## **Learning objectives**

* Understand the Auto Layout system
* Understand view constraints and relationships
* Understand how to define view hierarchies implied in a graphic of a view containing many subviews
* Understand how to create constraints **programmatically**

## **Instructions**

## You’ll have to write some code in the ViewController class that sets up some buttons and a big green view for you, as well as a method that changes the size of the box from square, to portrait to landscape so that we can test our AutoLayout constraints.

You will need to implement a view hierarchy and Auto Layout constraints that will produce the following (sizes and constraints detailed below):

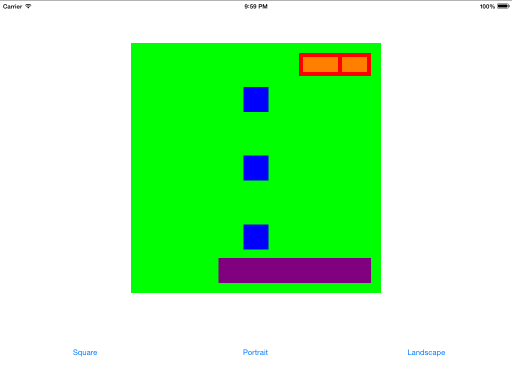
### 

### 

### 

### 

### **Square**

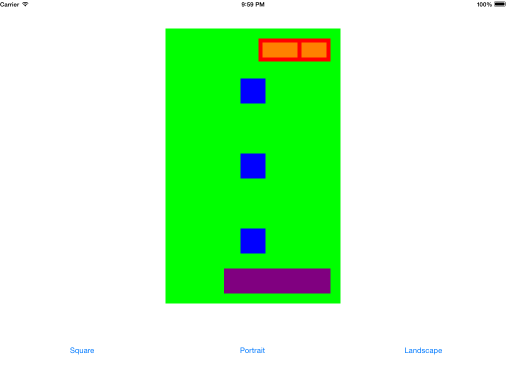
****

### 

### 

### 

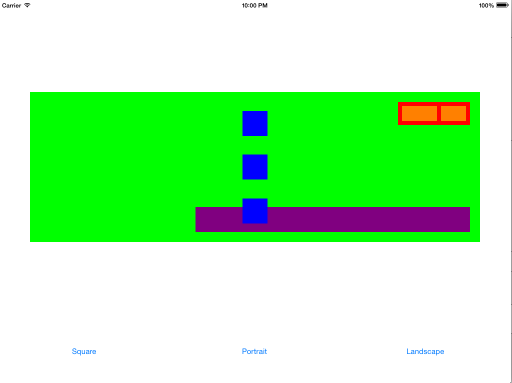
### **Portrait**

****

### 

### 

### **Landscape**

****

### 

### 

### 

### **Notes**

Let’s do the purple box together.

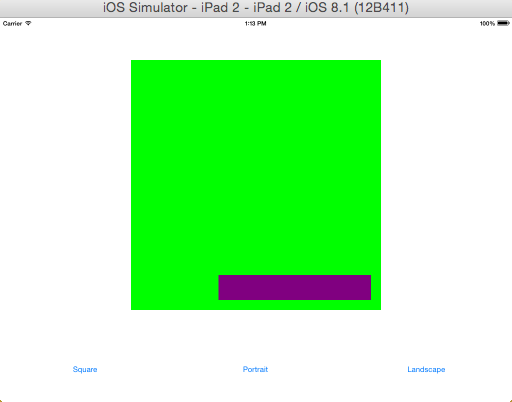
* Firstly, the goal we are trying to achieve is to have a purple box that changes width depending on our green box (mainView). To achieve this result, we add constraints on our purple box that change its width in proportion to green box's width. This is the beauty of AutoLayout.
* To start, we create a new UIView and set its frame to CGRect. We could have made the box with fixed values like (use CGFloat) CGRect(x: 0, y: 0, width: 100, height: 50). This creates a box at (0, 0) x, y coordinates and with 100 width and 50 height. However, we want to create something with dynamic properties. Notice that for the purple box x, y and width all change depending on if you are in square, portrait, or landscape mode. To make our size dynamic we set the frame to CGRect.zero, which is empty box (equivalent to CGRect(x: 0, y: 0, width: 0, height: 0)), and we assign layout constraints to change our x, y, and width depending on mainView's properties.
* Then we set translatesAutoresizingMaskIntoConstraints to false
* Set the view's background color to purple
* And add the purple box to our mainView.

Next, we will add some constraints (using NSLayoutConstraint).

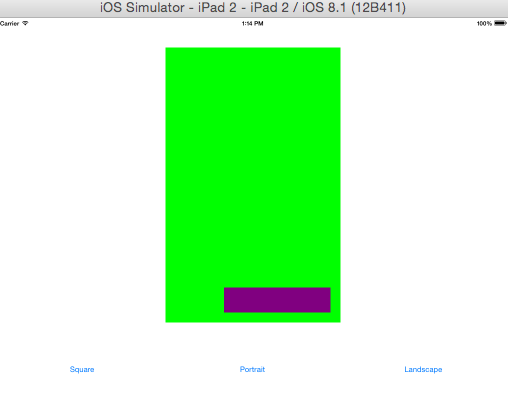
* [AutoLayout Programmatically (Docs)](https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/AutolayoutPG/ProgrammaticallyCreatingConstraints.html)
* [AutoLayout Programmatically (Article)](http://rshankar.com/how-to-programatically-add-autolayout-constraints/)

You should get the following results:

**Square- Purple Box**

****

**Portrait- Purple Box**

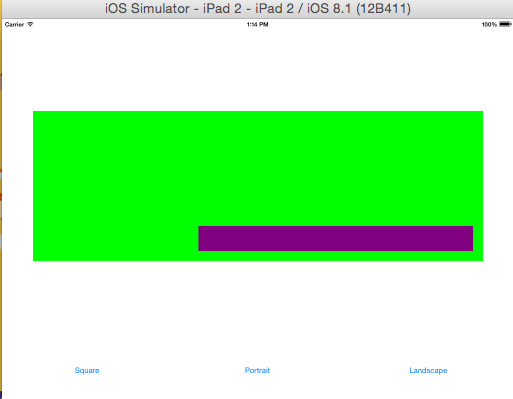
****

### 

### 

### 

### **Landscape- Purple Box**

****

## **References**

* [Auto Layout Guide](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/AutolayoutPG/index.html)
* [10 Things You Need To Know About Cocoa Auto Layout](http://oleb.net/blog/2013/03/things-you-need-to-know-about-cocoa-autolayout/)
* [Visual Format Language](https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/AutolayoutPG/VisualFormatLanguage.html)