Practical Course Visual Computing

WS 2013/14

Exercise 5: "Planetary Atmosphere and Terrain"

Michael Becher

Topic

Realtime rendering of planetary atmospheres and terrain.

Task Assignment

The goal of this assignment is to implement terrain rendering, focusing primarily on rendering planetary atmospheres to create a believable sky without the use of a high-res skybox. Although the implementation should be capable of rendering a planetary atmosphere from space, terrain rendering will be limited to a small, static area using a height-map based approach. The focus remains on rendering the planet's atmosphere by simulating/approximating Rayleigh and Mie scattering.

Introduction to space-lion

In order to concentrate on the core features and challenges of the assignment, a basic OpenGL framework/engine is used as a starting point. To that end, I created a new branch of the space-lion framework for this project, which is publicly viewable at `https://github.com/invor/space-lion/tree/fapra`. For the sake of completion and the for reviewers of this assignment, the notable features of space-lion are listed below.

- Context and window creation
- Wrapping/implementation of OpenGL core concepts, e.g. GLSL shaders, vertex buffer objects, texture objects and framebuffer objects
- Basic resource management including loading and management of shaders, textures, meshes and custom materials

- Basic scene handling
- Basic scene rendering using physically based shading
- Ability to add post processing steps to the render pipeline

As these basic features are mostly unrelated to the task given by this assignment, atmospheric scattering will be a noticeable and useful addition. Stability and reliability of space-lion has been previously field-tested by me in the sea-crossing[1] project.

Goals

1. Basic terrain rendering.
2. (Bonus) Use hardware tessellation to increase terrain detail. Skip this if it takes to long and gets in the way of more important goals.
3. Textures are applied to the terrain. Optimally this would include a diffuse albedo, specular colour, roughness and normal map.
4. Scattering integrals are precomputed at program start and stored in 2D/3D textures.
5. The atmosphere/sky is rendered using the precomputed lookup-tables.
6. Variable sun position for day/night cycles.
7. (Bonus) The sky is looking good, but the terrain is still a bit blank. Place a few static meshes in the scene.
8. (Bonus) Animate the camera to follow a path to create a benchmark scene.

[1] `https://github.com/chaot4/sea-crossing`

Modules

Module A: Atmosphere/Terrain Rendering

Check `src/fapra` and `resources/shaders/fapra` for the source code.
This module is expected to contain the main/render loop, classes for terrain and atmosphere, as well as a class for a planetary scene that features terrain and sky.
Furthermore, shaders necessary for pre-computing and rendering the atmosphere are part of this module (the terrain surface will probably use the space-lion default surface shader).
Files likely to exist are: `fapra_renderHub.{h,cpp}`, `planetaryScene.{h,cpp}`, `terrain.{h,cpp}`, `atmosphere.{h,cpp}`, `sky_{v,f}.glsl` ...

Module B: Space-Lion Core

Already existing module. Check `src/engine/core` for the source code.
Modifications to the sources in this module are expected depending on problems and requirement that will arise in the course of this assignment.

Module C: Space-Lion Fbx

Already existing module. Check `src/engine/fbx` for the source code.
Changes or frequent use of this module are not expected in the scope of this assignment.

Technical Overview

The application is written in C++ and makes heavy use of C++11, thus requiring a very recent compiler (e.g. Visual Studio 2013's compiler). A Visual Studio 2013 solution file is supplied with the project. Beware that the Linux makefile hasn't been touched in ages. While the application should run fine on Linux (sea-crossing is being developed on both Linux and windows), the makefile has to be updated/rewritten first.
A minimum of OpenGL version 3.3 is currently required/recommended and will be updated to version 4.x with the use of tessellation shaders. For the

OpenGL context creation and window handling, GLFW [2] -a lightweight, open source, portable framework for OpenGL application development- is used. Additionally, glew [3] is used to get access to the most recent OpenGL functionality. Out of convenience, space-lion currently uses glm [4] for vector and matrix calculations.
(At a later point, a readme will contain the details regarding the expected directories for include and lib folders.)

Approach

Start by pulling the latest commit from the fapra branch. See to it that all relevant classes of the framework are properly working.
To begin with, the RenderHub and Scene class need to be extended, since neither of them currently support handling and rendering of special scene elements, such as the terrain and the sky will be. Derive a new class FapraRenderHub and PlanetaryScene. Now create a class Terrain that contains the data and functionality for a height-map terrain and add an instance of it as a new member to the PlanetaryScene class. Do roughly the same for the sky/atmosphere.
Implement any method necessary in FapraRenderHub, PlanetaryScene and Terrain for rendering the terrain using instanced quads (space-lion already uses instancing per default if several objects share the same mesh and shader, but adding all quads for the terrain as separate objects doesn't seem like a good idea).
Parallel to implementing terrain rendering, research the details for rendering the atmosphere using -among others- the references listed below. As soon as the terrain reaches an acceptable stage (preferably by the beginning of the second week), start to implement the methods for pre-computing the necessary look-up tables (i.e. textures) using either CPU- or GPU-based implementations.
Following that, implement the methods for actually rendering the sky (this should happen some time during the third week).
If all normal goals are met by the fourth week, implement the remaining bonus features.

[2] `http://www.glfw.org/`
[3] `http://glew.sourceforge.net/`
[4] `http://glm.g-truc.net/0.9.5/index.html`

References

[1] S. Sperlhofer, *Deferred Rendering of Planetary Terrains with Accurate Atmospheres*, Master's Thesis: Vienna, 2011.

[2] O. Elek, *Rendering Parametrizable Planetary Atmospheres with Multiple Scattering in Real-Time*, CESCG: 2009.

[3] O. Elek, P. Kmoch, *Real-time spectral scattering in large-scale natural participating media*, SCCG: 2010.

[4] E. Bruneton, F. Neyret, *Precomputed Atmospheric Scattering*, Eurographics Symposium on Rendering: 2008.

Criteria of Grading

The overall 20 points given for this assignment are distributed to the following criteria:

Compulsory Criteria

1 Point	An original concept is handed in. The originality will be illustrated by two high resolution screenshots of the final product.
1 Point	The code is well structured and documented.
1 Point	The `proposal.tex` meets all requirements.
1 Point	The `readme.pdf` file contains a Section *MANUAL* that describes in detail the usage of the program.
1 Point	The `readme.pdf` file contains a section *IMPLEMENTATION* that describes the fundamental parts of the implementation.

Student's note: There already exits a README file for space-lion, which wouldn't be really suited for containing the information mentioned above. Therefore I would like to create an additional readme document (in Latex) and place it in the same directory as this document.

Self-defined Criteria

3 Points	Basic terrain rendering.
2 Points	Textured terrain.
2 Points	(Bonus) Tessellated terrain.
5 Points	Pre-computation of scattering integrals.
3 Points	Sky rendering with atmospheric scattering.
2 Points	Day/night cycle.
1 Point	(Bonus) Decorative objects are added to the scene.
1 Point	(Bonus) Benchmark sequence with animated camera.