Dominic Celiano

CS 385 – Mobile Computing

Final Project Design

Documentation: I have received no help in completing this assignment.

**Project Design**

1. **Project Description**

My project idea is a USAFA marching game, where the player act as a flight or squadron commander and march a flight or squadron on the terrazzo. The player will start out by marching a flight, and will rank up depending on how well his flight performs. The player will be able to give commands to his flight as well as correct his flight on things like distance and being out of step. These commands will be given via buttons in the lower portion of the screen. The better the player controls his flight, the quicker he will be able to rank up and take on more complicated tasks. Once the player becomes a cadet officer, he will be able to march a squadron and participate in events like noon meal formation.

1. **Functional Requirements/Frameworks**

Frameworks

* UIKit/Foundation
* SpriteKit
* NSUserDefaults

Functional Requirements

* March a flight/squadron on the terrazzo
* Be able to choose between marching a flight and squadron
* Have the flight/squadron make random mistakes
* Allow the player to rank up
* Allow the player to give commands to both control and correct his flight/squadron
* Save progress of the player and his flight/squadron
* Implement marching tasks/missions for the player to complete
* Implement a “settings menu” where the player can view both his progress and the progress of his flight

1. **Model-View-Controller Decomposition**

Model  
The model will handle all of the underlying properties of the marching game. The mistakes the flight makes will be handled by the model. What will also be handled by the model is the progress of both the player and the flight based on how well the player gives commands and correct his flight. There will be a lot of math behind tracking the progress of the player and the flight and all of this math will be handled by the model.

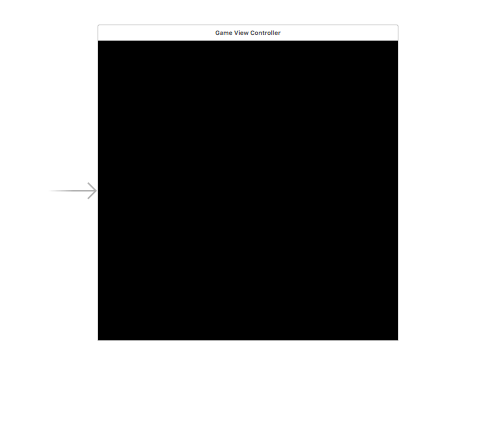
Location detection will also be handled by the model, as well as the various scenarios which are presented to the player. For example, if the player has a task to march his flight to Mitchell Hall, the model will know that this is the task and will use the coordinates of the player to determine when the player’s flight arrives at the entrance to Mitchell Hall.

View/Controller

The view and controller are where the front end of the game will be implemented. In SpriteKit, “scenes” are used to handle the different views. My game will have 3 different scenes. The first will be the MainMenuScene( ), where the player will start out when they open the application and will be able to choose whether to start a new game, and whether to march a flight or squadron. The beginnings of this scene can be seen in the picture below.



The next scene will be the actual GameScene( ). This is where the Terrazo environment will be implemented and the character will be allowed to march around. This scene will take the input from the MainMenuScene( ) and will create the flight or squadron accordingly and will display the appropriate settings. In SpriteKit, there is no way to show the storyboard for the entire game, but a GameViewController is used to contain all the game information, which can be seen below (this is a screenshot of the storyboard).



The last scene which will be implemented is the SettingsScene( ). In this scene, the player will be able to go from the main GameScene( ) and transition into this new scene where they will be able to do things like go to the main menu, start a new game, choose a new task to accomplish, or save the game for another time in which they wish to play. The information from the SettingScene( ) will be passed back to whichever scene is selected – either the GameScene( ) or the MainMenuScene( ).

Model and View/Controller Interaction

The controllers will interact with the model by passing back to the model which commands were pressed and which “foot” the commands were pressed on. From that information, the model will determine whether the command was called correctly and recalculate the resulting level increase or decrease for the player and flight. Then, the model will pass back the updated level information to the controller to update the progress of the player. There will be methods such as UpdateRank( ) and UpdateFlightSize( ) which will be called when certain conditions in the model are met.

1. **Implementation Plan/Test Plan/Division of Labor**

Implementation Plan (Intermediate Milestones)

* **T29** – sprites and environment textures found via internet
* **T30** – terrazzo environment created using SpriteKit Scene Builder
* **T31 –** able to march the flight with basic commands
* **T32** – flight will start to randomly makes mistakes
* **T33** – player is able to correct mistakes his flight makes
* **T34** – player is able to rank up and track his progress
* **T35 (Gate Check)** – player will be able to save game
* **T36** – player is able to track the progress of his flight
* **T37** – squadron marching implemented, different marching scenarios implemented
* **T39** – final tests conducted/attempts to “break the program” – final bug fixing

Test Plan

* **T29** – try to create the basic environment using the sprites/textures to see if they will work
* **T30** – run the environment created via Scene Builder on a target device to ensure it looks as expected
* **T31 –** create a game from scratch and give commands such as “column left/right”
* **T32** – march a flight around and see how whether mistakes are made as expected
* **T33** – have the flight make mistakes and see if the player can correct the mistakes
* **T34** – make sure the “settings” menu shows the level progress and actually play an example game and try to rank up
* **T35 (Gate Check)** – start a game from scratch, progress through some levels, and then quit the game completely and see if you pick up from where you left off
* **T36** – check to see if in the “settings” menu you can view how good your flight is and how much they are progressing. Make sure the better they are, the less mistakes they make.
* **T37** – Choose between a flight and squadron in the splash screen and make sure the game implements the choice which you selected.
* **T39** – Give the game to some of my friends to have them play it and “try to break it”.

Division of Labor

* Not applicable because I’m working alone.

1. **Critical Path Experiment**

While there are many critical hurdles to overcome for this project to be successful, I have chosen two which, if not reached, the project simply will not work. The first one was demonstrated by conducting an actual experiment, while in the 2nd critical path I simply made a design:

* 1. The Terrazo must be created and the player must be able to move around in it.
     1. Once the actual environment is created, this functionality will be tested by using a single “cadet” in the flight instead of a whole flight.
     2. I did an experiment to test the viability of reaching this hurdle by going through a SpriteKit Scene Kit tutorial located at <https://www.raywenderlich.com/118225/introduction-sprite-kit-scene-editor>. This tutorial went over the basics of how to create a scene and create the textures necessary. The environment which I created in this tutorial was very similar to the one I will create in my marching game. After understanding the tutorial and implementing it in XCode, I am confident that the Terrazo environment I create will work as necessary to make the project work successfully.
  2. The flight must be able to do basic marching functions, with the user controlling functions such as turns and stopping the flight.
     1. In order to test this, I first drew out how I was going to have the iPhone look when the user was controlling the flight. This picture can be seen on the next page, with the “Image View” representing the actual game board. I am confident this method of control will work because I have used buttons in many previous PEX’s and labs.

