A5 Project Proposal
Title: Forest Scavenger Hunt
Name: Orson Baines
Student ID: 20652798
User ID: obaines

Final Project:

Purpose :

To present some advanced graphics effects in a fun, interactive scavenger hunt game.

Statement :

My project will consist of a scavenger hunter game where the user will be given a list of clues and need to locate the required items. The user interface will be modelled after open-world adventure games like Elden Ring and Skyrim, with movements and actions controlled by the keyboard and the view controlled by the mouse. There will be a clock to show in-game time, which will control the direction of the sun and the amount of ambient light. There will be a slider or key-bindings to control the ratio of real-world to in-game time, which by default will be about 1 min : 1 hour (like Terraria). All objects will cast shadows.

The terrain will consist of a heightfield for the ground along with trees, rocks and puddles as static terrain. There will also be at least one campfire producing smoke.

As additional controls the user will be able to enter binocular mode where the field of view is reduced to a very tiny angle, and a flashlight mode that adds a spotlight to the scene pointing in front of them. Binocular mode will add crosshairs and the user can click to pick an item. If the item is on the scavenger hunt list then they will be notified and the item will be crossed off.

This format of game will be both challenging and interesting, because it will give me lots of room for creativity in modelling the scene, and designing the list of clue and objects to spot. It gives me the chance to use lots of graphics techniques including shadow mapping, texture mapping, bump mapping, transparency, Perlin noise, and others.

For libraries I will use SDL2 for window and OpenGL context creation, as will as for sound. I will also use GLM for matrix math, and GLEW for loading the OpenGL functions and extensions.

Technical Outline :

- **Terrain Heightfield:** I will use Perlin noise to generate a greyscale bitmap. Then in the vertex shader for the terrain I will read the texture value for the given point and use the gray scale texture to calculate the height at that point.
- **Bump Mapping:** I will use bump mapping to add a realistic surface to the tree trunks. I plan to create in software (GIMP, etc.) the heightfield for the surface of the trunk (with black lines in the grooves of the bark) and then calculate the texture derivatives to modify the normals. This will also rely on a cylindrical projection to map the vertices into UV coordinates. I'm referring to a tutorial on normal mapping [1], which is slightly different since it uses a normal map instead of a heightfield, but lots of the same ideas are still applicable.
- **Texture mapping:** For the objects I am modeling I will use at least one of each of the 3 generic projector functions we used in class (cylindrical, spherical, planar). Depending on the model, I will probably do some of the mappings in blender and some of the mappings in code. I will compress my RGB/RGBA textures using the DXT formats (DXT1, DXT3, and/or DXT5) which are supported by an ubiquitous OpenGL extension `GL_EXT_texture_compression_s3tc` and write code to load them from the .dds file into OpenGL.
- **Shadow Map:** I will need a shadow map for each light source, one for the sun and one for the flashlight. Since these lights are constantly changing, they will need to be updated each frame. This will involve creating a grey scale texture for each shadow map, and assigning the textures to a framebuffer, then binding the framebuffer an rendering the scene from each light's point of view with that light's framebuffer bound. To prevent aliasing effects I will need to use a bias to make the shadows slightly deeper than how their recorded value, and also use front face culling.

- **Transparency:** The smoke particles and water will be partially transparent. I will achieve that with alpha blending, which I will need to enable in OpenGL.
- **Particle System:** I will use a particle system to simulate smoke. [2] gives some examples on how to do a particle system in OpenGL, and [3] is a demo particle system.
- **Modelling the Scene:** I plan to use a combination of meshes I design in Blender exported to .obj files and simple geometric shapes to create the scene.

Bibliography :

References

[1] J. D. Vries. "Normal mapping." (), [Online]. Available: `https://learnopengl.com/Advanced-Lighting/Normal-Mapping`. [Accessed: Jun. 23, 2022].

[2] "Particles / instancing." (), [Online]. Available: `http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/`. [Accessed: Jun. 23, 2022].

[3] "Smoke particle system." (), [Online]. Available: `https://processing.org/examples/smokeparticlesystem.html`. [Accessed: Jun. 23, 2022].

Objectives:

Full UserID:_____ **Student ID:**_____

___ 1: Modelling the Scene.

___ 2: Use Perlin noise to generate terrain heightfield.

___ 3: Apply Bump Mapping using heightfield.

___ 4: Apply Shadow mapping with 2 light sources.

___ 5: Reflection using stencil buffer.

___ 6: Transparency with alpha blending.

___ 7: Apply Texture Mapping.

___ 8: Use skyboxes (one for day and one for night).

___ 9: Key frame animation.

___ 10: Static Collision Detection.