Computing for Animation Assigment Documentation - First Person Game
by Sergiy Abramov
s4914790

Introduction

For this assigment, I have set a task of creating a First Person Game - ideally an FPS, or First Person Shooter, with simple AI enemies, collision detection, first person camera, a simple level and working particle or raytracing sphere to sphere collision detection for shooting.

Implementation

During the process of creating the game, I have abandoned it many times to create other ideas that didn't work, and in the end I had very little of the features implemented or working. In this section I'll describe the workflow and how the current code works, and then describe how the features that I planned but did not implement should've ideally worked.

System Flow

The main.cpp creates a window and specifies GL format. NGLScene then draws everything else and interacts with all of the classes. InitializeGL function creates and loads all the shaders, loads the camera and its properties, loads the objects. A timer is started that checks for mouse movement and keyboard key presses, which are handled in their own functions, and the key commands that move the camera are handled in PaintGL. PaintGL draws the desired picture every few miliseconds, drawing objects with specific texture packs that are assigned to them, and also draws 4 spheres where the light source is coming from.

First Person Camera

NGLSceneMouseControls processes all the stored coordinates whenever the mouse is moved. FirstPersonCamera class has all the functions needed to set the camera's parameters and has functions that change the camera's vectors and position when they are called in the NGLScene when the mouse or keyboard triggers are activated.

The camera was also supposed to handle an FPS-style movement, where the player could only walk and not "fly", however I later found out it is impossible to implement without rewriting the entire camera to work that way.

Level Loading and Shading

The level is divided into 4 different objects that are loaded upon GL Initialization and are given VAOs, then the matrices and a set of textures are loaded onto the shader, which draws the objects with their respective textures. The textures are used to give color, roughness, metallic state and other parameters, and the shader also draws a light source

that hits the object. Originally a shader with environmental reflections was planned to be used, but ambient light has proven sufficient and actually looked better.

Collision Detection for Camera and Spheres

The collision detection runs every few moments based on a timer, and, in the case of the sphere, should send it on a different course if it finds a collision with the bounding box or other spheres. In the case of the camera it should've stopped the camera from moving. In the end it detected whenever collided with the bounding box, but didn't do anything else.

Gun Implementation

The gun would've been drawn very close to the camera and would've moved with it. Whenever a specific button would be pressed, a sphere would've been drawn with a direction vector set to be going straight from the camera. The sphere would be a bullet, and could be textured or be made into a model of a bullet later with ease in its class, given the right attributes.

Further Implementation

Once the basics would've been finished, I would've worked to add in a menu, UI, enemies in the form of target dummies, and interactions that would give the player a point each time they scored a hit on the dummies.

Conclusion

Overall I'm disappointed in the final work that I have produced. It does not fulfill the original goals, and can not be considered a game or satisfactory. I've faced difficulties with this project, but most of them stem from the fact that I did not devote nearly enough time to it and was indecisive during its production.

I've tried out several ideas such as Pong and Space Invaders before returning to the idea of a first person game, which took a lot of time away from me and I did not get a good grasp of NGL or OpenGL coding as a result. Most of the techniques I understood and used were for shading.

Althoguh unsuccessful this time, I learned from my mistakes and will be more decisive in the future, as well as put more time into programming to get better results.