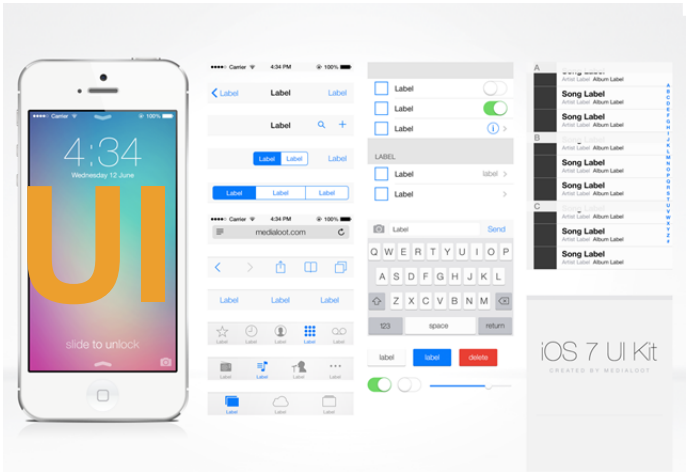
Chapter VI

Mastering Advance iOS UI Elements



VI.Mastering Advance iOS UI Element

In this chapter we will discuss detail for the following UI elements

TextField
ScrollView
TableView
CollectionView
Alert View Controller
Picker View

1. Advance iOS UI Elements

1. Textfield

A UITextField object displays an editable text area in your interface to gather text-based input from the user.

Useful Delegate Methods and Technique:

```
//After ViewController is adapted to UITextFieldDelegate

self.YOURTEXTFILENAME.delegate = self    //Set the Delegate in ViewDidLoad

//use following at suitable place

self.YOURTEXTFILENAME.resignFirstResponder( )       // To hide Keyboard
self.YOURTEXTFILENAME.becomeFirstResponder( )    // To show Keyboard

self.view.endEditing(true)                                    //To hide all keyboard

eg. textFieldShouldReturn delegated method is best place to hide keyboard
func textFieldShouldReturn(_ textField: UITextField) -> Bool {

        textField.resignFirstResponder( )
        return false
}

func textFieldDidEndEditing(_ textField: UITextField) {
        }
```

2. ScrollView

The UIScrollView class provides support for displaying content that is larger than the size of the application's window. It enables users to scroll within that content by making swiping gestures, and to zoom in and back from portions of the content by making pinching gestures.

Useful Properties and Delegate Methods

YOURSCROLLVIEW.frame -The Outerframe of Scroll View
YOURSCROLLVIEW.contentSize - To set content size
YOURSCROLLVIEW.contentOffset - Where to place content
YOURSCROLLVIEW.minimumZoomScale - Minimum zoomable scale
YOURSCROLLVIEW.maximumZoomScale -Maximum zoomable scale

YOURSCROLLVIEW.addSubView(anotherView) -Adding content view as a child

YOURSCROLLVIEW.autoresizesSubviews = true
YOURSCROLLVIEW.delegate = self

//Delegate Methods

viewForZoomingInScrollView -Asking for a view to scale when the scrollview is zoomed

3. TableView

To create a table view, several entities in an app must interact: the view controller, the table view itself, and the table view's data source and delegate. The view controller sets the data source and delegate of the table view and sends a reloadData message to it. The data source must adopt the UITableViewDataSource protocol, and the delegate must adopt the UITableViewDelegate protocol.

Useful Properties and method

YOURTABLEVIEW.indexPathForSelectedRow : used to refer the row selected by user

YOURTABLEVIEW . reloadData :user to reload data

Adding Refresh Control

```
let refreshControl = UIRefreshControl()
refreshControl.addTarget(self, action: #selector(updateNow), for:
UIControlEvents.valueChanged)
tableView.refreshControl = refreshControl
```

Useful Delegate Method and Technique

```
//After ViewController is adapted to UITableViewDataSource, UITableViewDelegate
//and create IBOutlet to TableView

self.YOURTABLEVIEW.delegate = self    //Set the Delegate in ViewDidLoad
self.YOURTABLEVIEW.dataSource = self

//Use following methods———————-
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int)
{
return    0 // NumberofRowAsPerYourData
}

func numberOfSections(in tableView: UITableView) -> Int {
        return 1
    }

func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
let cell = tableView.dequeueReusableCell(withIdentifier: "YourCellID") as?
YourCustomizedTableviewCellClass
return cell!                          //and return to table
}




func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath){

//Perform action when user click such as going to next detail view
//self.performSegueWithIdentifier("segueDetails", sender: self)
}
// set header title for each section if you use more than one section

 func tableView(tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
        if section == 0 {
            return "Section One" //etc… for section ….
        }
  return "Default Title"
}

//You may perform the following segue method before loading subsequent ViewController

 override func prepare(for segue: UIStoryboardSegue, sender: Any?)

//some where else, we can reference destination view controller
let destVc = segue.destination    as! DetailsViewController

}
```

4. Collection View

(See Demo of collection view)

5. Alert View Controller

1. Creating Alert Controller

UIAlertController(**title**: "Title Here", **message**: "Message Here", **preferredStyle**: Style)

Two Styles

UIAlertControllerStyle.alert
UIAlertControllerStyle.actionSheet

2. Adding Action Item

```
let action1 = UIAlertAction(title: "ITEMTITLE", style: UIAlertActionStyle.Default,
        handler:{ UIAlertAction in
})
```

Useful Styles

UIAlertActionStyle.default
UIAlertActionStyle.destructive
UIAlertActionStyle.cancel

3. Presenting Alert Controller

```
self.present(alertCtrl, animated: true, completion: nil)
```

6. Picker View

```
//After ViewController is adapted to UIPickerViewDataSource, UIPickerViewDelegate
//and create IBOutlet to PickerView

self.YOURPICKERVIEW.delegate = self        //Set the Delegate in ViewDidLoad
self.YOURPICKERVIEW.dataSource = self

//Use following methods———————-

func numberOfComponents(in pickerView: UIPickerView) -> Int
{
      return 2  //Number of Cols you want
 }

func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {

      if component == 0 {
         return  XX //Number of items you want in col 0

      }
      if component == 1  {
         return  XX //Number of items you want in col 1

      }
      return 0
   }

// title content for row in given column
   func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component:
Int) -> String? {

//User variable component and row to get value for title string
      if component == 0 {
         return "ITEM TEXT IN COLUM 0"
      }

      if component == 1 {
         return  "ITEM TEXT IN COLUM 1"
 }

         return "Invalid Row"
   }

func pickerView(pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int)
{
//User variable component and row to perform desired action
}
```

2. Interaction with Gesture

1. Gesture Types and Detection

Using UIGestureRecognizer, we can intercepts touches that are on their way to being handled by a view. When it recognizes a particular gesture, it calls a method on the object of your choice. There are several types of gesture recognizers built into the SDK.

Notes: ImageView are not able to accept gestures.

-UITapGestureRecognizer

-UIPinchGestureRecognizer

-UISwipeGestureRecognizer

-UIPanGestureRecognizer

-UIRotateGestureRecognizer

-UIScreenEdgeGestureRecognizer

-UILongPressGestureRecognizer

Properties

.numberOfTapsRequired

.delayTouchBegan

.requireGestureRecognizerToFail

.location(in:UIView?)

e.g

```
let tapGesture = UITapGestureRecognizer(target: self, action: "tap:")
tapGesture.numberOfTapsRequired = 2
self.view.addGestureRecognizer(tapGesture)

let pinchGesture = UIPinchGestureRecognizer(target: self, action: "pinch:")
self.view.addGestureRecognizer(pinchGesture)

let swipeGesture = UISwipeGestureRecognizer(target: self, action: "swipe:")
swipeGesture.direction = UISwipeGestureRecognizerDirection.Down
self.view.addGestureRecognizer(swipeGesture)

let rotGesture = UIRotationGestureRecognizer(target: self, action: "rotate:")
self.view.addGestureRecognizer(rotGesture)
```

```
        let panGesture = UIPanGestureRecognizer(target: self, action: "pan:")
         self.view.addGestureRecognizer(panGesture)

        let edgeGesture = UIScreenEdgePanGestureRecognizer(target: self, action:
"screenEdge:")
        self.view.addGestureRecognizer(edgeGesture)

        let lpGesture = UILongPressGestureRecognizer(target: self, action:
"longPress:")
        self.view.addGestureRecognizer(lpGesture)

//Example Events to intercept
func tap(gesture: UIGestureRecognizer) {
         print(gesture.locationInView(self.view))

}

func longPress(gesture: UIGestureRecognizer) {

         //Load and create Menu
         let menu = UIMenuController.sharedMenuController()

        let menuitem1 = UIMenuItem(title: "Help", action: "help:")
        let menuitem2 = UIMenuItem(title: "Delete", action: "del:")

        menu.menuItems = [ menuitem1,menuitem2]

        menu.setTargetRect(CGRect(x:gesture.locationInView(self.view).x,
y:gesture.locationInView(self.view).y, width:2, height:2),
          inView: self.view)
        menu.setMenuVisible(true, animated: true)

}

//To load menuController, this function is required
override func canBecomeFirstResponder() -> Bool {
        return true
    }
```

3. Handling iOS Hardware

1. Using Camera And Photo Library

Using UIImagePickerController, we can pickup video and photo from alblum and shoot ourself and store it in library. By incorporating protocols, UIImagePIckerControllerDelegate and UINavigationControllerDelegate, UIImagePickerController will perform transition and return the information as follow:
Note: Need to add permissions along with reason, to info.plist file as follows

Privacy - Camera Usage Description
Privacy - Media Library Usage Description
Privacy - Microphone Usage Description
Privacy - Photo Library Additions Usage Description

A. Setting up

```
import MobileCoreServices  //for constant KUTType

//adapt to protocols in addition to UIViewController

class ViewController: UIViewController,
UIImagePickerControllerDelegate, UINavigationControllerDelegate {


 let imagePicker = UIImagePickerController()

let imageMediaType = kUTTypeImage as String
let movieMediaType = kUTTypeMovie as String


// In Desired Triggered Action to pickup image or shoot

func pickFromCamera() {
        imagePicker.sourceType = .camera
        imagePicker.allowsEditing = true
        imagePicker.cameraFlashMode = .on
        imagePicker.mediaTypes = .availableMediaTypes(for: .camera)!
        present(imagePicker, animated: true, completion: nil)
 }
func pickFromLibrary() {
        imagePicker.sourceType = .photoLibrary
        imagePicker.allowsEditing = true
        present(imagePicker, animated: true, completion: nil)
 }
override func viewDidLoad() {
        super.viewDidLoad()
       imagePicker.delegate = self //****
  }
```

B. Getting Data

```
//Delegate Methods
func imagePickerController(_ picker: UIImagePickerController,
didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey :

Any]) {

let typeOfMedia = info[.mediaType] as! String
print(typeOfMedia)

switch typeOfMedia {

case movieMediaType: let tempVideoPath = info[.mediaURL] as? URL
        UISaveVideoAtPathToSavedPhotosAlbum((tempVideoPath?.path)!,
                                                        self,
        #selector(video(_:didFinishSavingWithError:contextInfo:)),nil)

 case imageMediaType: let image = info[.editedImage] as? UIImage
        picture.image = image
        print("User did shoot / Video")

        UIImageWriteToSavedPhotosAlbum(image!,
                                       self,
        #selector(image(_:didFinishSavingWithError:contextInfo:)),
                                       nil)
  default:()

   }
   picker.dismiss(animated: true, completion: nil )
}

@objc func image(_ image: UIImage, didFinishSavingWithError error:
Error?, contextInfo: UnsafeRawPointer) {
        if error == nil {
            print("Saved")
        }
        else {
            print(error?.localizedDescription)
        }
}

@objc func video(_ video: String, didFinishSavingWithError error:
Error?, contextInfo: UnsafeRawPointer) {
        if error == nil {
            print("Saved")
        }
        else {
            print(error?.localizedDescription)
        }
  }
```

2. Using TouchID

Sample Code:

```
import LocalAuthentication

//In event action

let ca = LAContext()
      var error:NSError?
      do {

         let status = try ca.canEvaluatePolicy(LAPolicy.deviceOwnerAuthenticationWithBiometrics,
                        error: nil   )

         let reason = "Testing TouchID"

          ca.evaluatePolicy(.deviceOwnerAuthenticationWithBiometrics, localizedReason: reason)
          {       (status, e) in
           if e != nil {
              print(e?.localizedDescription)
           } else {
              if status {
                 print("Confirmed!")
              } else {
                 print("Biometric failed try again")
              }
           }

        }
      } catch var error as Error {
          print("No Biometric supported")
          print(error.localizedDescription)
      }
```

3. Proximity Sensor, Accelerator Sensor and Gyroscope

Sample Code:

4. Multimedia

1. Playing Sound and Video

A. Useful Methods for Sound

```
//setup

var player: AVAudioPlayer = AVAudioPlayer()

or

let path  = Bundle.main.path(forResource: "01adele", ofType: "mp3")
let musicFileURL = URL(fileURLWithPath: musicFileLocation!)
player = AVPlayer(url: musicFileURL)
player.play()
player.pause()
```

B. Useful Properties for Sound

```
player.currentItem!.duration.seconds          -get the song length
player.currentTime().second                   -Set the location of Song
player.volume                                 -Set the volume

let cmt = CMTime(seconds:time, preferredTimescale: 1)
player.seek(to: cmt)
```

C. Useful Methods and Technique for Video

```
//setup

let moviePath = NSBundle.mainBundle().pathForResource("movie", ofType:
                    "mp4")!
let url = NSURL.fileURLWithPath(moviePath)
let player = AVPlayer(URL: url)

let playerViewController = AVPlayerViewController()
playerViewController.player = player

self.addChildViewController(playerViewController)
playerViewController.view.frame = CGRectMake(20, 50, 300, 300)
//playerViewController.view.frame = self.view.frame
self.view.addSubview(playerViewController.view)

player.play()
```

2. Using Timer

A. Setting up

```
timer = Timer.scheduledTimerWithTimeInterval(1.0, target: self, selector:
            "triggerEvent", userInfo: nil, repeats: true)

timer.fire( ) //Start the timer
```

```
func triggerEvent( )
{
        //To Do
}
```

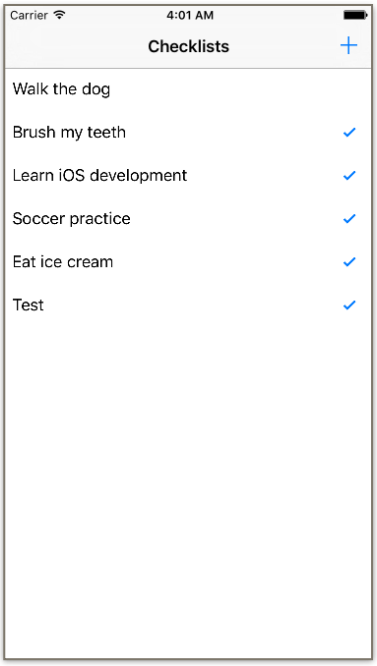
B. Methods

timer.invalidate() - To Stop Timer

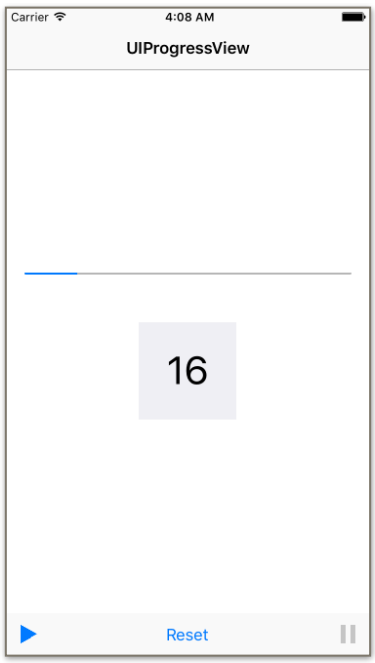
```
}
```

Chapter 6 Exercise:

1. Create a projects as shown in pic.1 using tableView



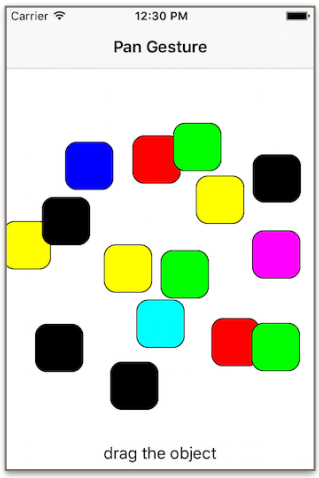
(pic.1)



(pic.2)

2. Create an app as shown in pic.2 using ProgressView and Timer, which let the progress view to fill up to 100% while counting the digit

3. Create an app that notify you at the designated time using Date and time picker and local notification.



(pic.3)

4. Create an app show in pic3 wherein the objects are able to move by a touch

5. Create an app that takes one photo from user album and the photo are later able to rotate , pinch for zooming effect.

6. Create a simple Music Player able to play designated songs from a List.

7. Type the following class definition and create additional two rectangles. rect2 and rect3

```
class rectangle {
var length: Double
var breadth: Double
init() {
length = 6
breadth = 12
}
}
var rect1 = rectangle()
print(rect1.length)
```

8. In previous example, modify the initialiser of class to accept length and breadth as a parameters. And put a function, able to compute an area of that rectangle and display.

9. Create a square class that inheritance from rectangle class in ex.1.But the propertie 'breadth' stored value will be initialise as the value in length properties

10. Create Two Initialiser in Class rectangle in exercise, one requiring with parameter to initialise length and breadth and other with default value as 12 for each properties.

11. Create a square objects, named square1, with its length as 5 unit. Assign new square object newSquare by square1. Print out both length. Change the original square square1 length to 7 and observe the newSquare's length again.

12. Create one array of square objects storing square1 and newSquare. and also create empty array to store square objects for future used.