**CS 4172 Final Project**

Project Title: FlappyBird3D

Submitted: May 4, 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**

Iphone 6

Google Carboard

**Asset Sources**

Source for Flappy Bird Game Object: Google SketchUp by Sr.Cookies <https://3dwarehouse.sketchup.com/model.html?id=7a65f8802fa5fad06a780ed54bbe4fec>

Source for Cloud Game Object: Google Sketchup by NumaNoxchi <https://3dwarehouse.sketchup.com/model.html?id=f11b4d4832089744d485372bb746f3c7>

**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 throughout 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 (no matter what level they’re on), 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 comes with the combination of 3D space and Google Cardboard interface. To achieve this, we wanted to make sure that in additional to having a fun game, 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 we 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 implemented 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 gameplay. Therefore, we decided to change the play mode into something that would allow more room for error while still promoting a challenge/incentive for the player. We drew inspiration from other games (we looked to 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. Initially, we decided that one hand would wield a selection wand tool, the other a travel ‘Atari-like’ joystick tool. The user can utilize the wand 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. After discussing through several options with the TAs and Professor Feiner, we decided that a better way to implement selection would be to utilize yet another degree of freedom where we could rotate the orientation of the Flappy Bird. This way, we could better exploit all the opportunities for user interaction that an interface such as AR/ Google Cardboard provides. So instead, in our final product, we allow the user to still wield the ‘Atari-like’ joystick in one hand, but instead of a wand in the other hand that serves as a selection tool, we have an image target that the user can turn (or rotate) to correspondingly rotate the Flappy Bird. To remove the clouds, the player would rotate the Flappy Bird to ‘aim’ at the cloud and ‘shoot’ (aim at it for 0.5s or so) it so to speak. As mentioned in the IEEE paper submitted to the 3D manipulation competition, entitled *Batmen — Hybrid Collaborative Object Manipulation Using Mobile Devices,* this if effective because “users understand the 1:1 mapping metaphor to manipulate objects: actions performed on the proxy are directly translated to the 3D world object being manipulated, achieving a 1:1 mapping.” In addition, we felt that it would be more seamless if both hands were utilized for manipulating the Flappy Bird. Having one hand used for a separate selection technique while the other controlled the bird seemed unnatural and made the game more difficult to play without making it more fun.

*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. People who are familiar with playing flying game should be able immediately know how to play our game.

*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 – The user is constantly aware of how he/she is doing in the game. Through the real time constant updates in the life bar score, the system provides sufficient user feedback to provide for an ame play experience.

Match between system and real world – As mentioned earlier in the write-up, we decided to model the controls off of aviation-like games because the controls (up on joystick is down, and vice versa) are mapped from real life flying controls. The decision to replace the magic want with a image target that rotates the Flappy Bird was due to the reasoning that it resembled the real world and so the user could easily understand the “1:1 mapping from proxy (image target) to game play”.

User control and freedom – User control and freedom typically refers to the implementation of an “emergency exit” or the support of undo and redo. In our game, we want the point is to play the game as perfectly as you can, running into as few obstacles as possible while collecting as many collectables as possible. Our decision to change our game from a one hit KO model to a life bar model allows the user to have mistakes and recover from them represents a type of “emergency exit” where users can be in a scenario where they decide it is worth running into an obstacle with the promise that there are many more coins behind it to collect and increase the life bar score.

Consistency and standards – The rationale behind our design decision regarding the Atari joystick and the image target utilized for Flappy Bird rotation was in large due to the reason of consistency and standards. We felt the infinite flying nature of the game was consistent with some well-known games in the past and wanted the user to be able to recognize the similarities and be able to learn the controls of the game instantly. As mentioned earlier, the joystick emulates the classic controls of aviation games and the image target for rotation emulates the real world manipulation techniques to help make the gameplay experience seamless and the learning curve nearly nonexistent.

Error prevention – In the case that the player does not know how to play the game, he and she will quickly learn as they run into the objects, that they are doing something wrong because their life bar will quickly drop from a certain percentage down towards 0%. This allows the user to quickly learn from their mistakes and prevent any future game play errors in the future.

Recognition rather than recall – Because VR provides such a new way to interact with gameplay objects (such as the scaling feature in the head tilt of the Google Cardboard), it is important that players can immediately recognize controls/features of the game that they have interacted with before. The Atari joystick has been around for almost as long as games have been around. Player will unmistakably recognize that the Atari-like joystick is for controlling the Flappy Bird and that the aviation game like controls similarly apply as well.

Flexibility and efficiency of use – At first, it make be hard for the player to truly take advantage of the mini maps that help with wayfinding and game strategy. However, once the user has gotten used to gameplay controls and can understand the intricacies of how to get the best score possible, he/she can use the minimaps as a feature of “acceleration, speeding up the interaction for the experienced user.”

Aesthetics and Minimalist Design – Our design features only what is necessary. The aesthetic of the track is minimal and well contained, allowing the user to focus on the task at hand without behind distracted by anything unnecessary to the goal of the game.

Help users recognize, diagnose, and recover from errors – The words ‘Game over’ are displayed when the user has failed to complete the goal of the game.

Help and Documentation – Refer to usage instructions.