RAMP

Rowan Academy of Mobile Programming

iOS Tic Tac Toe Game

Agenda – Day 3

- Introduction to Swift and Xcode
- Creating the Tic Tac Toe GUI
- Lunch Break
- Writing the Tic Tac Toe Game Code
- RAMP Wrap up

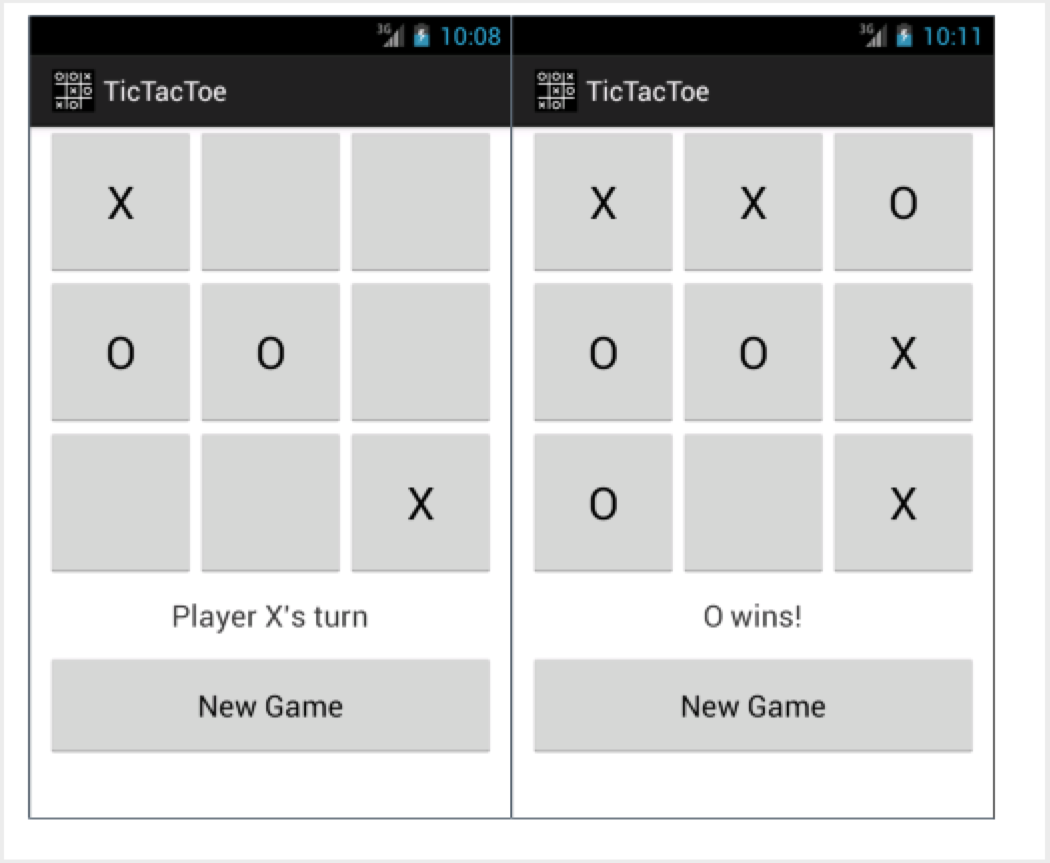
Process for Developing Mobile Applications

- Design and Build the Graphical User Interface (GUI)
- Create variables for the GUI objects the user will interact with
- Add listeners that capture the events generated by user interaction with GUI objects (i.e. Button Click)
- Write event handlers to handle the events

Tic Tac Toe GUI Description

In the first part of this project, you will build the GUI for a Tic-Tac-Toe game in Swift. You will use imageViews to represent the game board and a text control to display the status of the game at the bottom of the screen. The buttons will have no text on them when the game starts, but when the user clicks on a button, it will display an X, the computer will them move and turn the appropriate button text to an 0. The same goes if you use an imageView, no image will be displayed, but when you click on a square, either an x or o image will be displayed depending on who's turn it is. The label will indicate whose turn it is and when the game is over.

Tic Tac Toe GUI



Tic Tac Toe Game Description

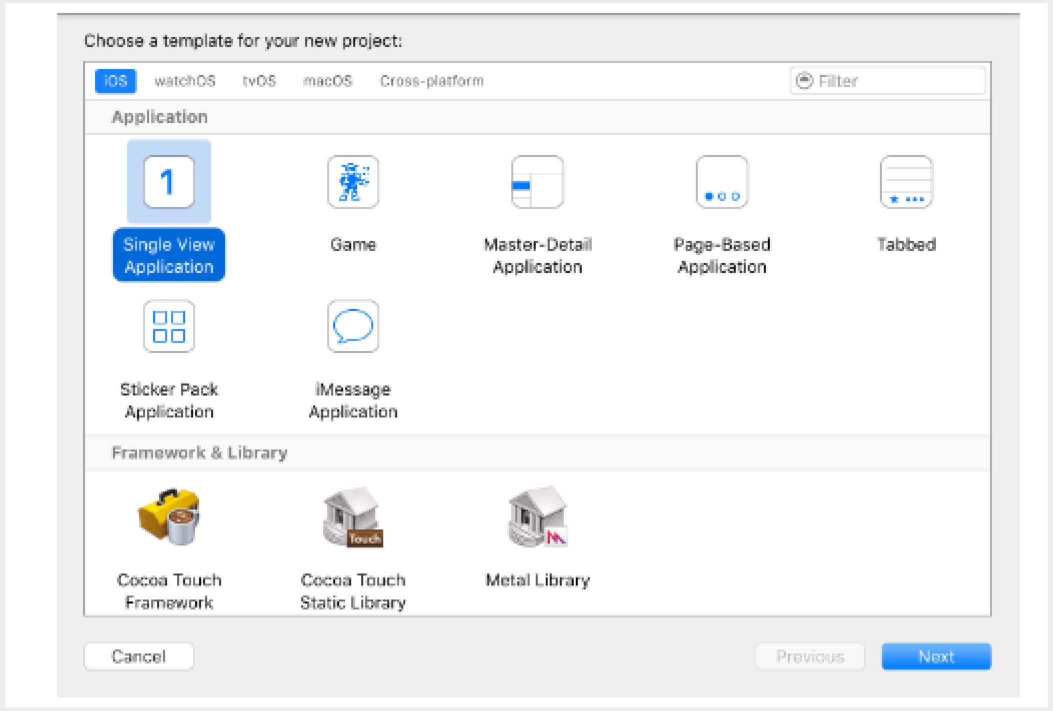
In the second part of this project, you will write the code for the Tic Tac Toe game in Swift. The game code consists of modules listed below that performs the task given by its name:

- **void getComputerMove( )**
- **void getUserMove( )**
- **int checkForWinner( )**
- **void displayBoard( )**
- **Void NewGame( )**

The game will work by the user touching a square on the gameboard that will place an X or depending on whose turn it is. We will check for a winner after the user places their mark on the board. If there is no winner, then we will update the text label to say that it is the next persons turn. We will also create a button for the user to reset the game.

Creating the Game Board (GUI)

Create a new Xcode **Single View Application project and name it “TicTacToe”.**



Creating the Game Board (GUI)

Open MainStoryBoard.storyboard.

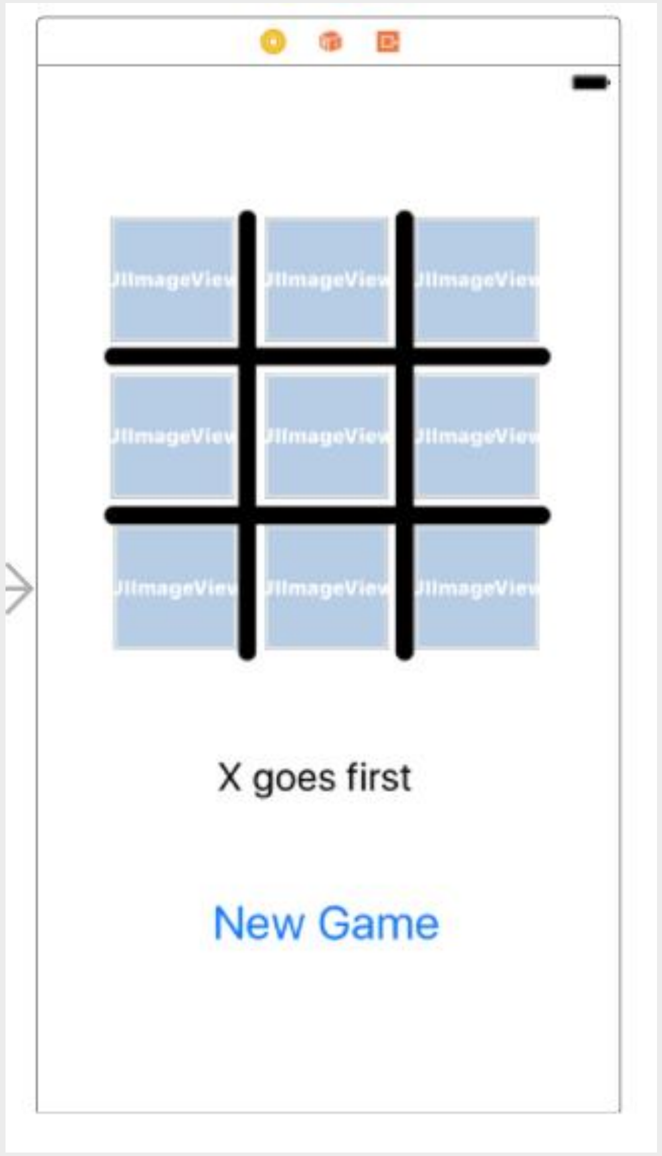
Drag a UIImageView to the window and resize it to be approximately 300 x 300 and set its image to Board image by using the Attributes Inspector.

Drag another UIImageView to the window and resize it to 80 x 80 then check the box for User Interaction Enabled.

Then copy and paste that 8 more times. Move those boxes to be hovering over the spaces of the gameboard lining them up.
Drag a UILabel to the window and position it under the board.

Drag a UIButton to the window and rename the text to Reset Game. The finished interface looks like the figure shown on the next slide:

Creating the Game Board (GUI)



Writing the Game Code

First thing you need to do is to create Outlets for all of the imageViews similar to this:

```
 @IBOutlet weak var s1: UIImageView!
```

Next create an Outlet for the label

```
@IBOutlet weak var whoseTurn: UILabel!
```

Next create the game variables:

```
 //Game Variables

   var mBoard: [String] = ["0", "1", "2", "3", "4", "5", "6", "7",
"8"]
   var mBoardArray: [UIImageView] = [];
   let BOARD_SIZE = 9

   let HUMAN_PLAYER = "X"
   let COMPUTER_PLAYER = "O"
   var turn = "X"
   var win = 0
   var move = -1
```

Writing the Game Code

Now that all of the GUI objects are wired, we can begin to write the game code.

You need to add listeners to the UIimageViews to capture when a click happens to make a move:

```
let tapGesture1 = UITapGestureRecognizer(target: self, action:
#selector(ViewController.img1Clicked))

    tapGesture1.numberOfTapsRequired = 1

    s1.addGestureRecognizer(tapGesture1)
```

Then we add the event handlers to handle a click on a UIimageView, this function sets the image for the UIimageView that was clicked and updates the board array:

```
 func img1Clicked(){

    if(player == 1){
      s1.image = #imageLiteral(resourceName: "x_img")
      board[0] = "X"
      player = 2
 }
```

Writing the Game Code

Finally, the code for the newGame function should reset everything:

```
@IBAction func NewGame(_ sender: UIButton) {
         s1.image = nil
        s2.image = nil
        s3.image = nil
        s4.image = nil
        s5.image = nil
        s6.image = nil
        s7.image = nil
        s8.image = nil
        s9.image = nil

        mBoard = ["1", "2", "3", "4", "5", "6", "7", "8", "9"]

        infoLbl.text = "Player X's Turn"

        win = 0

        turn = HUMAN_PLAYER
    }
```