









# TP 1 - Adaptive User Interfaces










Adrien Humilière

15/02/2017













## Part 1

- Open, build and run ( + ) the **Flashlight** project.
- Observe the size of the simulator on the screen. Use the menu item   to adjust the size of the simulator screen .
- This is a really basic application template. Use it to discover Xcode interface anatomy.
- Use  +  and  +  to switch to the simulator and back; and to stop the app from Xcode.

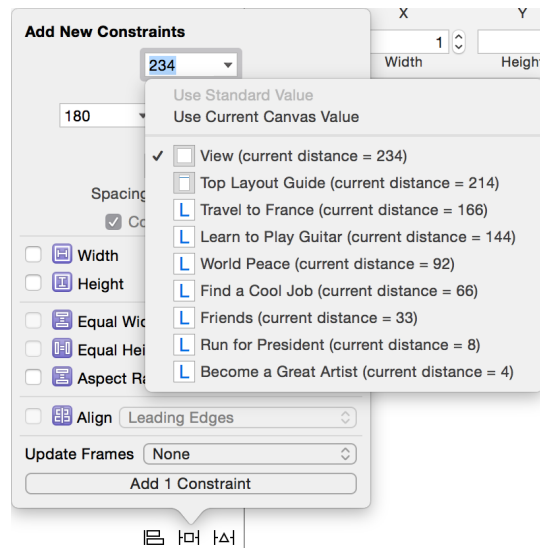
## Part 2

- Open and run ( + ) the **WordCollage** project.
- Using the Project Navigator ( + ), explore **Main.storyboard**.
- Using the Show Document Outline control () in the lower left corner of the canvas, ensure that the document outline is visible.
- Double-click a Label in the collage to change its contents.
- Run the app ( + ) , and witness the change in the iOS Simulator.
- Experiment with changing the content of the remaining labels to topics you care about.
- Run the app ( + ) , and witness the changes in the Simulator.

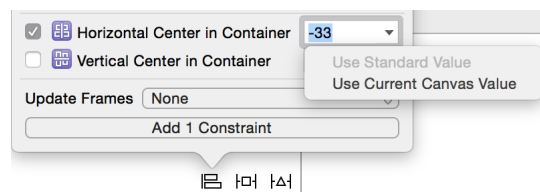
## Part 3

- Use the Project Navigator ( + ) to select Main.storyboard.
- Run the app ( + ) , and observe how the visual layout of the collage appears different in the iOS Simulator.
- Using the Object Library ( +  + ) , place a new Label on the interface. Change the Label contents (e.g. "Learn to Code") and use the Attributes Inspector ( +  + ) to change the font family, size and color (e.g. 51pt Avenir Next Ultra Light).
- Use the Label handles to expand its size, and adjust the Label position.
- Run the app ( + ) , and observe how the Label position appears differently in the iOS Simulator.

















- Position constraints must be added to the Label to influence its position.
- With the Label selected, use the Pin control to select a Vertical Space constraint relative to the View.























- Interface Builder displays a vertical blue bar representing the Vertical Space constraint. Missing constraints result in Interface Builder displaying Auto Layout issues in orange.
- With the Label selected, use the Align control to select a Center X Alignment constraint based on the current position of the Label.



- Interface Builder displays another vertical blue bar representing the Center X Alignment constraint.
- Using the Show Document Outline control (⌘⇧O) in the lower left corner of the canvas, ensure that the document outline is visible.
- Interface Builder displays one remaining Auto Layout issue in orange. Use the Issue Navigator (⌘⇧Y) or the Document Outline disclosure arrow (⊕) to observe the details of the remaining Auto Layout issue.
- With the Label selected, use the menu item **Editor > Resolve Auto Layout Issues > Update Frames** so the frame matches the constraint. Alternatively, use the menu item **Editor > Resolve Auto Layout Issues > Update Constraints** so the constraints match the frame.

- Run the app ( + ) and observe how the Label appears in a better position, but still appears somewhat different.
- Within the Interface Builder canvas, select the recently added Label, adjust its position, update the constraints ( +  + ) , and observe how the preview automatically reflects the change.
- Run the app ( + ) and observe how the Label appears as expected within the iOS Simulator.
- Rotate the app ( + ) within the iOS Simulator, and observe how the label appears in a different position when in a landscape orientation.
- Select the recently added Label, adjust its position, update the constraints ( +  + ) .
- Run the app ( + ) , rotate the app ( + ) in the Simulator, and observe the Label appearing in the expected position.

## Part 4

- Using Interface Builder and the Object Library ( +  + ) to place a Button on the interface.
- With the button selected, discover the Identity ( +  + ) , Attributes ( +  + ) and Size ( +  + ) Inspectors.
- Using Interface Builder, change the text of the button to "Change Background."
- Run the app ( + ) and observe how the button appears in a different location within the iOS Simulator.
- Using Interface Builder, Control-drag from the Button downward to the View, and select Bottom Space to Bottom Layout Guide to create a Vertical Space constraint.
- With the Button still selected, use the Align control and select Horizontal Center in Container to create a Center X Alignment constraint.
- Run the app ( + ) , tap the button, and observe that nothing happens.
- While viewing the storyboard in Interface Builder, open the Assistant Editor ( +  + ) .
- Using the Show Document Outline control () in the lower left corner of the canvas, ensure that the document outline is visible.
- Using the Document Outline, Control-click the button and drag a connection from the Touch Up Inside connection well to the controller, to create an Action connection. Use the name `changeBackgroundColor` and the Type `UIButton`.

---

```
1 @IBAction func changeBackgroundColor(sender: UIButton) {  
2  
3 }
```

---

- Drawing attention to the connection well next to the method, explain the how Interface Builder relies on the `@IBAction` attribute to establish connections between interface components and controller code.
- Experiment with removing the `@IBAction` attribute, and witness the connection well disappear. Undo the change, and witness the connection well reappear
- Implement the `changeBackgroundColor:` method.

---

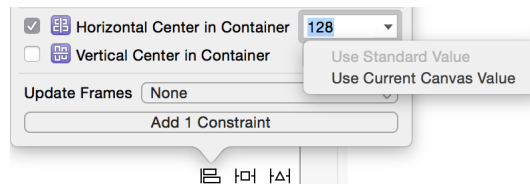
```
1 @IBAction func changeBackgroundColor(sender: UIButton) {  
2     view.backgroundColor = UIColor.blackColor()  
3 }
```

---




- Using the Xcode Documentation and API Reference ( $\uparrow + \mathbb{R} + 0$ ), discover the documentation for `UIColor` to discover other "easy" colors.
- Run the app ( $\mathbb{R} + R$ ), tap the button, and witness the background color change.

## Part 5

- Change the label of the existing Button contents to "Black."
- Using Interface Builder and the Object Library ( $\square + \mathbb{R} + L$ ), add a Button to the bottom left of the interface, labeled "White."
- Using Interface Builder, Control-drag from the Button downward to the View, and select Bottom Space to Bottom Layout Guide to create a Vertical Space constraint.
- With the Button still selected, use the Align control and select Horizontal Center in Container using the Current Canvas Value to create a Center X Alignment constraint.



- Add another button, labeled "Magenta," to the bottom right of the interface, and add constraints similar to the previous Button.

- Using Interface Builder and the Assistant Editor ( +  + ) , establish connections between each button and two new controller methods, `changeBackgroundColorToWhite:` and `changeBackgroundColorToMagenta:`.

---

```

1 @IBAction func changeBackgroundColorToWhite(sender: UIButton)↔
    {
2 }
3
4 @IBAction func changeBackgroundColorToMagenta(sender: ↔
    UIButton) {
5 }

```

---

- Implement the two methods.







---

```

1 @IBAction func changeBackgroundColorToWhite(sender: UIButton)↔
    {
2     view.backgroundColor = UIColor.whiteColor()
3 }
4
5 @IBAction func changeBackgroundColorToMagenta(sender: ↔
    UIButton) {
6     view.backgroundColor = UIColor.magentaColor()
7 }

```

---

- Rename `changeBackgroundColor:` to `changeBackgroundColorToBlack:`, and observe that the adjacent connection well appears hollow.
- Run the app ( + ) , tap the Black button, and witness the app crash. Stop the app ( + ) .
- The app crashed because Interface Builder still tries to connect the button to the `changeBackgroundColor:` method, which no longer exists.
- Using Interface Builder and the connection overlay, delete the old connection, establish a new connection to `changeBackgroundColorToBlack:`, and observe the connection well reappear.
- Run the app ( + ) , tap the buttons and witness the background color changing.