Dylan Hellems
djh5sc

Device: CCQPR2J6GGK5 (iOS)

Project Title: King of O'Hill

Project Pitch:

King of O'Hill is a relatively simple mobile game with an interesting and UVA-centric gimmick. When you open the app, you are faced with a menu screen. Here the user finds a play button and a leaderboards button. Pressing the play button loads the game screen, and, if this is the first time the app has been opened, the user is prompted to enter a nickname. This nickname is used for the leaderboards, and is stored on the app once entered, so it isn't asked for again unless that specific username has already been selected or the database has been wiped. Once the game screen loads, the user is prompted to tap the screen to begin. The game itself is probably a familiar one, akin to Doodle Jump and Winter Bells, it requires that the player "steer" their character, or in this case plate, using the accelerometer to collect pizza. Every time a pizza is collected, the player is boosted high enough to reach the next one, and so on. If a player misses a pizza, or for any reason falls below the screen, the game ends and the player's score is posted to the online leaderboards. Here is where the gimmick comes in, because along with the player's score and nickname, the app also sends the name of the nearest dining hall, through the use of GPS, and the current "mealtime". Using this extra information, the server categorizes the player's score into a local and temporal leaderboard, so that that a player can be the King of O'Hill at Breakfast or the King of Runk at Dinner. Navigating back to the main menu, and then on to the leaderboards page, the user is presented with a web view detailing all of the leaderboards, so that they may view their own, and others, rankings.

Platform Justification:

I created this application for iOS for a few reasons. Firstly, I own an iPhone, so the thought of creating an a-application that I could use regularly and show people excited me. Secondly, I plan on developing further iOS applications in the future, and potentially releasing them on App Store, so I thought that getting more experience with native development would be useful. Lastly, I really enjoyed using Xcode to create the earlier application that was assigned. I found testing and deployment on my device to be extremely quick and simple, and I liked working with Swift. Also, the UI builder made it easy to create sleek and modern UIs; more so, I thought, than Android Studio's UI Builder. Unfortunately, making a game meant working with SpriteKit and not so much the UI Builder, but it was still an enjoyable, and relatively painless, experience.

Key Features:

- GPS: I use location services to find the nearest UVA dining hall to the user, for the purpose of categorizing leaderboards by location
- Time: I use system time to find the correct "mealtime" (Breakfast, Lunch, Dinner, or Midnight Snack), for the purpose of categorizing leaderboards by timeslot
- On-device Storage: The app stores the user's nickname, and other necessary information for dealing with the web service, locally in order to enforce consistency in the leaderboards

Dylan Hellems
djh5sc

- Rotation: Due to the nature of the game in the application, rotation is locked to portrait mode. I do not believe it makes sense to allow landscape mode for a game of this type, and it doesn't make sense to allow rotation on some pages and not others
- Interruption Handling: The game pauses when interrupted by a phone call, clicking the home button on hardware, etc and can be resumed by tapping the screen
- Web Service: For this application, I developed and deployed a custom web service using the Django web framework and the hosting service, Heroku. This web service stores and serves users nicknames and scores, and presents the various leaderboards at https://king-of-ohill-web.herokuapp.com. Due to having only a free account with Heroku, the service will "fall asleep" after periods without use. Because of this, certain elements of the app may require a "warm up" period of a few seconds. Visiting the URL linked above will give a good idea of the status of the page.

Testing Methodologies:

To test my application, I mainly used the Xcode simulator and my device to test functionality. I worked through various edge cases to test the limits of my application, and to see where improvements could be made. I tested my web service similarly, using the Chrome app PostMan to send phantom POST and GET requests.

Usage:

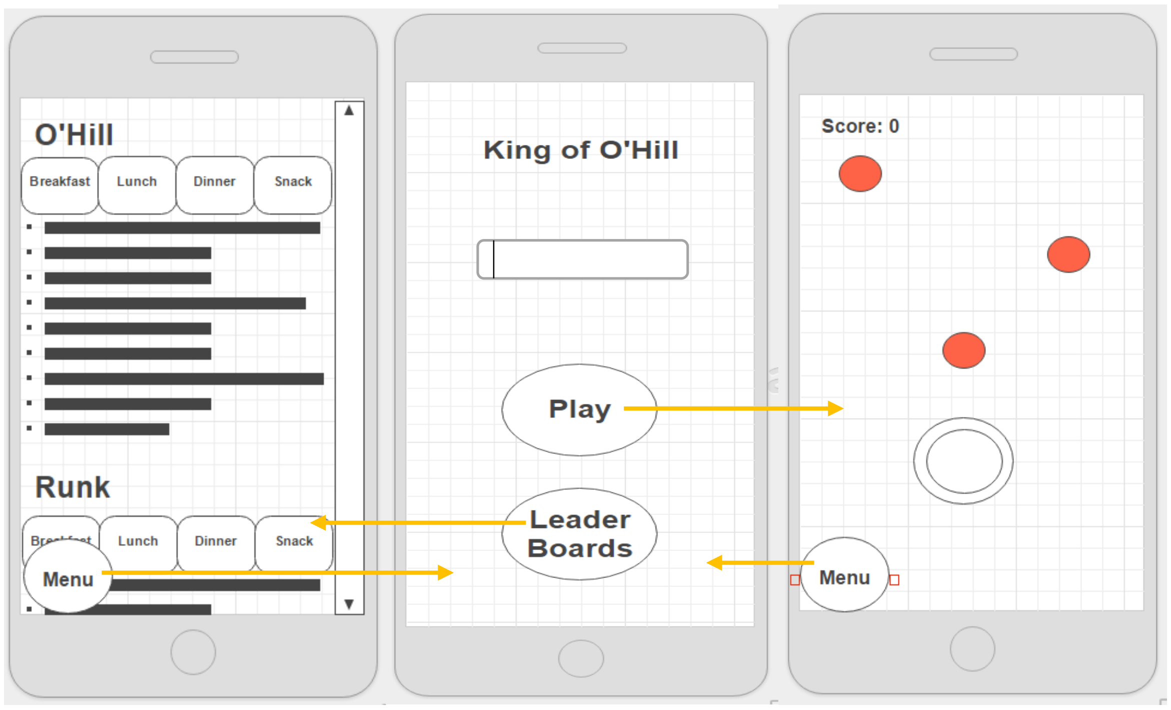
You shouldn't need any special instructions to run the app. The app has three screens, a main menu, the game, and leaderboards. The game and leaderboards can be reached through the main menu, and the main menu can be reached through either other page. The main thing to remember, when running the app for the first time after a period without use, is to go to the leaderboards page first and wait for it to load, that way you know the web service is up and running.

Lessons Learned:

The main thing I learned, while building this application, was how to use SpriteKit to build a mobile game sans UI Builder. With the help of some tutorials and previous game development experience, I was able to work with the engine quite easily. SpriteKIt made the physics components of the game extremely simple to implement. Navigation was also simple, though when I needed to navigate to a UIViewController and not a SpriteKitScene, I ran into some trouble. Unfortunately, getting this to work required some ugly code. The other main things I learned during this project, were how to use CoreLocation and CoreMotion in order to take advantage of location services and the accelerometer. These libraries were surprisingly easy to work with. The only real issues I ran into arose while dealing with permissions. Though requesting permissions In code is not difficult, it required me to fiddle around with the .plist file and change some variables, to ensure that the device showed the request to the user. I later had to do this again to lock rotation, so it ended up being good experience. Lastly, I learned a lot about how difficult Apple likes making using their products. During the last week of this project, I ran into multiple compatibility issues between the iOS version on my device and the version of Xcode on various public workstations around campus. Eventually, I had to borrow a friend's mac to actually build my app to a device. These kinds of issues, and the fact that Apple requires you to have a mac to build an application natively, shows just how protective they are of their platform.

Dylan Hellems
djh5sc

Wireframe:


O'Hill
Breakfast
Lunch
Dinner
Snack
Runk
Breakfast
Lunch
Dinner
Snack
Menu
King of O'Hill
Play
Leader Boards
Score: 0
Menu