Real-time Volumetric Fog Simulation

Milestone Presentation

Lucia Tódová

25.04.2020

Original steps and goals:

- Modeling (noise generation)
- Visualization (ray marching)
- Lighting (emission, absorption, scattering)
- Optimization

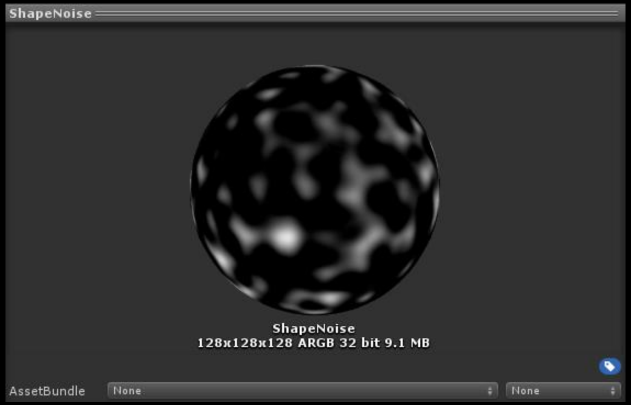
Done so far

- Implementation of Worley and Perlin noise
- Ray marching
- Part of light marching
- Unity GUI Inspector

Noise Generation

- Worley and Perlin noise functions
- Possibility of multiple octaves (Fractal Brownian noise)
- Setbacks:
  - Compute shader implementation and dispatching
  - Unity's RenderTexture and Texture3D classes and their conversion
  - Complicated testing in 3D
  - Tiling of the noises

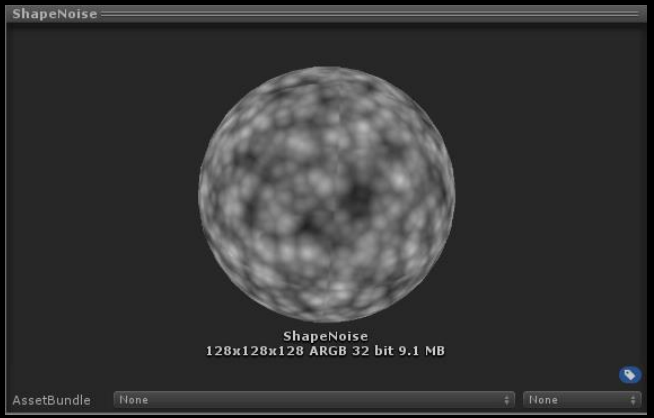
Perlin Noise



ShapeNoise
128x128x128 ARGB 32 bit 9.1 MB

AssetBundle None None

Inverted Worley Noise



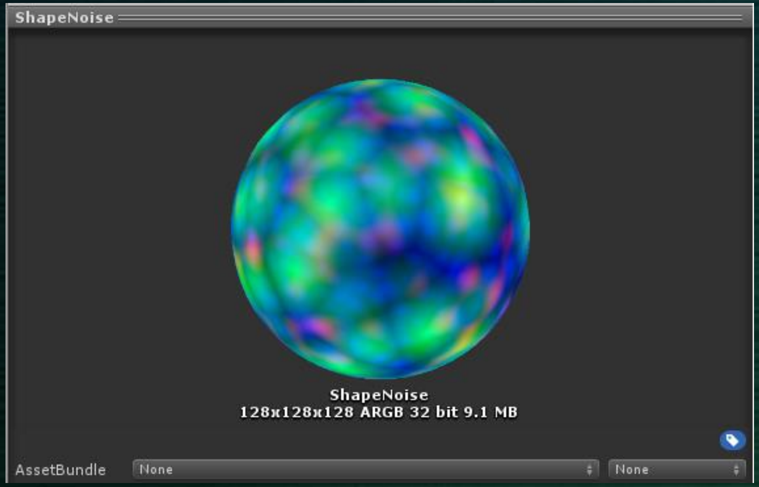
ShapeNoise
128x128x128 ARGB 32 bit 9.1 MB

AssetBundle None None

Noise Textures for Cloud Rendering

Shape Texture

- 128 x 128 x 128 resolution
- Red channel – Perlin noise
- Other channels – Worley noise
- Density: R * (B + G + A)

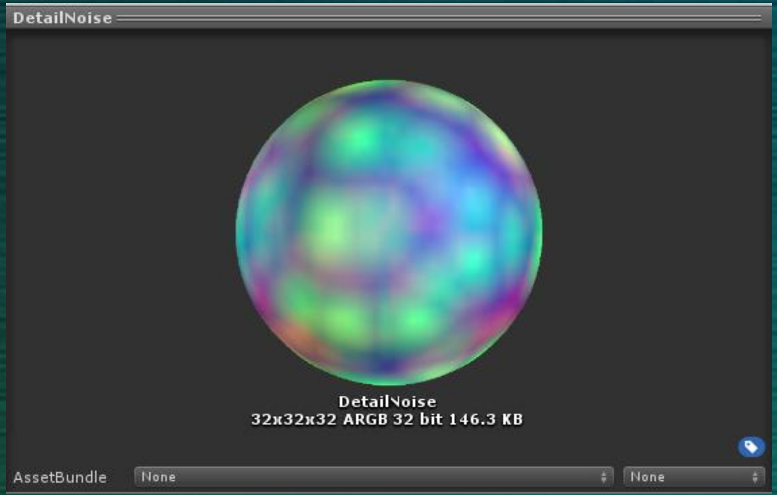


ShapeNoise
128x128x128 ARGB 32 bit 9.1 MB

AssetBundle None None

Detail Texture

- 32 x 32 x 32 resolution
- Alpha channel – 1 (max value)
- Other channels – Worley noise
- Density: (B + G + R)/4

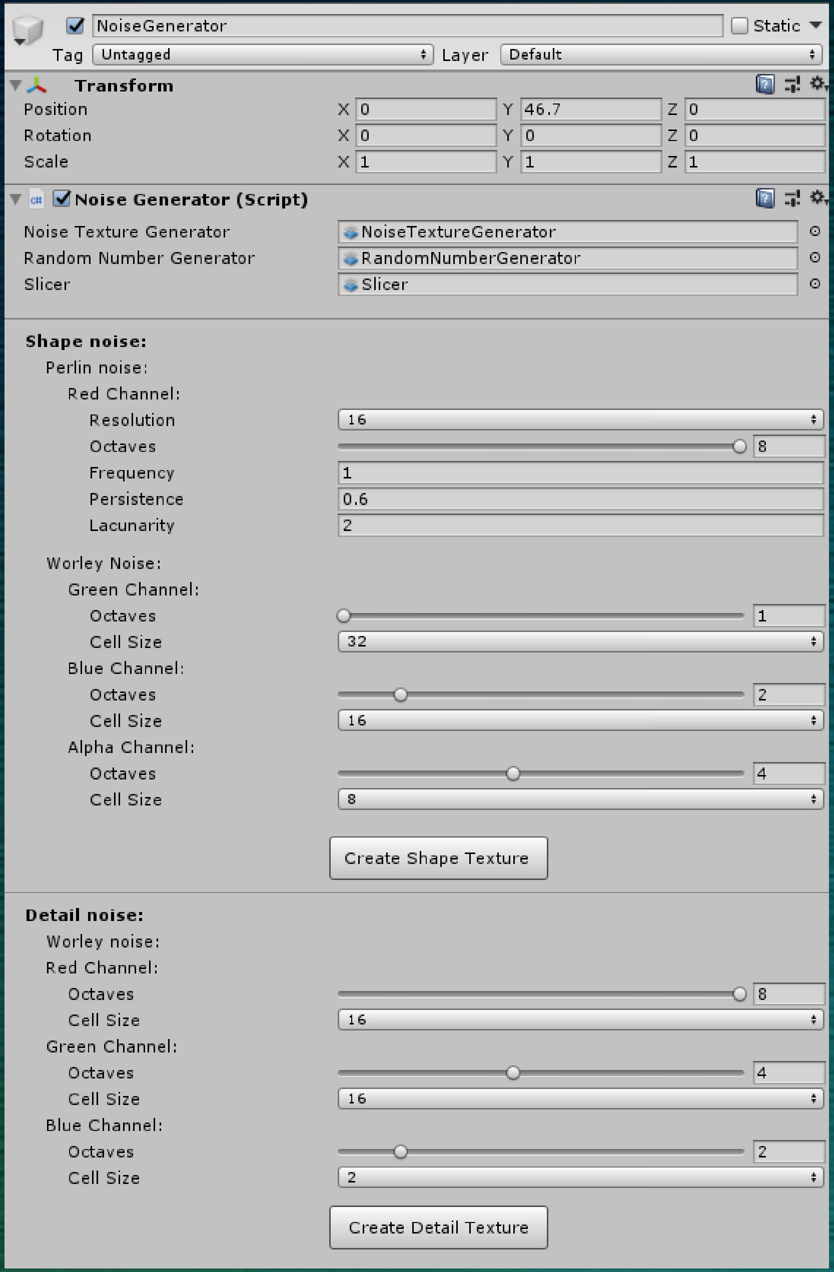


DetailNoise
32x32x32 ARGB 32 bit 146.3 KB

AssetBundle None None

Noise Generator Script

- Custom editor
- Shape and Detail noise independent of each other
- Saves the textures to the Resources folder
- Advice for creating textures:
  - Use varying octaves of Worley noise
  - Ideal cell size for Worley is 8/16



☑ NoiseGenerator □ Static

Tag Untagged Layer Default

Transform

Position	X	0	Y	46.7	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

☑ **Noise Generator (Script)**

Noise Texture Generator: NoiseTextureGenerator
Random Number Generator: RandomNumberGenerator
Slicer: Slicer

Shape noise:
Perlin noise:
Red Channel:
Resolution: 16
Octaves: 8
Frequency: 1
Persistence: 0.6
Lacunarity: 2

Worley Noise:
Green Channel:
Octaves: 1
Cell Size: 32
Blue Channel:
Octaves: 2
Cell Size: 16
Alpha Channel:
Octaves: 4
Cell Size: 8

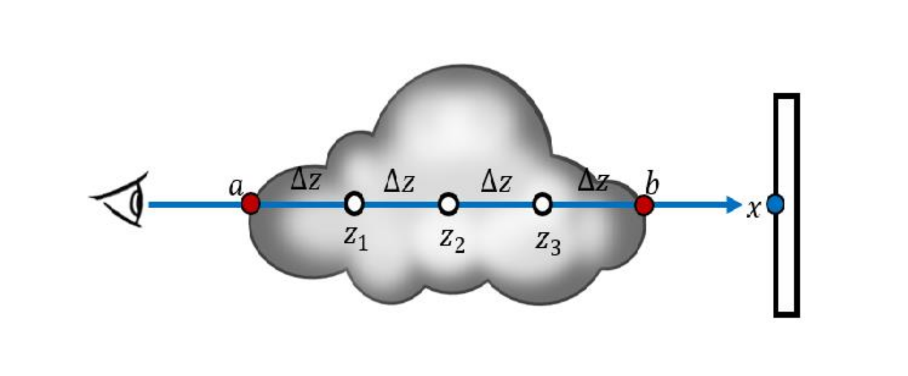
Detail noise:
Worley noise:
Red Channel:
Octaves: 8
Cell Size: 16
Green Channel:
Octaves: 4
Cell Size: 16
Blue Channel:
Octaves: 2
Cell Size: 2

Ray marching

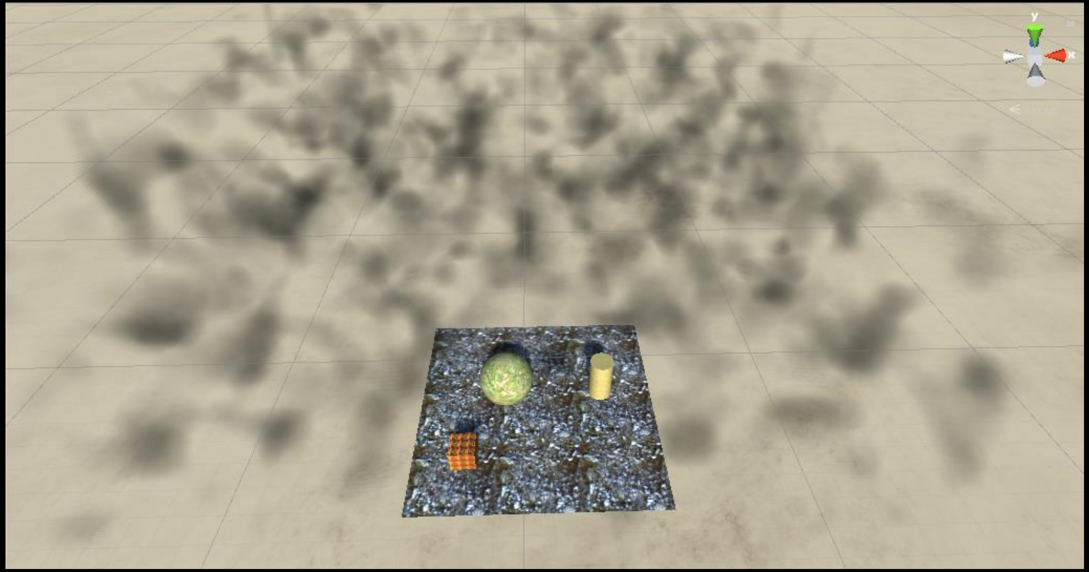
- March along a ray from A to B
- At each point, compute the density
- Sum densities up to get the density of point A
- Density computation:
  - Shape Texture Density – Detail Texture Density
  - Weight of the point along a ray determined by Beer's Law

$$T = \frac{I}{I_0} = e^{-\sigma_a l}$$

Transmittance at point dependent on absorption coefficient and current distance

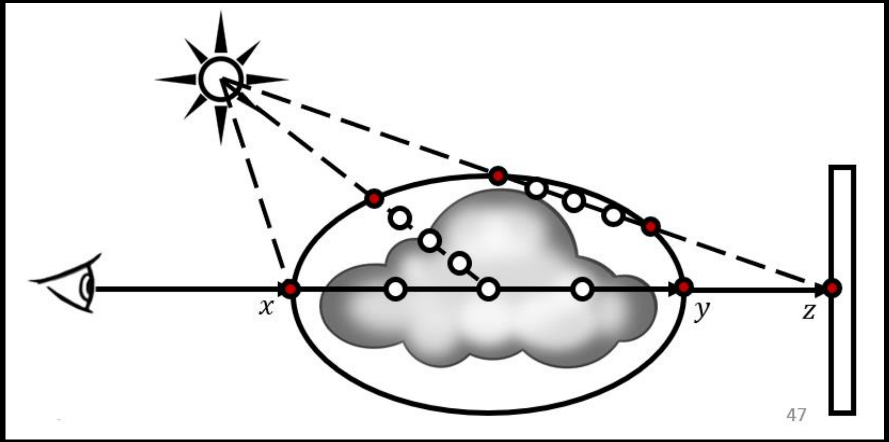


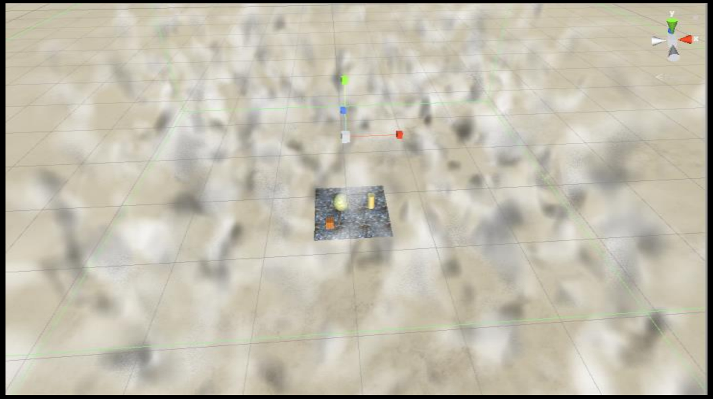
delta Z = step size; z1, z2, z3 – points of evaluation

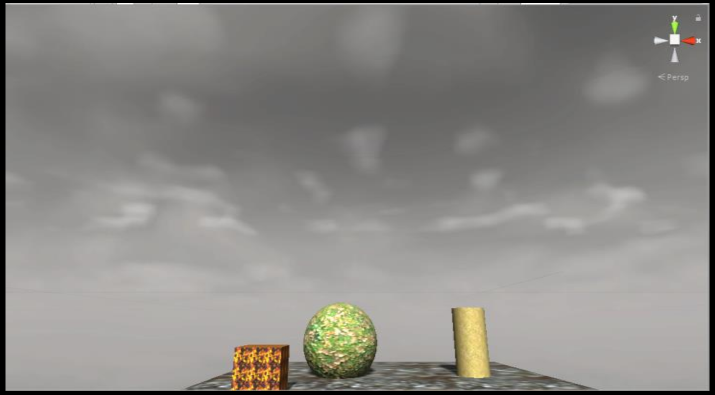


Light marching

- In progress
- Compute incident light at each point during ray march
- Ray march towards the light source
- Multiply the current density with total light

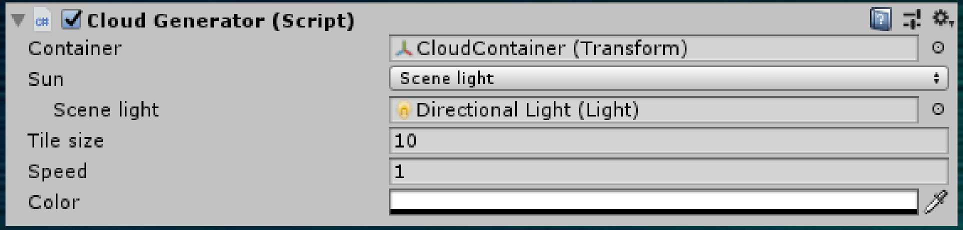






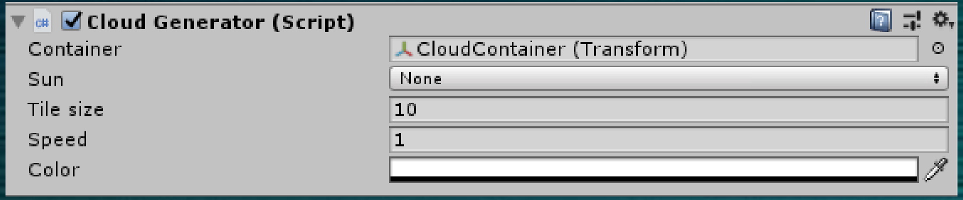
Cloud Generator Script

- In progress
- Option for choosing which light is used for light march
- Only one light to avoid performance problems



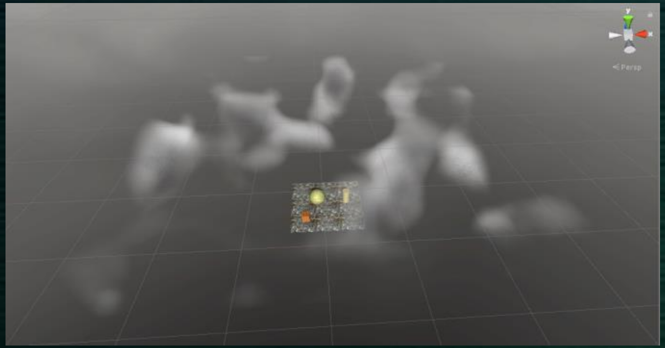
☑ Cloud Generator (Script)

Container	CloudContainer (Transform)
Sun	Scene light
Scene light	Directional Light (Light)
Tile size	10
Speed	1
Color	

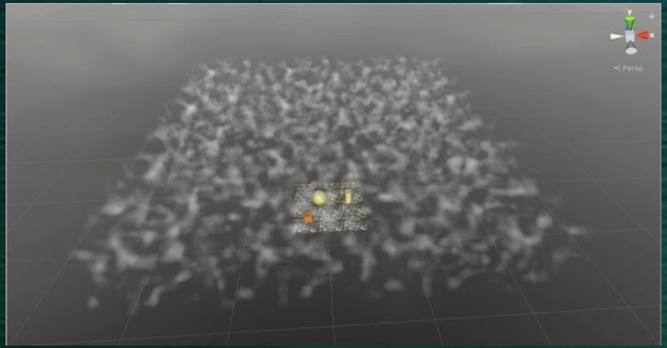


☑ Cloud Generator (Script)

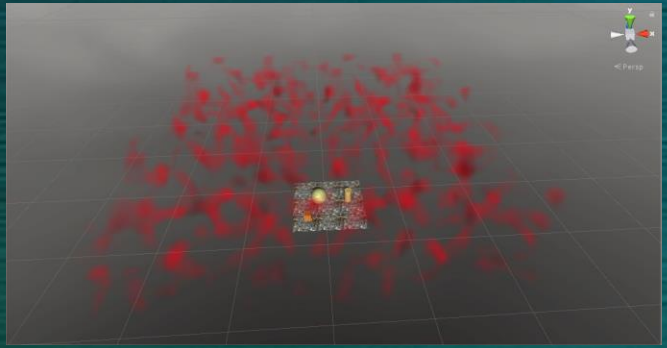
Container	CloudContainer (Transform)
Sun	None
Tile size	10
Speed	1
Color	



Tile Size: 2

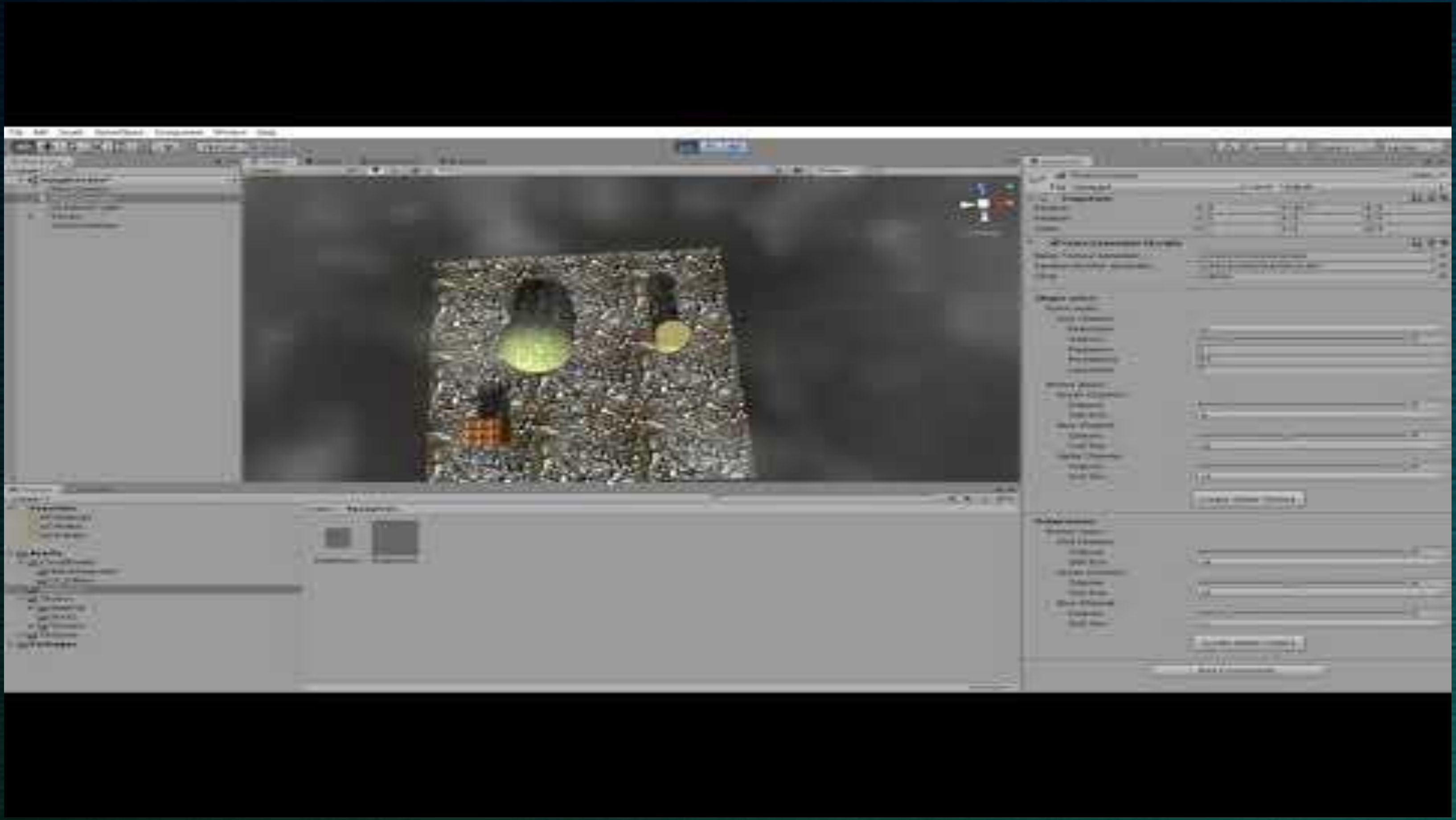


Tile Size: 10



Color: Red

Note: Color is not attenuated



Future work:

- Complete the incident light calculations
- Optimizations for Oculus Quest
  - Find optimal ray marching step size
- Scene light color and intensity dependent illumination

Nice to have:

- Weather map
  - Cloud density dependent on height
- Random number generation in Compute Shader
- Directional Light support
- User adjustable wind direction
- Better density calculations
  - Remap function

Thank you for your attention!

Lucia Tódová

25.04.2020