Implementation of deferred snow deformation

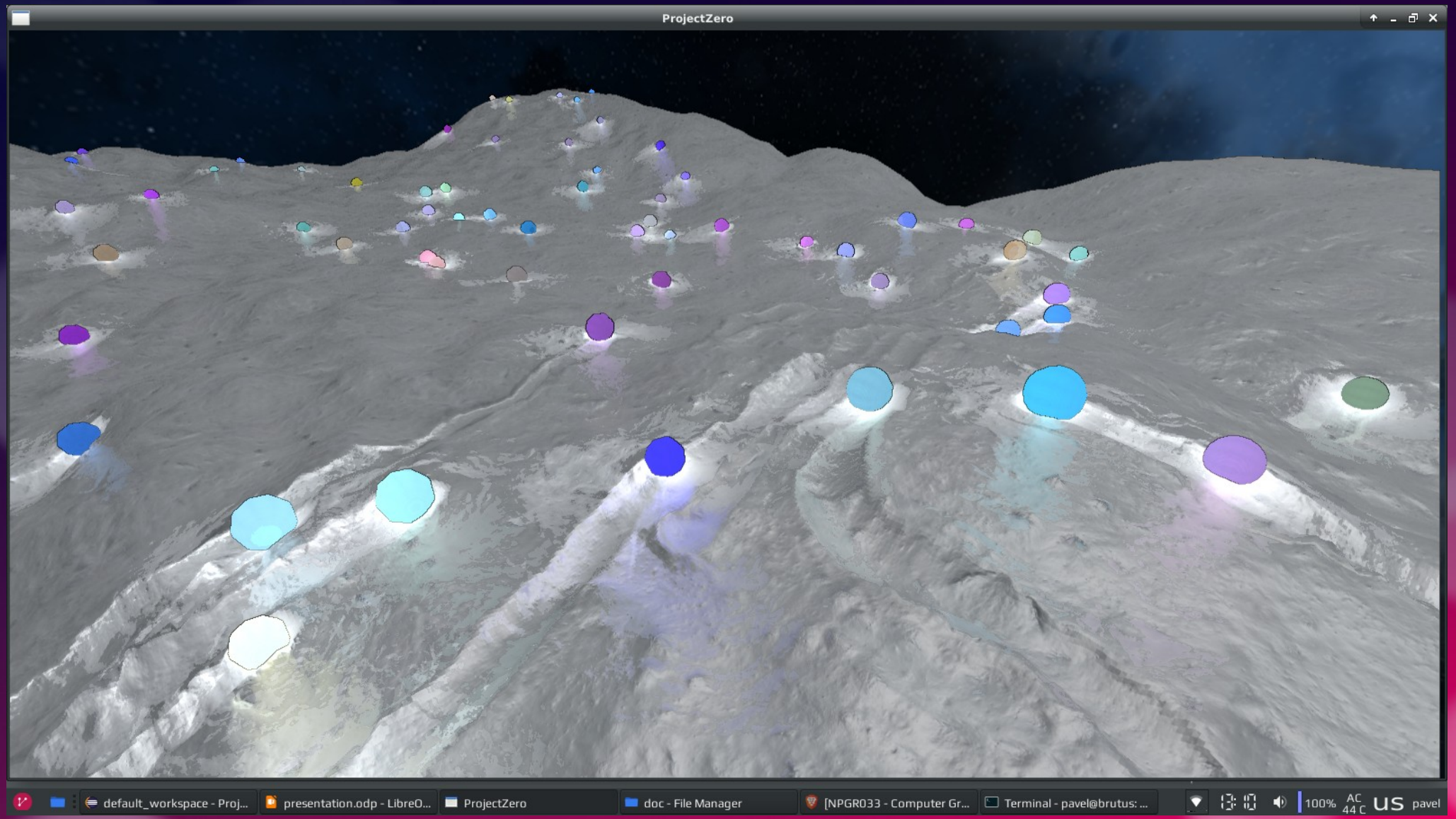
Based on the technique used in the Rise of the Tomb Raider

Why?

- Real-time game mechanic
- Compute shaders
- Mesh deformation
- Scalable complexity

SHOWCASE

How?


ProjectZero
default_workspace - Proj...
presentation.odp - LibreO...
ProjectZero
doc - File Manager
[NPGR033 - Computer Gr...
Terminal - pavel@brutus: ...
100%
us
pavel

Where I started:

Before first render pass

- OpenGL initialization
- Error-checking methods!
- Camera implementation
- Image loading
- Architecture - namespaces

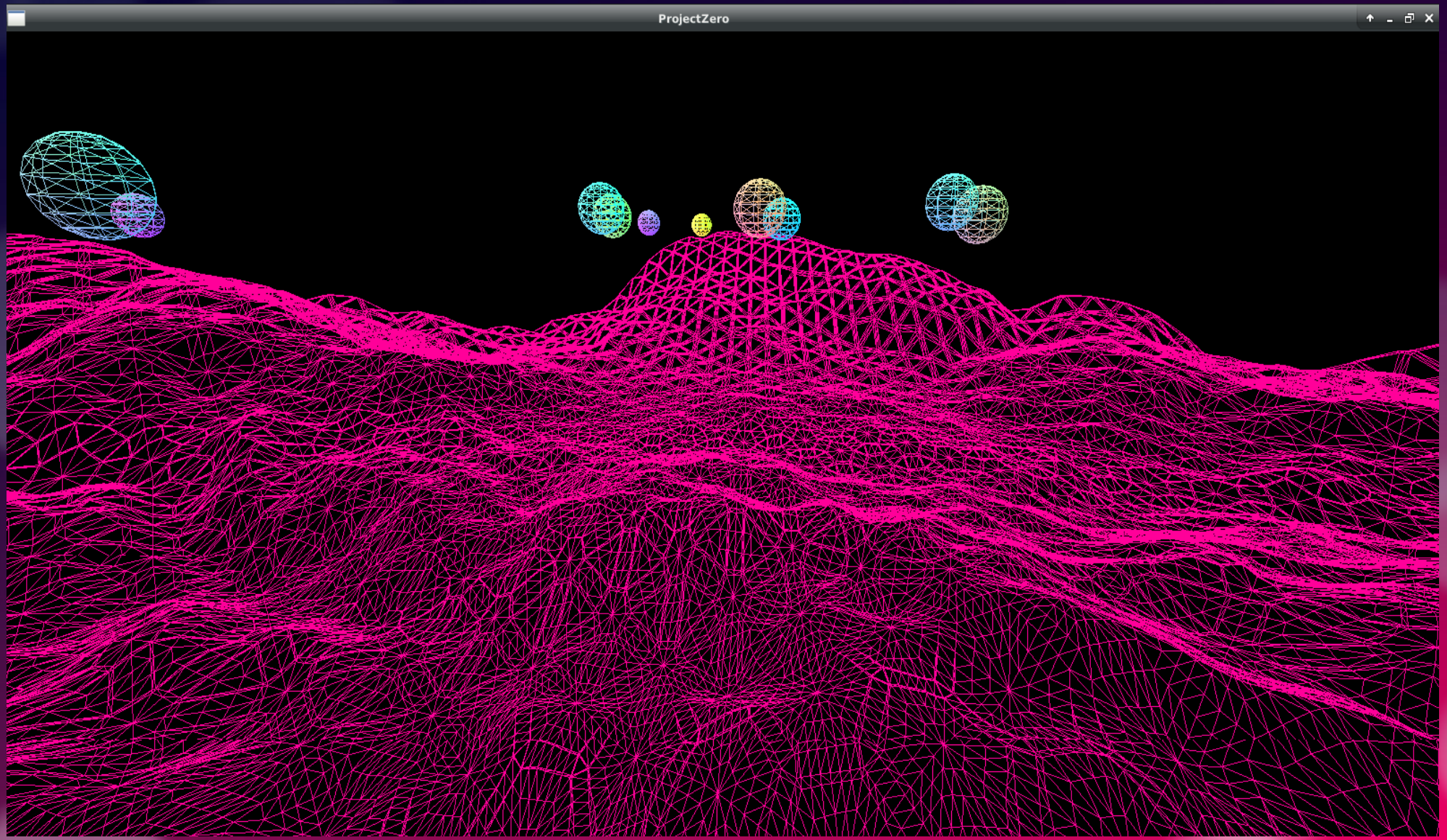
```
384
385 void setup() {
386     deffered_render::init();
387     config::defferedShading = true;
388     config::lightVolumes = true;
389     shaderProgram::createDefferedProgram();
390     shaderProgram::createTerrainProgram();
391     shaderProgram::createSpheresProgram();
392     shaderProgram::createSnowProgram(); //MUST BE AFTER TERRAIN
393     shaderProgram::createSkyboxProgram();
394     terrain::init();
395     spheres::init();
396     snow::init(); //MUST BE AFTER TERRAIN AND SPHERES
397     skybox::init();
398     checkGl();
399 }
400
401 }
402
403 int main() {
404     glfwContext::initGlfw();
405     rendering::lastTime = float(glfwGetTime());
406     rendering::currentTime = float(glfwGetTime());
407
408     rendering::setup();
409     fps::fps_start();
410     while (!glfwWindowShouldClose(glfwContext::window)) {
411         rendering::render();
412     }
413
414     glfwDestroyWindow(glfwContext::window);
415     glfwTerminate();
416     exit(EXIT_SUCCESS);
417 }
```

Tip: check your camera angle units

Render passes

- Deformation and physics
- Snow filling
- Terrain
- Snow
- Spheres
- Lights
- Skybox

Terrain



Terrain

Options:

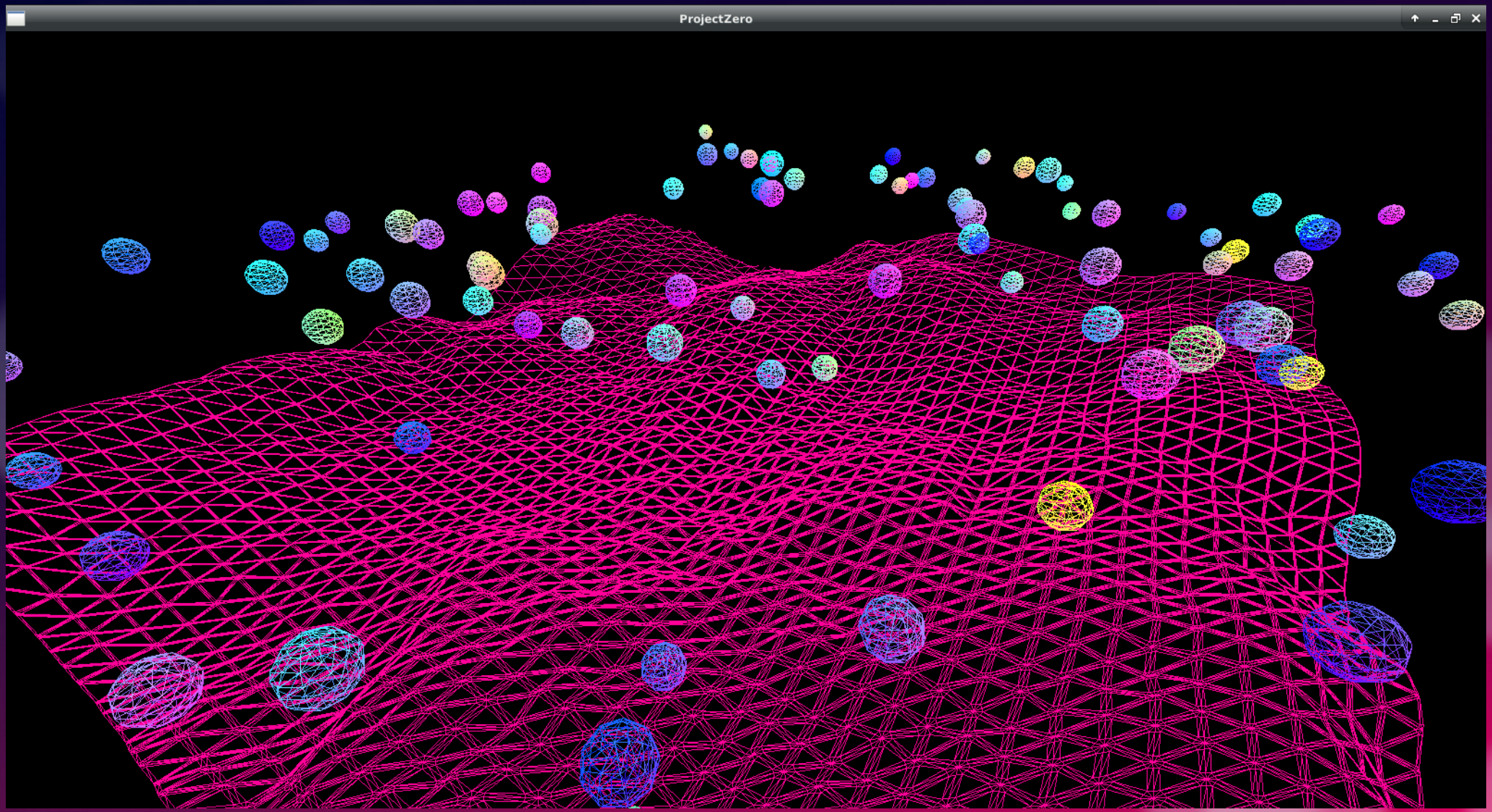
- Single dense mesh - demanding
- World chunks (with LOD) - complicated
- Single sparse mesh with tesselation

Height value:

- Generated from noise
- Heightmap

implemented sparse mesh with tesselation and heightmap

Spheres



Spheres

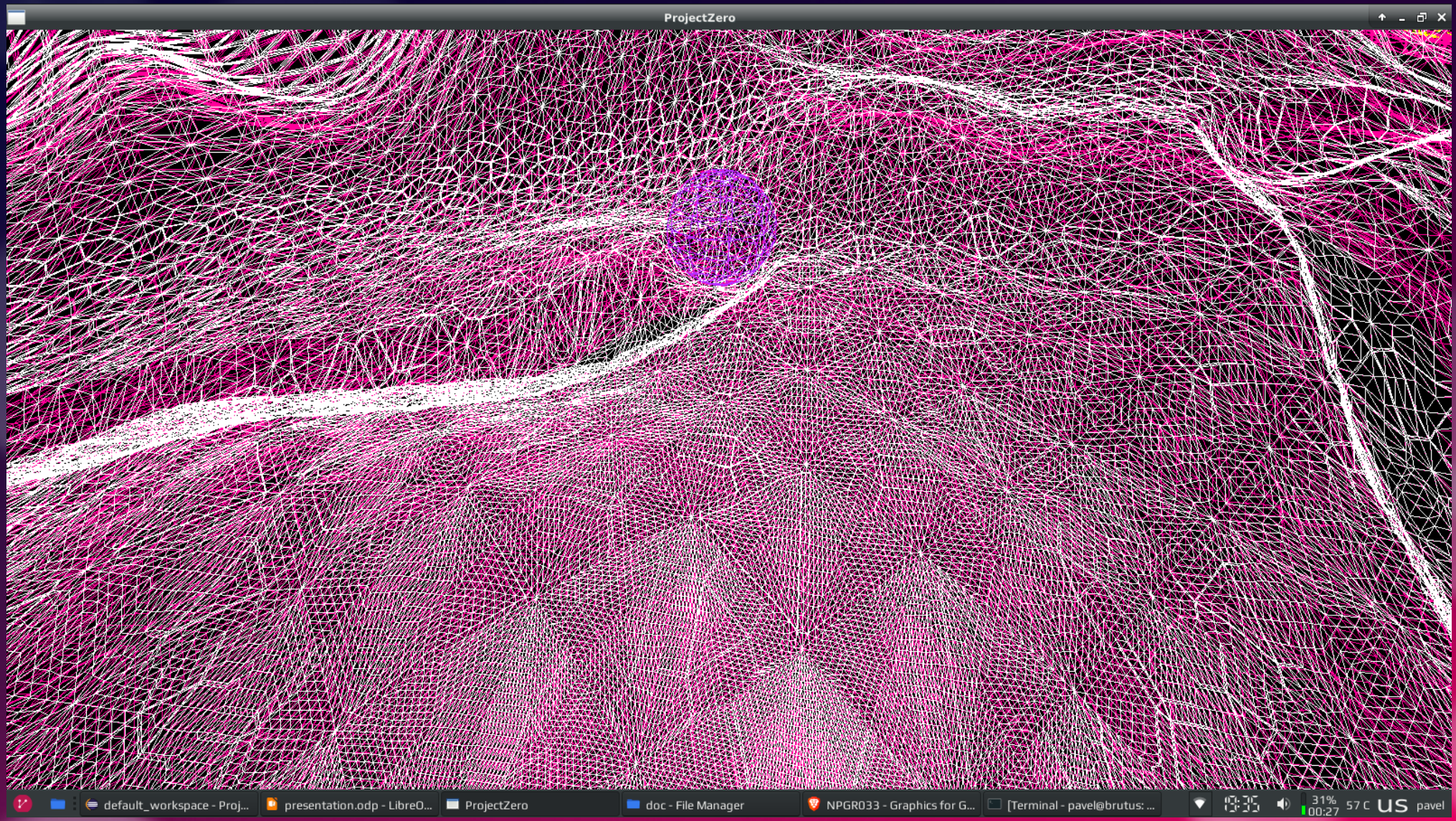
- Why?
  - Easy to generate
  - No animation during movement
  - Later reused for light volumes

- Procedurally generated mesh
- Instanced draw
- Data stored in SSBO
- Each sphere is a light source

Deformation and physics

- Compute shader dispatch
- Deformation
  - Atomic writes around the sphere location
  - World to texture mapping
- Physics
  - Data in SSBO – speed, velocity, size(mass)
  - Correct implementation – clustering
  - “broken” implementation used
  - Update of sphere position according to heightmap

Snow


ProjectZero
default_workspace - Proj...
presentation.odp - LibreO...
ProjectZero
doc - File Manager
NPGR033 - Graphics for G...
[Terminal - pavel@brutus: ...
31%
00:27
57 C
us
pavel

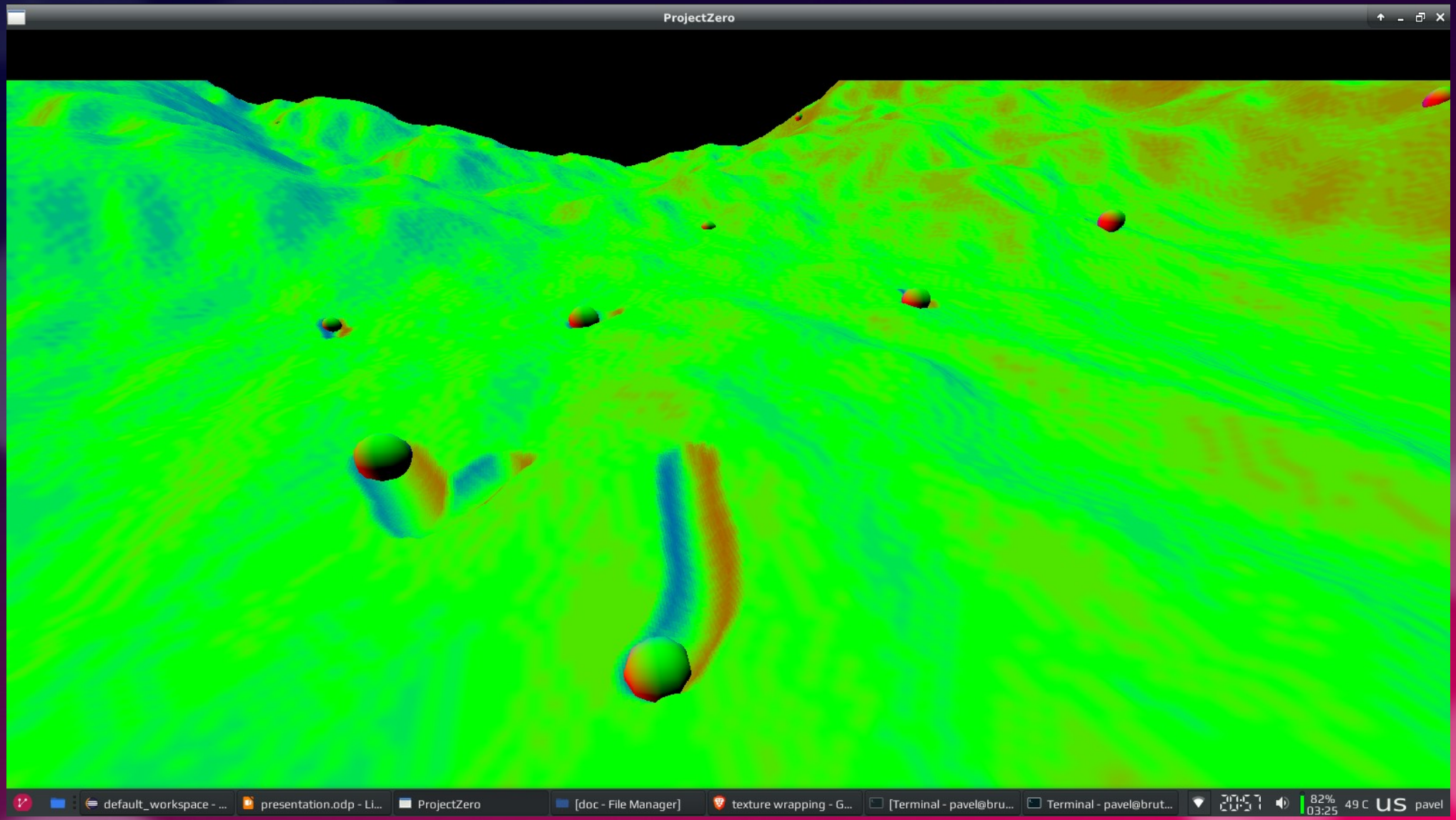
Snow

- Terrain lifted by certain amount
  - Inherited tesselation
- Integration of deformation texture
  - Texture to world mapping
  - Lowering deformed points
  - Computing the snow evaluation “at the edges”

Snow filling

- Compute shader dispatch
- Simple value addition for each pixel of deformation texture
- Except:
  - Map the camera position to the texture
  - Compute distance to this pixel as if it was in the middle
  - Create function that:
    - Has huge value at the most distant pixels from camera in the shifted space
    - Has almost zero value at the camera position

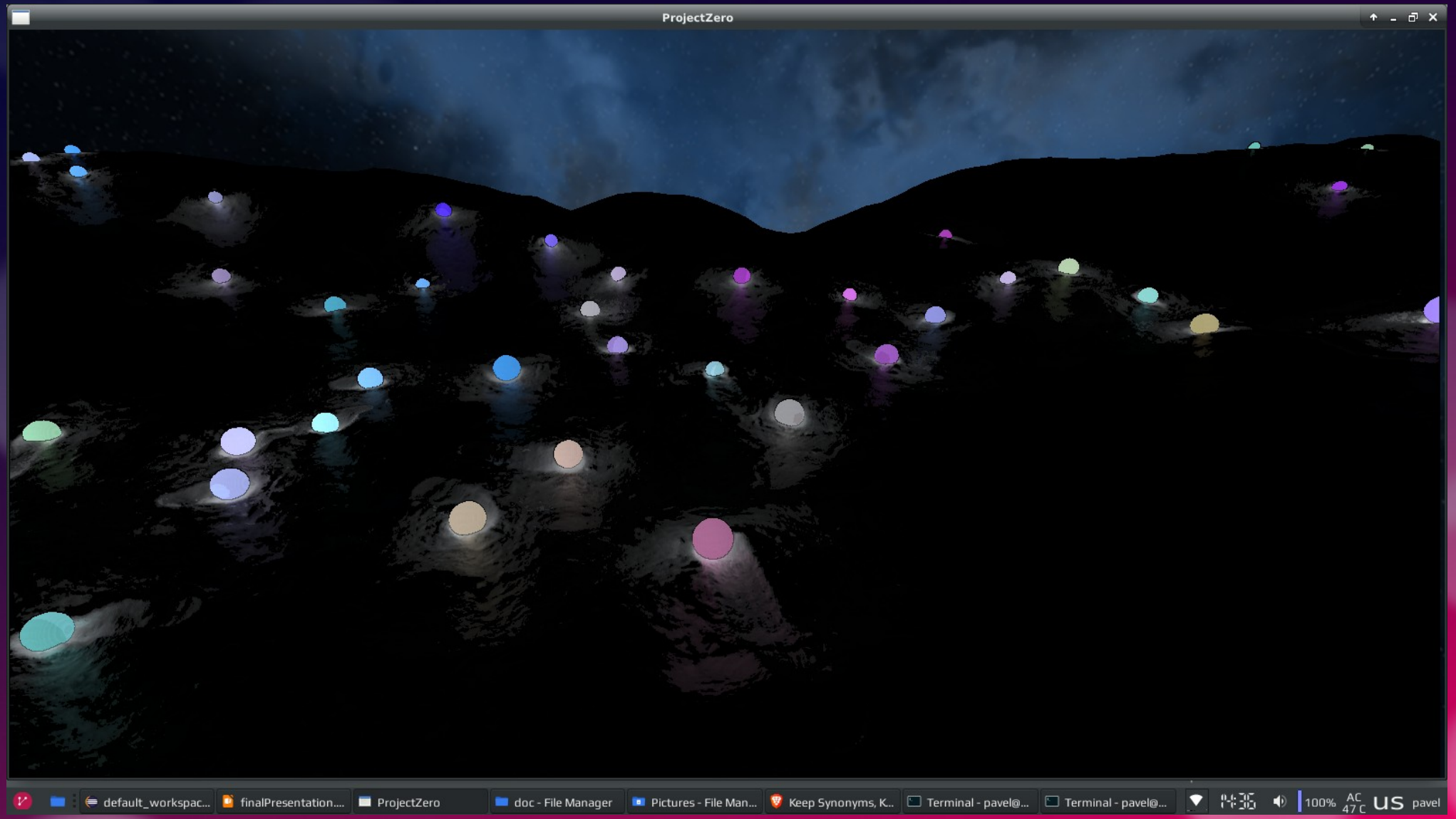
Deferred Pipeline



Deferred Pipeline

- G-Buffer:
  - Position
  - Normal
  - Color
- Changes in all fragment shaders
- Forward geometry pass kept for debug purposes
- Terrain and snow:
  - Computing normals from tangent and bitangent

Lights


ProjectZero
default_workspac...
finalPresentation....
ProjectZero
doc - File Manager
Pictures - File Man...
Keep Synonyms, K...
Terminal - pavel@...
Terminal - pavel@...
100%
us
pavel

Lights

- Each sphere is a light source
- Light volumes:
  - Each sphere is expanded to light volume
  - The size is computed from light attenuation equation

$$F_{light} = \frac{I}{K_c + K_l * d + K_q * d^2}$$

- Render setup:
  - The same geometry like in spheres render
  - Detph check and write off
  - Front-face culling enabled
  - Blending enabled with add_one_one function

Lights

- In vertex shader expand the light volumes
- In fragment shader:
  - Compute screen space coordinates
  - Read data from G-Buffer
  - Compute Blinn-Phong shading
  - Do gamma correction
- Problems:
  - Large light volumes = demanding, low fps
  - Small light volumes = very dark scene (no ambient)
  - Gamma correction breaks the smooth attenuation
  - Spheres are unlit – their faces are oriented away

Lights - hacks

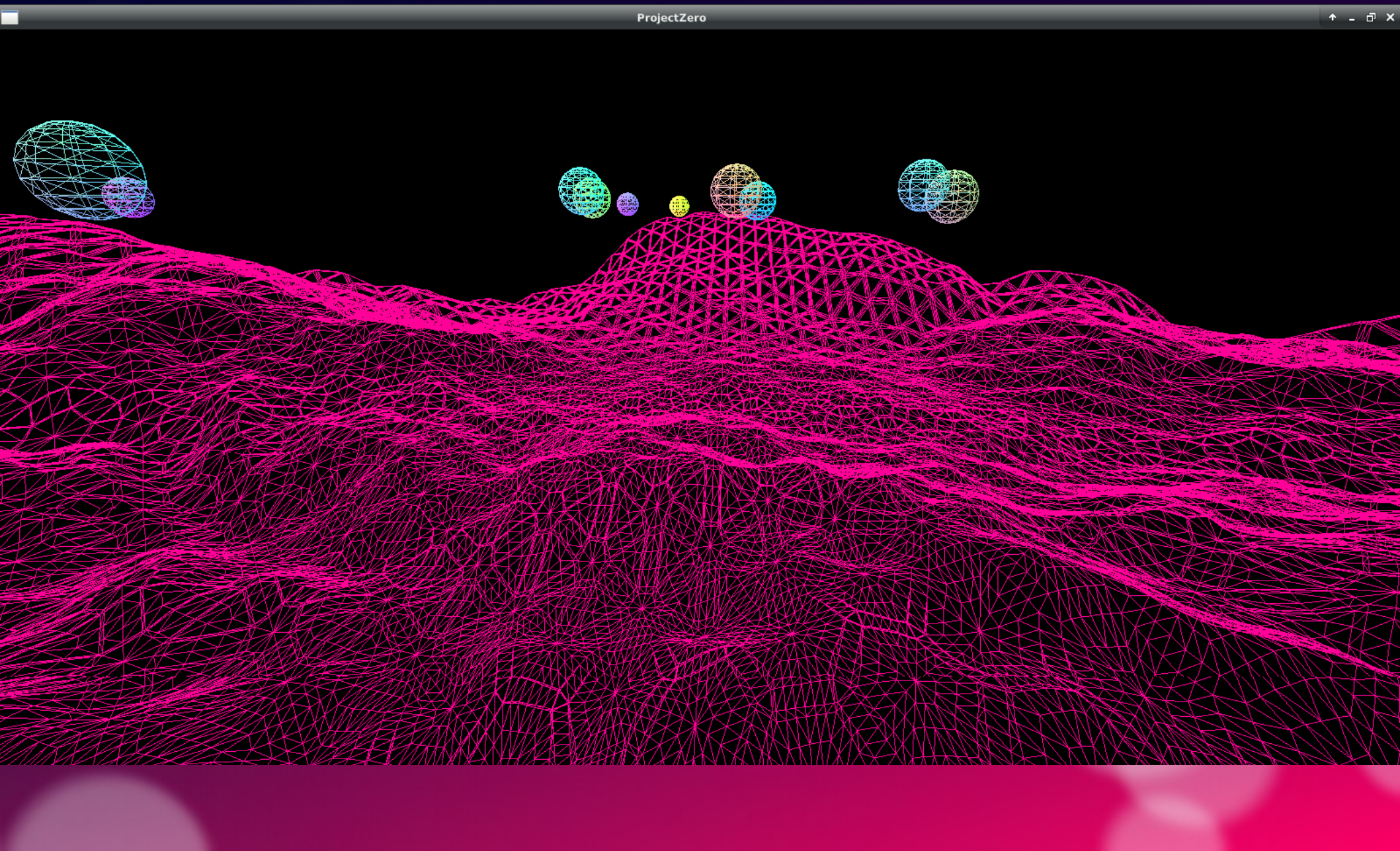
- Adding ambient light:
  - Another light volume centered on camera
  - Set constant attenuation
  - Switch off specular or set center above the scene
- Unlit spheres:
  - Set normals of spheres to (0,0,0)
  - In light pass, check for this, then set specular to 1
- Gamma correction:
  - Check for light intensity after correction and decrease

Skybox

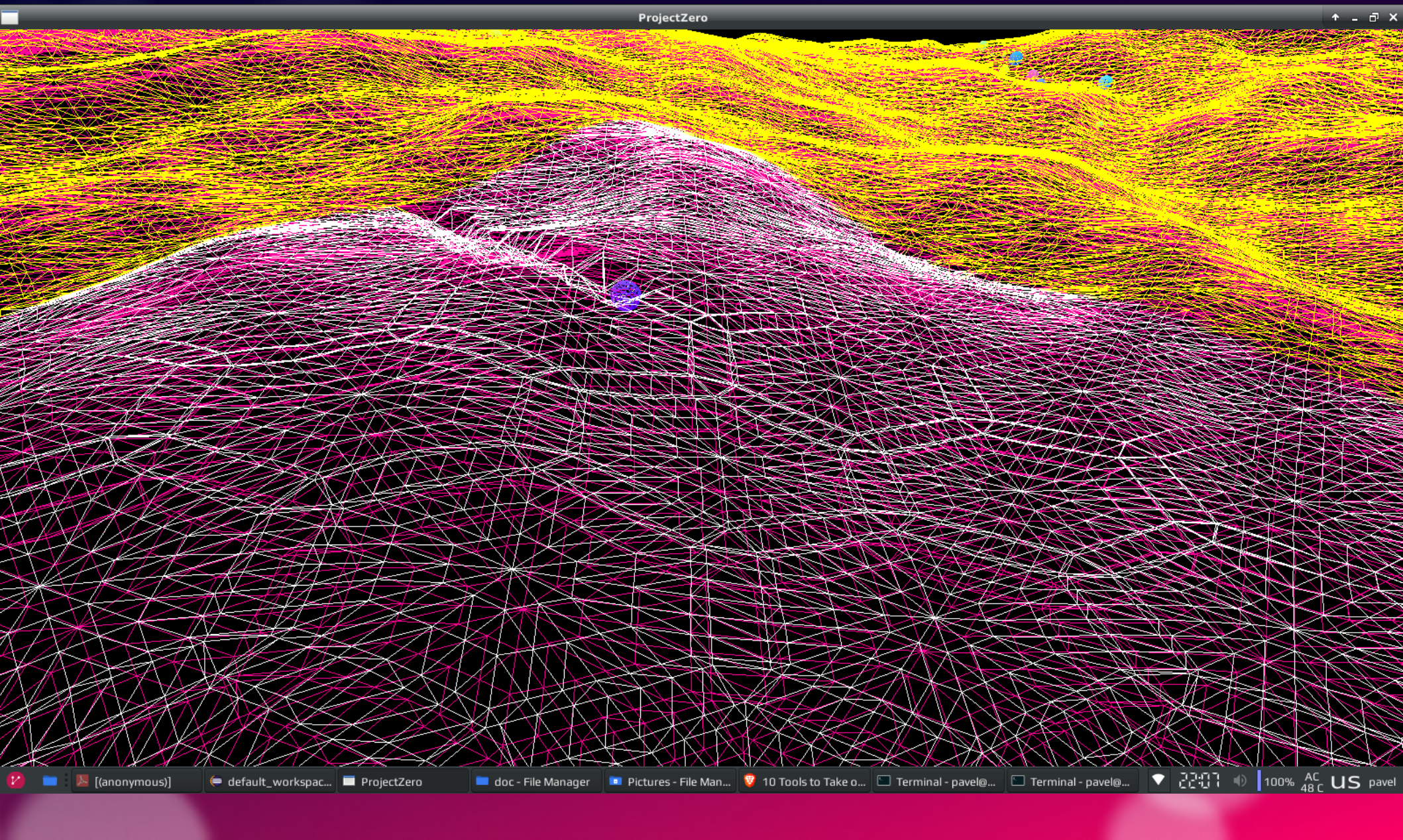
Skybox

- Texture sampled by direction
- Options:
  - Render first:
    - Depth must be 1.0
    - Propagated into g-color buffer
  - Render last:
    - Depth buffer is available
    - Check where depth = 1
    - Render sky, else discard fragment

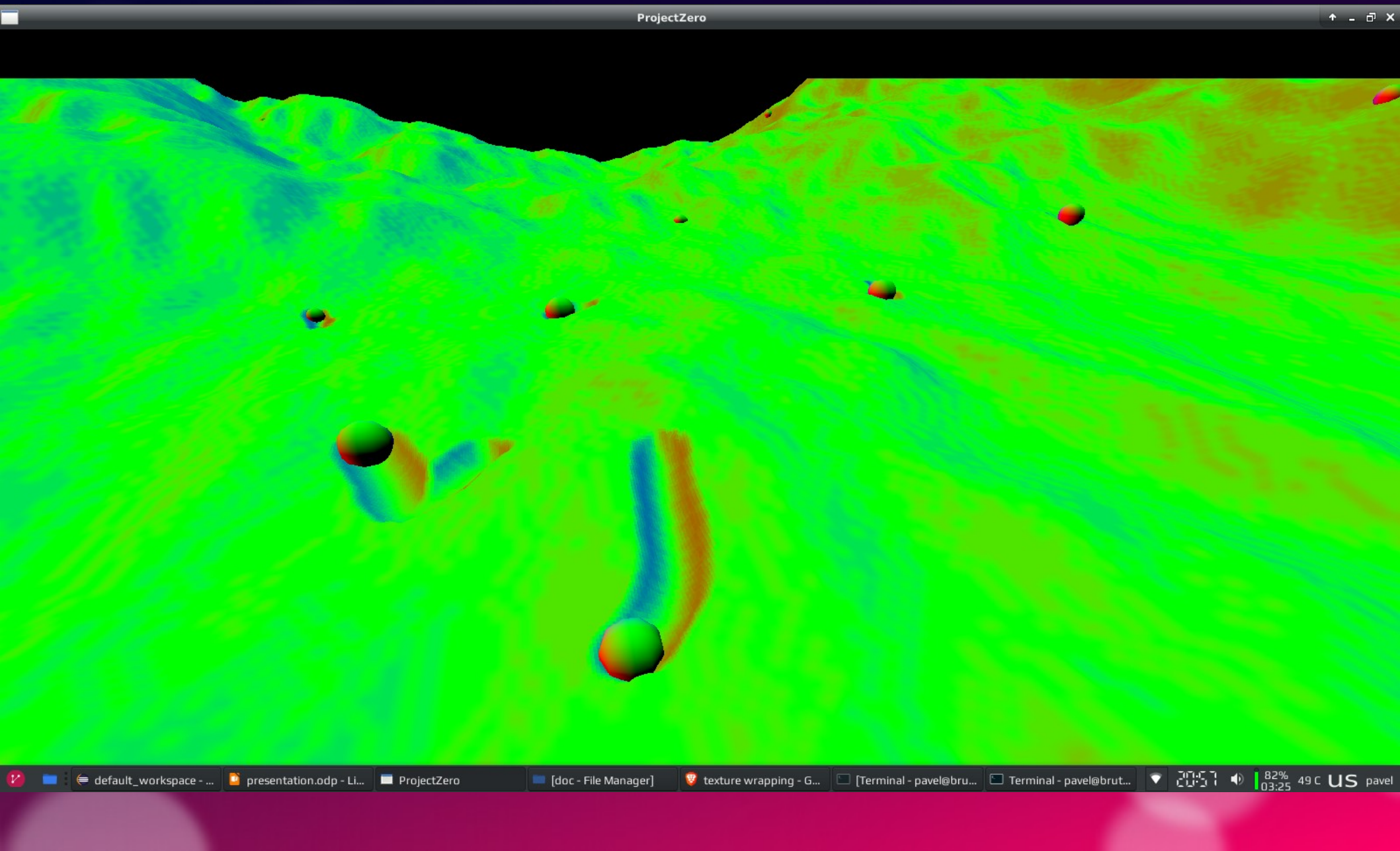
Progress samples



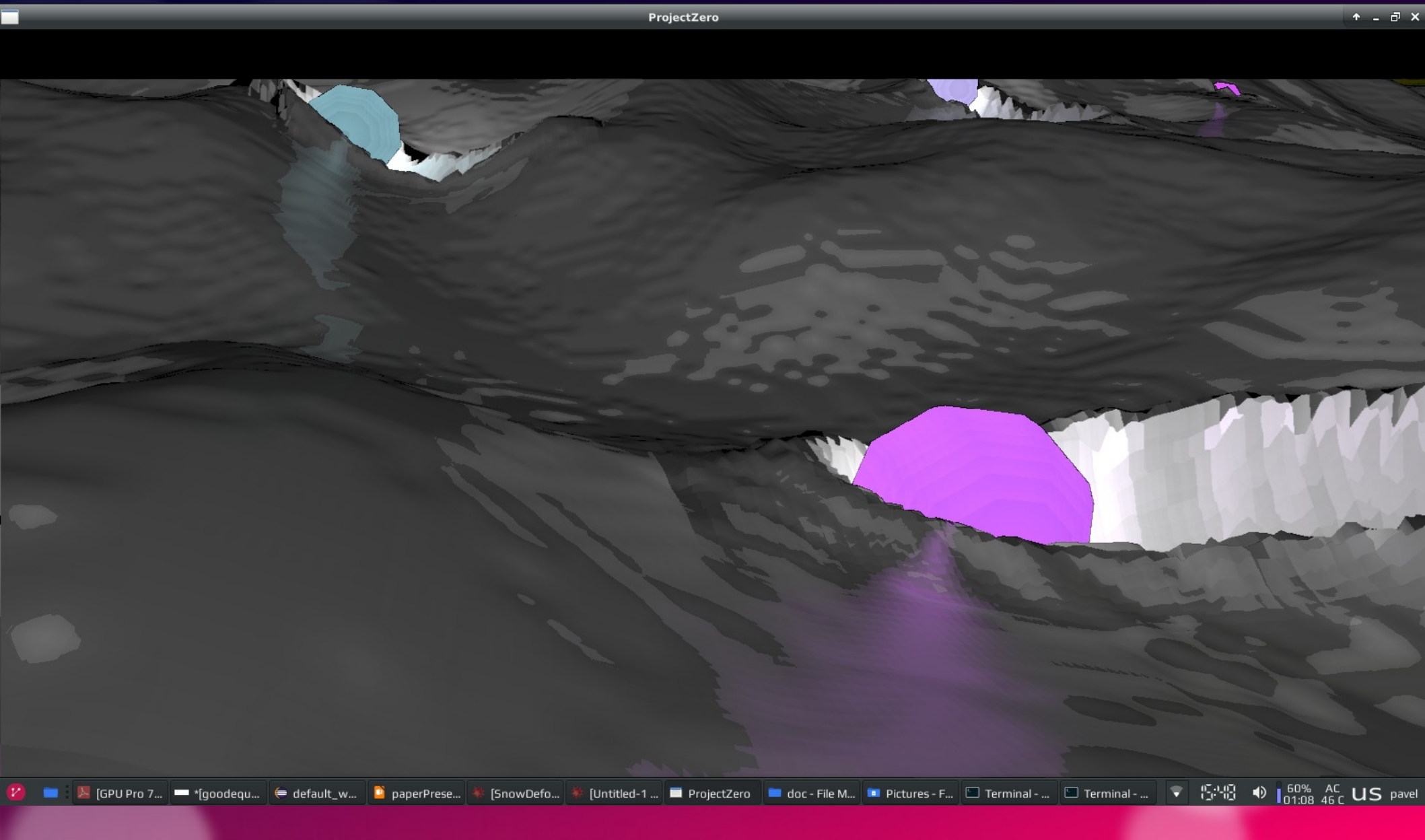
Progress samples



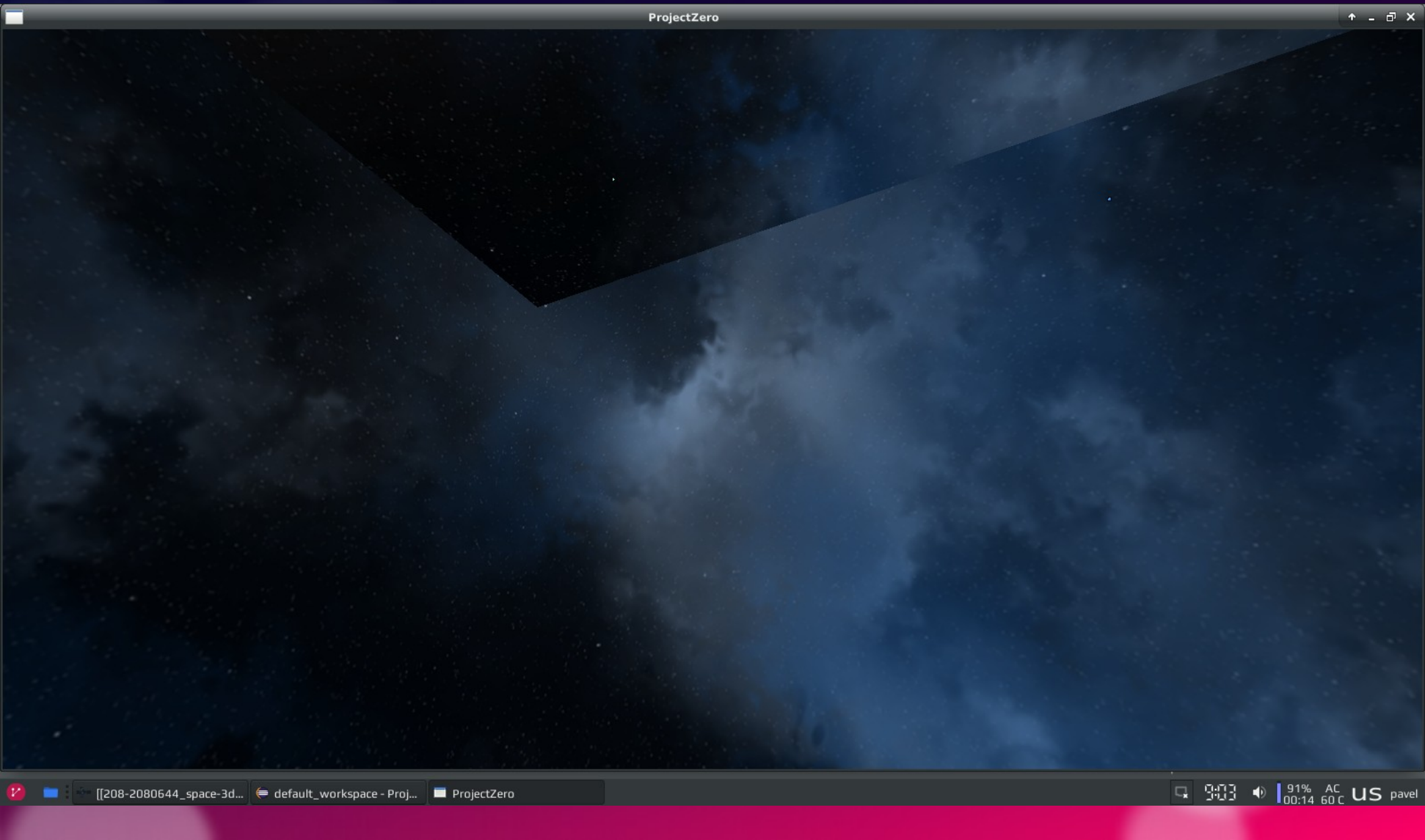
Progress samples



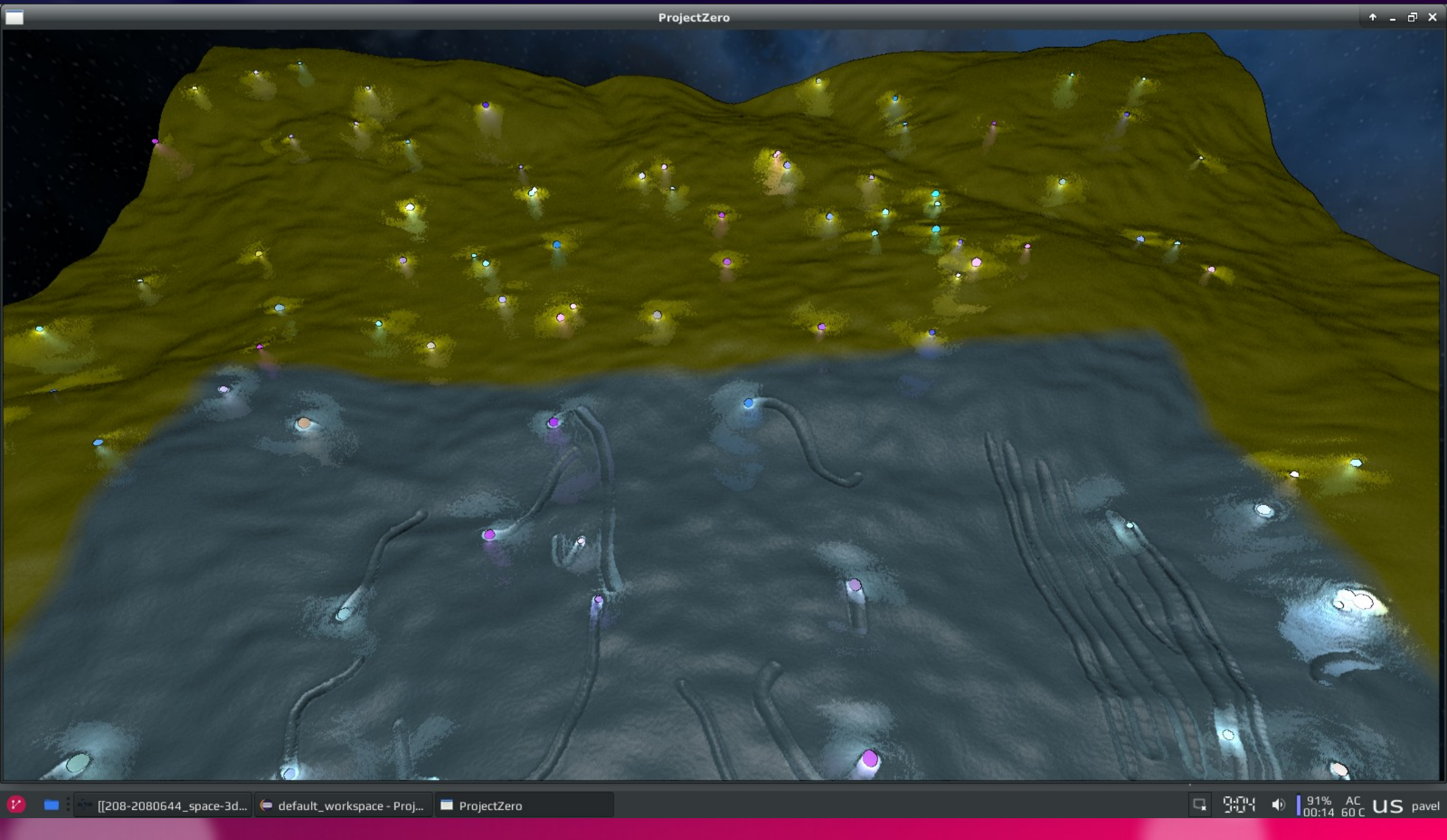
Progress samples



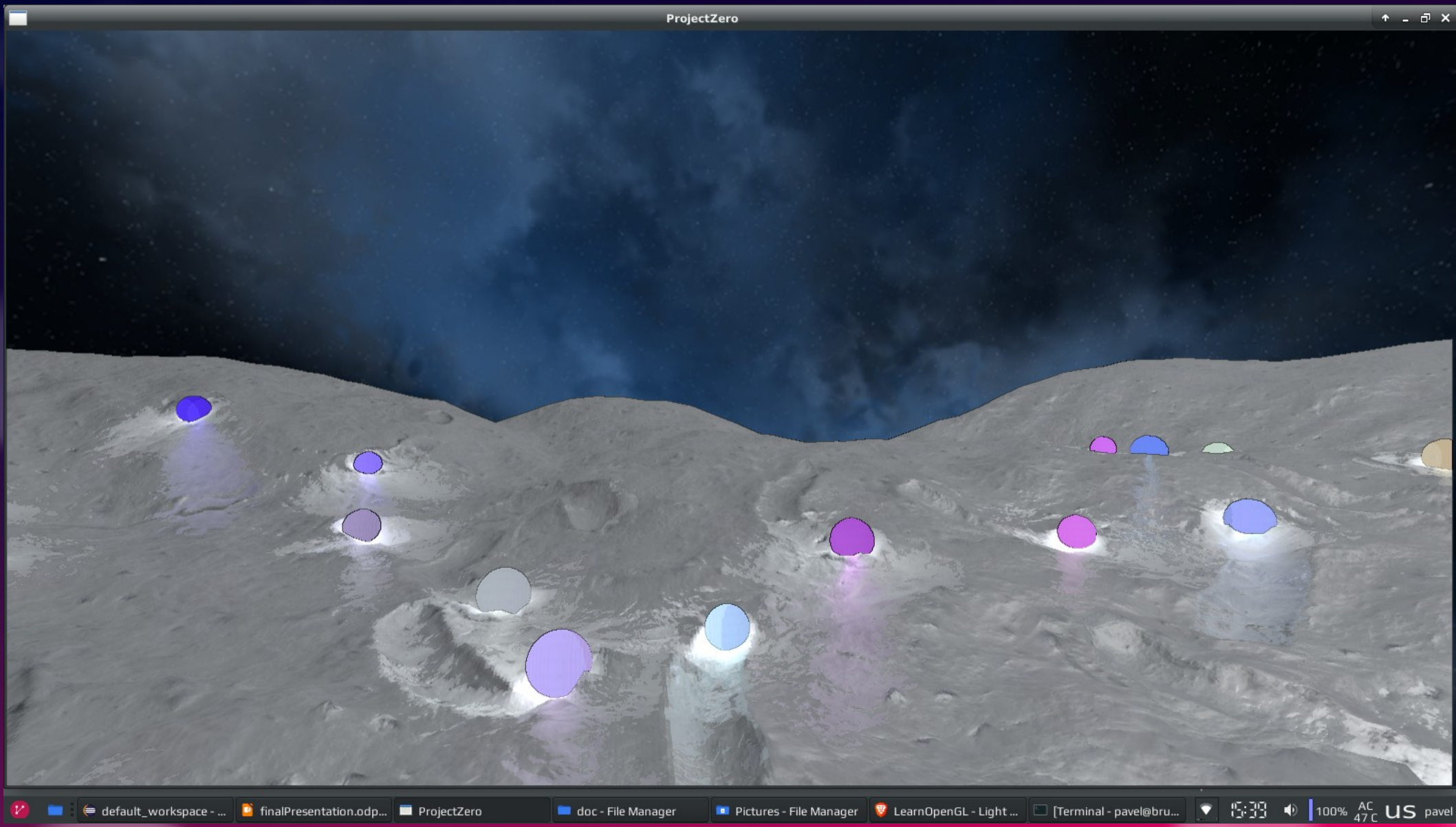
Progress samples



Progress samples



Result


ProjectZero
default_workspace - ...
finalPresentation.odp...
ProjectZero
doc - File Manager
Pictures - File Manager
LearnOpenGL - Light ...
[Terminal - pavel@bru...
100%
us
pavel

Future TODO

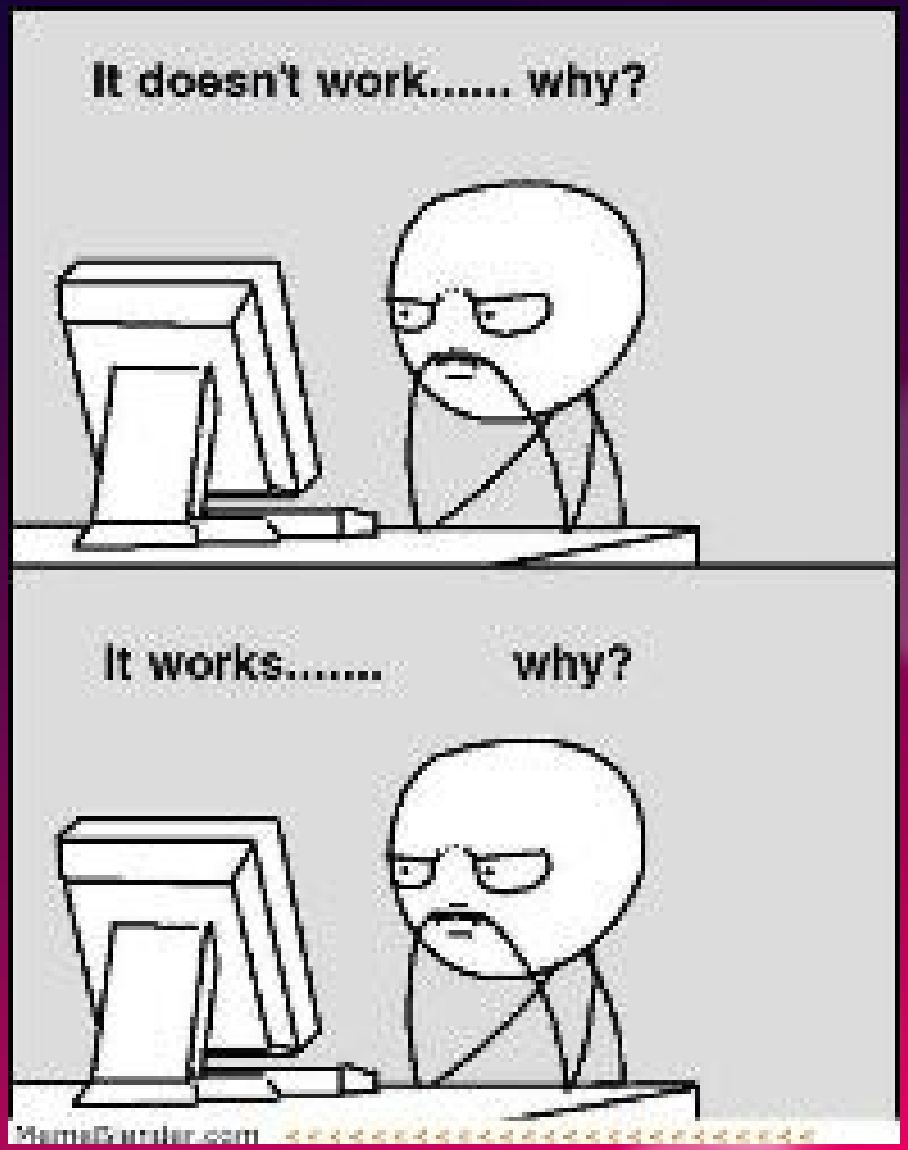
- Refactoring shaders – no copy-paste
- Add proper image loading
- Add material to the spheres
- Add spheres rotation
- Implement screen space ambient occlusion
- Add snowing
- Extend the terrain
- ...

Tips & Warnings

- Keep consistent units
- Plan unit size ahead when working with integers
- Don’t use magic numbers in shaders
- Try to check every OpenGL call
- Use some GLSL preprocessor to avoid copypaste
- Use VBO
- To get texture write access, use glBindImageTexture amd image2D instead of sampler
- In deferred renderer, create also specular buffer
- Keep geometry pass and G-Buffer passes usable, for debugging
- Structured SSBO – memory alignment!
- Don’t initialize camera to direction (0,-1,0) or similar
- OpenGL clamps values when rendering to GL_RGB type texture

The wisdom of the day

- GPU programming is HARD
  - No print
  - No debugger
  - No stacktrace
  - Errorcode if you are lucky
  - No direct memory access
  - Memory alignment problems



Q&A Time

- Some stuff:
  - 7 draw calls/dispatches
  - 19 shaders
  - 10 texture units
  - 17 unique uniforms
  - 23 configuration variables
  - 28.5 average fps (over 2 minutes)
  - 52.9 max fps
  - 21.9 min fps
  - 40 commits