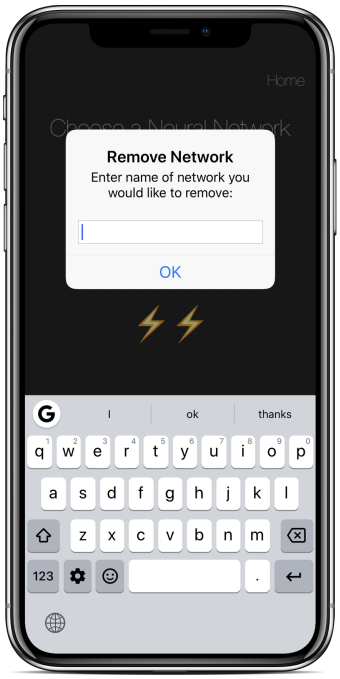
Criterion B: Design

Prototype

Main Menu Screen

Train Network Game Scene

Choose Network Scene



Remove A Network Scene

Design of Solution Overview

User Interface

- Each view is handled by a individual "Scene.swift" file
- User input views are handled by UIAlertController, they are presented using the helper file "Alertable.swift"
- All scenes are handled by central "GameViewController.swift" file
- Each object on screen is put on the screen as an SKNode

Game Design

- There are four objects in the game with the following properties:

1. **Main Character**

- Properties: Character_ WIDTH (CGFloat), Character_ HEIGHT (CGFloat), Character_Position (CGPosition), hasCollided (Bool), hasJumped (Bool)
- Actions: Can jump into the air, if character collides with enemy game ends

2. **Enemies**

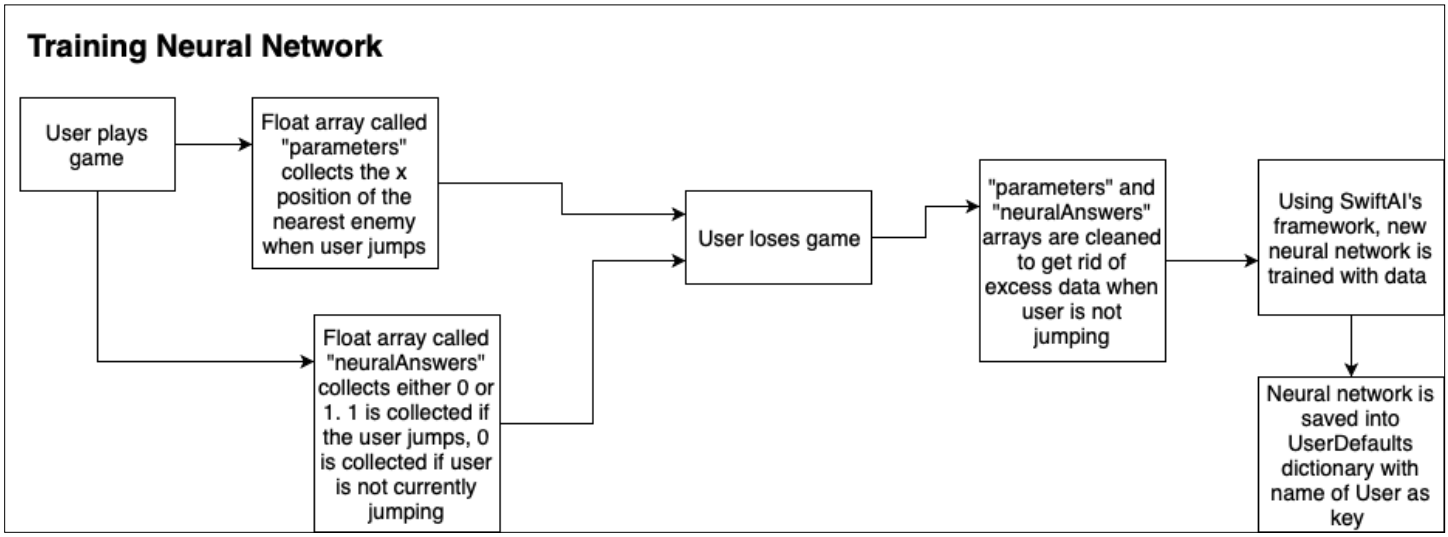
- Properties: Enemy_ WIDTH (CGFloat), Enemy_ HEIGHT (CGFloat), Enemy_Position (CGPosition)
- Actions: Randomly spawns on the floor and moves across floor

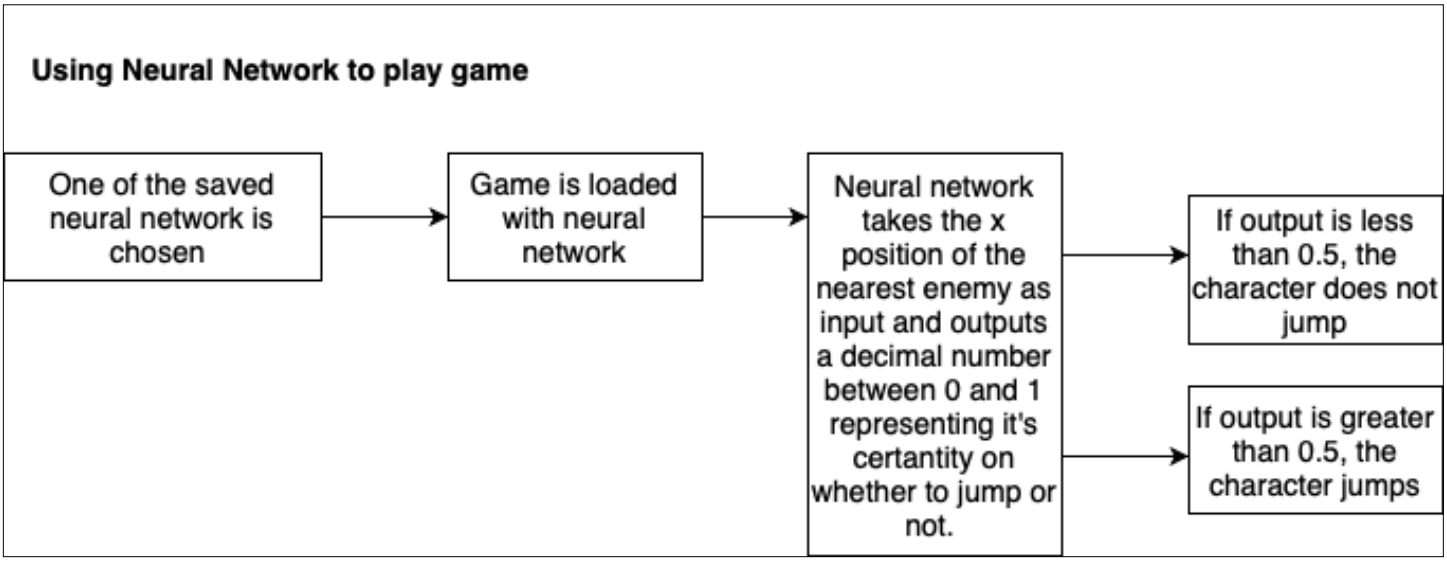
3. **The Floor**

- Properties: Floor_ WIDTH (CGFloat), Floor_ HEIGHT (CGFloat), Floor_Speed (CGFloat)
- Actions: Spawns on bottom of screen and moves across screen at Floor_Speed

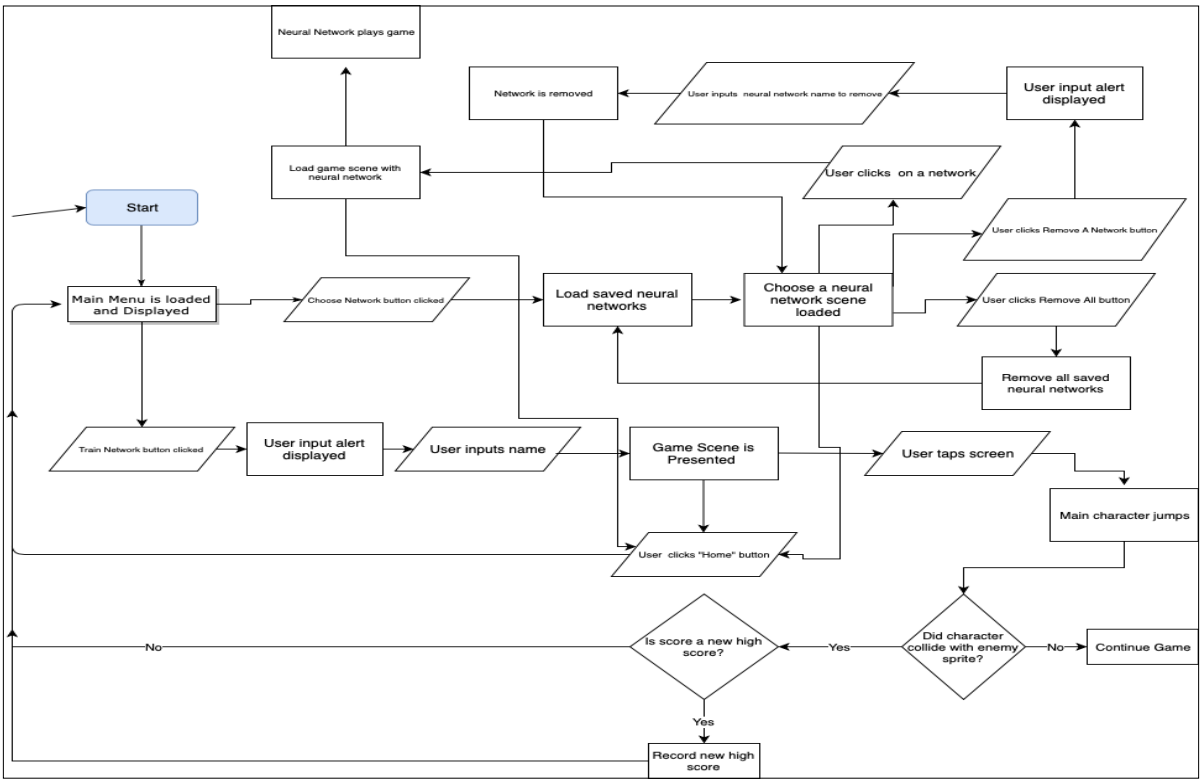
4. **Clouds**

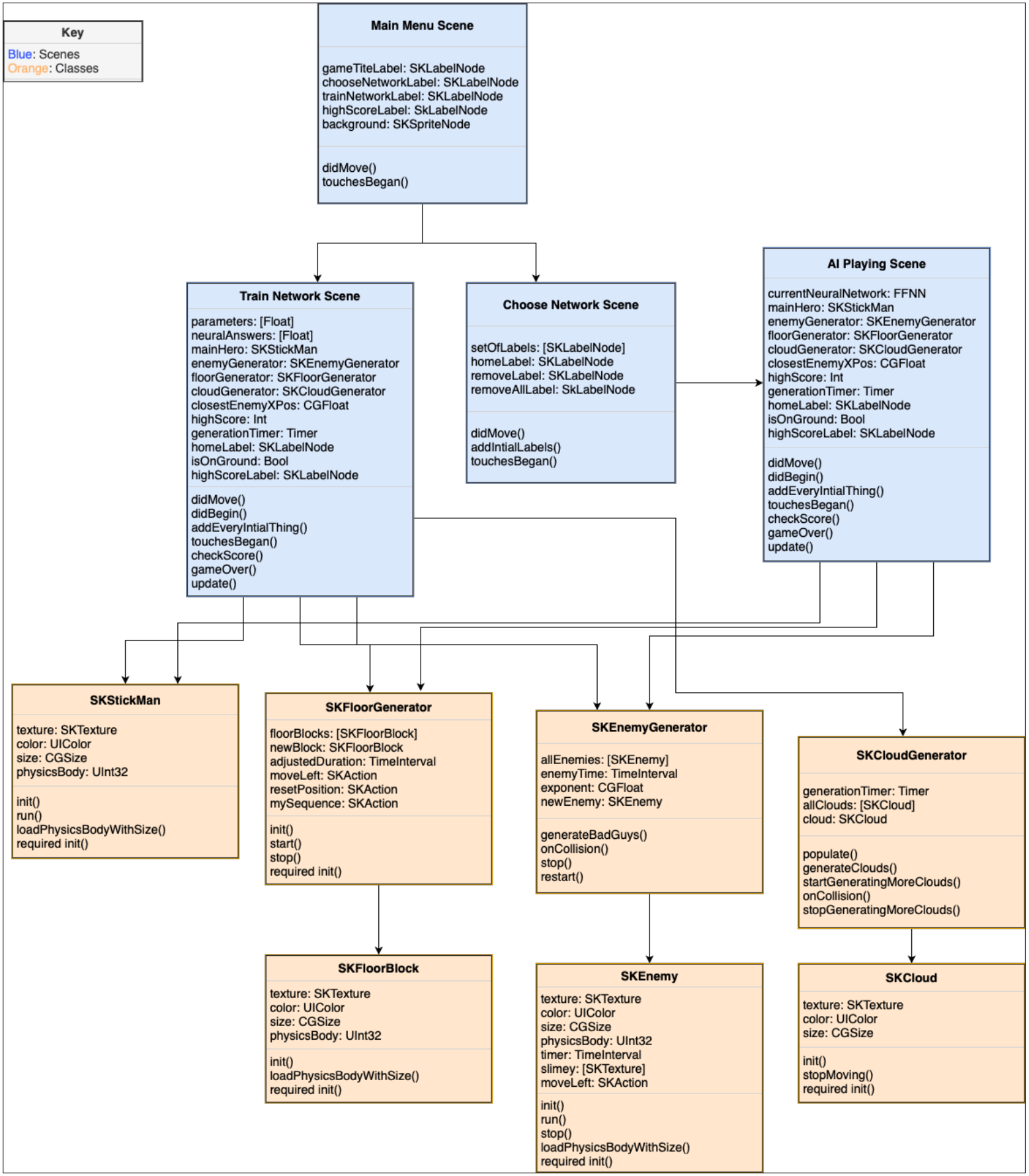
- Properties: Cloud_WIDTH (CGFloat), CLOUD_HEIGHT (CGFloat), CLOUD_POSITION (CGPosition), IsOnScreen (Bool)
- Actions: Crosses the screen, has no user interaction, is taken off the screen once it finishes crossing the screen





User Input Flowchart




Key
Blue: Scenes
Orange: Classes
Main Menu Scene
gameTiteLabel: SKLabelNode
chooseNetworkLabel: SKLabelNode
trainNetworkLabel: SKLabelNode
highScoreLabel: SkLabelNode
background: SKSpriteNode
didMove()
touchesBegan()
Train Network Scene
parameters: [Float]
neuralAnswers: [Float]
mainHero: SKStickMan
enemyGenerator: SKEnemyGenerator
floorGenerator: SKFloorGenerator
cloudGenerator: SKCloudGenerator
closestEnemyXPos: CGFloat
highScore: Int
generationTimer: Timer
homeLabel: SKLabelNode
isOnGround: Bool
highScoreLabel: SKLabelNode
didMove()
didBegin()
addEveryIntialThing()
touchesBegan()
checkScore()
gameOver()
update()
Choose Network Scene
setOfLabels: [SKLabelNode]
homeLabel: SKLabelNode
removeLabel: SKLabelNode
removeAllLabel: SkLabelNode
didMove()
addIntialLabels()
touchesBegan()
AI Playing Scene
currentNeuralNetwork: FFNN
mainHero: SKStickMan
enemyGenerator: SKEnemyGenerator
floorGenerator: SKFloorGenerator
cloudGenerator: SKCloudGenerator
closestEnemyXPos: CGFloat
highScore: Int
generationTimer: Timer
homeLabel: SKLabelNode
isOnGround: Bool
highScoreLabel: SKLabelNode
didMove()
didBegin()
addEveryIntialThing()
touchesBegan()
checkScore()
gameOver()
update()
SKStickMan
texture: SKTexture
color: UIColor
size: CGSize
physicsBody: UInt32
init()
run()
loadPhysicsBodyWithSize()
required init()
SKFloorGenerator
floorBlocks: [SKFloorBlock]
newBlock: SKFloorBlock
adjustedDuration: TimeInterval
moveLeft: SKAction
resetPosition: SKAction
mySequence: SKAction
init()
start()
stop()
required init()
SKEnemyGenerator
allEnemies: [SKEnemy]
enemyTime: TimeInterval
exponent: CGFloat
newEnemy: SKEnemy
generateBadGuys()
onCollision()
stop()
restart()
SKCloudGenerator
generationTimer: Timer
allClouds: [SKCloud]
cloud: SKCloud
populate()
generateClouds()
startGeneratingMoreClouds()
onCollision()
stopGeneratingMoreClouds()
SKFloorBlock
texture: SKTexture
color: UIColor
size: CGSize
physicsBody: UInt32
init()
loadPhysicsBodyWithSize()
required init()
SKEnemy
texture: SKTexture
color: UIColor
size: CGSize
physicsBody: UInt32
timer: TimeInterval
slimey: [SKTexture]
moveLeft: SKAction
init()
run()
stop()
loadPhysicsBodyWithSize()
required init()
SKCloud
texture: SKTexture
color: UIColor
size: CGSize
init()
stopMoving()
required init()

Test Plan

Test Type	Nature of Test	Example
Upon opening application, program should open up to main menu	Check that main menu is correctly displayed	App opens smoothly, launch screen is displayed, main menu is displayed with labels and buttons
All application buttons should successfully display intended scene and should pass data properly	Check that buttons are properly wired up	User clicks on "Choose Network" button, choose network scene is shown and saved networks are displayed
Game should allow the proper training of a neural network by the user	Game has implemented the proper system to save neural network	User clicks on "Train Network" button, alert to input name is shown, game saves the name of user, allows user to train network and saves neural network in UserDefaults
Game sprites should be properly animated	Enemy and main character sprites are animated and look like they are moving	Main character appears to be running when game is running
Application should allow users to view and choose from the saved neural networks	Game is properly saving neural networks and allows users to test the available networks	User opens "Choose Network" scene, is shown available networks to choose from and is able to select one and see it run
Neural network architecture is properly working	Neural network should properly mimic the circumstances it was trained under	A user trains a neural network by playing game, then when neural network is chosen to run, it knows how to play and mimics the style of the training
Users should be able to remove individual and all of the networks	Neural networks are saved into UserDefaults properly and able mutable, in the sense that they can be removed if desired	User clicks "Remove A Network" button in "Choose Network" scene, is shown a popup with the option to type name of network they want to remove.
Game scoring system works properly	If character jumps over enemy sprite, game detects that another point should be added	User is playing game to train network, clicks on screen to jump, jumps over enemy and is rewarded a point
Ground and cloud sprites are efficiently moved across the screen	The ground and cloud sprites move across screen. When they reach the end of the screen they are removed and replaced with new sprites at the other end of the screen in order to preserve memory.	A ground sprite reaches the end of the screen, that individual sprite is removed and a new sprite that starts at the beginning of the screen is added.

Word Count: 169