Hard Game

“Let’s play!”

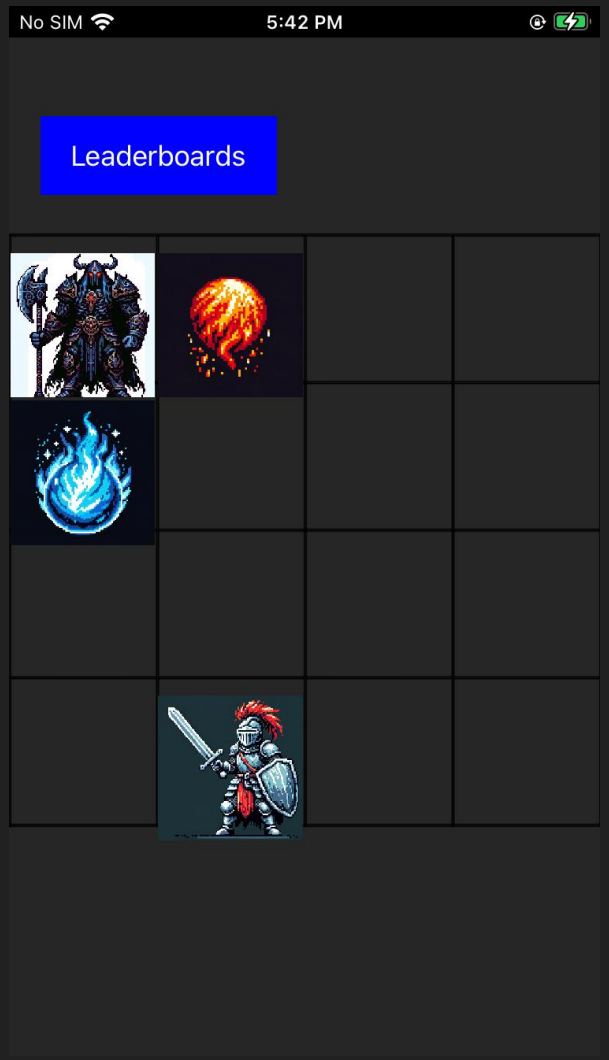
Oliver Raney
Fall 2023

What is Hard Game?

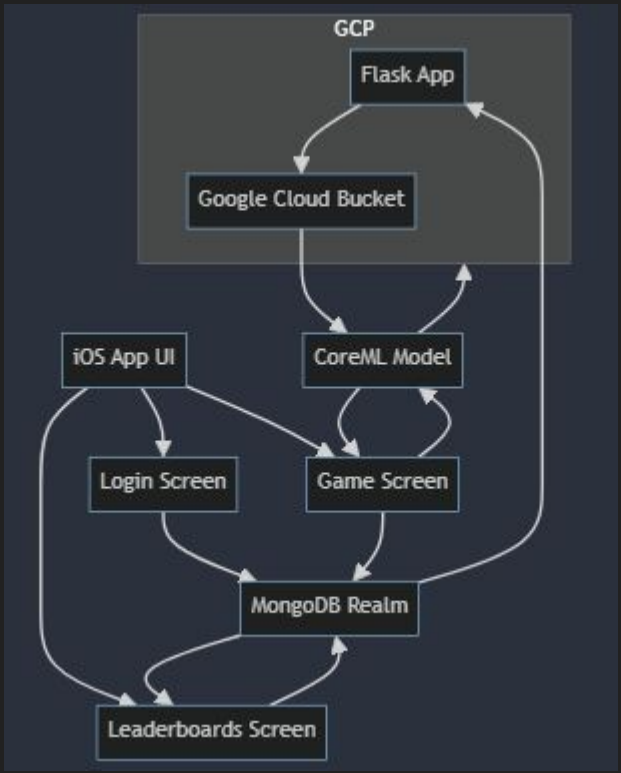
- You play a knight in shining armor, fighting an evil boss
- Each time you kill it, it “levels up”, retraining its Machine Learning model
  - Using scikitlearn and CoreML
- Fight for your leaderboard spot as the player to have killed the most well-trained version!
- Experience a challenge like never before
- The name “Hard Game” is intended to appeal to those gamers used to the ordinary “hard” AI.

Gameplay

- SpriteKit UI Scene
- Haptic Feedback
- Sound FX, Visual FX
- Tap to Move



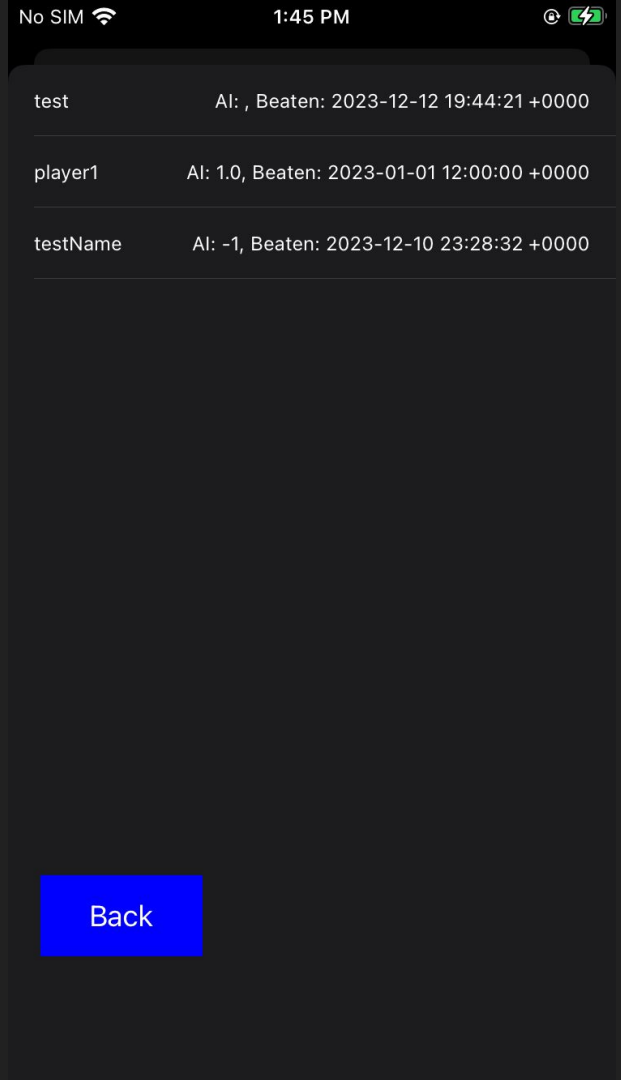
Architecture



- Swift App (iOS device)
- MongoDB
  - Authentication
  - Leaderboards
  - ML model metadata
- Google Cloud
  - Flask Python App
    - Trains ML models
  - Stores ML models

Leaderboards

- Dynamically updated using MongoDB requests
- Close and reopen app still remembers your wins
- Future state: show AI version beaten, on what date/time.
- These are real entries in the Mongo database



Google Cloud Flask App & Storage

The python Flask app on Google Cloud allows me to make HTTP requests:

```
@app.route('/upload_game_data', methods=['POST'])
```

- /upload_game_data: uploads the game data and retrains the model

```
@app.route('/get_latest_coreml_model', methods=['GET'])
```

- /get_latest_coreml_model: gets the latest model located in GC, converts it to CoreML, and sends it to the iOS app

The Swift App Itself

```
playerNode.position = CGPoint(x: size.width * 0.2, y: size.height * 0.2)
playerNode.size = CGSize(width: cellSize - 2, height: cellSize - 2)
playerNode.alpha = 1.0
playerNode.isHidden = false
playerNode.color = SKColor.red

addChild(playerNode)

playerNode.physicsBody = SKPhysicsBody(rectangleOf: playerNode.size)
playerNode.physicsBody?.categoryBitMask = PhysicsCategory.player
playerNode.physicsBody?.contactTestBitMask = PhysicsCategory.missile
playerNode.physicsBody?.collisionBitMask = PhysicsCategory.none
playerNode.physicsBody?.affectedByGravity = false
```

- Besides making HTTP requests, the app uses SpriteKit
- Here I am initializing the playerNode when the scene loads
- I also give this node a physicsBody so that missiles can hit it

App -> MongoDB

The app makes direct calls to the MongoDB realm, in multiple views. It uses a framework called “RealmSwift” which allows querying of the collections in the database.

In this code, I query for all of the documents in the “leaderboards” collection, which allows me to populate the leaderboards table in the App.

```
Task {
    let realm = try await openSyncedRealm()
    let subscriptions = realm.subscriptions
    let foundSubscription = subscriptions.first(named: "all_leaderboards")
    try await subscriptions.update {
        if foundSubscription != nil {
            foundSubscription!.updateQuery(toType: leaderboard.self)
        } else {
            subscriptions.append(QuerySubscription<leaderboard>(name: "all_leaderboards"))
        }
    }

    leaderboards = realm.objects(leaderboard.self)
    sortedLeaderboards = leaderboards.sorted(byKeyPath: "beatTime", ascending: true)
```

App -> Google Cloud

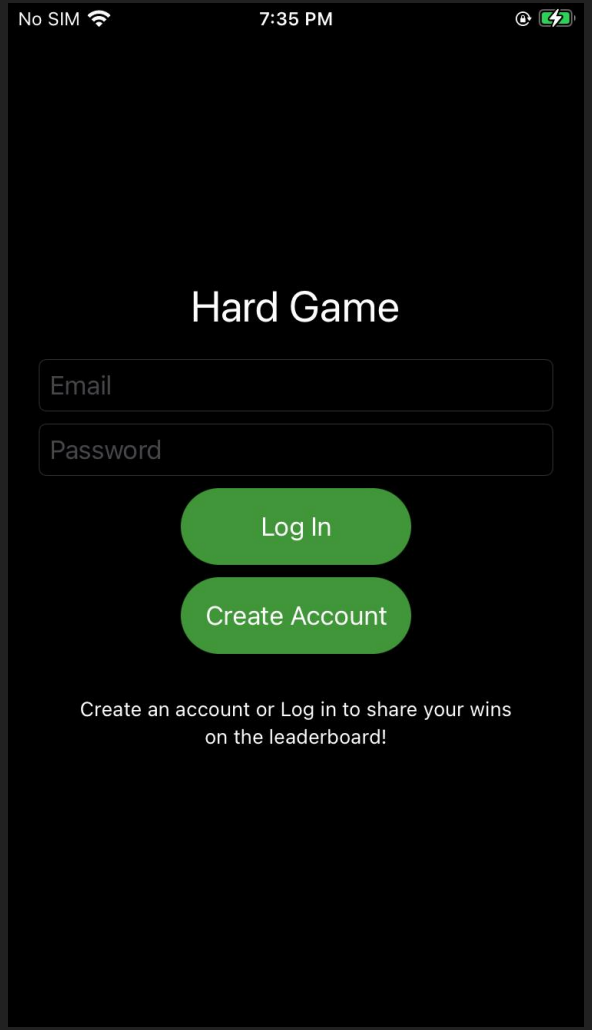
The iOS app also directly talks to GC via the HTTP requests shown before (e.g. /get_latest_coreml_model). This uses URLSession instead of RealmSwift.

```swift
func downloadLatestCoreMLModel(completion: @escaping (URL?) -> Void) {
    let url = URL(string: flask_app_url + "/get_latest_coreml_model")!
    let task = URLSession.shared.downloadTask(with: url) { localURL, _, error in
        guard let localURL = localURL, error == nil else {
            print("Model download failed: \(error?.localizedDescription ?? "Unknown error")")
            completion(nil)
            return
        }
        completion(localURL)
    }
    task.resume()
}
```

This function calls a GET request to grab the latest model in GC, which is a sklearn model, and convert to CoreML to download.

Users

- Hard Game keeps track of your wins
- Log in with the same account
- Create a new account
- Users uploaded to MongoDB
  - Realm-managed users table which includes email and passwords (I can’t see the passwords)
  - Dedicated “Users” collection which has “userName”, “accountCreatedDate” and other useful info



Constraints of the Project

- All of the functionality is coded, but some things aren’t working as of today
- In the future I want the app to run autonomously (i.e. all info stored on the cloud, no server that the user has to run locally)
- The game does utilize MongoDB as well as a scikitlearn classifier,
    - However, these are not working together and needs work in the future

Learning Outcomes

- Master of MongoDB Realms
- Master of SpriteKit scenes, nodes, and other settings
- Practiced some MVC techniques
  - Separate GameScene SKView file instead of logic in GameViewController
- Integrated non-storyboard elements with storyboard elements
- Grew my knowledge on lab-related material:
  - UI
  - ML
  - SpriteKit
  - HTTP Requests