Tutorial 1: Bull’s Eye

The objective of the game is to put the bull’s eye, which is on a slider that goes from 1 to 100, as close to a randomly chosen target value as you can.

When you’re confident of your estimate you press the **Hit Me!** button and a popup will tell you what your score is.

The closer to the target value you are, the more points you score. After you dismiss the alert popup by pressing the OK button, a new round begins with a new random target.

The game repeats until the player presses the **Start Over** button which resets the score to 0.

**Making a programming to-do list**

* A label with a random target value generated at the start of each round
* A slider that goes from 1 to 100
* A **Hit Me!** button to submit our choice and a popup will tell us our score
* A **Start Over** button that will reset the score and the round to 0
* A **Score** label: indicate the current score which is calculated by the target value minus the bull’s eye position
* A **Round** label: indicate the current round
* An **Info** button: shows the rules of the game

**The one-button app**

* Bulls Eye project is created

**Adding the button**

* Main.storyboard: **Hit Me!** button added

**The source code editor**

* ViewController.swift: **@IBAction func showAlert()** added

**View controllers**

The job of a view controller is to manage a single screen from your app.

* Main.storyboard file contains the design of the view controller’s interface.
* ViewController.swift contains its functionality

**Making connections**

* Click on the **Hit Me!** button to select it
* With the **Hit Me!** button selected, hold down the **Ctrl** key, click on the button and drag up to the **View Controller** item in the Outline pane
* Once you’re on View Controller, let go of the mouse button and a small menu will appear. It contains two section, **Action Segue** and **Sent Events**.

You’re interested in the **showAlert** option under Sent Events.

* Click on **showAlert** to select it.

This intructs Interface Builder to make a connection between the button and the line @IBAction func showAlert()

That is how you make buttons and other controls do things: you define an action in the view controller’s Swift file and then make the connection in Interface Builder.

**Acting on the button**

* ViewController.swift: added lines to showAlert()

The code in showAlert creates an alert with a title “Hello world”, a message “This is my first app!” and a single button labelled “Awesome”.

**How does an app work?**

An app is essentially made up of objects that can send messages to each other.

These objects communicate by passing messages to each other.

Everything your app does is triggered by some event.

**Working our way down the to-do list**

Let’s add the rest of the controls!

**Portrait vs landscape**

iPhone SE Portrait: 320 points horizontally and 568 vertically (640x1136 pixels)

iPhone SE Landscape: 568 points horizontally and 320 points vertically (1136x640 pixels)

**Converting the app to landscape**

* Main.storyboard: In the **View as:** **iPhone SE** panel, change **Orientation** to landscape
* Move the button back to the center
* Click the blue **BullsEye** project icon at the top of the **Project Navigator**
* Select **General** tab
* In the section **Deployment Info** -> **Device Orientation**
* Check only the **Landscape Left** and **Landscape Right** options and leave the Portrait and Upside Down options unchecked

**Objects, data and methods**

An object is a building block of your program.

Each object takes care of a specific part of the program.

An object can have both *data* and *functionality*:

* Data *contains* something, for example, the view controller contains the button
* Functionality *does* something, for example, the *showAlert* action respond to taps on that button

**Adding the rest of the controls**

* 8 Labels
* 1 Slider: value from 1 to 100
* 2 Buttons

**The slider**

“Read the value of the slider after the user presses the Hit Me button.”

* ViewController.swift: **@IBAction func sliderMoved(\_ slider: UISlider)** added

**Strings**

**Introducing variables**

* ViewController.swift:

- **@IBAction func sliderMoved(\_ slider: UISlider)** modified

- **@IBAction func showAlert()** modified

The **showAlert()** method shows now the current value of the slider.

**Your first bug**

* ViewController.swift: change the currentValue to 50

**Outlet**

The ViewDidLoad() message is sent by UIKit as soon as the view controller loads its user interface from the storyboard file.

There are three possible scope levels of a variable in Swift:

* Global scope. These objects exist for the duration of the app and are accessible from anywhere.
* Instance scope. These objects are alive for as long as the object that owns them stay alive.
* Local scope. These objects only exist for the duration of a method. As soon as the execution of the program leaves this method, the local objects are no longer accessible.
* ViewController.swift: **@IBOutlet weak var slider: UISlider!** added
* Main.storyboard: connect the slider object from the storyboard to the view controller’s slider outlet

**Generating the random number**

* ViewController.swift: **targetValue = 1 + Int(arc4random\_uniform(100))** added
* ViewController.swift: **var targetValue: Int = 0** added

**Adding rounds to the game**

A new round:

* Calculate a new random number
* Reset the slider to the halfway position
* ViewController.swift: **func startNewRound()** added

**Putting the target value into the label**

* ViewController.swift: **@IBOutlet weak var targetLabel: UILabel!** Added
* Main.storyboard: connect the label object from the storyboard to the view controller’s label outlet
* ViewController.swift: **func updateLabels()** added

Action methods vs Normal methods

An action method is the same as any other methods.

The only special thing is the @IBAction specifier. This allows Interface Builder to see the method so we can connect it to our buttons, sliders, etc…

If it is not marked as @IBAction, Interface Builder can’t see it.

**Calculating the score (p.71)**

If the slider’s value is greater than the target value, then the difference is: slider value minus the target value.

However, if the target value is greater than the slider value, then the difference is: target value minus the slider value.

Otherwise, both values must be equal, and the difference is zero.

* ViewController.swift: **@IBAction func showAlert()** modified

**What’s the score?**

* ViewController.swift: **@IBAction func showAlert()** modified

**Keeping track of the player’s total score**

* ViewController.swift: **@IBAction func showAlert()** modified

**Showing the score on the screen**

* ViewController.swift: **@IBOutlet weak var scoreLabel: UILabel!** Added
* Main.storyboard: connect the label object from the storyboard to the view controller’s label object

**One more round…**

* ViewController.swift: **@IBOutlet weak var roundLabel: UILabel!** Added
* Main.storyboard: connect the label object from the storyboard to the view controller’s label object

**Polishing the game**

* ViewController.swift: **@IBAction func showAlert()** modified

**Waiting for the alert to go away**

* ViewController.swift: **@IBAction func showAlert()** modified

**Starting over**

* ViewController.swift: **func startNewGame()** added
* ViewController.swift: **@IBAction startOver()** added