Rectangular plasma (Tsoding)

This guide explains how to **recreate this project from scratch** in WebGPU Studio (without loading an example).

1) Goal and principle

We will create the buffers, paste the WGSL helper functions, write the compute shaders, then configure the Pass.

Steps (in order):

- **Pipeline 1**

2) Create a new project

1. Launch WebGPU Studio.
2. Click `New`.

3) Create the buffers (Buffers tab)

Create the following buffers (names must match exactly):

- **`texture1`**: size `512×256×1`, type `uint`, fill `random`

After each change, click `Apply`.

4) Add the helper library (Functions tab)

For each entry below:

1. Paste the WGSL.

Bibliothèque 1

```
const SXf = 512.0 ;
const SYf = 256.0 ;
const PI = 3.14159 ;
```

5) Create the compute shaders (Compute Shaders tab)

For each shader:

1. Click `+Add`.

2. Set the name.
3. Click `Apply`.
4. Paste the WGSL.

Shader `Compute2`

Workgroup: `8×8×1`

```
@compute @workgroup_size(8, 8, 1)
fn Compute2(@builtin(global_invocation_id) gid : vec3<u32>) {
    let index = gid.y * u32(SXf) + gid.x;
    let x = gid.x;
    let y = gid.y;
    let w = SXf ;
    let h = SYf ;
    let r = vec2<f32>(w, h) ;
    let t = (f32( step ) / 240.0) * 2.0 * PI;
    let FC = vec2<f32>(f32(x), f32(y));
    let p = (FC * 2.0 - r) / r.y;
    var l = vec2<f32>(0.0, 0.0);
    var i = vec2<f32>(0.0, 0.0);
    let s = 4.0 - 4.0 * abs(0.7 - dot(p, p));
    l = l + vec2<f32>(s, s);
    var v = p * l;
    // o accum
    var o = vec4<f32>(0.0, 0.0, 0.0, 0.0);
    for (var k : i32 = 1; k <= 8; k = k + 1) {
        let iy = f32(k);
        i.y = iy;
        let s4 = sin(vec4<f32>(v.x, v.y, v.y, v.x)) + vec4<f32>(1.0);
        o = o + s4 * abs(v.x - v.y);
        let c2 = cos(vec2<f32>(v.y, v.x) * iy + i + vec2<f32>(t, t)) /
iy + vec2<f32>(0.7, 0.7);
        v = v + c2;
    }
    let py4 = p.y * vec4<f32>(-1.0, 1.0, 2.0, 0.0);
    let num = 5.0 * exp(vec4<f32>(l.x - 4.0) - py4);
    let eps = vec4<f32>(1e-6);
    o = tanh(num / max(abs(o), eps));
    let R = u32(o.x * 255.0) ;
    let G = u32(o.y * 255.0) ;
    let B = u32(o.z * 255.0) ;
    texture1[index] = 0xFF000000u + ( R << 16) + (G << 8) + B ;
}
```

6) Configure the Pass (Pass tab)

Create the pipelines/steps in the following order:

- **Pipeline 1**: dispatch `64×32×1`

7) Compile and run

1. In the `Buffers` tab, select **`texture1`**.
2. View it in `2D` or `3D`.
3. Click `Compile`.
4. Click `Run` (or use `Step`).

8) Quick checks (if it doesn't work)

- `Console` tab: read WGSL errors.
- Check buffer **names** match the WGSL code.
- Check buffer sizes (X/Y/Z) and Pass dispatch.

9) Save

Click `Save` to export the project as a `.wgstudio` file.