**Computer Animation and Gaming Project:**

**Roller Ball Minigame**

Michael Del Rosario

David Zweig

**Table of Contents**

1. Problem Summary...…………………………………...………………………………….3
2. Description of Work…...…………………………...……………………………………..5
3. Results…..…………………………………………………………………………………5
4. Analysis of Work....……………………………………………………………………….7

V. Works Cited……………………………………………………………………………….8

**Problem Summary**

Description

We wanted to create a simple video mini-game that uses a combination of various computer animation techniques such as Reverse Kinematics, Forward Kinematics, and rotation of objects along both the Local Coordinate System and World Coordinate System.

Importance

The topic is interesting because it allowed us to delve into one type of game in the platforming genre of video gaming. The project is an animation project, so the game was focused with the computer animation concepts we have learned over the course in mind. We took inspiration from the “Rollgoal” minigame in the title *The Legend of Zelda: Twilight Princess*.

Proposal

For this project, we planned to implement a game that features a rolling ball along a narrow platform with the player being in control of the platform. The platform should tilt left or right, according to the respective arrow key pressed on the keyboard, and the ball would roll along the tilt of the platform. There are various objects that move in the ball’s way that the player must make it avoid. These objects have properties to them such as a “tower” object that waves back and forth, or an object that moves along a fixed perpendicular path. The goal of the game was to keep the ball on the platform for as long as the player can. The ball must always be moving as the game progresses so that it adds a challenge to the gameplay. Optionally, we would have liked to add Nintendo Wii Remote functionality to the project to give the player more than one way to play. The project was to be created using C++ code using Visual Studio and Eclipse IDE.

**Goals**

There are a few goals we had in mind, which we hoped to accomplish by the end of the semester. At base, we wanted to make it possible for the keyboard to be used so that the player has buttons to control the platform’s orientation. We would have also wanted to implement the Wii Remote as a controller by using the tilt control built into the device, which would then affect the platform’s orientation. The platforms in the game needed to be surfaces that connect with each other and get drawn as they are in range of the camera view, which should gradually move along with the ball.

**Work Breakdown**

* Player Objects: The ball and the player-controlled, tilting platform.
* Obstacles: Various objects that the player must have the ball avoid to prevent a Game Over. These objects will use various computer animation techniques such as rotation on a Local Coordinate System. The player is sent back to a result screen upon a Game Over.
* Title Screen and Result Screen: The player is given the control scheme and must press a button to begin. The player reaches the result screen on Game Over which displays how far the player went without hitting an obstacle.
* (Optional) Wiimote Controls: Add Nintendo Wii Remote functionality to the game, and allow the player to choose which mode to play on the title screen.

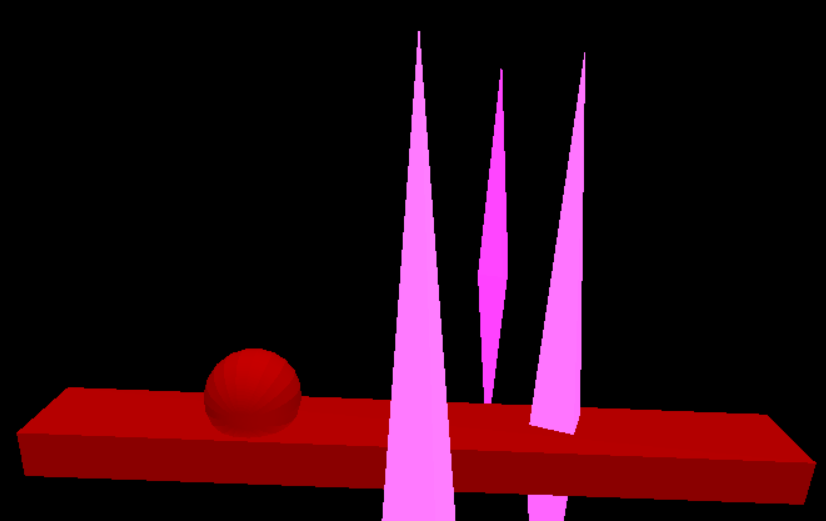
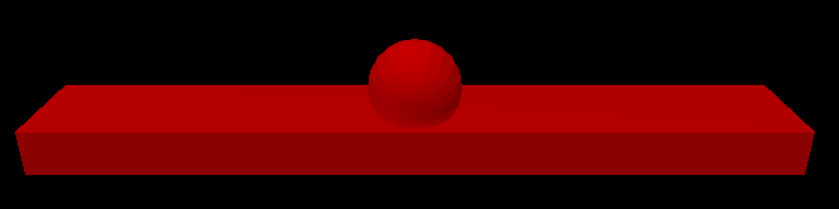
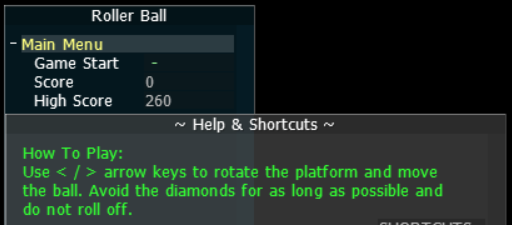
We planned on setting 1-week deadlines every Saturday for each activity. We intended to work on each activity together. If the workload became too tough, then we would replace the week to add Wiimote Functionalities to reduce workload stress.

**Description of Work**

The project worked on C++ with GLU, Glut, and AntTweakBar files. Michael Del Rosario worked primarily on the player platform and the rolling ball, and David Zweig worked on obstacles, the Anttweakbar GUI, and keyboard control. We both combined our efforts on collision detection and other game components such as scoring and game progression. One major dead end of research was implementing Wii Remote motion tilt controls into our project. When researching how to integrate the Wii Remote with C++ and OpenGL, we were able to connect our Wii Remotes to our computers via BlueTooth, but we had trouble finding appropriate external files that allowed our program to implement the Wii Remote; because of time constraints, we were forced to skip the Wii Remote functionality for the project.

**Results**

Below are a few screenshots of the program:



The first image shows the GUI for the main menu with AntTweakBar. When the game is not playing, the GUI is visible and gives the player the option to start the game by pressing the ‘-’ empty field to the right of the label “Game Start”. Below the first variable under the “Main Menu” group are the “Score” and “High Score” fields. The score variable represents the current game score and is initially 0 upon startup and after a Game Over. The high score is the top score that has been achieved since the program was run. The high score does not change until the current game score exceeds the previous high score, in which case the top high score will increase at the same rate as the current score. If the user presses the help guide button below at the bottom-left corner of the screen, the Help & Shortcuts GUI window will appear. At the top is an explanation of how to play the game; after beginning, the player controls the platform with the left and right arrow keys. The GUI will not be visible while the game is going, but upon a game over, the field will return and show the results of the game last played.

Featured on the second image is the ball and platform with a rotation angle set to 0. As the game is running, the aforementioned arrow keys will be activated, allowing the player to adjust the angle of the rectangular prism. The ball will rotate accordingly based on the state of the platform, and it rolls based on the rotation angle. If the ball’s distance value goes past certain maximum or minimum values (representing how far it can go on the platform), reset functionality is called to indicate a Game Over. The platform is assumed to have maximum friction, so the ball does not roll when the platform lays flat (i.e. perpendicular to the Y-axis).

The third image is a screenshot of the gameplay, where the platform is being controlled to move the ball away from the obstacles. The obstacles slowly translate toward the platform after first being placed a set distance away on the Z-axis, and then at a random position along the X-axis within the size of the platform along said axis. As the game progresses the obstacles speed on the Z-axis will increase. The obstacles are shaped like diamonds and have enlarged height to also resemble pillars. The pillar shape makes it so that it does not look awkward if the ball and obstacle effectively collide based on their X- and Z-axes. Regarding collision detection: if the obstacle intersects the platform, and the distance between the ball and obstacle X-coordinates is less or equal to the difference of their radii, then a collision must has occurred, and the game ends.

**Analysis of Work**

New Results

During original implementations, we had the platform move in one direction while the obstacles were stationary. However, we had trouble having the camera follow the platform. To remedy this, we switched functionalities and allowed the obstacles to move towards the platform while the platform is stationary. This changed our scoring system which was based on how far the platform moved on the Z-axis to a time score of how long the player avoids collision and falling off. The camera was also set with a 20 degree angle of rotation on the X-axis, 8 units above zero on the Y-axis, and 1 unit back from the camera’s original planned placement. Those settings create a new perspective to get a better look at the obstacles’ distance from the platform. The original placement of the camera was not angled, and this creates problem with the player judging the distance between the ball and the obstacles.

Meeting Goals

We were able to finish the basic version of the Roller Ball game. The basic version includes rotating the platform using the arrow keys and the ball rolls accordingly, obstacles for the ball to avoid, and a scoring system. We would like to add additional obstacles using more complicated algorithms such as a tower obstacle that tilts from one side to another using the reverse kinematics algorithm, but we did not have enough time to implement these obstacles reliably within our program. We did not implement Wii Remote control since there was trouble with finding the appropriate files that were compatible with C++ and time constraints.

Works Cited

Blando, Jenaro, director. “Zelda Twilight Princess - The Rollgoal Minigame”. YouTube, 19 Dec.

2006, www.youtube.com/watch?v=1qyB946B-UI.