Hide ‘n’ Seek: The Game

CS39440 Major Project Report

Author: Petter Vang Brakalsvålet (Pev2@aber.ac.uk)

Supervisor: Dr/Prof. Chris Price(cjp@aber.ac.uk)

24th March 2021

Version 0.7 (Draft)

This report is submitted as partial fulfilment of a BSc degree in
Computer Science (With Integrated Year In Industry)
(G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work, I understand and agree to abide by the University's regulations governing these issues.

Name Petter Vang Brakalsvålet

Date 29.04.2021

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name Petter Vang Brakalsvålet

Date 29.04.2021

Acknowledgements

I am grateful to my partner Alexandra S. Murphy for her love and support during this period and especially during the period where my code didn't work. I am also grateful to my parents for pushing me to do what I want and my brother, his partner, and my nephews for believing in me.

I'd like to thank my supervisor Chris Price for guiding me and for being patient with me.

I would also like to thank Stack overflow for the millions of answers they have given me during my period at Aberystwyth University.

I am grateful to monster energy drinks for keeping me awake during the long nights.

Abstract

This project is a 2D game for iOS which is based on a combination of the classic game “Hide and Seek” and the newly popular “Among Us’. It has been developed using Swift in XCode and is optimised for all iPhone and iPad devices.

After spending time in national lockdowns, Among Us is one of the many games and apps which became popular online, as it allowed multiplayer games which stretched across the world. I admired the layout, functionality, and creativity of the game. This, combined with the simple rules and fun gameplay of “hide and seek”, inspired my own creation of “Hide ‘n’ Seek: The Game” (a working title).

After research and experimentation, I have created an intricate and beautifully designed game which gives the player options for customisation, as well as a fun gameplay experience.

This report will go into further detail about the process of this project, and how I got to the final deliverables.

Contents

1. Background, Analysis & Process

1.1. Background

How I started:

Upon deciding that I wanted to make a 2D game with swift, I started by researching online to figure out where to start with my project. I started by watching a YouTube video playlist called “Beginning SpriteKit” by raywenderlich.com [1] to give myself an introduction to this type of iOS development.

I was also given a book by my supervisor called “2D Apple Game Development” by Tutorials [2]. This book walked me through how to create a simple game and gave me the starting point I needed to begin designing and developing.

Research:

To find out what the target audience already thought of this type of game, I investigated pre-existing Hide and Seek games on the AppStore. Here I found that there were a few games that fit into the same niche as mine, and only a couple of them had good reviews. However, I discovered that those with bad reviews were more helpful for my research, as they gave feedback on which features worked and which didn’t (from a user’s point of view). I found that all the games were “fun” and “addictive”. I also discovered that those which allowed customisation of the game had a better reception from the users than those which didn’t offer this feature. Most of the games locked the player into one role (hider or seeker) and they had a pre-defined game duration. It appeared that these features were the main turn-offs for the users, as they gave a limiting game experience.

Motivation:

My motivation for creating this game was that I had some experience with iOS app development, but I had never done any game development. I thought this would be a good challenge for me and it would be an opportunity for me to develop my skills whilst experiencing a new industry of mobile development. I was also inspired by Among Us [3] (a multiplayer game that rose in popularity during the COVID-19 Pandemic) which I spent a considerable amount of time playing.

Among Us is an online game where the players will be randomly assigned the role of imposter or crewmate. The crewmates have tasks they need to do, and the imposter's task is to kill all the crewmates. During the game, any player can call a crew meeting by clicking the panic button or if they find a dead body. Players are not allowed to talk outside of a crew meeting, but they can talk in crew meetings. In crew meetings everyone will vote on whom they think is the imposter and the person with the most votes will be killed. If the imposter is killed the game is won by the crewmates, but if the imposter kills every crewmate the imposter wins.

This game was the main inspiration for me to create a game. However, I wanted to put a spin on it. Instead of doing the same as Among Us, I wanted my game to be a variant of the classic game hide and seek, and I wanted the game to take place at a campsite in a forest.

My aim:

When I'm creating this game, I have set these aims for myself. I want to create a Hide and seek game where one person or bot is chosen to be the seeker and the rest of the players are hiders. The seeker's job is to catch every player and bot before the time runs out, and the hiders job is to outlast the seekers. If a hider is caught another hider can free them if they want to risk leaving their hiding spot. I don't want the hiders to stay in one place for too long, so to handle this I want them to take environmental damage or have the hiding spot make a noise every so often to call the seeker closer to them. I also want the seeker to be able to check a hiding spot, and if someone is hiding in it, they will be revealed and caught by the seeker. To make the game more challenging I want the game to take place at night and have each player and certain objects emit light. I hope that this will make the game more interesting and have the players more on edge when they are hiding. One important feature for games is customisability, so I want the player to set the duration of the game, the number of bots, what role they have and much more.

I dislike ads and especially ads forced on me when I'm doing something, so I don't want any ads in my game. However, I might add an optional ad button saying support the development.

1.2. Analysis

My original task was to create a hide and seek based game where the player would play against bots. The player would be able to choose what role they would play (hider or seeker) and how long the game would last.

After I did some research, I found out that customisability of the game and player interaction with the world is very important to create a successful game. I decided to make this my focus. To make my game more customisable, I gave the players the option of changing settings when they start a game (these settings include: the player's role; the reach, speed; the difficulty level and how many bots the game is going to have). By doing this, I ensured that the players could play a game personalised for their ideal gameplay as opposed to a universal standard gameplay.

I also discovered that I could use SpriteKit (and their integrated physics engine) or create my own parameters to track interference and node interaction. This meant I could handle player interaction much more efficiently by utilising SpriteKit.

Through using SpriteKit's physics engine, I had the ability to assign physics bodies to nodes and specify what each node will collide with. If the nodes are in the same height layer, they will collide into each other (automatically). It has the functionality to handle all the collisions for me while also allowing me to specify if I want something else to happen by changing SKPhysicsContactDelegate.

However, there is also the opportunity for me to handle these situations manually. This would be done by checking if a node is intersecting with another node and then handling the collisions and interactions. This is a less intensive way, which is better for the device running the game, but it wouldn't be as reliable as using SpriteKit's built-in features. I decided to use a combination of both these methods.

My combination approach was as follows: the physics engine handled collisions between nodes, but I manually handled interactions between the player, bots, and hiding spots (interactions include hiding, catching, or freeing of bots and players). By doing it like this, I created a less intense game which maintained the robustness given by SpriteKit and its physics engine.

1.2.1. Review of similar games

Hide 'N Seek!

This is a hide and seek game that takes place in a maze. This game has different mazes which you unlock by playing the game. It doesn't have any hiding spots, so you must hide around corners. This game gives you the option of playing as a seeker or a hider. As a seeker, you must run around in a maze and try to catch people, and as a Hider, you need to run away from the seeker. If you find a caught player, you can free them. Both seekers and hiders can pick up coins while playing the game. To make the game more fun and to give the players an incentive to play the game the developers have implemented an in-game shop where you can buy new skins for yourself with the coins you pick up during the game.

Positive	Negative
The game is fun and addictive, and people can be freed by other players when they have been captured. You can customise the character and earn in-game money.	The games ads aren't age appropriate for a 4+ game. The game has a lot of data tracking and most of the data collected is linked directly to you.

https://apps.apple.com/gb/app/hide-n-seek/id1491654968 [4]

Hide'N Find

This is a hide and seek game that takes place in a box with some walls randomly placed around the map. The hiders need to hide from the seeker by going around these walls or outrun them. The game gives you the option of being the seeker or the hider and depending on what you chose the duration of the game will change. When playing as a seeker your task is to catch all the hiders before the time runs out. The game has a shop where you can buy different skins for yourself. This game has coins placed randomly on the map so you can collect them while you play.

Positive	Negative
The game isn't tracking any data nor linking any data to you. The game is fun	The games seeker will only target you when you are the hider, and plainly ignore every other player. The seekers in this game can also catch you through walls.

https://apps.apple.com/gb/app/hiden-find/id1509189851 [5]

Hide & Seek Fun Game

This is a hide and seek game where you can choose between two maps (jungle or pyramided). The games allow you to play as a seeker or a hider. The seekers task is to catch all the hiders, and the hider's role is to run away from the seekers until the time runs out. When the player starts a game, they can set up how smart the seeker is. The players can choose between free outfits or they can use real money to buy new outfits.

Positive	Negative
This game gives you a lot of customisability in the form of changing controls, outfit, maps, roles, and game difficulty.	The game has no indication of who the seeker is. It has a messy user interface where everything is moving constantly. When the game is paused for any reason, you will be prompted with an ad.

https://apps.apple.com/gb/app/hide-seek-fun-game/id1495168019 [6]

Hide and Seek..!

This is a hide and seek game where you play inside of a house, and you are the seeker trying to find the hiders. The game is using physics and you can move furniture around. This game takes place in the dark and you can't see any hiders unless they get inside of your field of view, or they are caught.

Positive	Negative
The game looks very nice and has a good user interface. The game also included physics in the game with object falling when moving too close.	The game forces you to be the seeker and it doesn't have any customisability. The field of view is very small which makes it very hard to find someone.

https://apps.apple.com/gb/app/hide-and-seek/id1537051480 [7]

From resourcing these games, I have learned a few things.

- it's not important to have specific hiding spots, but you need to have some way of hiding. This could be just hiding behind another object or inside of objects.
- The seeker is difficult to get right and might require a lot of work, and a lot of testing. The seeker must try to catch everyone and not just the player.
- Having a field of view can make your game worse than not having it. However, I would like to try to restrict the field of view if I have the time for my game.
- The UI is very important for the game, and if the UI has a lot of animations it can make users not want to play your game simply because too much is happening.

1.3. Process

I tried to use an agile approach to develop this game, where I split my time into separate sprints. Each sprint gave me an opportunity to divide up the work and prioritise them based on importance. I also used stories to track the work and measure my progress. Each sprint was a 2-week period, beginning with a small planning session and culminating in a review and retrospective. This gave me opportunities to review and improve my working practices, while also ensuring all the work was completed by the respective deadlines.

Here is a brief overview of what happened in each sprint:

1.3.1. Sprint 1 & 2

In the beginning, I focused on learning how to develop 2D games for iOS and to build a library of game assets. I started by creating assets for the game, and by the end of Sprint 2 I had created most of the assets needed. Through Sprint 2 to Sprint 4, I worked through Chapter 1 to 3 of 2D Apple game by tutorials [2]. This gave me a basic understanding of Game development for iOS.

1.3.2. Sprint 3 & 4

In Sprint 3, I investigated similar games and in Sprint 4, I tested the most popular games. This gave an idea of which features to include and which to leave out. I found that most of the popular games had lots of customisation for the player, and that they give the player the option to choose which role they will have.
In Sprint 3, I also started to create the game. I began by creating the main menu and setting up a help page, but I didn't add any content to the help page. After this, I created a game setup page, and I added a simple game scene.

1.3.3. Sprint 5

During Sprint 5, I finished the last chapters of the "Getting Started" section of the 2D Apple games book [2]. After this point, a lot of technical work started to get done and my focus shifted to building a "functioning" game.

1.3.4. Sprint 6

In Sprint 6, I added the players, tents and house and I expanded the environment. Since I now had players in the game, I worked on developing the player interaction. I also gave the player the option to hide (if they are a hider) and to catch other players (if they are a seeker). Due to this addition, I added functionality to the game setup page. The player would now be able to set their role, the number of bots and many more settings. One problem was that the players would always see the full screen; to fix this, I added a camera and made the camera move with the player.

1.3.5. Sprint 7

In Sprint 7, I added bots and collision detection to the game. This made the player interaction less predictable, which would need to be improved in the future. I now had a functioning game, but I didn't have any bot movement. I decided to prioritise adding automatic tests, and I added a victory check to the game instead.

1.3.6. Sprint 8 (Final Sprint)

In the final sprint, I focused on the documentation instead of programming as I had most of the functionality complete. However, I added some content to the help page, and I added simple bot movement. I had some tasks left in the backlog which I didn't have time to implement. Some of the features I didn't have time to add were pausing, trees, complex bot movement, reduction of field of view, implementation of light and shadows, and the finished page.

2. Design

2.1. Overall Architecture

For my project, I used SpriteKit architecture. I did this because SpriteKit is Apple's framework developed for 2D game development, and it can handle different actions, physics, and constraints. SpriteKit runs in a loop and does different steps (see image below).

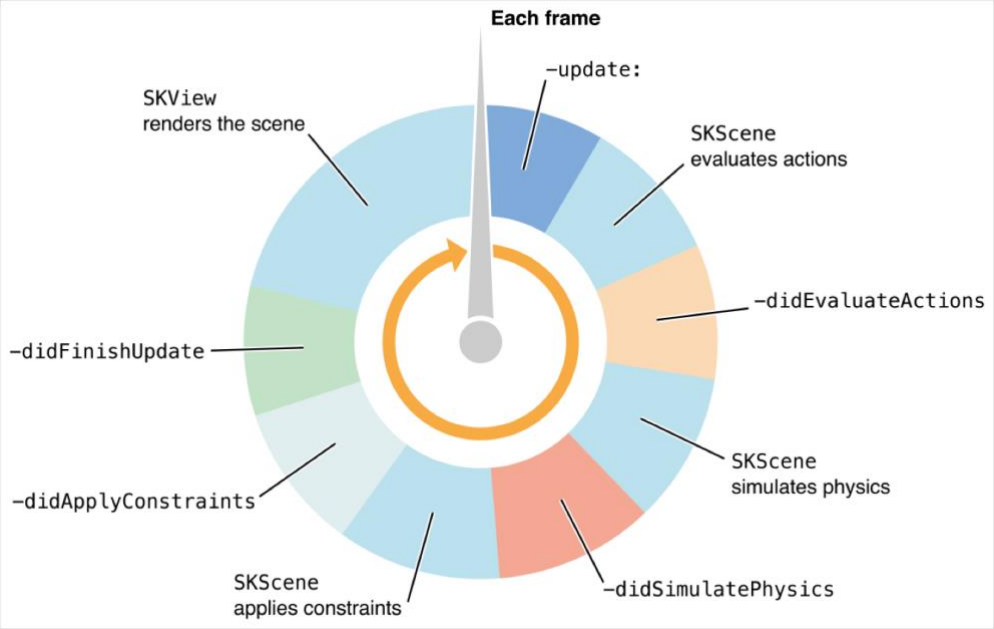


Figure 1 SpriteKit loop. See developer.apple.com [8]

I split the functions up into four groups which I will explain in the sections below.

2.1.1. Update

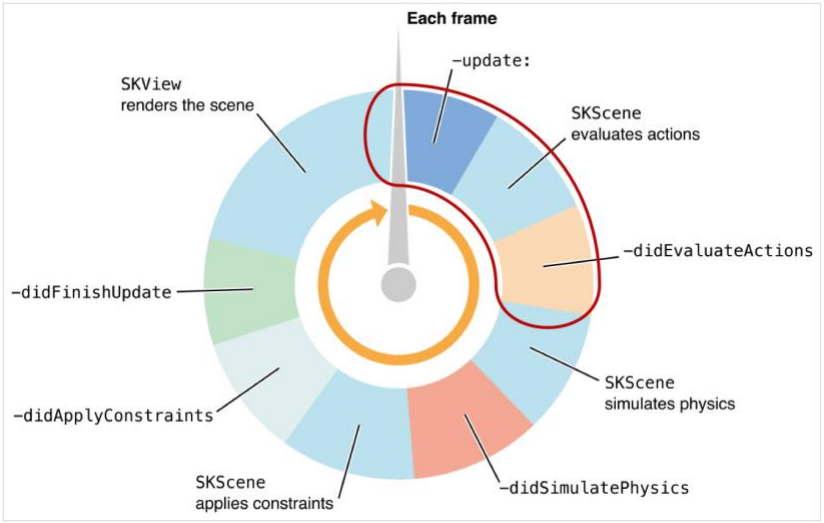


Figure 2 SpriteKit loop. See developer.apple.com [8]

The first section of functions to be run in the update loop is updating the section. This section is where most of the code for the game is performed. I used this section for updating the movement for all nodes, and I used this to handle the player interactions.

2.1.2. Physics

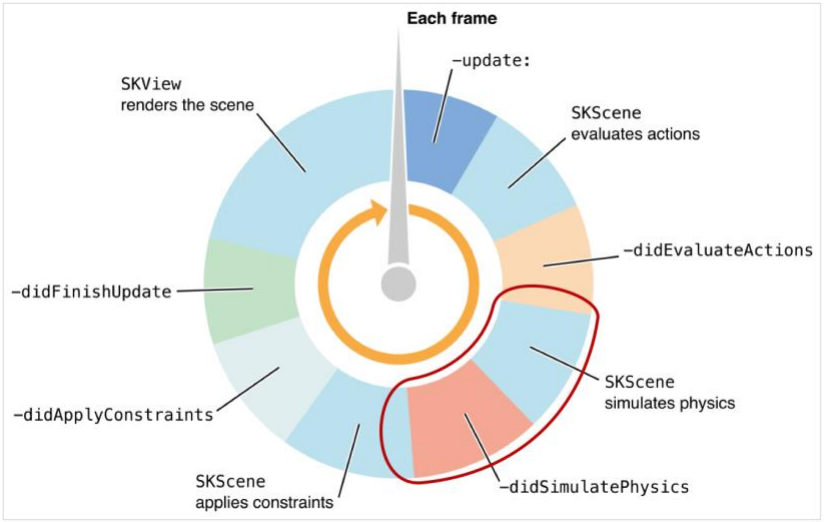


Figure 3 SpriteKit loop. See developer.apple.com [8]

This section handles the simulation of physics. SpriteKit can simulate some real-world physics like gravity and collisions, and this happens in this section. I'm using the physics simulation to handle the collisions in my game.

2.1.3. Constraints

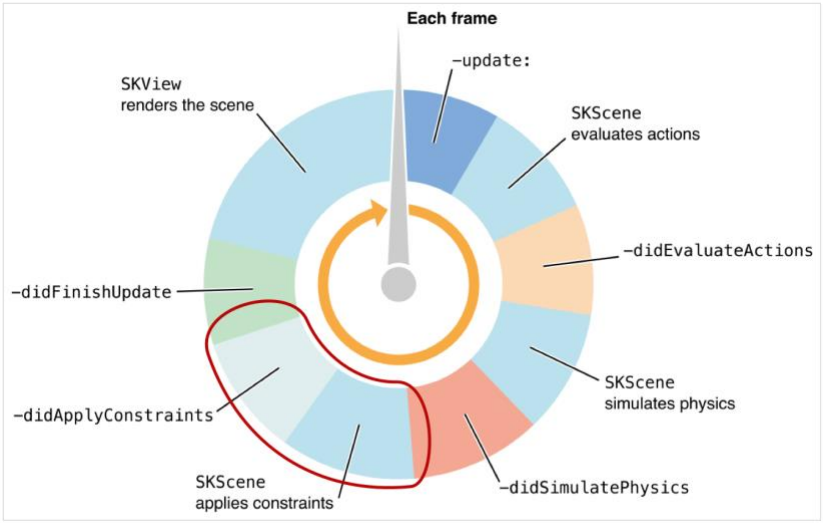


Figure 4 SpriteKit loop. See developer.apple.com [8]

In this section, constraints will be applied to the nodes of the game. I didn't use any constraints for my game. I handled this by moving all nodes separately. However, this section can be used to set relationships between nodes. This means that I could have multiple nodes pointing in the same directions and use constraints to make this happen. I have not used this in my code.

2.1.4. Rendering

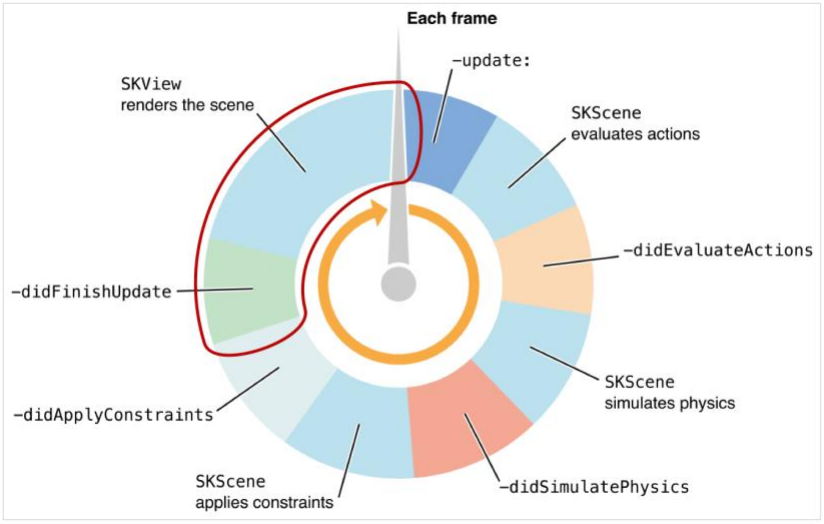
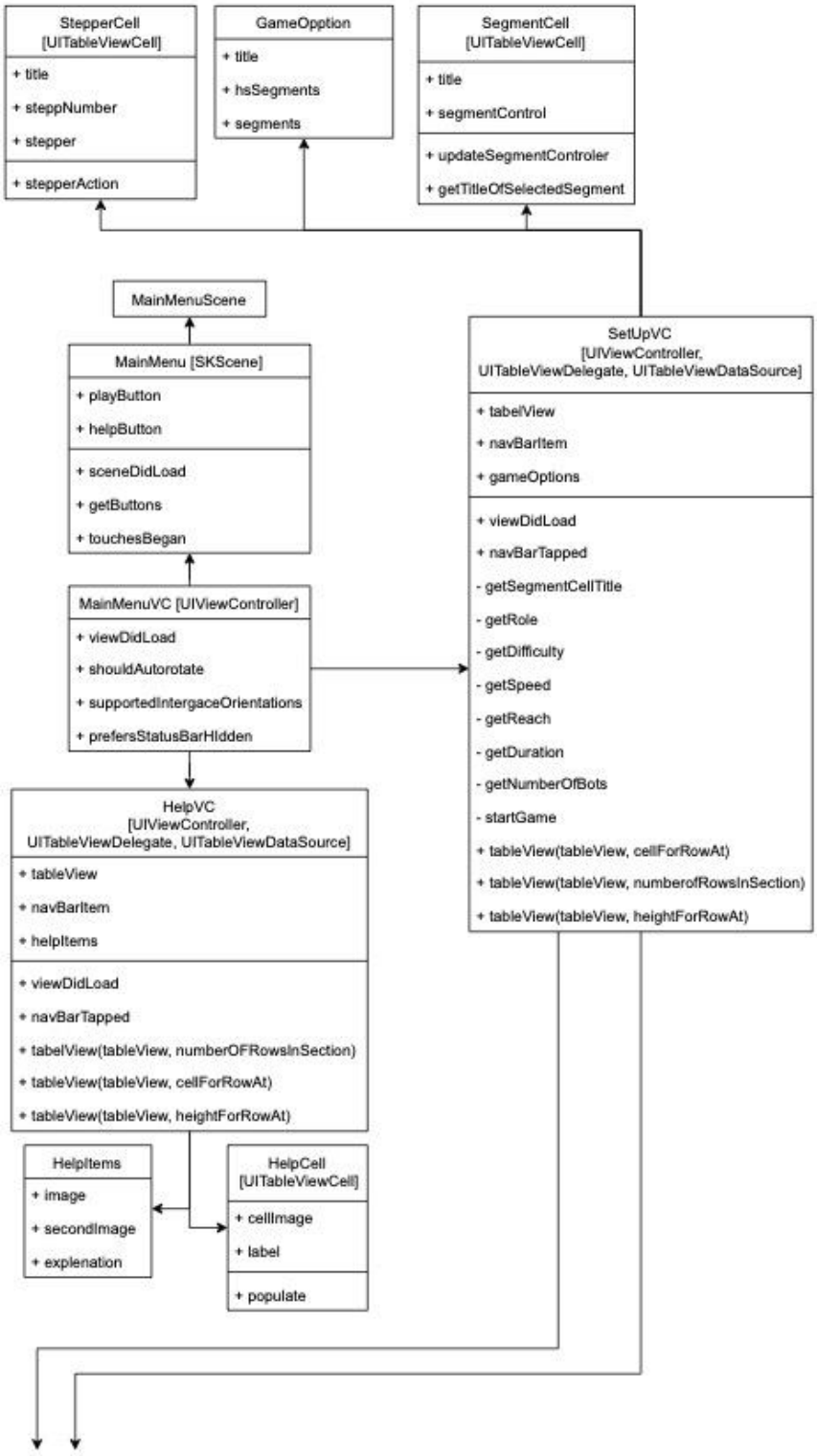
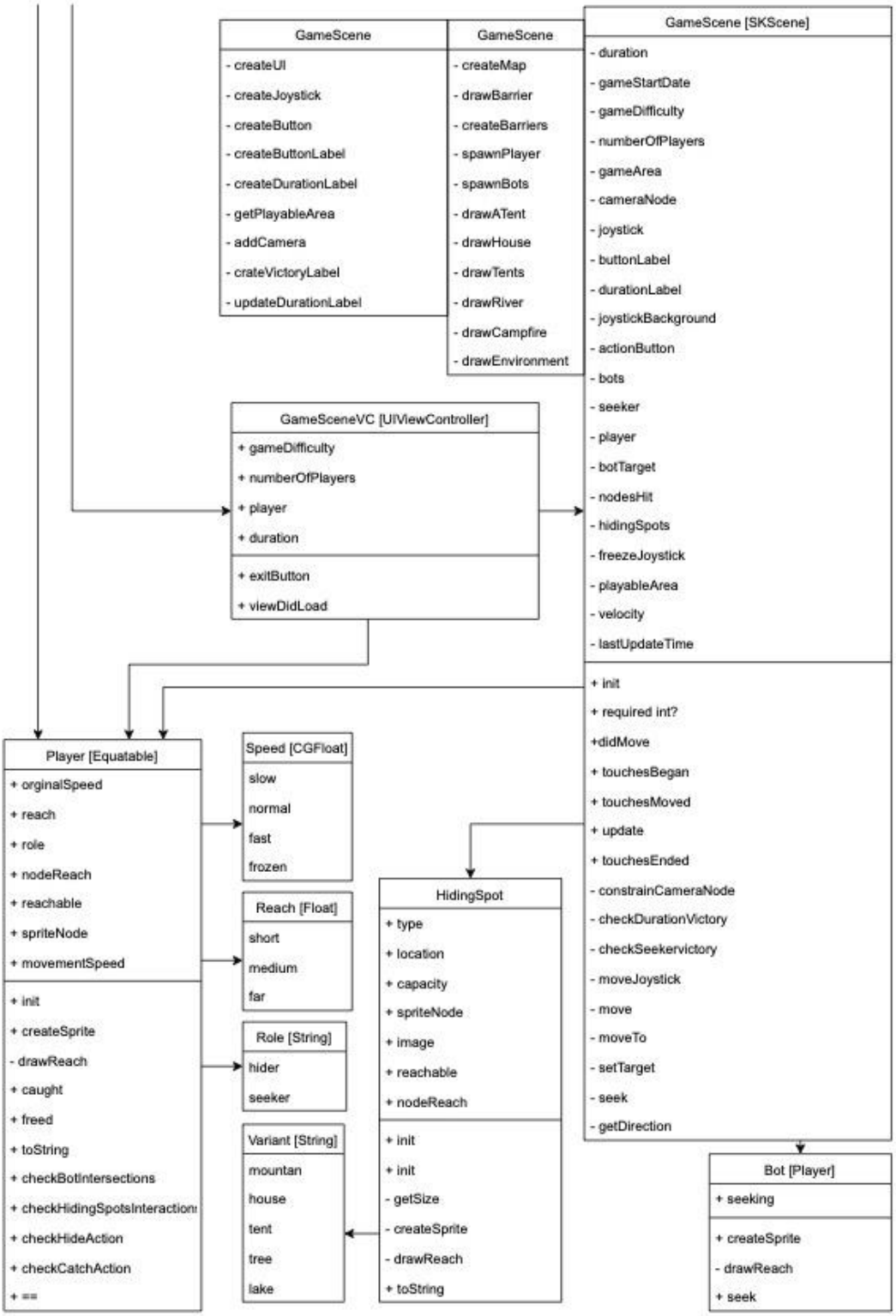


Figure 5 SpriteKit loop. See developer.apple.com [8]

This section is responsible for showing the texture for each node in a scene and the scene itself.

2.1.5. UML Diagram




GameScene
- createUI
- createJoystick
- createButton
- createButtonLabel
- createDurationLabel
- getPlayableArea
- addCamera
- crateVictoryLabel
- updateDurationLabel
GameScene
- createMap
- drawBarrier
- createBarriers
- spawnPlayer
- spawnBots
- drawATent
- drawHouse
- drawTents
- drawRiver
- drawCampfire
- drawEnvironment
GameScene [SKScene]
- duration
- gameStartDate
- gameDifficulty
- numberOfPlayers
- gameArea
- cameraNode
- joystick
- buttonLabel
- durationLabel
- joystickBackground
- actionButton
- bots
- seeker
- player
- botTarget
- nodesHit
- hidingSpots
- freezeJoystick
- playableArea
- velocity
- lastUpdateTime
+ init
+ required int?
+didMove
+ touchesBegan
+ touchesMoved
+ update
+ touchesEnded
- constrainCameraNode
- checkDurationVictory
- checkSeekervictory
- moveJoystick
- move
- moveTo
- setTarget
- seek
- getDirection
GameSceneVC [UIViewController]
+ gameDifficulty
+ numberOfPlayers
+ player
+ duration
+ exitButton
+ viewDidLoad
Player [Equatable]
+ orginalSpeed
+ reach
+ role
+ nodeReach
+ reachable
+ spriteNode
+ movementSpeed
+ init
+ createSprite
- drawReach
+ caught
+ freed
+ toString
+ checkBotIntersections
+ checkHidingSpotsInteraction
+ checkHideAction
+ checkCatchAction
+ ==
Speed [CGFloat]
slow
normal
fast
frozen
Reach [Float]
short
medium
far
Role [String]
hider
seeker
Variant [String]
mountan
house
tent
tree
lake
HidingSpot
+ type
+ location
+ capacity
+ spriteNode
+ image
+ reachable
+ nodeReach
+ init
+ init
- getSize
- createSprite
- drawReach
+ toString
Bot [Player]
+ seeking
+ createSprite
- drawReach
+ seek

2.2. Detailed Design

I have split my project into multiple folders to structure the project and make it more readable. I will explain here what each folder contains and what the files do.

2.2.1. MainMenu

Figure 6 A snapshot of the main menu of the game

This is the main menu screen where the player can choose to start the game or go to the help page. This scene is designed to show some of the aspects of the game, such as light coming from the campfire, tents to hide in and the mountain in the background which works as a barrier on the west side of the map.

The mainMenu folder contains all files needed to display the screen and to handle the inputs that might come from the screen.

MainMenuVC.swift

This file will load a SpriteKit scene and display it in the view. This file is inheriting from UIViewController so that I can display and handle the data.

MainMenuScene.sks

This file is a SpriteKit scene file, and here I have created the main menu scene using the user interface.

MainMenu.swift

This file is a SpriteKit scene file. It's here where all the programming for the main menu is located, and it's here, I'm handling actions from the scene. Since this file is handling actions happening in MainMenuScene.sks. It is inheriting from SKScene so that I can override some of the functions to handle the data.

2.2.2. Help

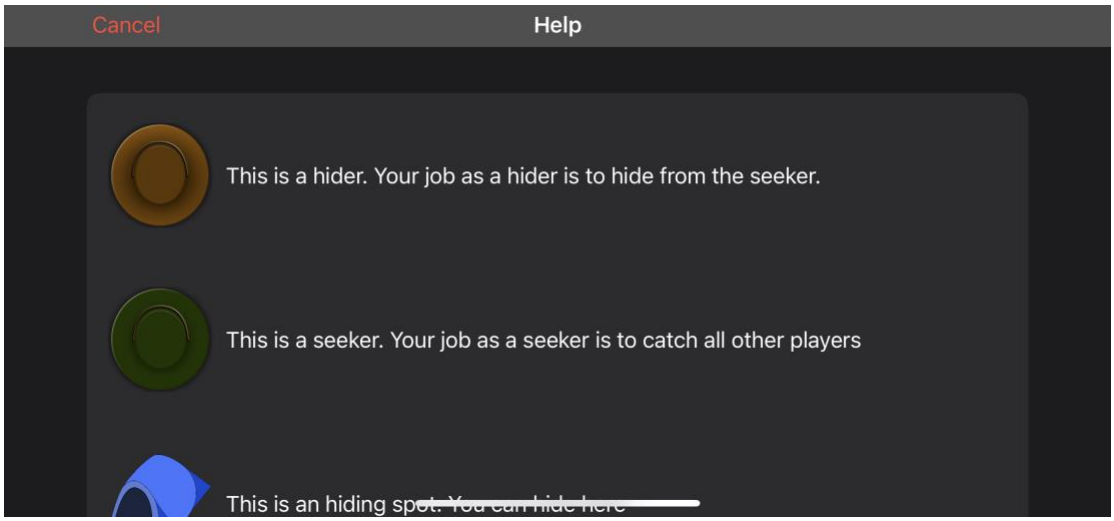


Figure 7 A snapshot of the help screen of the game

This is the help page. It has a brief description of what some items are and their intended purpose.

This folder contains the files related to the Help screen.

HelpVC.swift

This file will generate the help items and display the help data.
The help data is created by using a struck called HelpItem, this struck specifies the data needed for each HelpItem. Each HelpItem needs to have one image and an explanation of the image, but they have an optional second image. The second image will be displayed on top of the first image if used.
This file is inheriting from UIViewController, UITableViewDelegate and UITableViewDataSource. I'm using a UITableView to display and organise the data, and since I'm using a UITableView I need to override some functions so that I can supply the data to the UITableView and so I can handle user input.

HelpCell.swift

This file is my custom cell, and it's inheriting from UITableViewCell. The cell is created by the UITableView located in HelpVC.swift. This file will populate the cell with data passed to the cell.

2.2.3. SetUp

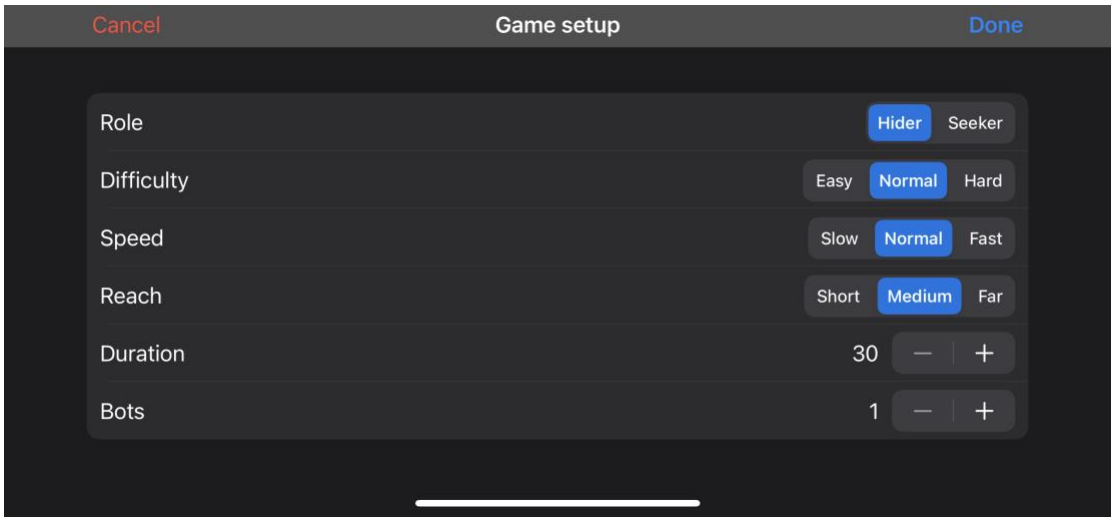


Figure 8 A snapshot of the setup screen of the game

This scene is where the user can customise the game to their preferences. The player can set all of these to what they prefer, but the duration and bots options are limited to a minimum of 30 seconds and a maximum of 6 minutes and a minimum of 1 bot and a maximum of 5 bots.

SetUpVC.swift

This file will generate the different options the user has to customise the game and display the options. The game options are created by using a struck called GameOption, this struck specifies the data needed for each GameOption. Each GameOption needs to have a title and specify if it has segments or not. It also has an optional segment variable; this is an array of all the segments needed for this GameOption.

This file is inheriting from UIViewController, UITableViewDelegate and UITableViewDataSource. I'm using a UITableView to display and organise the data, and since I'm using a UITableView I need to override some functions so that I can supply the data to the UITableView and so I can handle user input. This UITableView can have two different cells to display the data. Which cell depends on if the GameOption has segments or not.

SegmentCell.swift

This file is my custom cell, and it's inheriting from UITableViewCell. The cell is created by the UITableView located in SetUpVC.swift. This file will populate the cell with the data provided.

StepperCell.swift

This file is my custom cell, and it's inheriting from UITableViewCell. The cell is created by the UITableView located in SetUpVC.swift. This file will populate the cell with the data provided.

2.2.4. GameScene

Figure 9 a snapshot of the game scene of the game

This is the game scene. The scene contains some static user interface (the joystick to the left of the screen and the button to the right of the screen). The player is in the centre of the screen (The brown hat). There are some tents and a house where the hiders can hide. The green hat on the screen is the seeker, and in this case, it's played by a bot. The player has a set reach, and if the players' reach intersects with another node, it will reveal a label under the button on the right stating the action available. At the top of the screen, there is a timer counting down how much time you have until the game is finished.

GameSceneVC.swift

This is the view controller file for the game scene and It's inheriting from UIViewController. It will handle the closing of the game scene and load the game scene.

GameScene.swift

This is the game scene itself, unlike the main menu, this scene is created programmatically. This file will display and handle the game and its mechanics. This file will handle all the touches and how this will affect the nodes in the scene.

2.2.5. Models

HidingSpot.swift

This is a class to store all the data and functions needed for a hiding spot. This class contains an Enum called Variant which all the different hiding spots in the game. This Enum can return a string of the Variant. This class will create an SKSpriteNode and assign the values to it automatically.

Player.swift

This is a class to store all the data and functions need for a player. This class contains three Enums called Role, Reach, Speed. These Enums are created to make the class more readable and stores certain values for the different options. This class will create an SKSpriteNode and assign the values to it automatically. This class contains functions for checking what actions can be performed, running animations, and checking if the players reach is intersecting with any others.

Bot.swift

This is a class to store data relating to a bot. This class is inheriting from the Player class since a bot and a player is very similar. However, this class is overriding some of the functions so a bot is created and not a player.

2.2.6. Extensions.swift

This is a file containing any Enums or any other functions that need to be reached by the whole target. The Enums here are ChallangeRating and ColliderType these Enums store specific values for different cases. In this file, I'm also extending on SKSpriteNode with a function that can change the size of an SKSpriteNode but also keep the aspect ratio of the texture used by the SKSpriteNode.
This file contains some functions from 2D Apple Games by Tutorials [2] These functions are only to simplify mathematics in the code are copyrighted. The copyright is stated before the functions in question. See Appendices A.A

2.2.7. Main.storyboard

This file is where I have created the view controllers with storyboards. It contains all the different scene and, it has some dummy data for some of the scenes.

2.3. User Interface Design

2.3.1. Start screen assets

Unless otherwise stated, all assets have been designed and produced by myself using Affinity design.

Play button

Figure 10 play button

This is a button for the main menu, It's a plank with the written option on it.

Help button

Figure 11 help button

This is a button for the main menu, It's a plank with the written option on it.

Stick

Figure 12 stick

This is a stick for the main menu. I have the other main menu options on it as a sign with different labels.

Start screen

Figure 13 start screen

This is the background image for the start screen. This is a snapshot of the game in action and is designed to show what can happen.

2.3.2. Game scene assets

Campfire

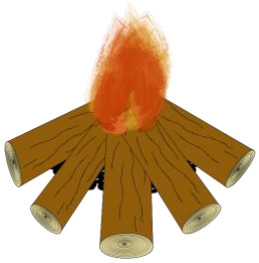


Figure 14 campfire

This is a simple campfire.

Mountain

Figure 15 mountain

This is a mountain created for the left side of the map.

Game background

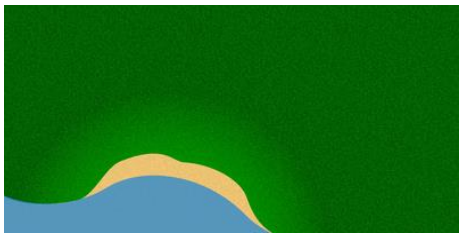


Figure 16 game background

This is the background image for the game scene.

River

Figure 17 river

This is a simple river to cut off the map.

Hider

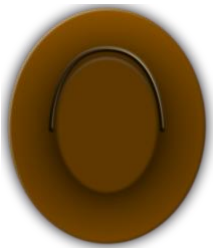


Figure 18 hider

This indicates a “hider”. It’s supposed to look like a hat from top down.

Seeker

Figure 19 seeker

This indicates a “seeker”. It’s supposed to look like a hat from top down.

Tent new

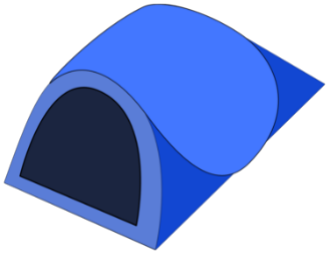


Figure 20 tent new style

This is a tent where a player can hide.

Tent old

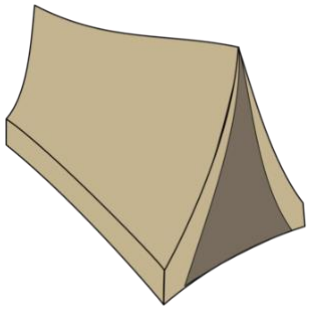


Figure 21 tent old style

This is a tent where a player can hide.

House

Figure 22 house

This is a house where multiple people can hide.

2.3.3. User interface

Button

Figure 23 button

This is a generic button. I found it on pixabay.com [9] under the license of **Pixabay License (https://pixabay.com/service/license/)**

Joystick background

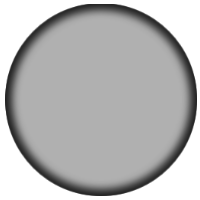


Figure 24 joystick background

This is a plain background I created for the joysticks.

Joystick node

Figure 25 joystick node

This is the joystick itself.

3. Implementation

As I don't have any game development experience, I needed to go through a few tutorials to figure out how to develop the game. While I was going through some tutorials on game development for iOS, I was also creating most of the game assets needed for the game. I did this so when I started to develop the game, I could use the assets I had created, and I wouldn't need to stop to create new assets.

3.1. Main menu

I started by creating the main menu screen as a SpriteKit scenes and creating it using the UI. I Added a background image as a SKSpriteNode, and I wanted to have two buttons (a play button and a help button). Instead of having a plain button I decided on creating a sign with two planks and have play and help engraved on it (This would also go with the theme of the game). These planks would act as buttons, and I wrote a function that would check where the user clicks and if the user clicked on one of the buttons it would navigate to a the play or help screen.

3.2. Set-up

To create the set-up screen, I decided that I would do this by creating it as an app screen and by using storyboards. Since this was going to be a settings page, I thought it would be a good idea to use a table view and create two different custom cells for the different types of settings. I also added a navigation bar and displaying the screen title. I added a cancel and done button in the navigation bar so that the user could navigate backwords and forwards.

When using a table view and custom cells it is required for me to create supporting code for this to work. I would also need to create code to retrieve the data from each cell after the users are done editing the settings. This data would then be sent to the game scene when that created.

3.3. Help

To create the help screen I wanted to do it the same way I created the set-up screen with storyboards and table views. I created a custom cell and decided that each cell should contain an image and some text next to it. The text would be used to explain how an item would work and the image would display the item in question. I added a navigation bar with a title and a back button for the user to navigate back.

Since I'm using a table view with custom cells, I would need to create supporting code for this. I created a struck for what information was need for each help item and I made an array containing these help items specifying the images and the explanation for them.

3.4. Game

I decided to create the game scene programmatically rather than making it whit the UI. I did this because the game is going to have many sprites moving around and because this would improve the loading time for the game.
When I started to create the game, I had the whole map visible so that I could see where I added sprites, and since I didn't have movement, yet I would be able to see the sprites otherwise.

I created the movement for the player by using a joystick, but I didn't want to use a third-party joystick and decided to create my own. My joystick works by calculating what direction it has moved and then add this to the movement speed and then add this to the sprite's location, and the sprite's location would be updated in the update stage of the SpriteKit loop. Since I was now able to move around, I wanted to add a camera node to the scene and zoom in on the map, and by doing this I would restrict the player's field of view.

I would now start to work on player interaction with objects. I did this by calculating the distance between the player and any hiding spots and if a hiding spot was within the reach of the player, I would update the button label with the action available. Calculating the exact distance between every object in a scene is very intensive on the device, and this would create a potential problem when I would add more elements to the game. I changed this later by using node intersections instead, and this could have resolved a potential problem in the future.

At this point, I started adding bots into the games and working on player and bot interactions so that seekers can catch hiders and hiders can free captured hiders. I implemented player and bot interaction by using node intersection; The same way I solved player and object interactions.

At this point, I had bots and hiding spots, so it was time to add collisions and boundaries to the game. I researched how I could handle collision detection, and my first thought was to use node intersections. I would check when node intersection occurred and then limit the direction the player could move. This approach was challenging, and I decided to try to use SpriteKit's built-in physics engine instead. I assigned physics bodies to all nodes that can collide. Physics bodies worked great bought for collision detection. I decided to create a wall on each side of the map and then giving it a physics body to create the boundaries. By utilising SpriteKits physics, I would have a reliable barrier around the map.

I would now attempt to add bot movement to the game, and this proved to be more difficult than I thought. I didn't manage to add any movement for the hider bots. However, I managed to add a simple seeking movement for the seeker bots. I used the sprites built-in run action to move the seeker bot towards the target. I used the move action, which makes a sprite move to a position over some time, and I gave it the location of one of the hiders. The seeker was now capable of seeking, but they couldn't catch hiders. I tried to use the player and bot interaction I had already created to capture a hider if they were close enough. At this point, I was happy with what I had done, and I started to refactor the code and write documentation for all my previous code.

3.5. Bugs

The game has some bugs that I know about, and there are probably some bugs that I don't know about yet. I will go into detail about the bugs I have, why they happen and how I would fix them if I had more time.

3.5.1. SetUp

When setting up the game in the set-up scene, the game might crash; This happens when some of the cells aren't visible, and the player clicks done. This happens because when the done button is pressed, the game will gather all the data from the table view and send them to the next scene. The problem is that when a cell is outside of the screen, the program can't find one of the cells and so can't retrieve the data for that cell, and then result in the game crashing. This problem should be an easy problem to fix, and to my understanding, I could resolve this by adding an observer to each cell. The observer would look for changes in the cell, and it would trigger an event. I would then retrieve the data when this event was triggered. If I implemented this the data needed to start the game would be updated whenever the user changed a value.

3.5.2. Game

In the game, there are some bugs.

- When a bot is controlling the seeker, they might jump around if there is an object blocking their path. This happens because of how I have set up bot movement and could be fixed by using GameplayKit. GameplayKit is Apple's framework for bots and Artificial game intelligence, and it includes pathfinding and agent behaviours among other things. If I could use pathfinding to handle this bug, and I could expand on this by implementing agent behaviour. Agent behaviour can be used to create more intelligent bots by giving the bot goals and making it react to its surroundings.
- When a bot is controlling the seeker, the seeker might catch itself. I'm not sure why this happens, but I think I could resolve the problem by checking who the seeker is trying to catch, and then make sure it can't try to catch itself.
- If the user is playing as seeker and they catch every bot, they don't automatically win the game, but they must wait until the time runs out. This could be fixed by checking adjusting the parameters for triggering a victory.

4. Testing

4.1. Overall Approach to Testing

I have chosen to use some automatic test to validate the data, but to test the game I'm using manual testing. I decided to do manual testing for the game because I'm not interested in what happens for specific actions, but when obscure user input happens and how my game handles that.

4.2. Automated Testing

I have implemented some automated unit tests for the game, but I don't have any automated test for the game mechanics or the UI. The tests I have written will test the data used for each table view, and it will validate the number of children in each scene. My tests will check if each cell of the tables is the same as expected.
You can find my tests in the appendices A.C as well as in my technical work.

4.3. Manual Testing

4.3.1. MainMenu

Id	Description	Outcome	Pass/Fail
1	When launching the game, them main menu should appear.	When clicking on the game the main menu appears.	Pass
2	When clicking on the Play button, the setup screen should appear.	When clicking the Play button, the setup scene appeared.	Pass
3	When clicking on the Help button the help scene should appear.	When clicking the Help button the help scene appeared.	Pass
4	When clicking on any other object nothing happens.	When clicking anywhere except for the button, noting happens.	Pass

4.3.2. Help Test

Number	Description	Outcome	Pass/Fail
1	When scrolling in the list the list moves up or down.	When dragging up or down the list moves up or down revealing more content.	Pass
2	When clicking on an item in the list it becomes highlighted.	When tapping on any of the rows in the list that list becomes highlighted.	Pass
3	When clicking on the cancel button the scene closes	When cancel is clicked the scene is closed and the previous scene is showed.	Pass

4.3.3. SetUp

Number	Description	Outcome	Pass/Fail
1	When clicking on one of the segments in a row the selection is moved to that one.	When clicking on the seeker segment the selection highlight is moved to that one.	Pass
2	When clicking on the plus or minus the number next to it increases or decreases.	When clicking plus the number increases and when clicking the minus, the number decreases.	Pass
3	The cells with a stepper should have a minimum value.	When clicking the minus, the number won't be lower than the value showed when the screen is loaded.	Pass
4	The cells with a stepper should have a maximum value	When clicking the plus, the number will increase until the number equals to 5 for the cell with bots and 360 for the duration cell	Pass
5	When scrolling the list moves	When dragging the list up or down the list moves up or down.	Pass
6	When clicking on the cancel button the scene closes	When cancel is clicked the scene is closed and the previous scene is showed.	Pass

7	When clicking on the Done button a new scene will show	When clicking the Done button, the game scene is loaded and showed. However, if some of the items of the list is not showed the app crashes	Partial Pass

4.3.4. GameScene

Number	Description	Outcome	Pass/Fail
1	When the game starts the time, left is decreasing.	When the game starts the time lefts label at the top of the screens slowly decreases to 0	Pass
2	When the time left hit 0 the game ends	When the time left label hits 0 a label is showed stating, who won the game, and the game goes back to the main menu. But if the seeker catches the last player after the time runs out the seekers will also win.	Partial pass
3	When a bot is the seeker, the seeker will chase the player.	The seeker starts moving after 5 seconds and will chase the player.	Pass
4	When all hiders are caught the game ends.	When all the hiders are caught a label is showed stating who won, and the game goes back to the main menu.	Pass
5	When clicking on the button when close to a player a player is caught or freed.	When clicking the button and a player is close the player will be caught or freed depending on if the player is a hider or a seeker.	Pass
6	When clicking on the button when close to a hiding spot the player will be hidden.	When clicking the button and a hiding spot is close. The player will hide if they are a hider, if not nothing will happen.	Pass
7	When clicking the button far away from	When clicking the button and noting is close to the player nothing will happen	Pass

	everything, nothing happens.		
8	When moving the joystick in any direction the player moves in that direction.	When moving the joystick in any direction the player moves in that direction	Pass
9	When moving the joystick and the player is hidden, the joystick doesn't move	When moving the joystick and the player is hidden, nothing happens.	Pass
10	When moving the joystick and the player is caught nothing happens.	When moving the joystick and the player is caught nothing happens.	Pass
11	When clicking anywhere outside of the joystick the joystick doesn't move	When clicking on the screen the joystick doesn't move, but sometimes when clicking and dragging on the button the joystick might move.	Partial pass
12	When the player is standing next to an object, it shouldn't move through it.	When the player is next to an object and the player tries to move in the same direction the player won't move through it.	Pass
13	When a seeker is close to a hider the hider will be caught	When the seeker is close to a hider they are caught.	Pass
14	The seeker will look for any hider	When the seeker is controlled by a bot it will only look for the player.	Partial Fail
15	When starting a game all hiders will be free.	When starting a game with more than 2 player one hider will be caught from the start.	Partial Fail
16	When a Bot ties to move in the direction of an object it will move around it.	When a bot tries to move in any direction and an object is blocking it's path the seeker will jump around multiple times until it's on the other side.	Fail

17	When any player is moving out of the map they will be stopped.	When trying to move outside of the map the players are stopped by an invisible wall.	Pass
18	When a player is hidden the seeker can't catch them	When a player or bot is hidden the seeker can't catch them.	Pass
19	Hiders can't hide forever	The hiders aren't limited on how long they can hide at one place.	Fail
20	If the player is caught the seeker will try to catch someone else	The seeker will only try to catch the player	Fail
21	When clicking the exit symbol, the game is closed.	When the exit symbol in the top left corner is clicked the game is closed and the previous scene is showed.	Pass

5. Critical Evaluation

5.1. Process

I have tried to use an agile approach to this project, where I split my time into sprints. I have used stories to track the work and measure my progress through the project. Each sprint was two weeks, beginning with a small planning session and ending in a review and retrospective. This should have ensured that all the work was completed by the respective deadlines.

Did this approach work? I would say that my agile approach worked, but I didn't follow the process that I planned to. I was supposed to have planning and review sessions, and this didn't always happen. Through the weeks I have been developing and researching game development I have been lacking when it came to planning what should be completed during each sprint. The result of this is that I have a lot of features that I wanted to include in the game, but my lack of planning has resulted in a backlog of features. In the future, I will need to be stricter with myself and set realistic deadlines for tasks to be completed.

5.2. UI evaluation

Main menu

First is the main menu needed for a game, NO it's not needed for any game, but it is expected to have one. The main menu screen gives the player a landing page. They can decide if they what to change any setting or they might need some help to understand what's going to happen when the game starts. If the game had started as soon as the game was launched, the user might be caught off guard.

Then have I created a good main menu? I think I have. I have created a simple main menu, where the users can start a game, or they can look at some help. The menu will also introduce the player to the game, by seeing some aspects of the game. The background shows the user the different tents available and gives the users an idea of the location of the game. One problem with this menu is that it gives the impression that it's going to be a 3D game. The player might be disappointed because the game is in 2D and not the immersive 3D that it gives the impression of.

Overall, I think this screen is very good and it gives the player what they need and it's offering help, but it's lacking some bits that would have improved it.

Help screen

I have created a very simple help screen, where the users will see a scrollable list where every row has an image and an explanation next to it. This screen gives the user some help on what each node is and how to use it, and mostly it does so in a very nicely structured way.

So, is this a good help screen? Well, it might be an okay way of doing it, but it's not ideal. The information is given to the users in a very clear way, and it has a nice structure to it. This might be good for some users, but it doesn't look nice, and it doesn't give enough information. I think a better way of doing this would have been to show the user how the game looks when you are playing the game. This could have been done with a screenshot of the game and speech balloons for each node explaining what it is and how to use it, and in addition to this, it could have had a list underneath the screenshot giving a more detailed explanation of how the game works and the rules of the game. This would have given the users a lot more information and would have made the learning curve very small.

Setup screen

Is a setup screen needed for a game? I would say yes. I think every game should have some sort of setup screen so the user can make some changes to the game or change the controls for the game. However, most game doesn't need to dedicate a separate screen to do this, most games could integrate it with the main menu or the start screen.

Have I created a good setup screen? I would say that yeah, it's a good setup screen and it gives the player what they need, but it doesn't look nice. I have created a simple list with different options. The user can change most of the aspects of the game and it's very easy to understand what each setting does, but it might be too simple, and it could do with some design improvements. Instead of having text on a plain background, the list could contain images of what would change in the game, and some symbols as well as the text.
For example, the speed setting could have had a symbol of a person running on the left and a label underneath saying speed, and the options could have a symbol of a tortoise, a person walking and a hare running with labels underneath saying the different speeds. This would have made each setting more understandable, and it would have made it easier for the user to see the changes.

Game screen

The user interface is arguably the most important part of any game. The user interface I have created for my game is very easy to understand for any person who has played any games before. However, people who haven't played any games before might need to look at the help page to understand how to use it.

The user interface of the game consist a joystick, button with a button label and a duration label at the top of the screen. When creating my user interface, I have had reachability in my mind, and I have only put interactable objects in areas where the users can reach them easily. (See Figure 26 Phone reachability in portrait and landscape)

Figure 26 Phone reachability in portrait and landscape [10]

I decided on creating a joystick for the player to use instead of moving after the player's finger. When the player has to point where the character should go, I have found out that the controls can be very difficult to use, so I chose to use a joystick to do this. I placed the joystick in the bottom left corner so that it would be easy to reach when playing the game, and because of its position it won't block the player's view.

I have created a context-dependent button. I chose to place this button in the bottom right corner so that it would be easily to reach for the player when they are playing the game, and since I only have one button, I have have a label underneath the button explaing the available actions, and if there are no actions available the label will be blank.
I chose to have context-dependent button instead of having multiple different buttons for different actions, so that the user won't be confused by the options available. If I had 2 or 3 different buttons, I would need to show which button to use at what point instead of only giving the user the ability to click on one button and

the option change deepening on what the character can reach. If I had multiple buttons, I would also need to be careful of the spacing between them and make sure that all of them where just as easy to reach.

5.3. Game mechanics

Overall, I'm quite happy with the game mechanics I have implemented and how the game works. The game works and it's playable, but there are some bugs in the game that I haven't been able to fix. There are some mechanics I haven't been able to implement and given more time I would implement bot movement for the hiders and implement a better field of view. I will go into more detail about each section.

Movement

There are two different movements happen in the game. Either the player makes a character move, or a bot is moving on its own. I'm very happy with how movement work when the player controls a character, but I haven't implemented the best bot movement.

The player's movement works very well, but there is one slight issue, the player can only move at full speed. This is something I could fix by checking how far the joystick has been moved and have that affect the speed of the player. However, I didn't manage to do this in time.

When it comes to bot movement there is a lot which needs to be improved. Firstly, the bots can only move if they are a seeker and not if they are a hider. This is because I didn't have time to implement movement for the hiders and focused on creating movement for the seekers so that the users could play as a hider. However, I would implement movement for the hiders if I had more time.

The seeker will select a target to catch and then go directly towards it. This work great when there is a clear path between the target and the seeker, but it can be a big problem if there is an object in the way. The seeker will completely ignore this and try to move through it, but because my collision detections work very well the seeker will jump around side to side until it's supposed to be on the other side of the object. I would improve this by using GameplayKit. GameplayKit is Apple's framework for bots and artificial intelligence, and I would implement this instead of creating my own.

Actions

The actions in the game work grate and they are reliable, but the user can only do one thing at a time. The user can walk, or they can do an action, not both. This is something I would want to improve by allowing the user to be able to do actions while they are walking. Implementing multiple touch inputs at once should be very easy, but I haven't had the time to investigate this.

Collisions

Collision detection is very important, and this makes sure that the player can't walk though object nor walk outside of the screen. I have not created collision detection for my game since I found this to be too difficult, but I have used the collision detection already available in SpriteKit, and this works perfectly. I don't think the collision detection could be improved in the game.

6. Annotated Bibliography

[1] raywenderlich.com, “Beginning SpriteKit,” youtube.com, 20 06 2017 . [Online]. Available: https://www.youtube.com/watch?v=wrEEIX2fUxE&list=PL23Revp-82LIKiBNSJaF311ixNvSzk8LC. [Accessed 01 2020].

A playlist of from YouTube that goes through the basics of game development for iOS with swift.

[2] C. Begbie, M. Berg, M. Briscoe, A. Hafizji, M. Todorov and R. Wenderlich, 2D Apple Games by Tutorials Second Edition, raywenderlich.com: Razeware LLC, 2017.

This is a book that contains tutorials where you will learn the skills need to develop games for Apple devices.

[3] InnerSloth LLC, “Among us,” 09 08 2019. [Online]. Available: https://apps.apple.com/us/app/among-us/id1351168404. [Accessed 11 2020].

This is an investigation game where you can play as a crewmate or an imposter, and you need to figure out who is the imposter before the imposter kills you.

[4] SUPERSONIC STUDIOS LTD, “Hide 'N Seek!,” 2020 06 24. [Online]. Available: https://apps.apple.com/gb/app/hide-n-seek/id1491654968. [Accessed 08 03 2021].

A hide and seek based game that I investigated to see what works and what doesn't work.

[5] W. Zhong, “Hide'N Find,” - - -. [Online]. Available: https://apps.apple.com/gb/app/hiden-find/id1509189851. [Accessed 08 03 2021].

A hide and seek based game that I investigated to see what works and what doesn't work.

[6] Y. Majeed, “Hide & Seek Fun Game,” - - -. [Online]. Available: https://apps.apple.com/gb/app/hide-seek-fun-game/id1495168019. [Accessed 08 03 2021].

A hide and seek based game that I investigated to see what works and what doesn't work.

[7] funcell Games Pvt Ltd, “Hide and Seek..!,” 30 10 2020. [Online]. Available: https://apps.apple.com/gb/app/hide-and-seek/id1537051480. [Accessed 08 03 2021].

A hide and seek based game that I investigated to see what works and what doesn't work.

[8] Apple Inc, “Responding to Frame-Cycle Events,” 2021. [Online]. Available: https://developer.apple.com/documentation/spritekit/skscene/responding_to_frame-cycle_events. [Accessed 24 04 2021].

An image showing the different stages in SpriteKit's life cycle

[9] TheDigitalArtist, “generic button,” pixabay, 29 04 2016. [Online]. Available: https://pixabay.com/illustrations/generic-button-button-3d-sign-1357003/ . [Accessed 03 2021].

This is a link to the website where I found a generic button.

[10] A. Dori, “Game Design UX Best Practices,” UX Planet, 16 04 2019. [Online]. Available: https://uxplanet.org/game-design-ux-best-practices-guide-4a3078c32099. [Accessed 29 04 2020].

This is an article talking about good UI design for games and it contains an image of a phone’s reachability.

7. Table of figures

8. Appendices

A. Third-Party Code and Libraries

I have used some third-Party code in my project. The code I have used is functions to handle simple mathematics in a more user-friendly way. The code and the copyright associated with the code I have used can be found bellow.


```
func + (left: CGPoint, right: CGPoint) -> CGPoint {
        return CGPoint(x: left.x + right.x, y: left.y + right.y)
}
func += (left: inout CGPoint, right: CGPoint) {
        left = left + right
}
func - (left: CGPoint, right: CGPoint) -> CGPoint {
        return CGPoint(x: left.x - right.x, y: left.y - right.y)
}
func -= (left: inout CGPoint, right: CGPoint) {
```

```
        left = left - right
}
func * (left: CGPoint, right: CGPoint) -> CGPoint {
        return CGPoint(x: left.x * right.x, y: left.y * right.y)
}
func *= (left: inout CGPoint, right: CGPoint) {
        left = left * right
}
func * (point: CGPoint, scalar: CGFloat) -> CGPoint {
        return CGPoint(x: point.x * scalar, y: point.y * scalar)
}
func *= (point: inout CGPoint, scalar: CGFloat) {
        point = point * scalar
}
func / (left: CGPoint, right: CGPoint) -> CGPoint {
        return CGPoint(x: left.x / right.x, y: left.y / right.y)
}
func /= ( left: inout CGPoint, right: CGPoint) {
        left = left / right
}
func / (point: CGPoint, scalar: CGFloat) -> CGPoint {
        return CGPoint(x: point.x / scalar, y: point.y / scalar)
}
func /= (point: inout CGPoint, scalar: CGFloat) {
        point = point / scalar
}
extension CGPoint {
        func length() -> CGFloat {
          return sqrt(x*x + y*y)
        }
        func normalized() -> CGPoint {
          return self / length()
        }
        var angle: CGFloat {
          return atan2(y, x)
        }
}
let π = CGFloat.pi
func shortestAngleBetween(angle1: CGFloat, angle2: CGFloat) -> CGFloat {
        let twoπ = π * 2.0
        var angle = (angle2 - angle1).truncatingRemainder(dividingBy: twoπ)
        if angle >= π {
          angle = angle - twoπ
        }
        if angle <= -π {
          angle = angle + twoπ
```

```
        }
        return angle
}
extension CGFloat {
        func sign() -> CGFloat {
          return self >= 0.0 ? 1.0 : -1.0
        }
}
extension CGFloat {
        static func random() -> CGFloat {
          return CGFloat(Float(arc4random()) / Float(UInt32.max))
        }
        static func random(min: CGFloat, max: CGFloat) -> CGFloat {
          assert(min < max)
          return CGFloat.random() * (max - min) + min
        }
}
```

B. Ethics Submission

Image not found or type unknown

11/03/2021

For your information, please find below a copy of your recently completed online ethics assessment

Next steps

Please refer to the email accompanying this attachment for details on the correct ethical approval route for this project. You should also review the content below for any ethical issues which have been flagged for your attention

Staff research - if you have completed this assessment for a grant application, you are not required to obtain approval until you have received confirmation that the grant has been awarded.

Please remember that collection must not commence until approval has been confirmed.

In case of any further queries, please visit www.aber.ac.uk/ethics or contact ethics@aber.ac.uk quoting reference number 19208.

Assesment Details

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
pev2@aber.ac.uk
Full Name
Petter Vang Brakalsvalet

Please enter the name of the person responsible for reviewing your assessment.
Neil Taylor

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment nst@aber.ac.uk
Supervisor or Institute Director of Research Department
cs

Module code (Only enter if you have been asked to do so) CS39440
Proposed Study Title
2D game construction in Swift

Proposed Start Date 25 January 2021
Proposed Completion Date 1 June 2021
Are you conducting a quantitative or qualitative research project? Mixed Methods
Does your research require external ethical approval under the Health Research Authority? No
Does your research involve animals? No

Does your research involve human participants? Yes
Are you completing this form for your own research? Yes
Does your research involve human participants? Yes
Institute
IMPACS

Please provide a brief summary of your project (150 word max)
My aim is to create a hide and seek style 2D game for iOS. This game will take place at night in a forest. The players will have multiple places to hide. However, some of the hiding places will have some form of punishment depending on how good of a hiding spot it is. For example, if a player hides in a tent the player would rustle the tent and make a noise, or if they are hiding in a tree they will get tired and eventually fall down. When a player takes the role of a seeker, they will have the option to use a flashlight to extend their field of view, and if another player is within reach the seeker can catch them. When a player is caught they will be frozen in place until another player frees them or the game finishes. When the game finishes they will be greeted with a new page where they will be able to see who was caught and who wasn't. I might ask other people to test my game and to provide some feedback.

I can confirm that the study does not involve vulnerable participants including participants under the age of 18, those with learning/communication or associated difficulties or those that are otherwise unable to provide informed consent? Yes
I can confirm that the participants will not be asked to take part in the study without their consent or knowledge at the time and participants will be fully informed of the purpose of the research (including what data will be gathered and how it shall be used during and after the study). Participants will also be given time to consider whether they wish to take part in the study and be given the right to withdraw at any given time. Yes
I can confirm that there is no risk that the nature of the research topic might lead to disclosures from the participant concerning their own involvement in illegal activities or other activities that represent a risk to themselves or others (e.g. sexual activity, drug use or professional misconduct). Yes

I can confirm that the study will not induce stress, anxiety, lead to humiliation or cause harm or any other negative consequences beyond the risks encountered in the participant's day-to-day lives. Yes
Please include any further relevant information for this section here:
Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material? Not applicable
Will appropriate measures be put in place for the secure and confidential storage of data? Yes
Does the research pose more than minimal and predictable risk to the researcher? No
Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to? No
Please include any further relevant information for this section here:
Is your research study related to
COVID-19? No
If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one. Yes
Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change. Yes
Please include any further relevant information for this section here:

C. Code Snippet

Here is my automated test for my game:

8.1.1. MainMenu

```
class MainMenuTests: XCTestCase {
    var mainMenu: MainMenu!

    override func setUpWithError() throws {
        // Put setup code here. This method is called before the invocation of each test method in the
class.
        mainMenu = MainMenu(fileNamed: "MainMenuScene")
    }

    override func tearDownWithError() throws {
        // Put teardown code here. This method is called after the invocation of each test method in the
class.
    }

    func testPlayButton(){
        // This tests if the play button has been found in the scene.
        let playButton = mainMenu.playButton
        XCTAssert(playButton != nil, "Play button was not found")
    }

    func testHelpButton(){
        // This tests if the help button has been found in the scene.
        let helpButton = mainMenu.helpButton
        XCTAssert(helpButton != nil, "Help button was not found")
    }

    func testNumberOfChildrenInScene() {
        // This tests if the scene has the expected amount of children.
        let children = mainMenu.children
        XCTAssert(children.count == 2, "Number of children in main menu is not what's expected")
    }
}
```

8.1.2. Help

```
class HelpTests: XCTestCase {
    var helpVC: HelpVC!
```

```
    var defaultHelpItems = [HelpVC.HelpItem(image: "Hider", explenation: "This is a hider. Your job as a hider is to hide from the seeker."),
                            HelpVC.HelpItem(image: "Seeker", explenation: "This is a seeker. Your job as a seeker is to catch all other players."),
                            HelpVC.HelpItem(image: "tentNew", explenation: "This is an hiding spot. You can hide here."),
                            HelpVC.HelpItem(image: "tentOld", explenation: "This is an hiding spot. You can hide here."),
                            HelpVC.HelpItem(image: "house", explenation: "This is an hiding spot. You can hide here."),
                            HelpVC.HelpItem(image: "joystick_background",secondImage: "joystick" , explenation: "This is an joystick. You can use this to move around on the map."),
                            HelpVC.HelpItem(image: "button", explenation: "This is the action button. I will have a label stating the action yo can preform.")]

    override func setUpWithError() throws {
        // Put setup code here. This method is called before the invocation of each test method in the class.
        helpVC = HelpVC()
    }

    override func tearDownWithError() throws {
        // Put teardown code here. This method is called after the invocation of each test method in the class.
    }

    func testNumberOfHelpItems() {
        XCTAssertEqual(helpVC.helpItems.count, defaultHelpItems.count, "The number of help items is not as expected.")
    }

    func testHiderHelp() {
        if helpVC.helpItems[0].image == defaultHelpItems[0].image &&
            helpVC.helpItems[0].explenation == defaultHelpItems[0].explenation{
            XCTAssert(true)
        } else {
            XCTAssert(false,  "Hiders item is not as expected.")
        }
    }

    func testSeekerHelp() {
        if helpVC.helpItems[1].image == defaultHelpItems[1].image &&
            helpVC.helpItems[1].explenation == defaultHelpItems[1].explenation{
            XCTAssert(true)
        } else {
            XCTAssert(false,  "Seeker item is not as expected.")
```

```
    }
}

func testTentNewHelp() {
    if helpVC.helpItems[2].image == defaultHelpItems[2].image &&
        helpVC.helpItems[2].explenation == defaultHelpItems[2].explenation{
        XCTAssert(true)
    } else {
        XCTAssert(false, "TentNew item is not as expected.")
    }
}

func testTentOldHelp() {
    if helpVC.helpItems[3].image == defaultHelpItems[3].image &&
        helpVC.helpItems[3].explenation == defaultHelpItems[3].explenation{
        XCTAssert(true)
    } else {
        XCTAssert(false, "TentOld item is not as expected.")
    }
}

func testHouseHelp() {
    if helpVC.helpItems[4].image == defaultHelpItems[4].image &&
        helpVC.helpItems[4].explenation == defaultHelpItems[4].explenation{
        XCTAssert(true)
    } else {
        XCTAssert(false, "House item is not as expected.")
    }
}

func testJoystickHelp() {
    if helpVC.helpItems[5].image == defaultHelpItems[5].image &&
        helpVC.helpItems[5].secondImage == defaultHelpItems[5].secondImage &&
        helpVC.helpItems[5].explenation == defaultHelpItems[5].explenation{
        XCTAssert(true)
    } else {
        XCTAssert(false, "Joystick item is not as expected.")
    }
}

func testButtonHelp() {
    if helpVC.helpItems[6].image == defaultHelpItems[6].image &&
        helpVC.helpItems[6].explenation == defaultHelpItems[6].explenation{
        XCTAssert(true)
    } else {
        XCTAssert(false, "Button item is not as expected.")
```

```
    }
  }
}
```

8.1.3. SetUp

```
class SetUpTests: XCTestCase {
  private struct GameOptions {
    var title: String
    var hasSegments: Bool
    var options: [String]?
  }
  var setUpVC: SetUpVC!
  private var defaultOptions = [
    GameOptions(title: "Role", hasSegments: true, options: [Player.Role.hider.rawValue,
Player.Role.seeker.rawValue]),
    GameOptions(title: "Difficulty", hasSegments: true,
          options: [ChallangeRating.easy.rawValue,
              ChallangeRating.normal.rawValue,
              ChallangeRating.hard.rawValue]),
    GameOptions(title: "Speed", hasSegments: true, options: ["Slow", "Normal", "Fast"]),
    GameOptions(title: "Reach", hasSegments: true, options: ["Short", "Medium", "Far"]),
    GameOptions(title: "Duration", hasSegments: false),
    GameOptions(title: "Bots", hasSegments: false)]

  override func setUpWithError() throws {
    // Put setup code here. This method is called before the invocation of each test method in the
class.
    setUpVC = SetUpVC()
  }

  override func tearDownWithError() throws {
    // Put teardown code here. This method is called after the invocation of each test method in the
class.
  }

  func testNumberOfSetUpOptions() {
    XCTAssert(setUpVC.gameOptions.count == defaultOptions.count, "Number of game options is
not what's expected")
  }

  // Test position of game options
  func testPositionOfRoleOption(){
    let gameOptions = setUpVC.gameOptions
    let index = gameOptions.firstIndex(where: {$0.title == defaultOptions[0].title})
    XCTAssertEqual(index, 0, "Role option is not at expected index")
```

```
    }

    func testPositionOfDifficultyOption(){
        let gameOptions = setUpVC.gameOptions
        let index = gameOptions.firstIndex(where: {$0.title == defaultOptions[1].title})
        XCTAssertEqual(index, 1, "Difficulty option is not at expected index")
    }

    func testPositionOfSpeedOption(){
        let gameOptions = setUpVC.gameOptions
        let index = gameOptions.firstIndex(where: {$0.title == defaultOptions[2].title})
        XCTAssertEqual(index, 2, "Speed option is not at expected index")
    }

    func testPositionOfReachOption(){
        let gameOptions = setUpVC.gameOptions
        let index = gameOptions.firstIndex(where: {$0.title == defaultOptions[3].title})
        XCTAssertEqual(index, 3, "Reach option is not at expected index")
    }

    func testPositionOfDurationOption(){
        let gameOptions = setUpVC.gameOptions
        let index = gameOptions.firstIndex(where: {$0.title == defaultOptions[4].title})
        XCTAssertEqual(index, 4, "Duration option is not at expected index")
    }

    func testPositionOfBotsOption(){
        let gameOptions = setUpVC.gameOptions
        let index = gameOptions.firstIndex(where: {$0.title == defaultOptions[5].title})
        XCTAssertEqual(index, 5, "Bots option is not at expected index")
    }

    // Test number of segments of game options
    func testNumberOfSegmentsOfRoleOption(){
        let gameOptions = setUpVC.gameOptions
        XCTAssertEqual(gameOptions[0].segments, defaultOptions[0].options, "Number of segments of
role option is not as expected")
    }

    func testNumberOfSegmentsOfDifficultyOption(){
        let gameOptions = setUpVC.gameOptions
        XCTAssertEqual(gameOptions[1].segments, defaultOptions[1].options, "Number of segments of
difficulty option is not as expected")
    }

    func testNumberOfSegmentsOfSpeedOption(){
```

```
        let gameOptions = setUpVC.gameOptions
        XCTAssertEqual(gameOptions[2].segments, defaultOptions[2].options, "Number of segments of speed option is not as expected")
    }

    func testNumberOfSegmentsOfReachOption(){
        let gameOptions = setUpVC.gameOptions
        XCTAssertEqual(gameOptions[3].segments, defaultOptions[3].options, "Number of segments of reach option is not as expected")
    }

    func testNumberOfSegmentsOfDurationOption(){
        let gameOptions = setUpVC.gameOptions
        XCTAssertNil(gameOptions[4].segments, "Number of segments of duration option is not nil")
    }

    func testNumberOfSegmentsOfBotsOption(){
        let gameOptions = setUpVC.gameOptions
        XCTAssertNil(gameOptions[5].segments, "Number of segments of bots option is not nil")
    }

    func testRoleOption(){
        let gameOption = setUpVC.gameOptions[0]
        XCTAssert(
            gameOption.segments?[0] == defaultOptions[0].options?[0] &&
                gameOption.segments?[1] == defaultOptions[0].options?[1], "The role options isn't matching the expected options.")
    }

    func testDifficultyOption(){
        let gameOption = setUpVC.gameOptions[1]
        XCTAssert(
            gameOption.segments?[0] == defaultOptions[1].options?[0] &&
                gameOption.segments?[1] == defaultOptions[1].options?[1] &&
                gameOption.segments?[2] == defaultOptions[1].options?[2], "The difficulty options isn't matching the expected options.")
    }

    func testSpeedOption(){
        let gameOption = setUpVC.gameOptions[2]
        XCTAssert(
            gameOption.segments?[0] == defaultOptions[2].options?[0] &&
                gameOption.segments?[1] == defaultOptions[2].options?[1] &&
                gameOption.segments?[2] == defaultOptions[2].options?[2], "The speed options isn't matching the expected options.")
    }
```

```
    func testReachOption(){
        let gameOption = setUpVC.gameOptions[3]
        XCTAssert(
            gameOption.segments?[0] == defaultOptions[3].options?[0] &&
                gameOption.segments?[1] == defaultOptions[3].options?[1] &&
                gameOption.segments?[2] == defaultOptions[3].options?[2], "The reach options isn't
matching the expected options.")
    }
}
```