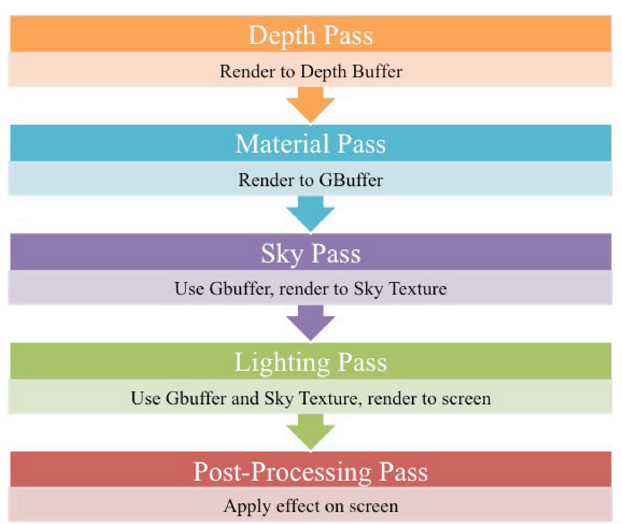
Realistic Procedural Environment with Atmospheric Scattering

Tae Hoon Kim, Raphael Marinheiro, Grant Mulitz, Sam Nelson, Lu Yang

CS 4621

December 15, 2014

Framework & Deferred Shading Pipeline

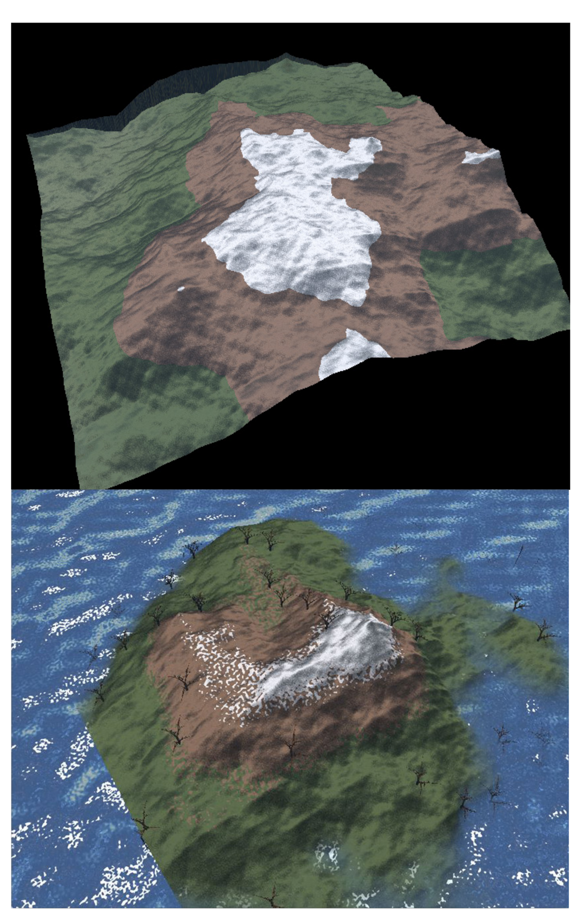


A visualization of the Deferred Shading Pipeline, detailing the sequential stages.

- We built our framework code using OpenGL in GLSL and C++. We chose C++ because it is reliably fast, and it is the standard when combined with GLSL for fast graphics rendering.
- We implement a Deferred Shading Pipeline as part of this project.
- As opposed to shading all at once, keeping separate parts of shading modular makes working on separate parts of the project more convenient.
- Our GBuffer stores 48 bytes per pixel: We accumulate the global position, the normal, the albedo and material parameters during the Material Pass and then we use these parameters during the lighting pass.

Terrain Generation

- We use Procedurally Generated Landscapes in our project
- We first generate a height map is generated using the Diamond-Square Algorithm, first introduced by Fournier et al in 1982.
- Given a height map, we define regions based on the height: water, grass fields, rocky cliffs and snow mountains.
- We apply a Perlin-based displacement to the normals add roughness to the surface and a Perlin-based noise to the Albedo.
- Added noise to the tops of the mountains to make the snow look more realistic, almost melted at some peaks.

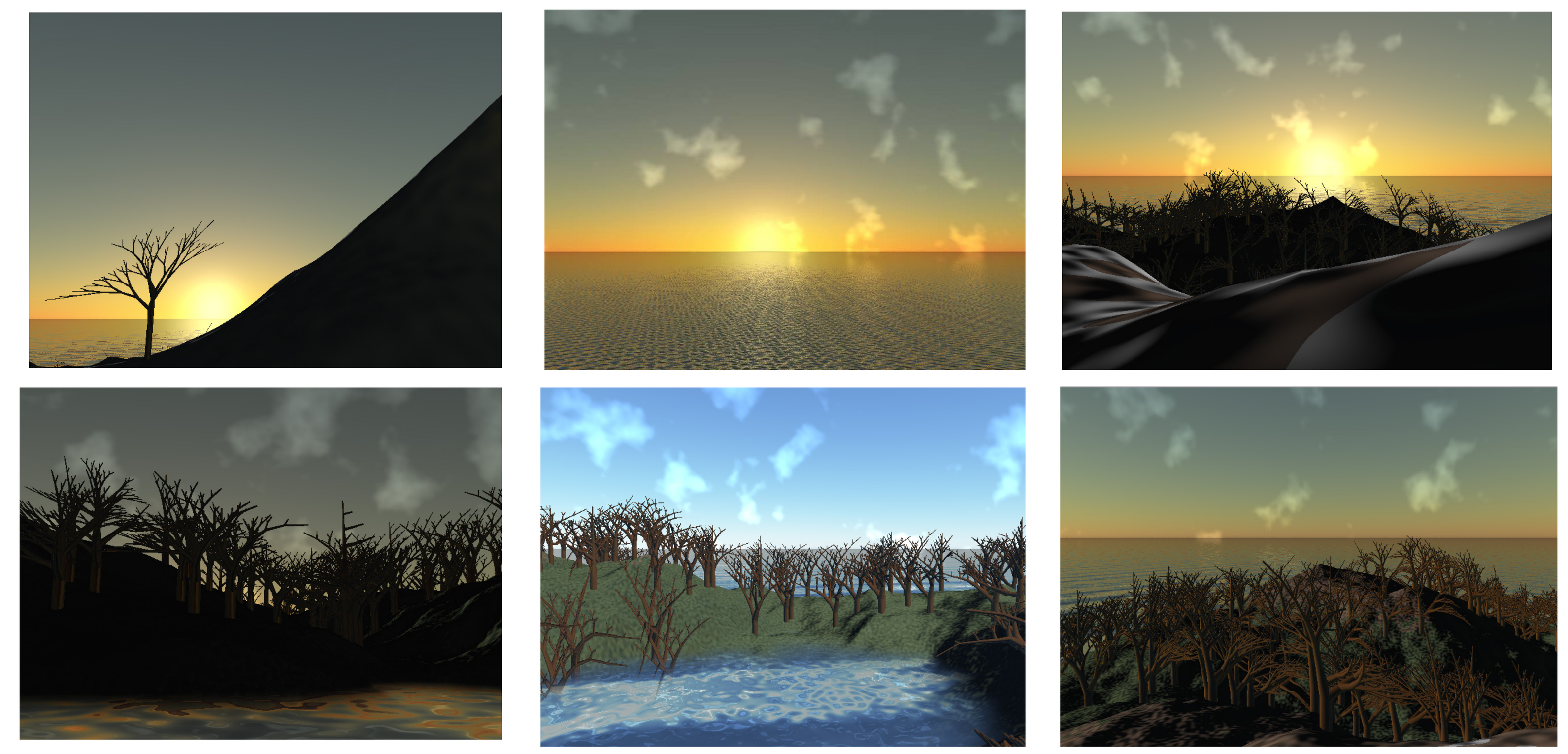


The Diamond-Square heightmap without water and snow noise (above), and with water and snow noise (below).

User Interaction

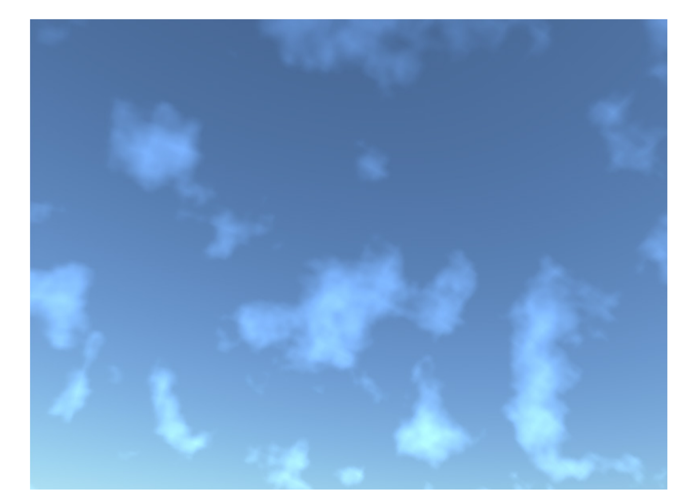
- The user can interact with the scene via the W/S keys (forward and backward) and mouse (directional).
- The user can change the position of the sun with the up and down arrow keys

Finished Product

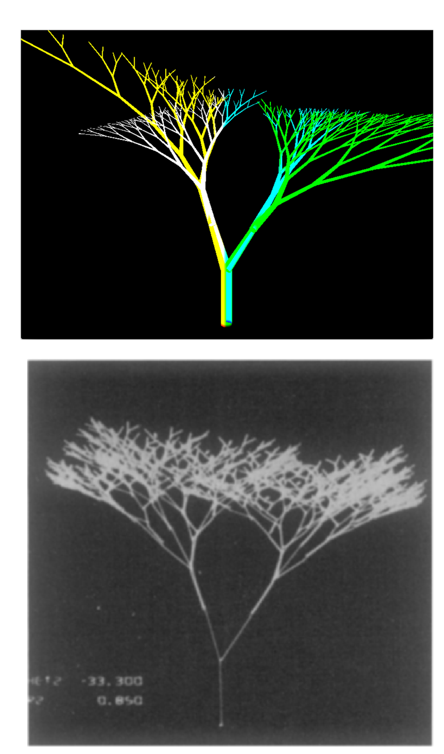


Cloud Generation

- Use Perlin Noise to create a noise texture.
- Noise texture smoothed and modified to wrap around seamlessly.
- We apply Inverse Stereographic Projection to the texture, and multiply it by the Sky texture to arrive at the cloud texture.



Tree Modeling

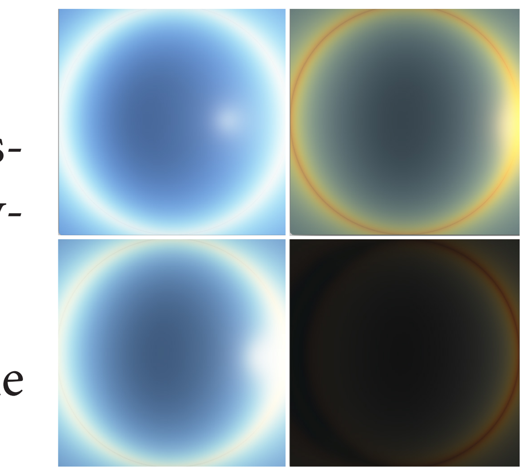


Above: The tree generated from our implementation
Below: The tree generated by Honda.

- Trees are generated procedurally according to a fractal pattern, based off of Hisao Honda's work describing the effects of branching angle and branch length.
- Branch angles, ratio of branch to children specified at start. Children branch in plane containing parent at that angle, with that length.
- Equivalent to deterministic L-system. Added randomization on the angles produces nicer effect.
- "Twigging": each branch is a conical frustum, shrinking from base to tip.This gives the appearance of branch children being smaller, but still connected.
- Base to tip ratio of a frustum is determined as $0.7(1 - 2^{-r})$, where r is the maximum recursion depth of the tree.

Atmospheric Scattering

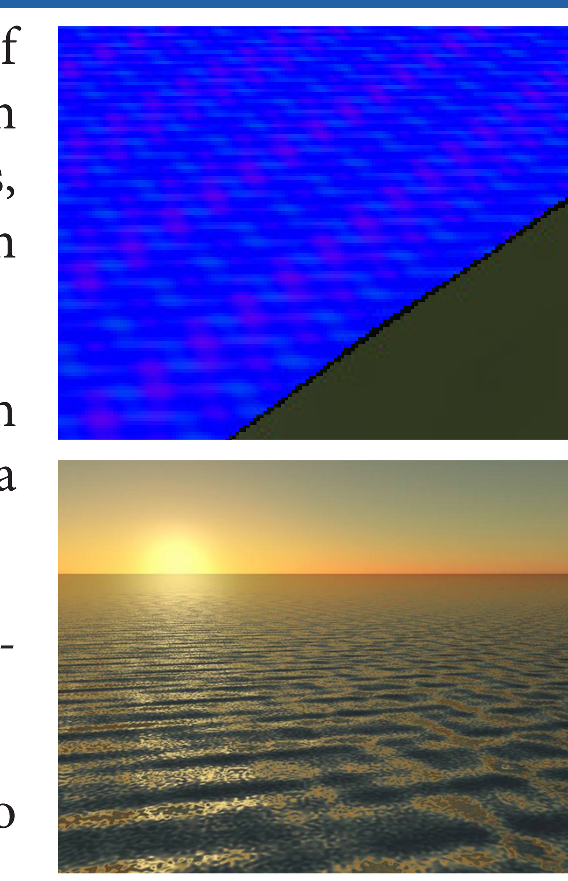
- We have implemented the Precomputed Atmospheric Scattering approach by Bruneton et Al.
- Using some precomputed Scattering and Transmittance tables, we are able to render a realistic Skymap without performance hits.
- We render the Skymap in a texture during the Sky Pass.
- During the Lighting Pass and Post-Processing Pass, we fetch the Skymap to add reflection effects.



Examples of the skymap for different positions of the sun

Wave-based Water Rendering

- The height of the water surface is a sum of 8 sine waves. Each is a function of xy position and time. We can specify number of waves, amplitude, wavelength, speed and direction of each wave.
- At each point, the partial derivatvies with respect to X and Y are summed to produce a normal vector.
- The normal vector is used for skybox texture coordinates.
- We add noise to the normals in order to make water look realistic.
- We specify the alpha value for the water in order to make it transparent at the shoreline.



Above: An illustration of the water normals, shaded as their intensity.
Below: The end result of applying the wave-based shader

Acknowledgements

- Bruneton, E. et al. (2008). *Eurographics Symposium on Rendering*. Vol. 27 No. 4.
- Fournier A. et al. (1982). *Communications of the ACM*. 371-384.
- Honda, H. (1971). *Journal of Theoretical Biology*. Vol. 31, 331.
- McEwan, I. et al. (2012). *Journal of Graphics Tools*. Vol. 16, No. 2, 85-94.