Camera App Part 3 - Zooming

We are going to create the ability to zoom in and out on the selected photo by tapping twice on it. One double-tap to zoom in. One double-tap to zoom back out.

Embed the Image View in a Scroll View

1. Select the *UIImageView* on the Storyboard
2. Embed the Image View in a Scroll View by selecting ***Editor -> Embed In -> Scroll View***
3. Add an **IBOutlet** for the Scroll View and name it *scrollView*

This will remove your existing constraints on the Image View, so you will have to add them back. First add constraints to the ScrollView

Add Constraints to the Scroll View

1. Add Constraints for Leading, Trailing, Top and Bottom for the *UIScrollView*
2. Set the values for all of them to *zero*

Add Constraints to the Image View

1. Add constraints for Leading, Trailing, Top, and Bottom of the *UIImageView* to make it the same size as the Scroll View (its container).
2. Set the values for all of them to *zero*

In order for a Scroll View correctly size it's internal size, it needs to know more about the position of it's contents.

Add Constraints for the Image View for the Scroll View

1. Add two constraints so the Scroll View knows the location of the Image View inside of it.
2. Add *Center Horizontally in Container* and *Center *Vertically in Container* constraints to the *UIImageView*

> Be sure to *Update Frames* to ensure that there are no warnings or errors in the Auto Layout

Add a Gesture Recognizer to Zoom the Image

Let's create the Gesture Recognizer in the *viewDidLoad* method. We need to tell it that the *MainViewController* is the target class, and we want it to call a method we are going to create named *zoomImage*. We also need to set the number of taps to *two* so it only executes on a double-tap.

1. In the *MainViewController* class, add the following code in the *viewDidLoad* function to create a *UITapGestureRecognizer*

```
override func viewDidLoad() {
    ...

    let gesture = UITapGestureRecognizer(target: self, action: "zoomImage")

    gesture.numberOfTapsRequired = 2

    self.scrollView.addGestureRecognizer(gesture)

    ...
}
```

Create the *zoomImage* function

The zoomImage function will toggle the scroll view to zoom its contents between 1x and 2x.

Before creating the function, we will need a property to keep track of the current zoom setting.

1. Create a private property named *currentZoom* of type *CGFloat* and initialize it to 1.0

```
private var currentZoom : CGFloat = 1.0
```

2. Create a function and name it *zoomImage*
3. In the function, check the currentZoom. If it is equal to 1.0, then set it to 2.0. Otherwise, set it back to 1.0

```
if self.currentZoom == 1.0 {
    self.currentZoom = 2.0
}
else {
    self.currentZoom = 1.0
}
```

4. Set the Scroll View's zoom properties to the currentZoom

```
self.scrollView.minimumZoomScale = self.currentZoom
self.scrollView.maximumZoomScale = self.currentZoom
self.scrollView.zoomScale = self.currentZoom
```

Make the Main View Controller a Delegate of the Scroll View

1. Set the *MainViewController* to implement *UIScrollViewDelegate*

```
class MainViewController: UIViewController, UIImagePickerControllerDelegate, UINavigationControllerDelegate,
UIScrollViewDelegate {
...
}
```

2. Set the Scroll View's *delegate* property in the *viewDidLoad* method

```
override func viewDidLoad() {
        ...
    self.scrollView.delegate = self
        ...
}
```

3. Implement the *viewForZoomingInScrollView* function of the delegate and return the Image View from it.

```
func viewForZoomingInScrollView(scrollView: UIScrollView) -> UIView? {
    return self.displayImageView
}
```

> You should be able to run the app now.

Add a Little Animation to the Zoom

1. Wrap the setting of the Scroll View's zoom properties in a closure that is passed to the *animateWithDuration:animations:* function of *UIView*

```
UIView.animateWithDuration(0.5, animations: { ()-> Void in

    self.scrollView.minimumZoomScale = self.currentZoom
    self.scrollView.maximumZoomScale = self.currentZoom
    self.scrollView.zoomScale = self.currentZoom
})
```

2. Let's reduce the Closure to it's simplest form

   a. Since the Closure is the last parameter, we can remove the named property and move it outside of the parenthesis.

```
UIView.animateWithDuration(0.5) { () -> Void in

    self.scrollView.minimumZoomScale = self.currentZoom
```

```
    self.scrollView.maximumZoomScale = self.currentZoom
    self.scrollView.zoomScale = self.currentZoom
}
```

b. Since there are not any parameters to the Closure and it doesn't return anything, we can remove the *() -> Void in*

```
UIView.animateWithDuration(0.5) {

    self.scrollView.minimumZoomScale = self.currentZoom
    self.scrollView.maximumZoomScale = self.currentZoom
    self.scrollView.zoomScale = self.currentZoom
}
```

Prevent Strong Reference Cycle

A *Strong Reference Cycle* is caused by two things having a Strong reference to each other which prevents them from being able to release their memory. You can find out more about them and how memory management works here: https://developer.apple.com/library/prerelease/ios/documentation/Swift/Conceptual/Swift_Programming_Language/AutomaticReferenceCoun

1. We created a Strong Reference Cycle by calling self inside of the Closure. We can remedy that by using a keyword called *unowned*. Add *[unowned self] in* to the beginning of the closure statement

```
UIView.animateWithDuration(0.5) { [unowned self] in

    self.scrollView.minimumZoomScale = self.currentZoom
    self.scrollView.maximumZoomScale = self.currentZoom
    self.scrollView.zoomScale = self.currentZoom
}
```