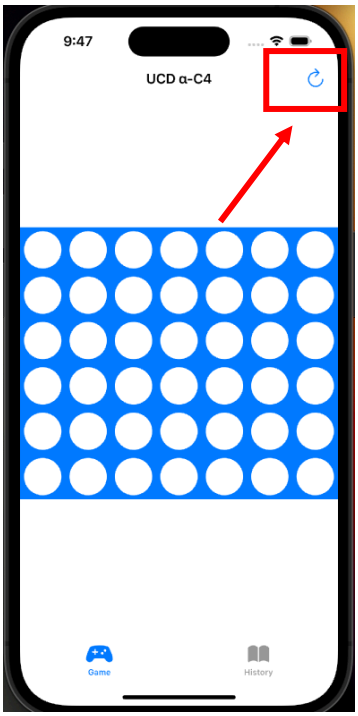
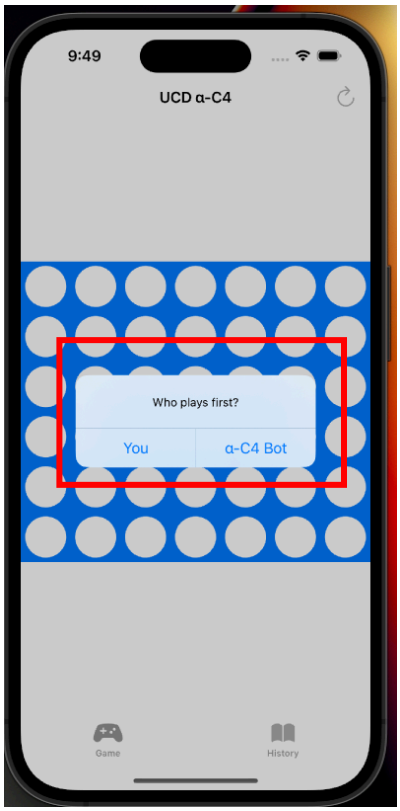
Game running：

1. Click the **Refresh button** （ ↻ ） in the top right corner to start the game

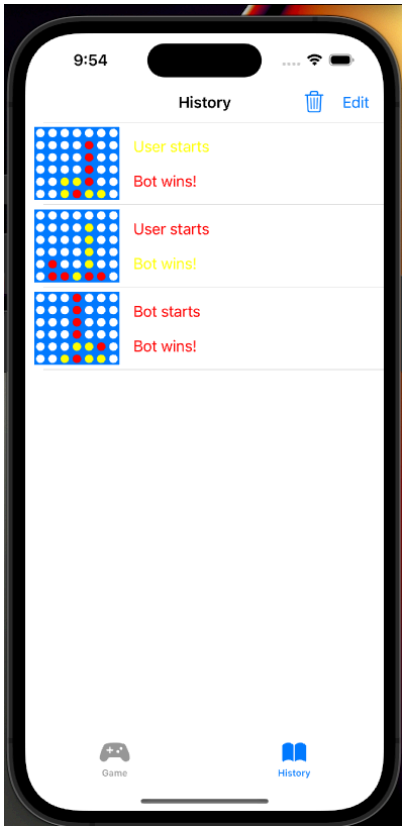


2. Choose who will start the game first



3. End of the game

4. Historical records



Code explanation：

Set the board pattern with seven holes per row

```
func setupUI()  {
    //The entire upper play area
    self.view.addSubview(self.topView)
    topView.frame = CGRect(x: 0, y: 0, width: KscreenW, height: KscreenH * 0.7)
    for i in 0..<7 {
        //Seven columns in total
        let discsColumnView = DiscsColumnView(frame: CGRect(x: discsWH *
            Double(i), y: 0, width: discsWH, height: topView.frame.height))
        //discsColumnView.backgroundColor = UIColor.randomColor
        discsColumnViewList.append(discsColumnView)
        topView.addSubview(discsColumnView);
    }

    //Setting up the game board
    let gameBoardH = KscreenW * 6/7.0
    self.topView.addSubview(self.gameBoard)
    gameBoard.frame = CGRect(x: 0, y: self.topView.frame.size.height -
        gameBoardH, width: KscreenW, height: gameBoardH)

    resultLab.text = ""


}
```

Set the board to skeleton

```
//MARK: Setting up the game board skeleton
func setGameBoard() {
    //7 per row
    let maskPath = UIBezierPath(rect: self.gameBoard.bounds)
    //Create 42 holes
    for i in 0..<42 {
        let rect = CGRect(x: 5 + discsWH * Double((i%7)), y: 5 + discsWH *
            Double(i/7), width: holeWH, height: holeWH)

        let holePath = UIBezierPath(roundedRect: rect,cornerRadius: discsWH/2.0)
        maskPath.append(holePath)
    }
    let mask = CAShapeLayer()
    mask.fillRule = .evenOdd
    mask.path = maskPath.cgPath
    self.gameBoard.layer.mask = mask
}
```

Start game session with random bot parameter

```
// Start game session with random bot parameter
private func newGameSession() {
    // Print game layout
    print("CONNECT4 \(gameSession.boardLayout.rows) rows by
        \(gameSession.boardLayout.columns) columns")

    // Start game, resuming with some discs
    // set initialMoves to [(Int, Int)]() to start with clear board
    //let initialMoves = [(row: 1, column: 4), (row: 2, column: 4)]

    //Start the game
    self.gameSession.startGame(delegate: self, botPlays: self.botColor, first:
        self.isBotFirst,initialPositions: [(Int, Int)]())
}
```

ensure click on the game board to drop the ball

```
105     //MARK: - Event response
106     override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
107         // The bot is playing, or it hasn't started yet, or it's over
108         if self.canPlay == false {
109             return
110         }
111         // ensure click on the game board to drop the ball
112         if let touche = touches.first,
113            let result =  touche.view?.isDescendant(of: self.gameBoard),
114            result == true{
115             //Click on the game board
116             let point = touche.location(in: self.gameBoard)
117             self.calculationPosition(point: point)
118         }
119         // dropBubble()
120     }
121
```

Calculate which column to drop in

```
//Calculate which column to drop in
func calculationPosition(point: CGPoint) {
    var column:Int = -1
    //Circulation
    repeat{
        column += 1
    } while !((point.x >= discsWH * Double(column)) && (point.x <= discsWH *
        Double(column+1)))

    column += 1
    //Tell the bot which column I have placed it in
    if gameSession.isValidMove(column) {
        // Drop disc
        gameSession.dropDiscAt(column)
    }

}
```

After the drop show the disc

```
140     //MARK: After the drop show the disc
141     private func showDropDiscs(location:(row: Int, column: Int), color: DiscColor
            ,index: Int) {
142         // Remove this column
143         let discsColumnView = discsColumnViewList[location.column-1]
144         var frame = CGRect()
145         frame.origin = CGPoint.zero
146         frame.size = Constants.bubbleSize
147         var discsView: DiscsView!
148         if color == .yellow {
149             //Create a yellow ball, record the number of the ball and the column
150             discsView = DiscsView(frame: frame, borderColor: yellowColor2,
                    contentColor: yellowColor1,text: "\(index)", column:
                    location.column)
151         } else {
152             discsView = DiscsView(frame: frame, borderColor: redColor2,
                    contentColor: redColor1,text: "\(index)",  column: location.column)
153         }
154         self.discsViewList.append(discsView)
155         discsColumnView.addSubview(discsView)
156         //Add gravity drop animation
157         discsColumnViewList[location.column-1].addAnimationItem(discsView:
                discsView)
158     }
```

Choose who goes first

```
    //Choose who goes first
    func firstChoice() {
        showAlertVC(message: "Who plays first?") { index in
            // Copy random colours to the bot
            self.botColor = Int.random(in: 0..<2) == 0 ? .red : .yellow
            if index == 2 {
                self.isBotFirst = true
                self.resultLab.text = "Bot (\(self.botColor == .red ? "Red" : "Yellow")) Turn"
            }else {
                self.isBotFirst = false
                self.resultLab.text = "Your (\(self.botColor == .red ? "Yellow" : "Red")) Turn"
            }

            self.newGameSession()
        }
    }

    @IBAction func reStartGame(_ sender: UIBarButtonItem) {
        firstChoice()
    }

}
```

Winning side colour

```
        // Winning side colours
        if let color = sessionItem.winningColor {
            let botColor = DiscColor(rawValue: Int(sessionItem.botColor))
            cell.startLab.text = sessionItem.botIsFirst ? "Bot starts" : "User starts"
            cell.startLab.textColor = sessionItem.botIsFirst ? (botColor == .red ? redColor1 : yellowColor1) : (botColor == .red ? yellowColor1 : redColor1)
            if botColor == color {
                cell.winLab.text = "Bot wins!"

            }else {
                cell.winLab.text = "User wins!"
            }
            cell.winLab.textColor = color == .red ? redColor1 : yellowColor1

        }
        drawGameBoard(sessionItem: sessionItem, cell: cell)
        return cell
    }
```

Drawing a game board

```
    //Drawing a game board
    func drawGameBoard(sessionItem: CoreDataSession,cell: HistoryTableViewCell)  {
        print(sessionItem.log)          ⚠ Expression implicitly coerced from 'String?' to 'Any'
        let gameBoardView = cell.gameBoardView!
        let maskShapeLayer = cell.maskShapeLayer
        //Since cells are reused, it is important to remove the last
        maskShapeLayer.removeFromSuperlayer()
        maskShapeLayer.frame = gameBoardView.bounds
        //Create 42 holes
        let itemW = (gameBoardView.bounds.width)/7.0
        let itemH = (gameBoardView.bounds.height)/6.0
        //Frame height and weight of the disc
        let discsW = itemW - 4
        let discsH = itemH - 4
        var subMaskShapeLayerList = [CAShapeLayer]()
        for i in 0..<42 {
            let subMaskShapeLayer = CAShapeLayer()
            let rect = CGRect(x: 2 + itemW * Double((i%7)), y: 2 + itemH * Double(i/7), width: discsW, height: discsH)
            let discsPath = UIBezierPath(roundedRect: rect,cornerRadius: discsW/2.0)
            subMaskShapeLayer.path = discsPath.cgPath
            //Set all to white first, then yellow or red as required
            subMaskShapeLayer.fillColor = UIColor.white.cgColor
            maskShapeLayer.addSublayer(subMaskShapeLayer)

            //Record to data, find it and set the colour below
            subMaskShapeLayerList.append(subMaskShapeLayer)
        }
```

Get disc position data and determining colour

```
        }
        // Get disc position data
        if let discs = sessionItem.discs?.objectEnumerator().allObjects as? [CoreDataDisc]{
            //First player colour
            var firstColor: UIColor!
            //secound player colour
            var secondColor: UIColor!
            let botColor: DiscColor? = DiscColor(rawValue: Int(sessionItem.botColor))
            //bot first, bot colour is first colour
            if sessionItem.botIsFirst {
                firstColor = botColor == .red ? redColor1 : yellowColor1
                secondColor = botColor == .red ? yellowColor1 : redColor1
            } else {
                //Player's first hand, bot colour not first hand colour
                firstColor = botColor == .red ? yellowColor1 : redColor1
                secondColor = botColor == .red ? redColor1 : yellowColor1
            }
            for disc in discs {
                // 36
                //Reference conversion disc.row disc.column is from the bottom left corner, to convert to top left index
                let i = ((6 - disc.row) * 7 + disc.column) - 1
                if i < 42 {
                    let myColor =  disc.index%2 == 1 ? firstColor : secondColor
                    subMaskShapeLayerList[Int(i)].fillColor = myColor?.cgColor
                }
            }
        }
```

Restart game in history

```
    @IBAction func reStartGame(_ sender: UIBarButtonItem) {
        //Remove gravity ball effect
        for (_, discsView) in self.discsViewList.enumerated() {
            let discsColumnView = self.discsColumnViewList[discsView.column-1]
            discsColumnView.removeAnimationItem(discsView: discsView)
            discsView.removeFromSuperview()
        }
        //Delete the array
        self.discsViewList.removeAll()
        //Animation re-add
        animationShowDetail()
    }

}
```

The balls in each column are placed on this view and adding special effects

```
// The balls in each column are placed on this view

class DiscsColumnView: UIView {
    public var gravity = UIGravityBehavior()//Gravity effects
    public var collider = UICollisionBehavior()//Boundary collision effects
    public var animator: UIDynamicAnimator?

    override init(frame: CGRect) {
        super.init(frame: frame)
        //gravity.angle = 2 Spin on descent
        self.animator = UIDynamicAnimator(referenceView: self)
        self.animator?.addBehavior(self.gravity)
        self.collider.translatesReferenceBoundsIntoBoundary =  true

        self.animator?.addBehavior(self.collider)
    }

    required init?(coder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }


    func addAnimationItem(discsView: DiscsView) {
        collider.addItem(discsView)
        gravity.addItem(discsView)
    }

    func removeAnimationItem(discsView: DiscsView) {
        collider.removeItem(discsView)
        gravity.removeItem(discsView)
    }

}
```

Get the program's main window

```
//Get the program's main window
func getRootWindow() -> UIWindow {
    guard let windowScene = UIApplication.shared.connectedScenes.first as? UIWindowScene,
          let delegate = windowScene.delegate as? SceneDelegate, let window = delegate.window else { return UIWindow() }
    return window

}
```

Alert pop-ups

```
//Alert pop-ups
func showAlertVC(title: String, message: String)  {
    //Creating a pop-up controller
    let  alertController = UIAlertController.init(title: title, message:message , preferredStyle: .alert);
    // Creating actions
    let actionOne = UIAlertAction.init(title: "Cancel", style: .default)

    // Adding confirmation actions to pop-up controllers
    alertController.addAction(actionOne)
    // Main window display (modal) pop-up controller
    getRootWindow().rootViewController?.present(alertController, animated: true, completion: nil)
}
//Alert pop-ups
func showAlertVC(title: String, message: String, handler: (()->())?)  {
    //Creating a pop-up controller
    let  alertController = UIAlertController.init(title: title, message:message , preferredStyle: .alert);
    // Creating actions
    let actionOne = UIAlertAction.init(title: "Cancel", style: .cancel)

    let actionTwo = UIAlertAction.init(title: "Sure", style: .default){
      _ in
        handler?()
    }
    // Adding confirmation actions to pop-up controllers
    alertController.addAction(actionOne)
    alertController.addAction(actionTwo)
    // Main window display (modal) pop-up controller
    getRootWindow().rootViewController?.present(alertController, animated: true, completion: nil)
}
```

Main