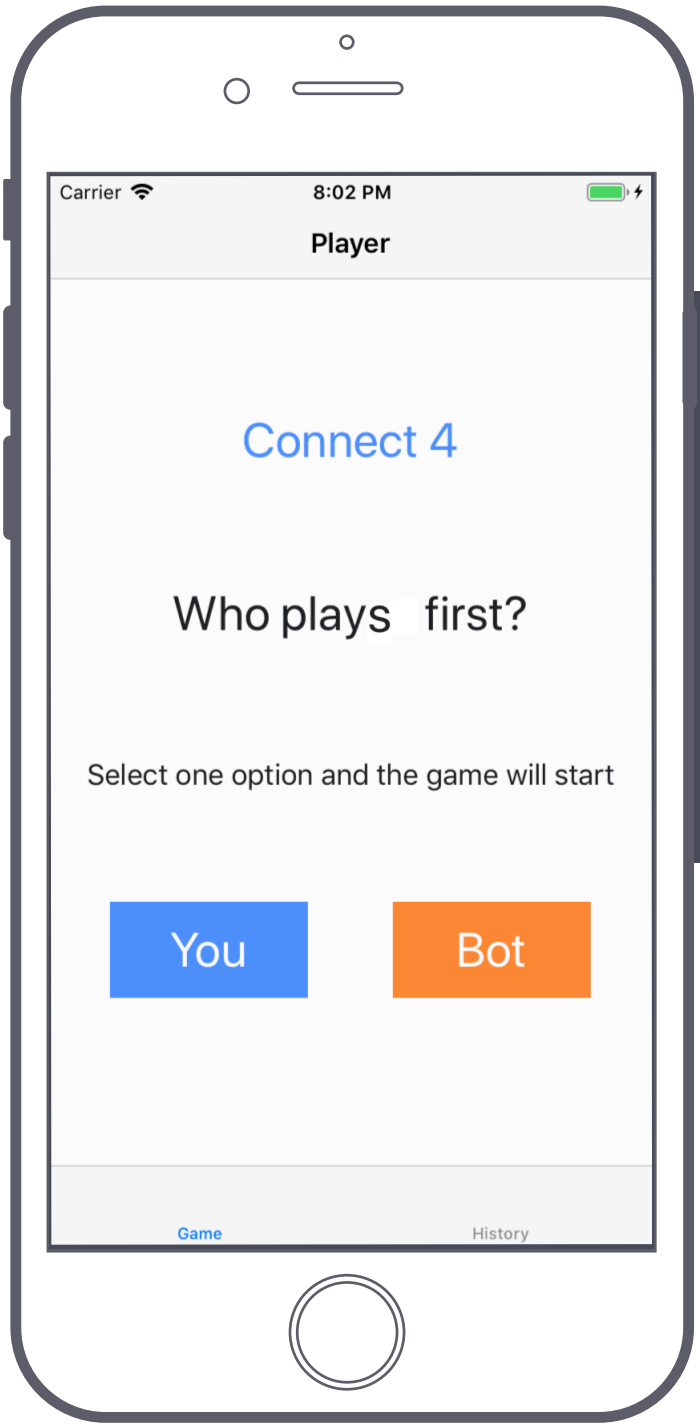
IOS: Connect4

2019

Student Number: 18204188,
Student Email: gregoire.cousin@ucdconnect.ie



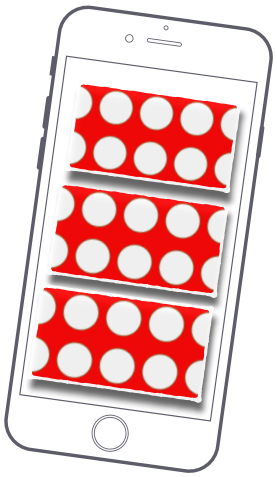
Connect4

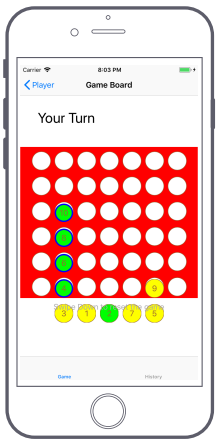
A digital version of the puzzle game: Connect4 with a deep learning bot

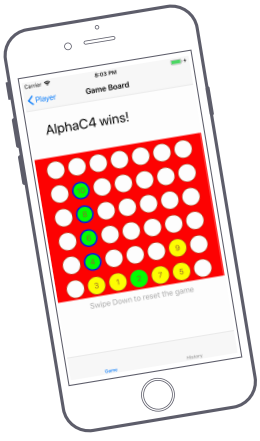
Table of Contents

Connect4 Introduction

The objective of this assignment was to design a responsive, dynamic user interface for the game Connect4/4-in-a-Row to facilitate users to play against a deep learning bot. The nature of the game itself is a two player connection game which allows users to pick a color: yellow or red, and drop their own colored disc in a seven-column, six-row suspended grid at each turn. The pieces fall straight down occupying the lowest available space within the column. The winner of the game is the first to form a vertical, horizontal or diagonal line of four of one's own discs. The design features mentioned below were decided upon after careful consideration of the architecture of the program.







Technology Used

Connect4 is for iOS and created with Swift 4 and Xcode

Design

The base design of the application is applied in the storyboard of *Xcode* while the rest of the game board is programmed throughout separate files.

The Main storyboard shows two *main screens* which are: *User Select* and the *History screen*. The Main *storyboard* consists of a *Tabbar controller* which show the two screens.

App Design

UserSelect ViewController

User Select is essentially a UIViewController embedded in a UINavigationController. The screen allows the user to select who is going to play the game first, either the user or the bot. User Select has two buttons and based on what the user chooses to click, a segue will direct the user to the Game screen.

Game ViewController

The *Game screen* is the heart of the application where the user plays the game. The screen consists of a *game board* which is designed programmatically and allows the user to play the game with the deep learning bot.

This screen also shows has the functionality of the replay feature from the users previous games.

History TableViewController

The *History screen* is made with a *TableViewController* that shows the history of each games played. This screen is embedded in a *UINavigationController* and has one cell. Clicking on the cell will perform a *show* segue to the Game screen where the replay of the game is played.

Component

An iOS application consists of different components which work together to make it work. Connect4 components are as below

- *UITabbarController*
- *UINavigationController*
- *UITableViewController*
- *UITableViewCell*

UITabbarController

UITabbarController allows an application to have tabs at the bottom of the display. It embeds up-to **5** *ViewControllers* and a user can easily switch between them. This class is generally used as-is, but may also be subclassed.

In Connect4, there are two tabs which are the *UserSelect screen* and the *History screen.*

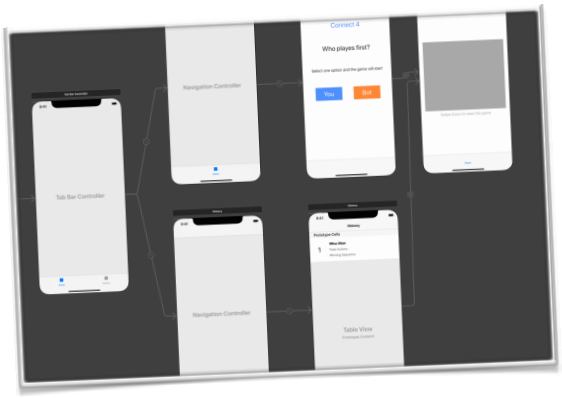
UINavigationController

A navigation controller is a container view controller that manages one or more child view controllers in a navigation interface. Connect4 has two *UINavigationController* which can show the *UserSelect screen*, *Game screen and History screen*. This class is generally used without customization, allowing basic functionality and can be subclassed. In the game, whenever a user moves from one screen to another, the *UINavigationController* is used to perform this functionality. ?

UITableViewController

A table view displays a list of items in a single column. The *UITableViewController* is a subclass of the *UIViewController* which allows the users to scroll through the content.

The *History screen* uses the *UITableViewController* and shows "*search history"* performed by the user in a cell. These are classes of *UITableViewCell*. Clicking on a cell allows the user to segue to the *Game screen. This screen* acts as a *Replay* showing an animated replay of the game. The game data is sent from the *History screen* to the *Game screen* through a "*prepareForSegue*" method. This method allows the sending of information across the two view controllers.



UITableViewCell

This component includes properties and methods for setting and managing **cell text**, **images**, and **custom views**.

Connect4's History screen uses the *UITableViewCells* to show the information of a games history. In order to do that, subclassing the *UITableViewCells* to the HistoryCell is necessary. In the *cellForRow* method of *UITableViewController*, the data is assigned to the variable in the cell.

Code

This section will explain the key-code components used to develop the application.

UserSelectVC

the *UserSelect* screen has a class name of *UserSelectVC*. This screens sends a *C4GameUserType* enum object to the *GameVC* in the *prepareForSegue* method.

GameVC

The Game screen has a class named GameVC. It has three main objects that perform the entire functionality to play the game.The first is gameboard which is of type Gameboard creates the actual gameboard, the discs and the appropriate behavior allowing the animation of the discs after the game is reset. When the game is started, the gameboard is created by calling the method createOverlay.

```
    for i in 1...6{
      if(i == 1){
          yOffset += 10
      }else{

          yOffset += frame.height / CGFloat(6.0)
      }
    xOffset  = 0
      for j in 1...7{
            if(j == 1){
                xOffset += 25
            }else{
            xOffset += (frame.width - 40) / 7.0
            }
                        let diskFrame = CGRect(x: 0, y: 0, width: diskSize, height: diskSize)
                        let diskPath = UIBezierPath(ovalIn: diskFrame)
                        let diskShape = CAShapeLayer()

                        diskShape.path = diskPath.cgPath
                        diskShape.fillColor = UIColor.white.cgColor
                        diskShape.strokeColor = UIColor.brown.cgColor
                        diskShape.lineWidth = 1.0
                        name += 1

                        let diskView = UIView(frame: CGRect(x: xOffset, y: yOffset, width: diskSize, height:
                        diskView.tag = name
                        diskView.layer.addSublayer(diskShape)
                        boardView.addSubview(diskView)
            }
            boardView.backgroundColor = UIColor.red
        self.view.addSubview(boardView)
}
```

The following screenshot shows the relevant code for it. When the user and bot plays the game, the discs fall in the spaces on the gameboard by calling the method createdisc.

```
let diskFrame = CGRect(x: 0, y: 0, width: diskSize, height: diskSize)
let diskPath = UIBezierPath(ovalIn: diskFrame)

let diskShape = CAShapeLayer()
diskShape.path = diskPath.cgPath
diskShape.lineWidth = 1.0
if(userType == .user){
      diskShape.fillColor = UIColor.yellow.cgColor
      diskShape.strokeColor = UIColor.brown.cgColor
}else{
      diskShape.fillColor = UIColor.green.cgColor
      diskShape.strokeColor = UIColor.brown.cgColor
}
```

```
diskShape.name = String(tag)

let diskView = UIView(frame: CGRect(x: frame.origin.x, y: -200, width: diskSize, height:
               diskSize))
```

```
diskView.clipsToBounds = true
diskView.tag = diskName

diskView.layer.addSublayer(diskShape)

let diskLabel  = UILabel(frame: CGRect(x: 0, y: 0, width: diskSize, height: diskSize))
diskLabel.text = String(index)
diskLabel.textAlignment = .center
diskLabel.textColor = UIColor.brown
```

```
diskView.addSubview(diskLabel)

boardView.addSubview(diskView)

gameDisks.append(diskView)

let position = CGPoint(x: frame!.midX , y: frame!.midY)
diskBehavior = DiscBehaviour(disk: diskView, reference: self.view)

let snap = UISnapBehavior(item: diskView, snapTo: position)
diskBehavior.addChildBehavior(snap)

self.diskBehavior.animator.delegate = self;
```

The second object is *gameSession* of type *GameSession.* The Object is responsible for the actual game and all the decision making within the game. The object incorporates the functionality coming from the deep learning bot to let it play with the user. It is intentionally difficult for the user to defeat the learning bot.

The code allowing to play the game

```
if let move = gameSession.move {
      DispatchQueue.main.async {
         self.gameboard.createdisk(tag: move.action, index: move.index)
     }
}
```

After each move the status is checked to see if the game has been completed

```
if let outcome = gameSession.outcome {}
```

The third object is used to store the games session information in the core data. This object is called *c4GameSession* and is of type *C4GameSession*. *c4GameSession* stores the sequence of the game in which each game is played, the winning sequence as well as who won that game. After the game has been completed, if the user resets the game, it will drop all discs on the *gameboard* using the *Discbehaviour* class which is a subclass of *UIDynamicbehaviour*. This class is responsible for adding the *animations and gravity* features for each of the disc drop.

Below is relevant code for dropping discs so the game can than reset.

```
animator = UIDynamicAnimator(referenceView: reference)
let gravity = behavior as! UIGravityBehavior
let direction = CGVector(dx: 0.0, dy: 1.0)
gravity.gravityDirection = direction
gravity.magnitude = 1.0
gravity.action = {
    if(self.disk.frame.origin.y > 2000){
       self.animator.removeBehavior(gravity)
    }
}
animator.addBehavior(gravity)
```

HistoryTVC

The History screen has the class name “HistoryVC”. It’s fundamental purpose is to show the history of all previous games played by the user. A method getAllSessions in C4GameSession retrieves the history from the core data every time the user visits this screen and stores it in a sessions array.

The order of historyArray is reversed so that the user can see their latest games played at the top of the screen. Clicking on a cell will allow “prepareForSegue” to send the specific game session object to the GameVC and replay the game in an animated form.

Conclusion

AI > Connect4

The main objective of this project was to design an AI Connect4 game centered around a user interface where the user races against a deep learning bot in forming a horizontal, diagonal or vertical line of four discs. Throughout the process of conceptualization, design and programing of the game, I developed a deeper understanding of the animation of different sections of an application with Xcode, as well as components I had never used before.

The hardest achievement for this application has been the attempt to put into motion how I conceptualized the different functions of the application at the start of the project. Taking small steps and organizing certain behaviors of the game in separate files, sections, and objects has helped me tremendously.

I am very proud of the creation of the game board, discovering and applying animation, physics, and altering gravity to fit the needs of the games objective.

In conclusion, this project has been a very important step in learning how to deal with the complexity of MVCs, display and screen transitions, manipulating table views, and using proper techniques while ‘animating’.