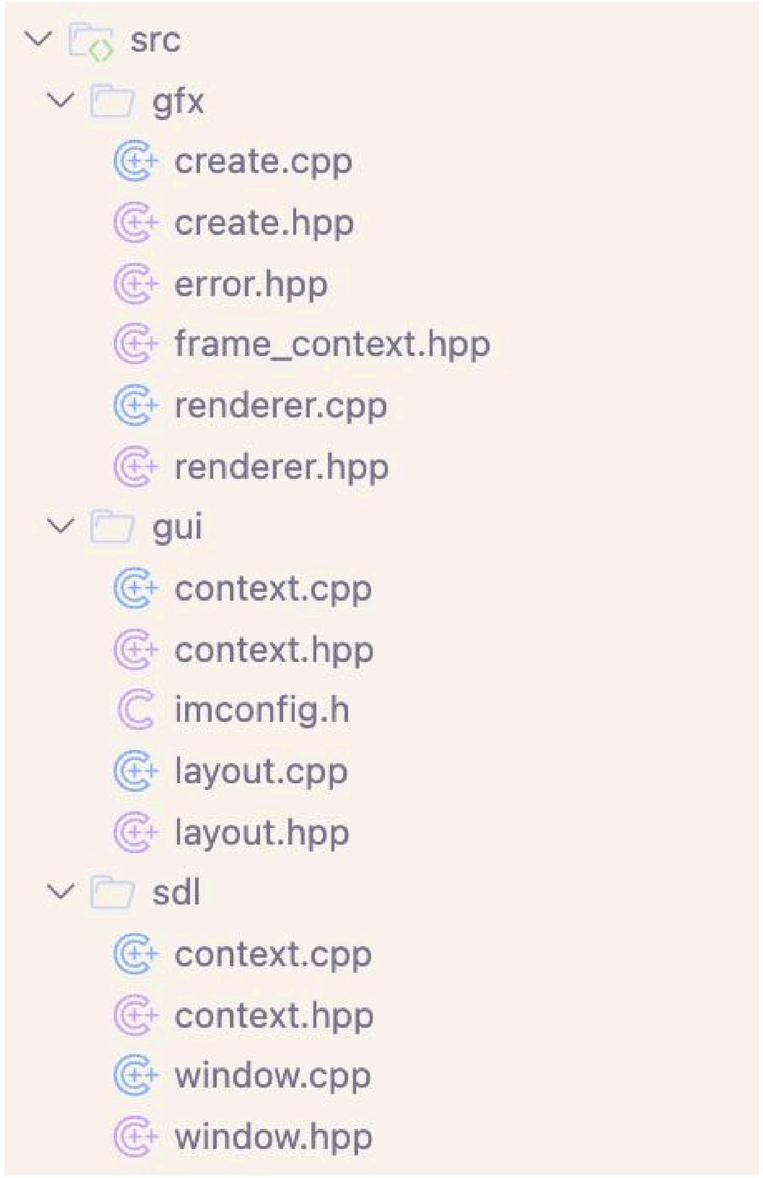

Shader History Examples
Hello World
Fire
Controls
Running | 8.333 (120.0 FPS)
Shader
Code Inputs
1/1 | 66 lines Compile
// ported from https://www.shadertoy.com/view/4ttGWM
fn rand(n: vec2f) -> f32 {
return fract(sin(cos(dot(n, vec2f(12.9898,12.1414)))) * 83758.5453);
}
fn noise(n: vec2f) -> f32 {
const d = vec2f(0.0, 1.0);
var b = floor(n);
var f = smoothstep(vec2f(0.0), vec2f(1.0), fract(n));
return mix(mix(rand(b), rand(b + d.yx), f.x), mix(rand(b + d.xy), rand(b + d.yy), f.x), f.y);
}
fn fbm(n: vec2f) -> f32 {
var total = 0.0;
var amplitude = inputs.size.x / inputs.size.y * 0.5;
var vn = n;
for (var i : i32 = 0; i <5; i++) {
total += noise(vn) * amplitude;
vn *= vn*1.7;
amplitude *= 0.47;
}
return total;
}
@fragment
fn fragmentMain(@builtin(position) pos: vec4f) -> @location(0) vec4f {
const c1 = vec3f(0.5, 0.0, 0.1);
const c2 = vec3f(0.9, 0.1, 0.0);
const c3 = vec3f(0.2, 0.1, 0.7);
const c4 = vec3f(1.0, 0.9, 0.1);
const c5 = vec3f(0.1);
const c6 = vec3f(0.9);
var iTime = inputs.time;
var fragCoord = vec2f(pos.x, inputs.size.y - pos.y);
var iResolution = inputs.size;
const speed = vec2f(0.1, 0.9);
var shift = 1.327+sin(iTime*2.0)
const alpha = 1.0;
var dist = 3.5-sin(iTime*0.4)/1.8
var uv = fragCoord.xy / iResoluti
var p = fragCoord.xy * dist / iR
p += sin(p.yx*4.0+vec2f(.2,-.3)*
p += sin(p.yx*8.0+vec2f(.6,.1)*i
p.x -= iTime/1.1;
var q = fbm(p - iTime * 0.3+1.0*s
var qb = fbm(p - iTime * 0.4+0.1*
README
MIT license
WgShadertoy
WGSL playground inspired by Shadertoy.
No packages published
Contributors 2
fralonra fralonra
MonsieurLanza Damien Brun
Languages
Rust 85.0%
Fluent 15.0%
@compute.toys
new
browse
Search ...
LOGIN / SIGN UP
1152×648
@compute @workgroup_size(16, 16)
fn main_image(@builtin(global_invocation_id) id: vec3u) {
// Viewport resolution (in pixels)
let screen_size = textureDimensions(screen);
// Prevent overdraw for workgroups on the edge of the viewport
if (id.x >= screen_size.x || id.y >= screen_size.y) { return; }
// Pixel coordinates (centre of pixel, origin at bottom left)
let fragCoord = vec2f(f32(id.x) + .5, f32(screen_size.y - id.y) - .5);
// Normalised pixel coordinates (from 0 to 1)
let uv = fragCoord / vec2f(screen_size);
// Time varying pixel colour
var col = .5 + .5 * cos(time.elapsed + uv.xyx + vec3f(0.,2.,4.));
// Convert from gamma-encoded to linear colour space
col = pow(col, vec3f(2.2));
// Output to screen (linear colour space)
textureStore(screen, id.xy, vec4f(col, 1.));
}
channel0
channel1
Language WGSL
HDR sRGB
Float32 Textures
main_image
FragCoord.xyz
Editor
Explore
Docs
v0.6
Sign In
Untitled Shader
Share
Export
LIBRARY
History 1
Common
Main
void main()
{
//Normalized screen uvs [0, 1]
vec2 uv = gl_FragCoord.xy / u_resolution;
//Centered, fit screen coordinates
vec2 fit = 0.5 + (gl_FragCoord.xy - 0.5 * u_resolution) / min(u_resolution.x,
//Output for demo
fragColor = vec4(fit, 0, 1);
}
736×473
TUNER
INSPECT
ERRORS
FRAMES
HEATMAP
BUILT-IN
u_date 2026, 1, 20, 72…
u_frame 401
u_main sampler2D (Main)
u_mouse 6, 82, -1, -1
u_resolution 736, 473
u_time 3.5430
u_time_delta 0.0080
+ Add uniform
Compile
157 chars ~7 instr
Format

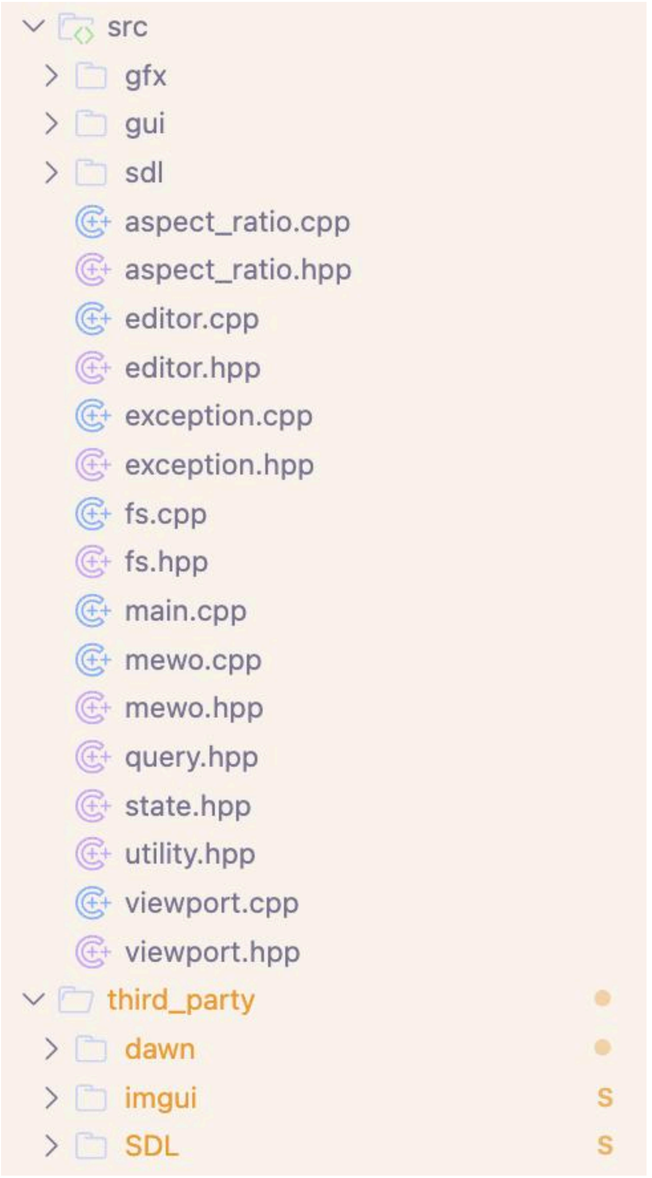
Goals

- Good good desktop user experience
- Not a competition!
- There are many stretch goals that would make Mewo be “better” but those will come naturally as I flesh things out
  - Multiple shader language support, additional channels/buffers, common functions library, porting to the web, etc.
- Feature parity with Shadertoy, and improve on some! (not a particularly high bar...)
  - Already have better resizing mechanics
- Feedback and feature suggestions are still much appreciated :)

In this alpha

- Basic project setup (macOS + Windows)
- Very simple fragment editor
- Viewport output
  - Two modes: aspect ratio and resolution
- Uniforms: time and resolution
- Resizable window


src
gfx
create.cpp
create.hpp
error.hpp
frame_context.hpp
renderer.cpp
renderer.hpp
gui
context.cpp
context.hpp
imconfig.h
layout.cpp
layout.hpp
sdl
context.cpp
context.hpp
window.cpp
window.hpp


src
gfx
gui
sdl
aspect_ratio.cpp
aspect_ratio.hpp
editor.cpp
editor.hpp
exception.cpp
exception.hpp
fs.cpp
fs.hpp
main.cpp
mewo.cpp
mewo.hpp
query.hpp
state.hpp
utility.hpp
viewport.cpp
viewport.hpp
third_party
dawn
imgui
SDL

```
while (!state_.should_quit) {
  while (SDL_PollEvent(&event)) {
    ImGui_ImplSDL3_ProcessEvent(&event);

    switch (event.type) {
    case SDL_EVENT_QUIT:
    case SDL_EVENT_WINDOW_CLOSE_REQUESTED: {
      state_.should_quit = true;
      break;
    }

    case SDL_EVENT_WINDOW_RESIZED: {
      auto [new_width, new_height] = window_.size_in_pixels();
      renderer_.resize(new_width, new_height);
      break;
    }
    }
  }

  const gfx::FrameContext frame_ctx = renderer_.prepare_new_frame();
  gui_ctx_.prepare_new_frame();
  viewport_.prepare_new_frame(state_, device, renderer_.queue());

  layout_.build(state_, gui_ctx_, device, editor_, viewport_);

  viewport_.record(frame_ctx);
  gui_ctx_.record(frame_ctx);

  static const wgpu::CommandBufferDescriptor CMD_BUF_DESC = { .label = "command-buffer" };
  wgpu::CommandBuffer cmd_buf = frame_ctx.encoder.Finish(&CMD_BUF_DESC);

  renderer_.queue().Submit(1, &cmd_buf);
  renderer_.surface().Present();
  device.Tick();
}
```

DEMO TIME 🙀

What's next

- Saving of projects to disk, and at minimum loading of projects via command line

Finishing this from alpha

Beta

- External media support (for now, just local images and videos)
- Hot reloading
- More polished user interface
- Finish loading and saving of projects
- Simple on-page website offering a description of the program and download links
- Documentation within the codebase (in Markdown)

Thanks!