**CS 4172 Final Project**

Project Title: FlappyBird3D

Submitted: May 3, 2016

**Team**

Christopher Chang

Josh Fram

Kyle Jackson

Cassie Wes

**Development Environment**

Unity 5.3.2

Google Cardboard

Vuforia 5.5

**Project Directory Overview**

**Mobile platforms, OS Versions, and Device Names**

**Asset Sources**

**Usage Instructions**

**Video Link**

**Written Description**

*Intro*

Initially, our project set out to mimic the already well-known game Flappy Bird. However, as we began to develop the game and explore the opportunities presented by having a 3D interface, we were able to add features and functions that make the game more interesting and challenging. The decision to implement Google Cardboard in the usage of our game added yet another dimension to user gameplay. So given an ecosystem where the user can operate within a 3D space while being able to manipulate objects with both hands, we developed a 3D version of Flappy Bird that we think best demonstrates our knowledge of 3D UI principles that allows players to have a fun and challenging experience.

*The Game*

Macintosh HD:Users:macbookpro:Desktop:D14D45EB-7ACC-40BE-8FA4-75E128F54C67.pdf

Macintosh HD:Users:macbookpro:Desktop:FF0BDCAD-9021-4336-88AF-586BBA14BA5F.pdf

Macintosh HD:Users:macbookpro:Desktop:05DDE0AE-8F90-4CDC-980C-D3F2B3A3A945.pdf

Macintosh HD:Users:macbookpro:Desktop:28F21D9A-E895-46B3-B8B3-5A888AC06B81.pdf

The four images are of the game’s aerial view, aerial view with mini map, play mode (third-person view), and life-bar (score tracker). As you can see, there are pipes, clouds, and collectable coins in the scene. The user will start out with a life bar represented in the form of a percentage. By collecting the coins through out the game a user will increase their life bar. However, every time a user runs into the clouds or pipes, they will decrease in their life bar score. Every time the user reaches 100 percent, they level up and try to each 100 percent again. However, once they reach 0 percent, they ‘die’ and the game is over. As the players level up the obstacles will appear more frequently and the game will move forward faster, increasing the difficulty.

*Thought Process and Design Decisions*

Now that you have a good idea of how the game operates, and how to play it, we can go in depth about why we decided to build the game in the way we did. As indicated in the title, our goal was to mimic the already well-known game Flappy Bird while introducing the additional DOF of manipulation that come with the combination of 3D space and Google Cardboard interface. To do achieve this, we wanted to make sure that in additional to having a fun game, that we implemented good UI3D practices that included methods for selection, wayfinding, travel, scaling, and of course follows the Neilson Heuristics.

Our initial goal was to mimic Flappy Bird as closely as possible, but quickly realized after testing our first build, that on an AR – Google Cardboard interface, it would be unreasonable to expect the user to be able to so precisely navigate the Flappy Bird in 3D space around all the obstacles we implements and at the same time collect the coins. The Flappy Bird too often ran into the obstacles, causing the Bird to die and quickly end the game. It was no fun and hard to get any continuity in the game. Therefore, we decided to change the play mode into something that would allow more room for error while still promoting a challenge as well as incentive for the player. We drew inspiration from other games (many first person shooter games), and decided that a game play set-up where the user had life bar was the best option because it would allow us to use the obstacles to no end the game, but either increase of decrease their life bar.

*Selection*

By introducing the Google Cardboard interface in addition to the 3D AR space, we were able introduce new forms of manipulation that utilize the user’s (now free) two hands. One hand will wield a selection wand tool, the other a travel joystick tool. The user can utilize the want to select clouds (an obstacle in our game) as they approach them and ‘swipe’ them out of the way to avoid losing points on their life bar score.

*Travel*

Travel is implemented through an Atari-like joystick that allows player to navigate the Flappy Bird either up and down or side to side in order to position it away from the obstacles and in line with the collectables. We wanted to mimic the playing style of other classic ‘flying’ games that people are familiar with so our game implements the controls where up on the joystick means moving the bird downward and vice versa.

*Wayfinding*

The way we implemented wayfinding in our game was by having two mini maps that appear on the screen during game play. As you can see from the pictures in the first section, these mini maps are of importance because they provide two additional views that allow the user to better understand where they are in the game and how to strategize further. The first view is a first person view, which may makes it easier for the user to see what items are coming up and what their relative positions are (especially height, sometimes hard to tell in default third person view). The other mini map allow the user to view the game from a bird’s eye view, which allows the player to be able to plan farther ahead, and to see where the collectables as well as obstacles are coming up next. We chose to implement wayfinding the way we did because these mini maps complement the way the game is played through our methods of selection, travel, and scaling; This allowing for an interesting dimension of strategic planning and gameplay.

*Scaling*

We implemented scaling in a way that would add a dimension of gameplay to Flappy Bird 3D. The user can control the scale of the bird by tilting his/her head either left or right, making the bird either bigger or smaller. How big the bird is affects how many points the user can score. These tradeoffs forces the user to make what they think is the right choice during gameplay; the bigger the bird the more coins it can collect, but at the same time, the more obstacles it can run into.

*Neilson’s Heuristics*

Visibility of System Status

Match between system and real world

User control and freedom

Consistency and standards

Error prevention

Recognition rather than recall

Flexibility and efficiency of use

Aesthetics and Minimalist Design

Help users recognize, diagnose, and recover from errors

Help and Documentation