**Term Project Report**

Christophe Savard

40017812

COMP-371 WW

For: C. Poullis & S. Mudur

Concordia University

April 12th

**The World**

The world space containing the project is made rather exactly as per the submission guidelines. The grid/plane is sized 100 units per 100 units, centered at the world origin, giving 50 units in every direction from the center. The world grid is normally composed of straight lines at 1 unit of interval, and do not cast shadows in this configuration, but when textures are applied, each unit square is represented as a pair of triangles rendering the texture, and cast a shadow is any object should be underneath it.

The X, Y, and Z axis are rendered with a single line, centered at the origin, with the X axis being red, Y axis being green, and Z axis being blue. The axis themselves are the only world objects to not be subjected to lighting, meaning they neither are subjected to the light source, nor do they cast shadows.

The light source is located 20 units above the world origin, and emits light in a point fashion using the Phong model. Different materials constants are implemented depending on the surface interacting with the light.

The shadows are implemented using a 3D Cube Map, rendering the point origin shadows as accurately as possible, with shadow edge smoothing (PCF) making the edges as smooth as possible in a simple fashion.

The camera’s control is slightly modified from the propositioned control to allow a better personal control flow. Instead of rotating the scene, the arrow keys translate the camera around the grid by increments of one unit on the X and Z axis. The rotation around the Y axis (left and right) is done by holding down the right mouse button, and the rotation around the X axis (up and down) is done by holding down the middle mouse button.

**The Horse**

The horses are composed of the required 11 cubes assembled in a hierarchal fashion, with bendable joints between each cube.

The mesh itself is loaded and assembled via a custom format implemented through a .mesh file, which indicates how to position, scale, and rotate each unit cube, and the parent/children links between them. This is merely directives to assemble the mesh through code rather than fixed hardcoding, and allows quick and easy modifications of the mesh without recompiling. Each horse is given a random position and rotation on spawn, and it is make sure that two horses are not already colliding in their spawn locations.

The animations are handled using a timer. Each horse has a set random speed, walk distance, turn speed, turn angle, and wait time, which are all carried in the fashion walk, turn, wait, repeat.

The collisions are handled via a spherical collider around the horse. Given the simple and slow movements, and to give the impression of a set distance before the horses stops to handle the collision rather than to just hug each other every time, this works very well in this situation, and means very little computational overhead to detect collisions. When a collision is detected, the horse the earliest in the rendering queue will rotate until it can move forward, then proceed normally, while the other colliding horse waits for the first to move out of the way.

**References**

[learnopengl.com](https://learnopengl.com/) – OpenGL implementation guides, shaders, lights and shadows implementations, etc.

[stackoverflow.com](https://stackoverflow.com/) – General C++ algorithm hints and implementations

Both textures are sourced from the Public Domain.