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

The FlappyBird3D directory contains all progress reports as well as the written report. You will also see a FlappyBird3DGame subfolder, FlappyBird3DGame2.0 subfolder, and a FlappyBird3DGame3.0 subfolder in the directory.

The FlappyBird3DGame directory contains all the assets for our initial version of game. (Before we talked to the professor and TAs and came to our finalized idea, refer to powerpoint progress reports for more detail).

FlappyBird3DGame2.0 is another iteration while

FlappyBird3DGame3.0 is the finalized version. This is the version you will want to deploy for testing/play. You will find contained the folders/files that make up this game.

Assets/_Scenes/ - contains all scenes used in the project

Assets/Materials/ - contains all materials used

Assets/Scripts/ - contains all necessary C# script files

Assets/Textures/ - contains all textures used

Assets/Vuforia/ - contains all AR related files, image targets

Assets/UI and Item Sound Effect/ - contains sound effects, menu UI files

Mobile platforms, OS Versions, and Device Names

Iphone 6

Google Cardboard

Mac OSX Yosemite Version 10.10.5

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

After you git clone from https://github.com/jpf2141/FlappyBird3D.git:

In Unity, go to File -> Open Project -> [FlappyBird3D directory] -> Deploy to mobile

To use the game, build the application onto an Android or iOS device from the Unity Game Engine. The user will need 3 image targets, which are the stones, leaves, and vortex images. The stones image target is used for the playing field, while the leaves and vortex images are used for the joystick and gun aimer, respectively. Along with the image targets, Google Cardboard is needed to allow for two free hands. To control the Flappy Bird, tilt the joystick in the respective direction and adjust its distance from the camera to move the bird up and down. To select and destroy clouds, adjust the rotation of the Flappy Bird using the gun aimer that is rendered on the vortex rectangular image target. The user may find it beneficial to refer to the first person way finding camera with crosshairs to help with this process. The objective of the game is to gather as many coins as possible to gain points and level up, while also dodging both clouds and pipes. The user has the option to select an easy or hard difficulty by tilting his or her head at the menu screen.

Video Link

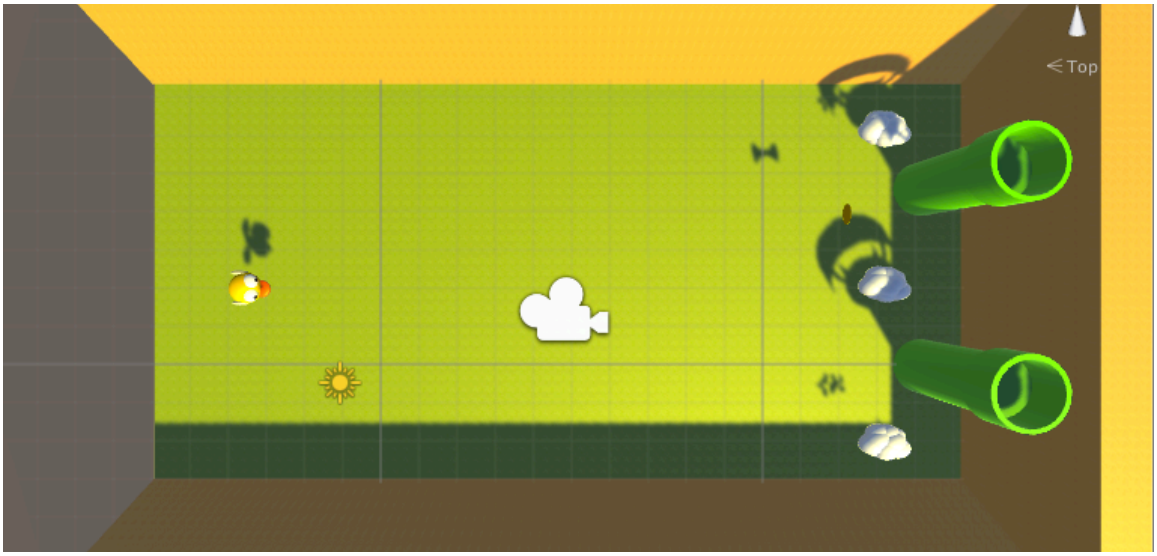
https://www.youtube.com/watch?v=_6W9Id0UBkU&feature=youtu.be

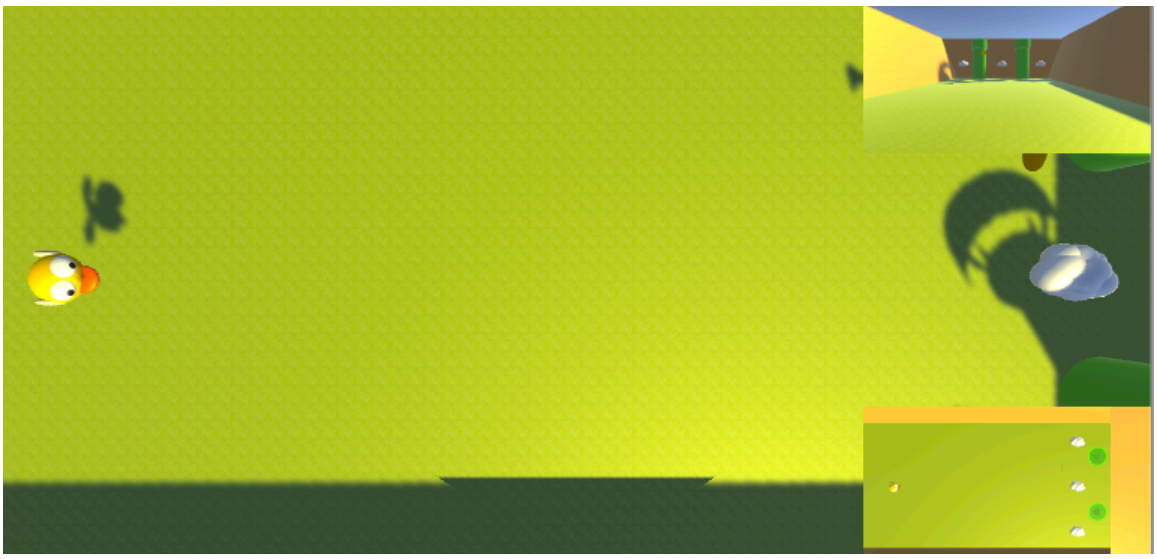
Written Description

Intro

Initially, our project goal was to mimic the already well-known game, Flappy Bird. Our first iteration of the project, though it incorporated some 3D features, did not take full advantage of all the concepts we learned through the course of this class. However, as we began to develop the game and explore the opportunities presented by creating a complete 3D experience, we added features and functions that made the game more interesting and challenging. This also allowed us to more fully incorporate the concepts we learned in class. Google Cardboard was essential in allowing us to accomplish this. Google Cardboard provides the user additional freedoms in playing the game – they can use both of their hands to play, instead of needing both to hold their phone. With this in mind, we created many different ways through which the user can interact with the game that allow us to show off what we have learned in the class.

The Game





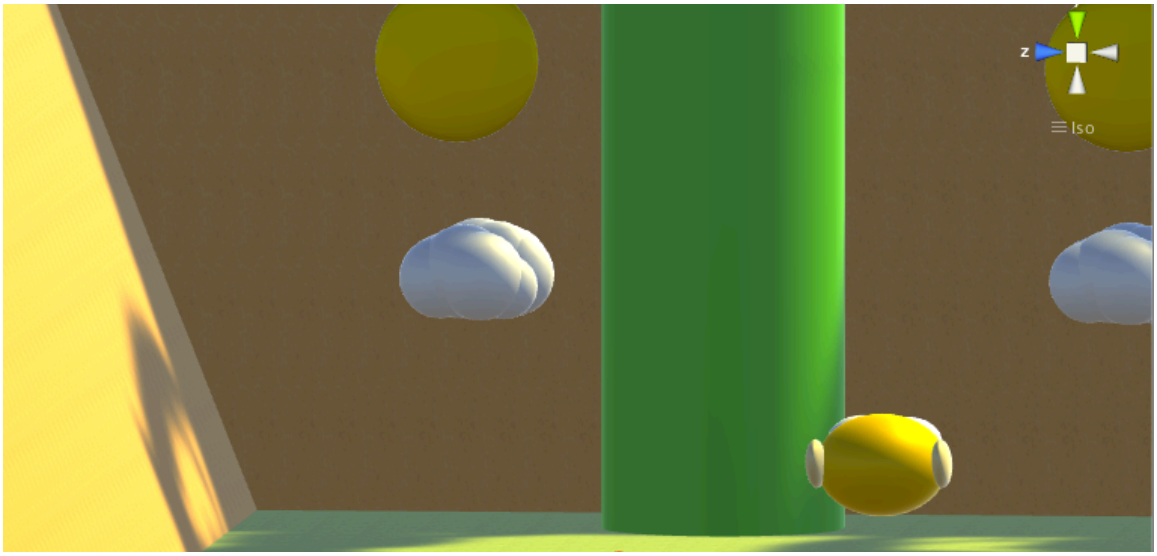

z
Iso

Image 1: A view of the game's map from an aerial view.
Image 2: A view from the same perspective, featuring the two mini-maps used for way finding and selection.
Image 3: A view from the perspective closer to what the user will see, directly behind the bird. You can see a pipe obstacle, a cloud obstacle, and a coin collectible.
Image 4: The user's current level, score, and health. The user progresses to the next level when their score increases to 100%, but the game ends when their health decreases to 0%.

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 DOFs of manipulation that comes with the combination of 3D space and Google Cardboard interface. To achieve this, we wanted to make sure that in addition 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 completely avoid every obstacle while attempting to 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 gameplay set-up where the user had a life bar was the best option because it would allow us to use the obstacles not to abruptly end the game, but to modulate the life bar (score).

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, while the other, a travel 'Atari-like' joystick tool. The user can utilize the wand to select clouds (an obstacle in our game) as the bird approaches them and 'swipe' them out of the way to avoid losing points on his/her 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 the 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 for a separate selection technique while the other navigated the bird seemed unnatural and made the game much more difficult to play without making it more fun.

Travel

Travel is implemented through an Atari-like joystick that allows a 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 already familiar with. Therefore, we implemented the controls where up on the joystick means moving the bird downward and vice versa. People who are familiar with playing flying games should be able to immediately learn 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 they can strategize further. The first view is a first person view, which may make 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 allows the user to view the game from a bird's eye view. From here, the user can more easily see where the collectables and obstacles are going appear on the track. This gives the user the ability to plan ahead and strategize. 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. The combination of these things allows for an interesting dimension of

strategy and gameplay.

Scaling

We implemented scaling in a way that would add another 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. The bigger the bird is, the more coins it can collect. However, it is also important to consider that the bigger the bird is, the more obstacles it will run into. These tradeoffs force the user to make scaling decisions throughout the course of the game.

Neilson's Heuristics

Visibility of System Status

"The system should always keep users informed about what is going on, through appropriate feedback within reasonable time" (Nielsen).

The user is constantly aware of how he/she is doing in the game. The system provides sufficient user feedback during gameplay by providing the user with a health bar. Whenever a user collides with a cloud or pipe, the health bar will decrease. Conversely, when a user collects a coin the health bar will increase. The health bar is located at the upper right-hand corner of the screen at all times.

Match between system and real world

"The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order" (Nielsen).

The game begins with a main home screen in which the user is able to select an easier or more difficult level. The instructions on the screen tell the user to either rotate their head to the left or to the right to access the different levels. The directions to do so are very

straightforward and concise: “Tilt left for easy mode” and “Tilt right for hard mode”. Once a user selects a mode, they will be taken into the gameplay scene in which they are able to control the FlappyBird by a “joystick” image target. We decided to implement joystick-like controls to emulate hand-motions that the user is already familiar with. The user is also supplied with a second “wand” in which they are able to point, select, and delete incoming clouds. We decided to implement this “point-and-shoot” functionality to mimic the pointing gesture that the user is already familiar with.

User control and freedom

“Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo” (Nielsen).

User control and freedom typically refers to the implementation of an “emergency exit” or the support of undo and redo. In our game, the goal is to run into as few obstacles as possible while collecting as many coins as possible. Our decision to change our game from a one-hit-kill model to a life bar model allows the user to make mistakes but still recover from them, which represents a type of “emergency exit” where users can be in a scenario where they decide that it is worth running into an obstacle with the promise that there are many more coins behind it to collect and increase their score.

Consistency and standards

“Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions” (Nielsen).

Because we implemented the Google Cardboard we were able to allow the user to use both hands to manipulate the Flappy Bird. By providing the user with both a joystick controller and a pointer controller, it eliminates any controller confusion. As mentioned

earlier, the joystick emulates the classic controls of aviation games and the image target used for rotation emulates the real world manipulation technique to help make the gameplay experience seamless and the learning curve nearly nonexistent.

Error prevention

“Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action” (Nielsen).

In the case that the player does not know how to play the game, he or 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.

Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate” (Nielsen).

Because AR/Google Cardboard provides new ways for users to interact with game 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. Players 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

“Accelerators -- unseen by the novice user -- may often speed up the

interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions" (Nielsen).

At first, it may 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

"Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility" (Nielsen).

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

"Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution" (Nielsen).

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.