Programming Project Report

Rhett Brandon
010722727

Problem Statement:

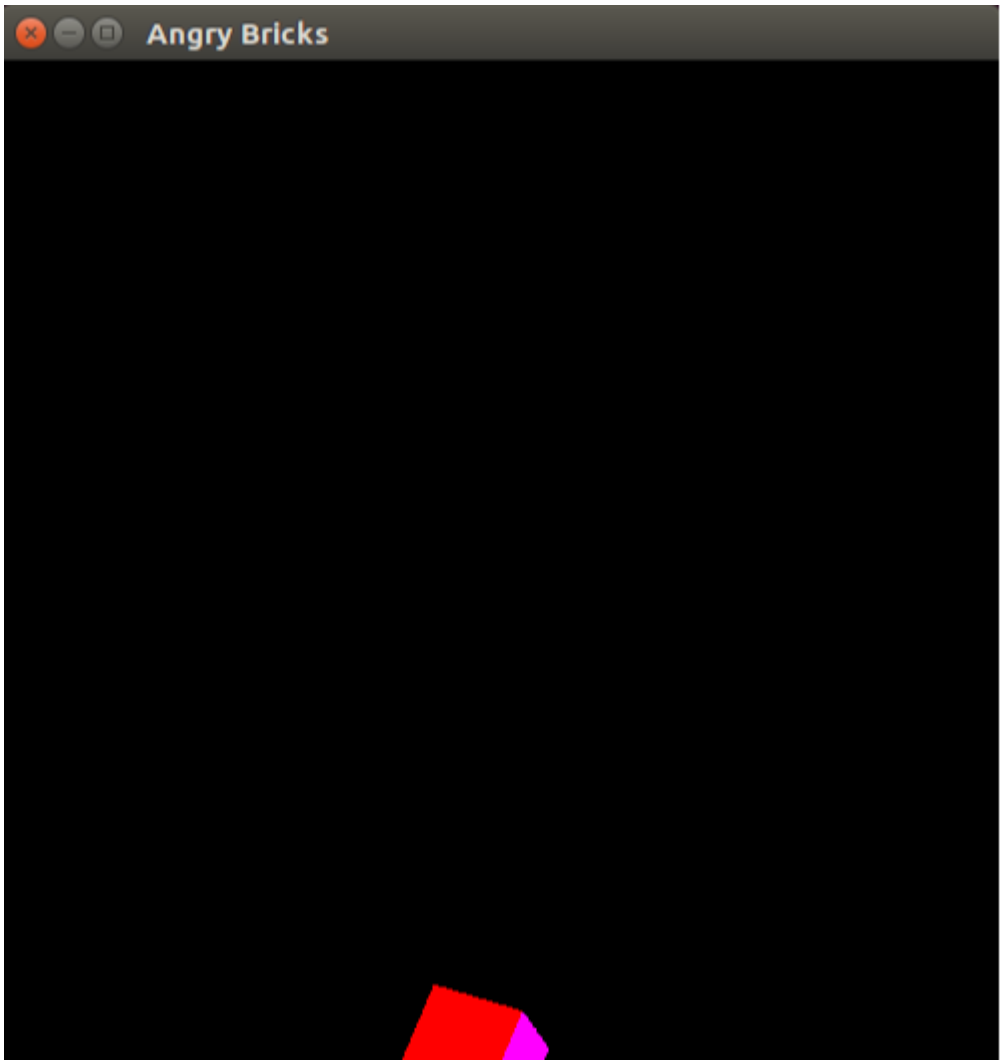
The goal of this project was to take project 2 and finish the “Angry Bricks” application. The project first required us to calculate the velocity between two mouse clicks, creating a slingshot effect. The brick should slingshot further and faster, the farther the mouse is release from its initial click. The next task we had to complete was to simulate physics within our view, this should make the brick drop from the top of the screen to the bottom. Next, we were asked to create a bounce effect when the cube hits either side of the view, this should reverse the velocity of the cube when a wall is hit. Finally, the brick should rotate slowly as it moves across the screen. No error handling was required.

Design:

During the designing of this project, I decided to use floats for the rotation, and velocity variables. Using floats allowed me to make small increments in order to test things like x and y velocity. For instance, I started testing the rotation of the cube by adding 15 degrees every time that the idle function is called. This large rotation ended up creating a cube that spun too rapidly, so I had to decrease the rotation to 0.1 degrees, allowing for a smoother rotation speed.

Implementation:

When implementing this project, I started off with the project_2.cpp project I turned in for the last project. The first thing I did was fix an issue I had where the cube would be slightly off-center from the mouse click when I redrew the cube. After fixing the small bug, I decided to start working on getting the cube to rotate idly. This was done using the GL Matrices. I started by first creating an identity matrix for the cube drawn and calculating what the cube would look like if I were to rotate it idly. Next, I decided to work on translating the cube back to the origin, then rotating the cube, and redrawing the cube back in the position of the mouse click. Presenting the geometric functions in this order prevented an issue where when clicked, the cube would orbit and rotate around the center of the mouse click. Upon getting that to work I created the slingshot velocity vectors and ensured that the cube would slide across the screen. The next implementation task asked us to implement gravity, this was simple, by subtracting from the y-velocity of the cube, until the cube reached the floor. I added a visual feature that stops the cube from rotating whenever it hits the floor, this looked better in my opinion.



Testing:

For the implementation of this project, I tested incrementally. As described above, the first things I began testing was to ensure that when the mouse is clicked, the cube's center is redrawn where it was clicked. Upon fixing this bug from the first part of "Angry Bricks" I moved on to trying to get my cube to idly rotate, and to not orbit when the mouse is clicked. Next was testing the slingshot functionality, with this I tried to ensure that both the x-velocity and y-velocity would change when the mouse was click and dragged farther away from the initial point. Next was testing the 'physics' this was an easy change as no user input was needed, as long as the cube fell to the floor this was completed. Finally, I tested the bouncing mechanic by throwing the brick across the wall and ensuring that the brick's velocity reversed.

Conclusions:

This project was a success when implementing. Creating the project was easy and overall took about 3 to 4 hours. The hardest part was making the cube rotate and not orbit when the mouse was clicked.