Austin Lucas

Project 6

Hot Mic

Final State Of System

For my final delivery, I have implemented almost all of the features that I intended to. I have implemented a round system for keeping track of the current player, current round, and total rounds. I have also implemented a leaderboard system. The way this works is, each individual player object keeps track of their own points. If a player wins a round, they are awarded points. In the player group class, I have Implemented an insertion sort algorithm to sort the players according to their score. Then, depending on the number of players in the game, I display the updated score on the results page. I have also implemented a set of classes for handling the tasks/content part of the game. These handle the "charade" part of the game. The classes are very simple, they are very similar to the player class with a few getters and setters and a member variable which holds the content.

I was unable to implement the audio recording part of my idea. There are a few reasons why I was unable to do this. First off, apple provides a framework called AVfoundation. This framework handles all AV tasks across apple software. I found that It was quite simple to capture audio from the user device. However, analyzing that audio to give you real time visual feedback is very hard. I found one reference online that did not provide a solution, but said that in order to do this I would need to first capture the audio, perform a series of calculations and conversions on the audio captured, and then come up with a way to display the results graphically. This was obviously out of my wheelhouse. Next I found out about another framework called AudioKit. AudioKit is an open source framework which uses Apples AVfoundation to provide tools for analyzing and generating audio in swift. After researching this for a little while I found that AudioKit could potentially be a good solution. After downloading the framework and trying to use it for a really simple test, I found that nothing was working like the documentation said. I couldn't get anything to compile, even after following the AudioKit documentation exactly. It seemed like Audiokit was a bit outdated or maybe I am working with a newer version of swift that AudioKit doesn't fully support yet. Either way, I was not able to find a solution for the audio Analysis part of my idea.

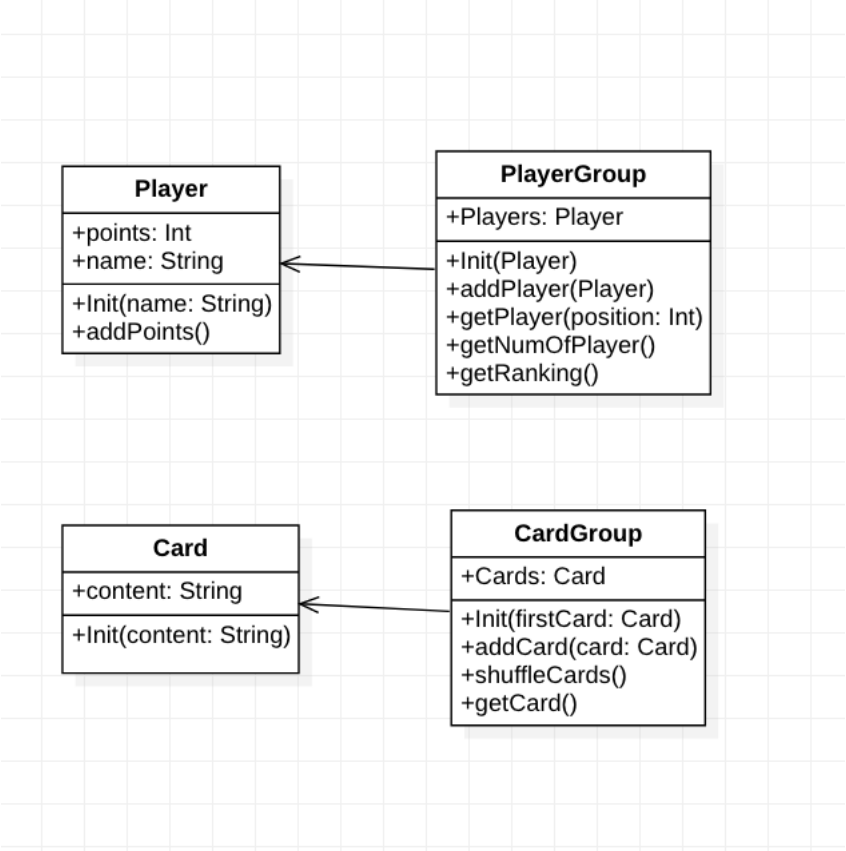
The lack of audio support in my app is the only thing that I was not able to do. It is really unfortunate, however, that the audio aspect is the part that makes my app unique and not just like any other charades game.

Final Class Diagram

My project consists of 14 different files. Most of the files are things called view controllers. View controllers in swift control what happens on the screen and they also handle how data gets passed between screens. But because they are not really classes I have created, more so just classes I have populated with my implementation, I have decided to keep those out of the UMl. The only classes I will show in the UML are the ones I created from scratch.

Austin Lucas

Project 6



Third-Party Code

I tried to use very little third party code. The only third party code I used was to implement a timer in my MicScene.swift file

(https://stackoverflow.com/questions/29374553/how-can-i-make-a-countdown-with-nstimer).

My project also includes an AppDelegate.swift and SceneDelegate.swift file which are included in every xcode project but I did not modify those.

I started learning xcode by watching a youtube series by a guy called CodeWithChris (https://www.youtube.com/user/CodeWithChris). This taught me the basic functionality of swift and how Xcode works but all ideas about the project were mine.

OOAD Process

At the end of the day, my project was fairly simple. The hardest part was learning how to use Xcode and Swift, as well as UI Kit to create the interactive UI. I honestly got caught up in that more than I did thinking about OO patterns. However, I was using the OO fundamentals the entire time. The entire composition of my project is classes using encapsulation to protect their own state. This actually made passing data between views harder than I expected. Also in the classes that I created. I was using inheritance to create subclasses which would hold groups of parent classes.