LemnosLife Rust

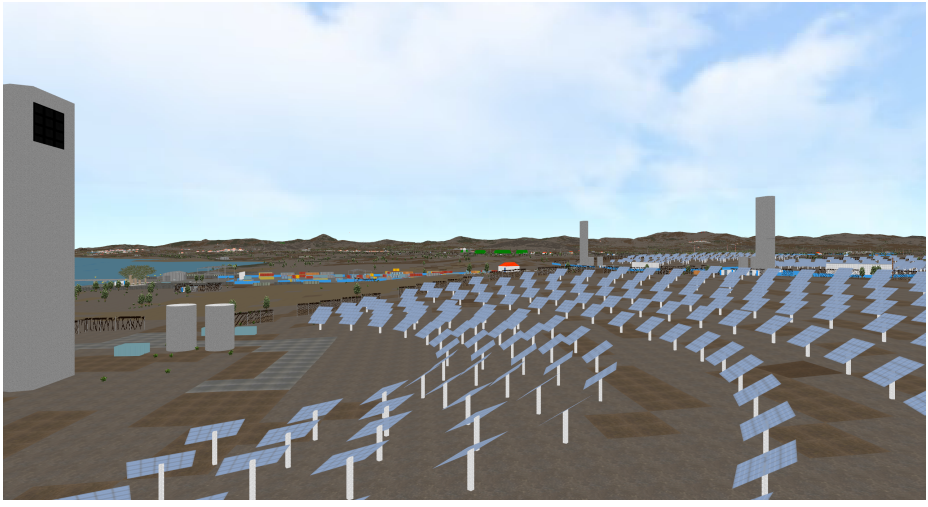
Benjamin Loison

November 2, 2022

Arma 3 (Bohemia Interactive, release date: 2013)

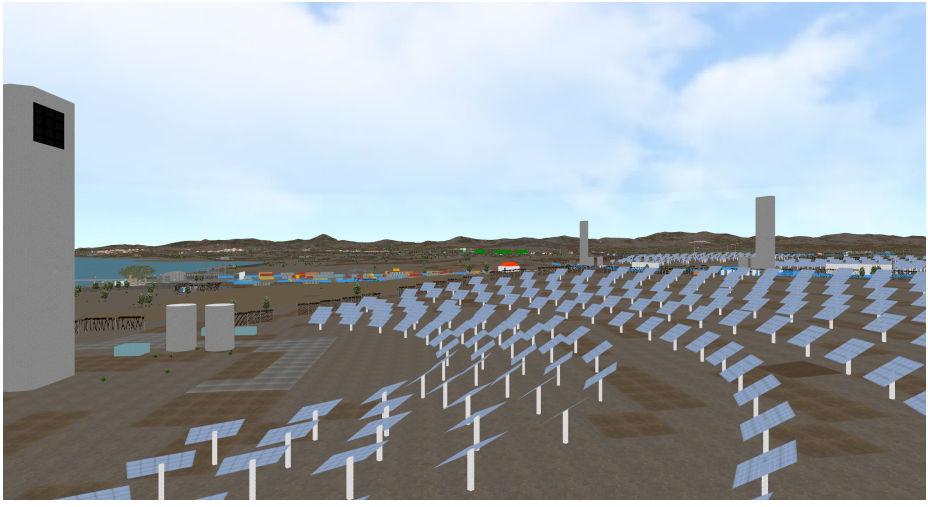
Open world realism-based military tactical shooter multiplayer video game (with vocal chat within the game).

LemnosLife (since 2017, 83 C++ KLOC)



Using SDL 2 and OpenGL.

LemnosLife (since 2017, 83 C++ KLOC)



Using SDL 2 and OpenGL.

Goal: Rust version using OpenGL of LemnosLife with ground and structures rendering.

Altis map

Modified version of the 476 km² Greek Lemnos island obtained with in-game scripts.

Altis map

Modified version of the 476 km² Greek Lemnos island obtained with in-game scripts. Only took following meta-data:

1. Ground type is defined every 8 meters, so they are 14,745,600 such information.
2. Ground altitude is defined every 4 meters, so they are 58,982,400 vertices for the ground.
3. 926 different structures with about 2,000,000 instances. I modeled 168 of them.

Programming choices

1. Rust once cleanly compiled is crash free in comparison with C++.

Programming choices

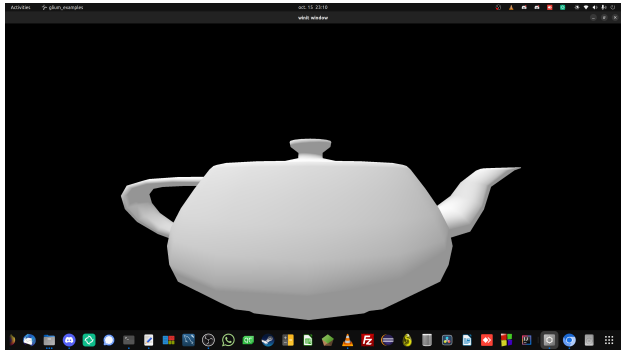
1. Rust once cleanly compiled is crash free in comparison with C++.
2. glium (Rust OpenGL wrapper) is no longer actively maintained but is easier to use than wgpu.

Programming choices

1. Rust once cleanly compiled is crash free in comparison with C++.
2. glium (Rust OpenGL wrapper) is no longer actively maintained but is easier to use than wgpu.
3. Use three files per 1 km^2 for ground altitudes, ground types and structures.

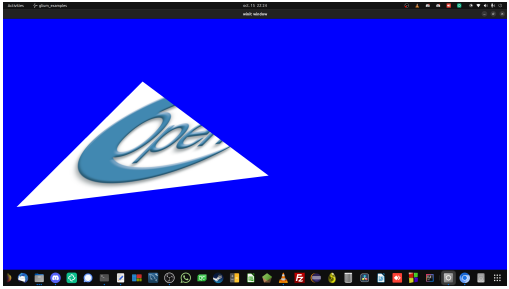
Camera with non rotative camera and specific OBJ rendering

Rust glium (OpenGL) ‘teapot‘ example:



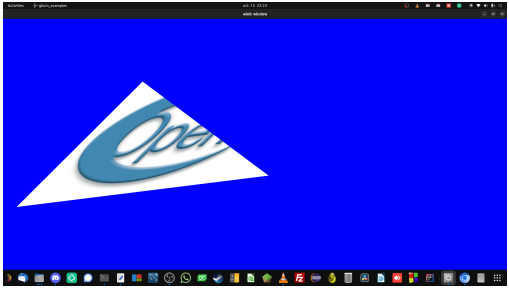
1. Modify OBJ rendering to render arbitrary white triangles (for the ground)
2. Added rotating left/right camera

Add a texture to the rendered triangles

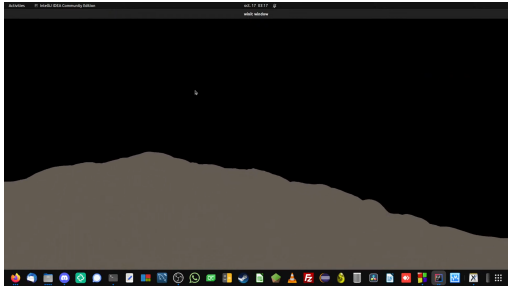


Rust glium example

Add a texture to the rendered triangles



Rust glium example



Rust LemnosLife single ground type chunk

Not able to pass an array of textures

In order to support to have a texture per ground type.

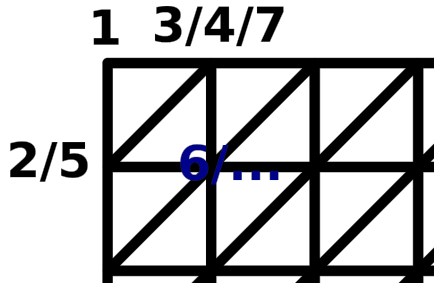
```
122             fragment: "
123                 #version 140
124
125                 in vec2 v_tex_coords;
126                 flat in uint v_biome;
127
128                 out vec4 f_color;
129
130                 uniform sampler2D tex0, tex1, tex2, tex3, tex4, tex5, tex6, tex7, tex8, tex9, tex10, tex11, tex12, tex13, tex14, tex15, tex16, tex1
    7, tex18;
131
132                 void main() {
133                     switch (v_biome) {
134                         case 0u:
135                             f_color = texture(tex0, v_tex_coords);
136                             break;
137                         case 1u:
138                             f_color = texture(tex1, v_tex_coords);
139                             break;
```

Optimization: don't consider redundant information

Caring about optimizations because facing out of memory error otherwise.

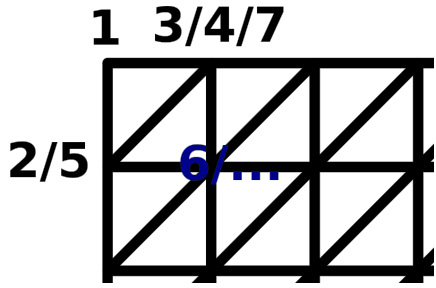
Optimization: don't consider redundant information

Caring about optimizations because facing out of memory error otherwise.



Optimization: don't consider redundant information

Caring about optimizations because facing out of memory error otherwise.



Instead of repeating redundant information to the GPU, we just provide for each vertex: its index within the chunk, its altitude and its ground type.

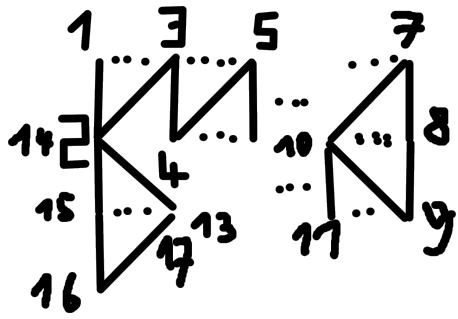
Optimization: don't consider redundant information

Fragment shader:

```
86                  uint index_mod = index % 6u;
87                  uint index_div = index / 6u;
88
89                  uint z = index_div / 250u;
90                  uint x = index_div % 250u;
91
92                  switch (index_mod) {
93                      case 0u:
94                          v_tex_coords = vec2(0, 0);
95                          break;
96                      case 1u:
97                      case 4u:
98                          v_tex_coords = vec2(0, 1);
99                          x++;
100                         break;
101                     case 2u:
102                     case 3u:
103                         v_tex_coords = vec2(1, 0);
104                         z++;
105                         break;
106                     default: // case 5u
107                         v_tex_coords = vec2(1, 1);
108                         x++;
109                         z++;
110                         break;
111                 }
114                 x *= TILE_SIZE;
115                 z *= TILE_SIZE;
```

Optimization: reduce number of vertices

Using triangle strip to reduce number of vertices.



Optimization: reduce number of vertices

Fragment shader:

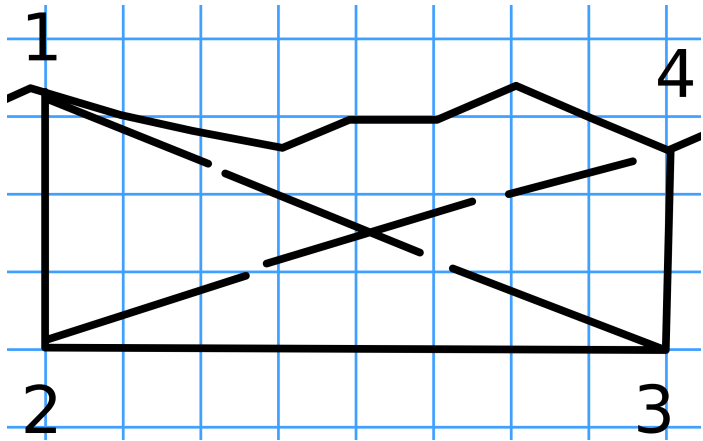
```
const uint TILES_PER_CHUNK_ROW = 250u;
const uint TRIANGLES_PER_CHUNK_ROW = TILES_PER_CHUNK_ROW * 2u + 2u;

void main() {
    uint index_mod_triangles_per_chunk_row = index % TRIANGLES_PER_CHUNK_ROW;
    uint index_div_triangles_per_chunk_row = index / TRIANGLES_PER_CHUNK_ROW;
    uint index_div_triangles_per_chunk_row_mod_2 = index_div_triangles_per_chunk_row % 2u;
    uint index_mod_triangles_per_chunk_row_div_2 = index_mod_triangles_per_chunk_row / 2u;

    uint x = index_div_triangles_per_chunk_row_mod_2 == 0u ? index_mod_triangles_per_chunk_row_div_2 : TILES_PER_CHUNK_ROW - index_mod_triangles_per_chunk_row_div_2;
    uint z = index_div_triangles_per_chunk_row + index % 2u;

    switch (index % 4u) {
        case 0u:
            v_tex_coords = vec2(index_div_triangles_per_chunk_row_mod_2, 0);
            break;
        case 1u:
            v_tex_coords = vec2(index_div_triangles_per_chunk_row_mod_2, 1);
            break;
        case 2u:
            v_tex_coords = vec2(1u - index_div_triangles_per_chunk_row_mod_2, 0);
            break;
        default: // case 3u
            v_tex_coords = vec2(1u - index_div_triangles_per_chunk_row_mod_2, 1);
            break;
    }
```

Optimization: problem across chunks

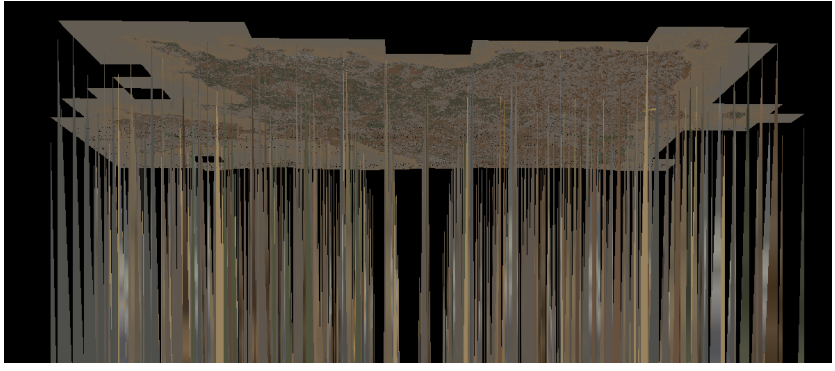
However can't lift drawing cursor between chunks (as we don't assume any order between them).

Optimization: problem across chunks

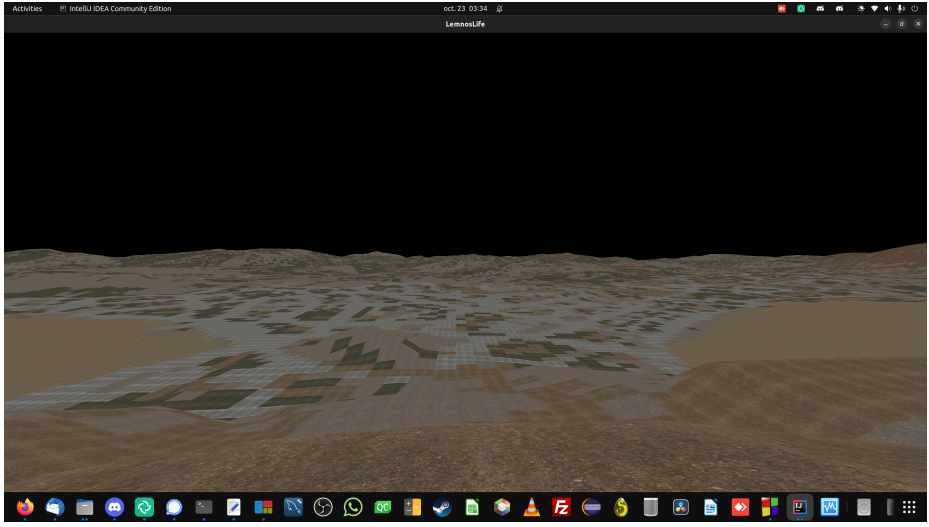
However can't lift drawing cursor between chunks (as we don't assume any order between them).



Optimization: problem across chunks



Shift chunks



Convert my structures encoded in my .struc format to .obj

My format:

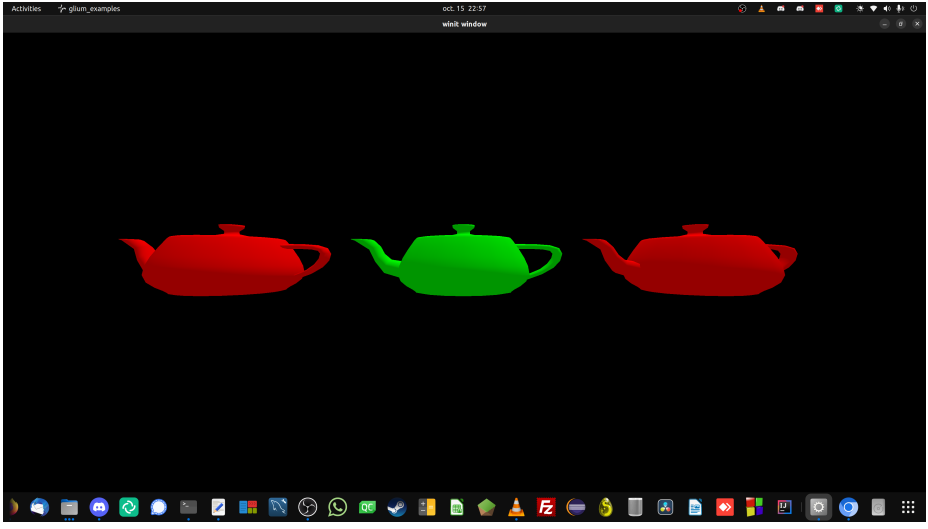
```
ficus.png
0 QUAD 1;0 1;1 0;1 0;0 -0.5;0;0 -0.5;0;1 0.5;0;1 0.5;0;0
1 QUAD 1;0 1;1 0;1 0;0 0;-0.5;0 0;-0.5;1 0;0.5;1 0;0.5;0
```

OBJ format (.obj and .mtl respectively):

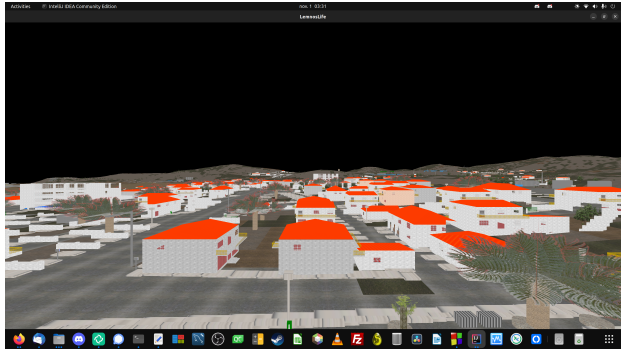
```
mtllib 24.mtl
v -0.5 0 0
...
vt 1 0
...
usemtl ficus
f 1/1 2/2 3/3 4/4
f 5/5 6/6 7/7 8/8

newmtl ficus
map_Kd ../../../Assets/Downloads/ficus.png
```

Multiple OBJ instances

Rust glium example:



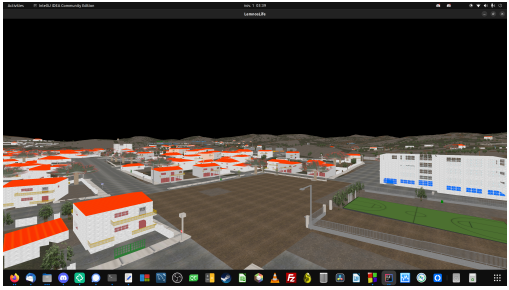
Add structures rotation



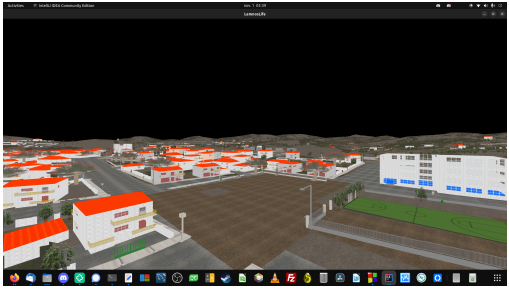
2D rotation matrix:

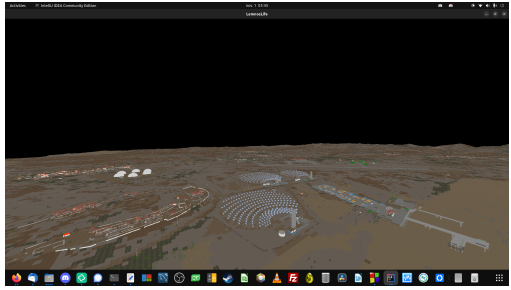
$$\begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$$

Results

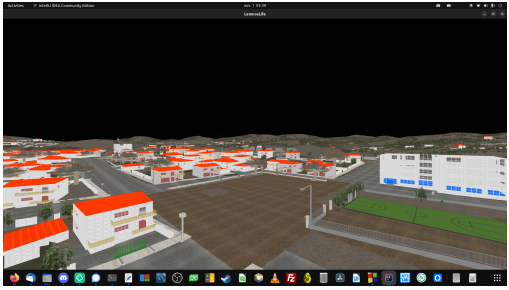


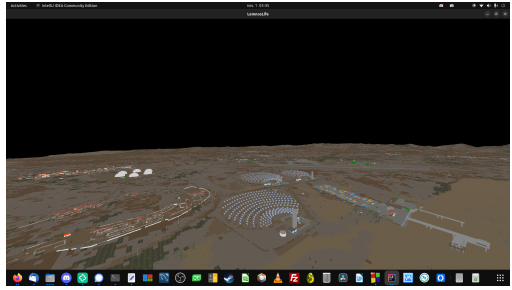
Results





Results

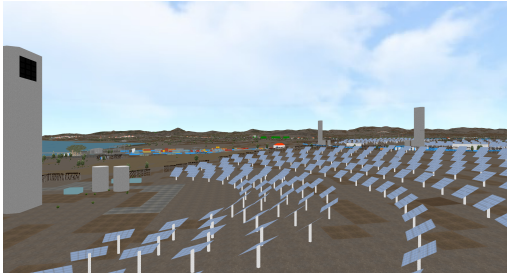




With whole map display:

1. Loading takes 8.5 seconds
2. In theory 160 FPS, according to my frame timer

Comparison C++/Rust LemnosLife

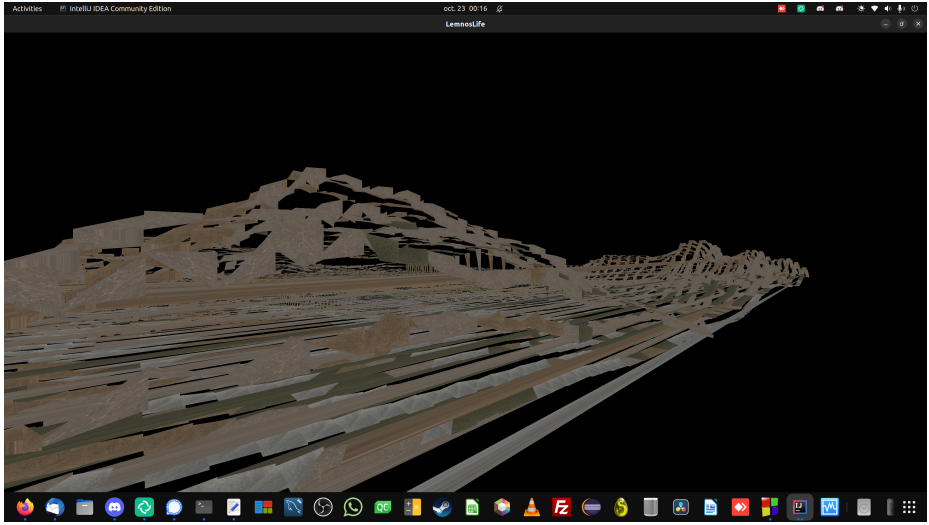


Goal

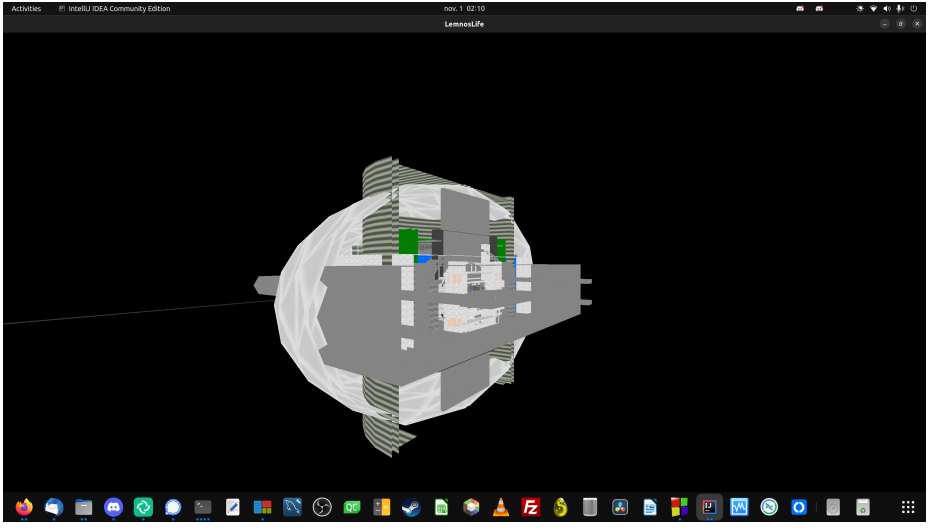
Result

Any questions?

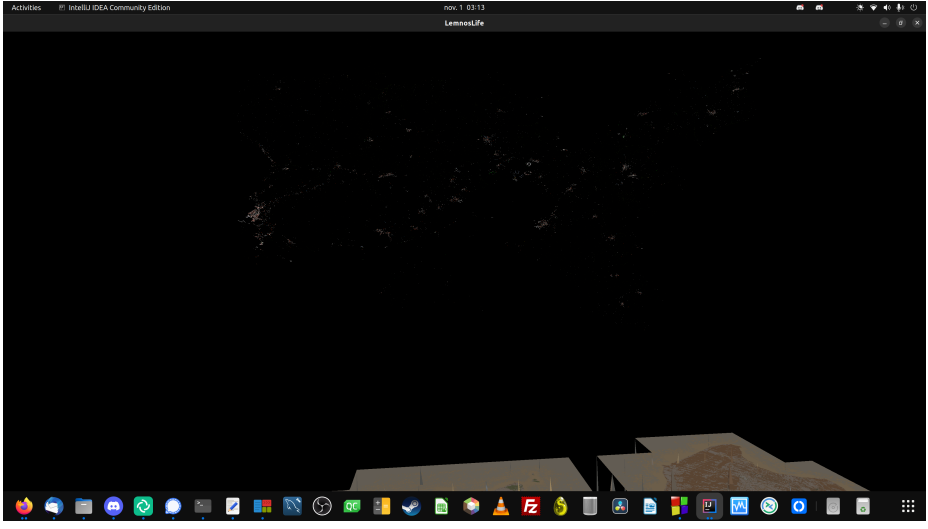
Blooper


Activities
IntelliJ IDEA Community Edition
oct. 23 00:16
LemnosLife

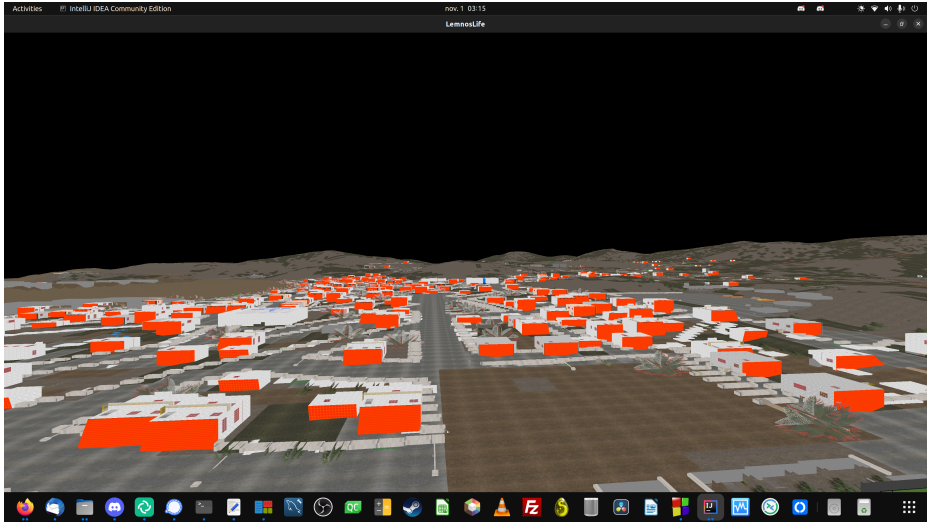
Blooper


Activities
IntelliJ IDEA Community Edition
nov. 1 02:10
LemnosLife

Blooper


Activities
IntelliJ IDEA Community Edition
nov. 1 03:13
LemnosLife

Blooper


Activities
IntelliJ IDEA Community Edition
nov. 1 03:15
LemnosLife

Blooper


Activities
IntelliJ IDEA Community Edition
nov. 1 03:33
LemnosLife