

Random walk

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):

- **Initialisation**
- **Choix Mvt**
- **ChoixAutorisation**
- **Mvt**
- **Cpy**
- **Render**

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):

- **render**: size `128×128×1`, type `uint`, fill `empty`
- **render_mvt**: size `128×128×1`, type `uint`, fill `empty`
- **render_auto**: size `128×128×1`, type `uint`, fill `empty`
- **particules**: size `128×128×1`, type `uint`, fill `random`
- **particules2**: size `128×128×1`, type `uint`, fill `empty`
- **mvt**: size `128×128×1`, type `uint`, fill `empty`
- **autorisation**: size `128×128×1`, type `uint`, fill `empty`

After each change, click **Apply**.

4) Add the helper library (Functions tab)

For each entry below:

1. Paste the WGSL.

Bibliothèque 1

```
const SX = 128 ;
const SY = 128 ;
fn hash3(u : vec3<u32>, step : u32) -> f32 {
    // 32-bit mix, renvoie dans [0,1)
    var x = u.x * 0x27d4eb2du + u.y * 0x85ebca6bu + u.z * 0xc2b2ae35u +
    step * 0x165667b1u;
    x ^= x >> 15;
    x *= 0x2c1b3c6du;
    x ^= x >> 12;
    x *= 0x297a2d39u;
    x ^= x >> 15;
    return f32(x) / f32(0xfffffffffu);
}
```

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 **Init**

Workgroup: **8×8×1**

```
@compute @workgroup_size(8, 8, 1)
fn Init(@builtin(global_invocation_id) gid : vec3<u32>) {
    let index = gid.y * SX + gid.x;
    if (step == 0) {
        if (u32( hash3(gid, step) * 100.0 ) <= 0 ) {
            particules[index] = 1;
        } else {
            particules[index] = 0;
        }
        particules2[index]= 0 ;
        render[index]= 0;
        render_mvt[index]= 0;
        render_auto[index] = 0
        mvt[index]= 0;
        autorisation[index] = 0;
    }
}
```

Shader ChoixMvt

Workgroup: 8×8×1

```
@compute @workgroup_size(8, 8, 1)
fn ChoixMvt(@builtin(global_invocation_id) gid : vec3<u32>) {
    let index = gid.y * SX + gid.x;
    if (step >= 1 && gid.x >= 1 && gid.x < SX-1 && gid.y >= 1 && gid.y < SY-1) {
        if( particles[index] == 1 ) { // particle
            mvt[index] = 1 + u32( hash3(gid, step) * 4.0 ); // choix de la case où aller
        } else { // no particle
            mvt[index] = 0 ; // pas de choix car pas de particule
        }
    }
}
```

Shader ChoixAutorisation

Workgroup: 8×8×1

```
@compute @workgroup_size(8, 8, 1)
fn ChoixAutorisation(@builtin(global_invocation_id) gid : vec3<u32>) {
    let index = gid.y * SX + gid.x;
    if (step >= 1 && gid.x >= 1 && gid.x < SX-1 && gid.y >= 1 && gid.y < SY-1) {
        if( particles[index] == 0 ) { // NO particle
            autorisation[index] = 1 + u32( hash3(gid, step) * 4.0 );
        } else { // particle
            autorisation[index] = 0 ; // interdit car deja occupee
        }
    }
}
```

Shader Mvt

Workgroup: 8×8×1

```
@compute @workgroup_size(8, 8, 1)
fn Mvt(@builtin(global_invocation_id) gid : vec3<u32>) {
    let index = gid.y * SX + gid.x;
    var vient = 0u ;
    if (step >= 1 && gid.x >= 1 && gid.x < SX-1 && gid.y >= 1 && gid.y <
```

```

SY-1) {
    if ( autorisation[index] == 1 && mvt[index+1] == 3 ) {
//-[1]-><=3=
        vient = 1 ;
    }
    if ( autorisation[index] == 3 && mvt[index-1] == 1 ) {
//=1=><-[3]-
        vient = 1 ;
    }

    if ( autorisation[index] == 4 && mvt[index-SX] == 2 ) {
//v-[4]-
        vient = 1 ;
//=2=^
    }

    if ( autorisation[index] == 2 && mvt[index+SX] == 4 ) {
//=[4]=v
        vient = 1 ;
//^-[2]-
    }
}
var part = 0u ;
if (step >= 1 && gid.x >= 1 && gid.x < SX-1 && gid.y >= 1 && gid.y <
SY-1) {
    if (mvt[index]==1 && autorisation[index+1] == 3 ) {
// =[1]=><-3-
        part = 1 ;
    }

    if (mvt[index]==3 && autorisation[index-1] == 1 ) {
// -1-><=[3]=
        part = 1 ;
    }
    if ( mvt[index] == 2 && autorisation[index+SX] == 4 ) {
// -4- v
        part = 1 ;
// =[2]=^
    }
    if (mvt[index]==4 && autorisation[index-SX] == 2 ) {
//=[4]=v
        part = 1 ;
// -2- ^
    }
}
if (vient == 1 ) { // On est d accord : on peut venir ici et une

```

```

particule veut venir ici
    particules2[index] = 1 ;
    return ;
}
if ( part == 1 ) { // La particule ici s'en va dans une autre case
    particules2[index] = 0 ;
    return ;
}
particules2[index] = particules[index] ;
}

```

Shader Cpy

Workgroup: 8×8×1

```

@compute @workgroup_size(8, 8, 1)
fn Cpy(@builtin(global_invocation_id) gid : vec3<u32>) {
    let index = gid.y * SX + gid.x;
    if (step >= 1 ) {
        particules[index] = particules2[index] ;
    }
}

```

Shader Render

Workgroup: 8×8×1

```

@compute @workgroup_size(8, 8, 1)

fn Render(@builtin(global_invocation_id) gid : vec3<u32>) {
    let R = 0xFFFF0000u ;
    let G = 0xFF00FF00u ;
    let B = 0xFF0000FFu ;
    let Y = 0xFF00FFFFu ;
    let Rs = 0xFF440000u ;
    let Gs = 0xFF004400u ;
    let Bs = 0xFF000044u ;
    let Ys = 0xFF004444u ;
    let Grey = 0xFF443344u;
    let index = gid.y * SX + gid.x;
    if ( particules[index] == 0 ) { render[index] = Grey; }
    else {render[index] = 0xFFFFFFFFu ; }
    // Mvt souhaite
    if ( mvt[index] == 0 ) { render_mvt[index] = Grey ; }
    if ( mvt[index] == 1 ) { render_mvt[index] = R ; }
}

```

```

if ( mvt[index] == 3 ) { render_mvt[index] = G ; }
if ( mvt[index] == 2 ) { render_mvt[index] = B ; }
if ( mvt[index] == 4 ) { render_mvt[index] = Y ; }
// autorisation souhaite
if ( autorisation[index] == 0 ) { render_auto[index] = Grey ; }
if ( autorisation[index] == 1 ) { render_auto[index] = Rs ; }
if ( autorisation[index] == 3 ) { render_auto[index] = Gs ; }
if ( autorisation[index] == 2 ) { render_auto[index] = Bs ; }
if ( autorisation[index] == 4 ) { render_auto[index] = Ys ; }
}

```

6) Configure the Pass (Pass tab)

Create the pipelines/steps in the following order:

- **Initialisation**: dispatch $16 \times 16 \times 1$
- **ChoixMvt**: dispatch $16 \times 16 \times 1$
- **ChoixAutorisation**: dispatch $16 \times 16 \times 1$
- **Mvt**: dispatch $16 \times 16 \times 1$
- **Cpy** : dispatch $16 \times 16 \times 1$
- **Render** : dispatch $16 \times 16 \times 1$

7) Compile and run

1. In the **Buffers** tab, select **render**.
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.