3D Nature Environment with Procedurally Generated Terrain and Trees

Liam Ozog
lozog
20515121

Final Project:

Manual :

Build :

Building this project is largely similar to the previous CS488 assignments. First, run premake in the top level CS488 project directory:

```
premake4 gmake
make
```

There is an additional library, SOIL, used for image loading. Copy the file *libSOIL.a* from the A5/ directory to the lib/ directory.
Next, build the actual project. Navigate to the A5/ folder and compile:

```
cd A5
premake4 gmake
make
```

Program Input :

Usage of the program is as follows:

```
./A5 [ input-param-file ]
```

Specifying the input file name is not required. If it isn't supplied, the program will load the file **inparams1**, which *must* exist.

The input file is used to specify program parameters. I chose not to use Lua because the input is simple enough to not require it. The input file assumes well-formed input. Each line contains a parameter name, then a value. Depending on the parameter, the value might be a number or a vector. Lines beginning with a # are ignored. The input system recognizes the following parameters:

- TERRAIN_SIZE
- WATER_HEIGHT
- NUM_OCTAVES
- REDIST
- GROUND_TEXTURE
- TREE_DENSITY
- GRASS_DENSITY
- GRASS_COLOUR
- LEAF_COLOUR
- SKYBOX_NAME
- SUN_DIRECTION
- SUN_COLOUR
- SUN_INTENSITY
- AMBIENT_COLOUR
- CAMERA_POS
- CAMERA_FRONT
- CAMERA_PITCH
- CAMERA_YAW
- CAMERA_SPEED
- LSYSTEM

The LSYSTEM parameter takes an integer n (n >0) that specifies the number of production rules associated with that L-system. The following n lines are then read as production rules.

Controls :

The program employs a standard first-person camera (controlled by the mouse) with the following keyboard shortcuts:

- W - Forward
- A - Left
- S - Backward
- D - Right
- Q - Down
- E - Up
- Z - Speed down
- X - Speed Up
- R - Reset camera
- P - print position information

In addition to the camera controls, the user may also alter some of the program parameters at runtime, and the results are immmediately applied. Here are the relevant keyboard shortcuts:

- 1 - Increase the Sun's X direction
- 2 - Decrease the Sun's X direction
- 3 - Increase the Sun's Y direction
- 4 - Decrease the Sun's Y direction
- 5 - Increase the Sun's Z direction
- 6 - Decrease the Sun's Z direction
- O - Increase the number of octaves used by terrain generation
- I - Decrease the number of octaves used by terrain generation
- G - Raise the water level
- H - Lower the water level
- M - Increase the distribution exponent used by terrain generation
- N - Decrease the distribution exponent used by terrain generation

Finally, the **B** key toggles display of the shadow map, and the **ESC** key quits the program.

Statement :

For my project, I intend to render a 3D environment with terrain, procedurally generated trees, and reflective water. Much of the inspiration for this project comes from David Whatley's chapter in GPU Gems 2, Ṫoward Photorealism in Virtual Botany; Many of the topics covered in that chapter are beyond the scope of this project, but it did pique my interest in nature scenes. The other inspiration was Ryan Geissćhapter in GPU Gems 3, Ġenerating Complex Procedural Terrains Using the GPU; which got me interested in procedural generation.

This project is more advanced than any real-time program I have ever attempted, so it will be an interesting challenge to try to achieve the objectives while maintaining a reasonable framerate. I also hope to achieve some interesting, perhaps even life-like trees when implementing L-systems to generate tree models.

All of these objectives are completely new topics to me. I've been playing videogames for much of my life, and am familiar with many of these effects in terms of the end result (how they look), but I haven't given much thought as to how they are implemented. A lot of ground will be covered - procedural

generation with L-systems, mapping of height and texture, and interesting lighting effects. As I noted above, this will be the most advanced real-time program I have attempted, so I expect to learn a lot in that area.

Technical Outline :

The terrain will be modelled from a height map, with a grayscale gradient representing height. The height map will be generated using Perlin noise.

I will implement a skybox in the scene using the technique of cubemapping.

The landscape will have water up to a fixed height. Using OpenGL's stencil buffer, I plan on adding reflections to the water.

The environment will be populated with foliage - grass and trees.

Since it's unrealistic to render each individual blade of grass, I will use billboards to render ćlumpsóf grass. Additionally, I plan on using a ścreen-dooréffect (also known as a dissolve effect) to simulate alpha transparency for grass - since calculating the attentuation of alpha transparency for the grass as distances increases is expensive.

I plan on making use of Lindenmayer systems to procedurally generate tree models to make the world seem more l̈ife-like.

The ground and models (grass and trees) will have textures applied to them using texture mapping.

Finally, I plan on making the scene look nicer (for some definition of nicer) by implementing high dynamic range (HDR) rendering and bloom effects, which both make use of framebuffers.

3D Nature Environment with Procedurally Generated Terrain and Trees

Name: Liam Ozog
User ID: lozog
Student ID: 20515121

Objectives

___ **1:** UI: Implement a first-person camera with associated controls to allow navigation of the scene, including movement in 3 axes, speed adjustment, and camera rotation.

___ **2:** Modelling: Add a skybox to the scene using cube mapping.

___ **3:** Implement reflections for water using OpenGL's stencil buffer.

___ **4:** Generate a pseudo-random terrain heightmap with Perlin noise.

___ **5:** Add grass to the scene using billboards to create the illusion of many blades of grass.

___ **6:** Add texture to the ground and foliage using texture mapping.

___ **7:** Use L-systems to procedurally generate trees.

___ **8:** Implement shadows using a depth map stored in an OpenGL frame buffer.

___ **9:** Implement bloom using framebuffers and Gaussian blur.

___ **10:** Objective ten. Use a "screen-door" effect to simulate alpha transparency for grass.