Pinball

Programming Assignment 10

CS 480 Computer Graphics

Team: Specular Bois
Members: Hadi Rumjahn, David Valenzuela, Jay Woo

Date Demonstrated
November 16, 2018

Table of Contents

Overview

Project Description

For this project we implemented a pinball simulation game. When the program runs a full pinball machine will be loaded in. The pinball machine includes models for a ball, board, two flippers, cylinder bumpers, a vertical back splashboard, and plunger. Each model has a texture to add more visual detail to the object. The viewpoint can be moved by pressing keys on the keyboard. A score is presented to the player via an in game menu and increases when the ball hits the bumper. The ball has proper physics interaction when colliding into other objects. Gravity causes the ball to always move downward. Also the ball never leaves the board. The game also features per vertex and per fragment lighting which can be changed during runtime. Ambient and specular lighting properties are adjustable. A spotlight continuously hovers above the ball.

Dependencies

For both of the operating systems to run this project installation of these three programs are required GLEW, GLM, and SDL2.

This project uses OpenGL 3.3. Some computers, such as virtual machines in the ECC, can not run this version. In in order to run OpenGL 2.7 follow the instructions at Using OpenGL 2.7

This project also uses Dear ImGui (https://github.com/ocornut/imgui), but all the necessary files are already included in the project.

This project also uses the Assimp, ImageMagick, and Bullet physics libraries.

Extra credit

Sound

Our pinball project has sound that plays while it is running. The sound starts once the program starts and continuously runs as background noise while the program is running. Adding sound contributes to a more fun player experience when playing our pinball simulation game.

Plunger Intensity changes via kbd

The plunger intensity increases the longer the player holds down the spacebar key on the keyboard. This means that the longer the player holds down the spacebar, the more power will be used to launch the ball. Also when the player holds down the spacebar, they will be able to see the plunger being pulled back. Once the space bar is let go the plunger will move forward and hit the ball.

User Manual

Build Instructions

Download the PA10 project folder from the following website, https://github.com/david4jsus/cs480Valenzuela. Open the terminal in the PA10 folder by double clicking the PA7 folder, right clicking inside the folder, and select the open terminal here option. Inside the terminal enter the following commands shown below.

```
mkdir build
cd build
cmake ..
make
```

Run Instructions

After building the project open the terminal in the build folder. To run the project enter the command ./Pinball into the terminal.

Keyboard Input

Camera movement

The camera can be freely moved throughout the environment using keys on the keyboard. Below is a list of keys which control camera movement.

W: Move forward
S: Move backward
A: Move left
D: Move right
Q: Move down
E: Move up

Up arrow: Rotate view upwards
Down arrow: Rotate view downwards
Left arrow: Rotate view left
Right arrow: Rotate view right

Figure 2. Two pictures of the pinball machine presented at different views.

Launch ball with plunger

The ball can be initially launched by pressing the spacebar. The ball will be launched with more power as the spacebar is held down longer.

Spacebar: Launch ball forward (can be held down)

Lighting

Lighting can be changed between per fragment lighting and per vertex lighting during runtime by pressing the tab key. The program starts on per fragment lighting. Ambient lighting can be adjusted by pressing the 8 and 2 number pad keys. Specular lighting can be adjusted by pressing the 4 and 6 number pad keys.

8 (number pad): Increase ambient lighting
2 (number pad): Decrease ambient lighting
6 (number pad): Increase specular lighting
4 (number pad): Increase specular lighting

Ball movement

Use the I, J, K, and L keys to move the ball around.

I: Move ball up
J: Move ball left
K: Move ball down

L: Move ball right

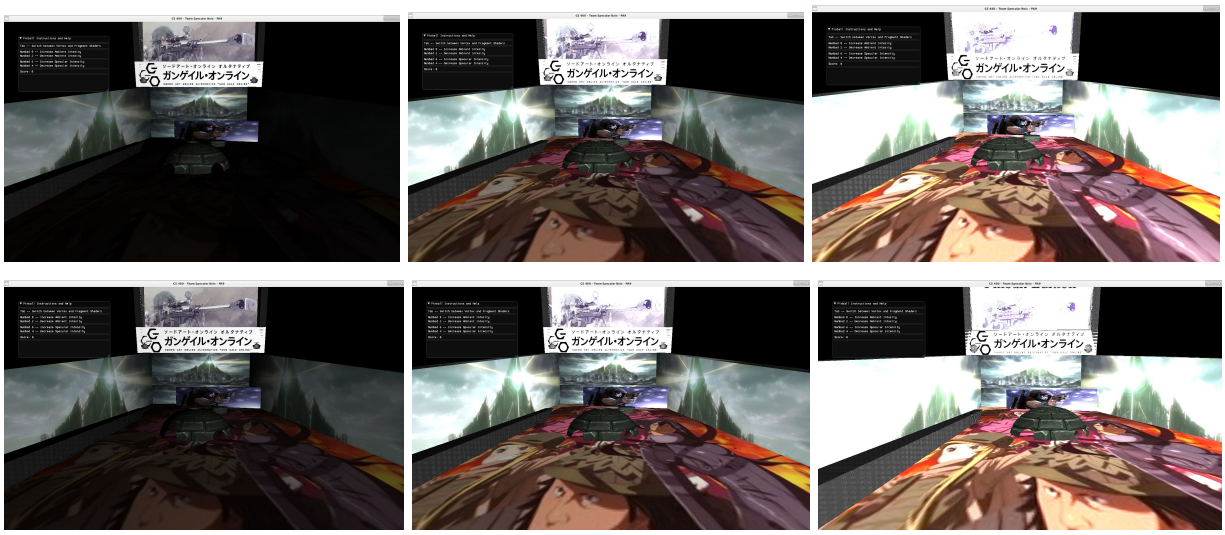

Figure 4. The top row shows per fragment lighting with different ambient and specular values. The bottom row shows per vertex lighting with different ambient and specular values.

Score

Score is displayed on the adjustable in game menu. The score gets increased by increments of 100 when the ball collides with the bumper.

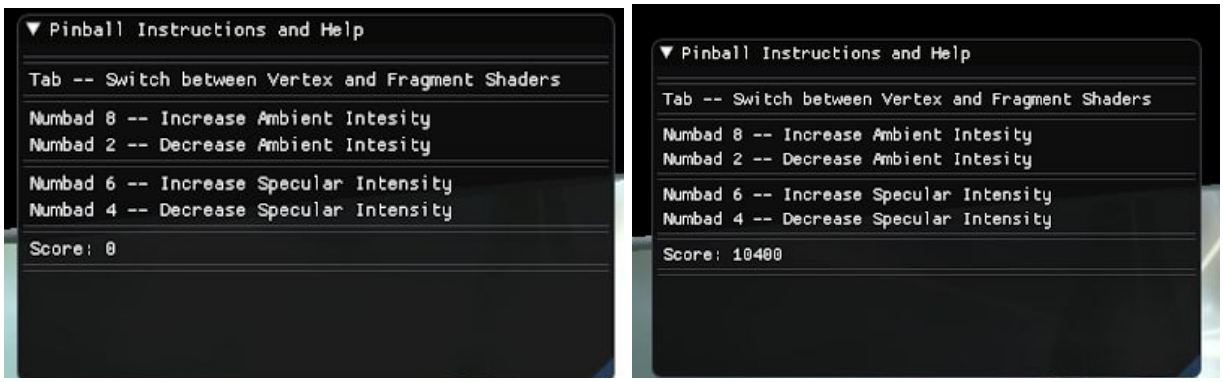

Figure 5. Two pictures showing different score values which are displayed on a menu in the pinball game.

Technical Manual

Issues

An issue with our pinball project is that the paddles do not work correctly. We were having problems being able to place a collision shape at the same position as a paddle. Our collision shapes would appear far above the position of the object. We found a solution to correctly position the collision shapes for all the objects except for the paddles. All methods we attempted to get the paddles to work correctly caused them to continuously bounce up and down.

Another issue with our project is that we have no game logic. To implement game logic we needed to detect a collision when the ball hits the bottom of the board and is out of play. We did manage to get find a way to detect collisions between specific objects but did not have enough time to implement it for game logic. We would have needed to create a new collision shape, properly position it, and then properly size it to create a zone in which is the ball goes out of play once the ball reaches it. We also did not have time to implement multiple lights and a plunger to start the game.

Changes

Since the date 11/14/2018 we have implemented a few changes to our project. One of the largest changes is that we are now loading in every object to create the entire pinball machine. Before we were still loading only a board, ball, cylinder, and cube. Our pinball project looks much more like a pinball project after loading in the entire pinball machine. To load in the pinball machine we had to load in pieces of the machine as separate objects.

Another change to our project is that we have fixed a lighting issue which would occasionally appear when running our pinball project. This lighting issue would appear when running the project multiple times. One of three outcomes would occur. One outcome was that all textures would get inverted on each object. This would cause all objects to look extremely bright on the outside bit appear normal when being viewed on the inside. The second outcome was that our moving spotlight would only affect objects outside the range of the spotlight. This would make most objects appear pitch black. The third outcome was that lighting would work perfectly and all objects would look

normal. This issue was caused by us not defaulting the values of a variable we were using for lighting.

Things we would have done differently

What we would have done much differently from the beginning was figure how to get implement complex collision shapes for use with bullet. We spent time on a previous project, PA8, trying to get complex collision shapes to work properly. We also spent time later on during the pinball project, PA10, to implement complex collision shapes. If we could have successfully implemented complex collision shapes we would have saved much time not having to manually position and size primitive collision shapes. Also we could have created a more complex and fun to play pinball machine layout.