UIView Animations with Options

We will consider animations customized using options. First of all we have the four options dealing with how the animation changes speed during its course:

- UIViewAnimationOptions.CurveEaseInOut
- UIViewAnimationOptions.CurveEaseIn
- UIViewAnimationOptions.CurveEaseOut
- UIViewAnimationOptions.CurveLinear (constant speed throughout)

Then we have the options:

- UIViewAnimationOptions.Repeat
- UIViewAnimationOptions.Autoreverse

dealing with repetition and auto reversal of animation. The two options can be combined to have an oscillating animation between initial and final states.

To deal with the two sets of options, we create a tabbed application. The first tab is used for animation curve options. A screen shot is shown below:

This has the controls on the interface:

- A UIView to be used in the animation
- A UISegment control with four segments for the four options
- A UILabel control

There is an outlet for the UIView and a variable for the original position of the view:

```
@IBOutlet weak var myView: UIView!

//variables for original state of myView
var originalPosition:CGPoint!
```

myView is centered horizontally and the viewDidAppear method initializes its vertical position and assigns the center of myView to the variable originalPosition:

```
override func  viewDidAppear(animated: Bool) {
    myView.center.y = 100
    //Stores myView's initial position
    originalPosition = myView.center
}
```

The method animate(option:UIViewAnimationOptions) provides the animation block:

```
func animate(option:UIViewAnimationOptions){
    UIView.animateWithDuration(2.0,delay: 0.5, options:option,
        animations: {
            var center = self.myView.center
            center.y += 600
            //Move myView center by 600 vertically down
            self.myView.center = center
        }, completion:nil)
}
```

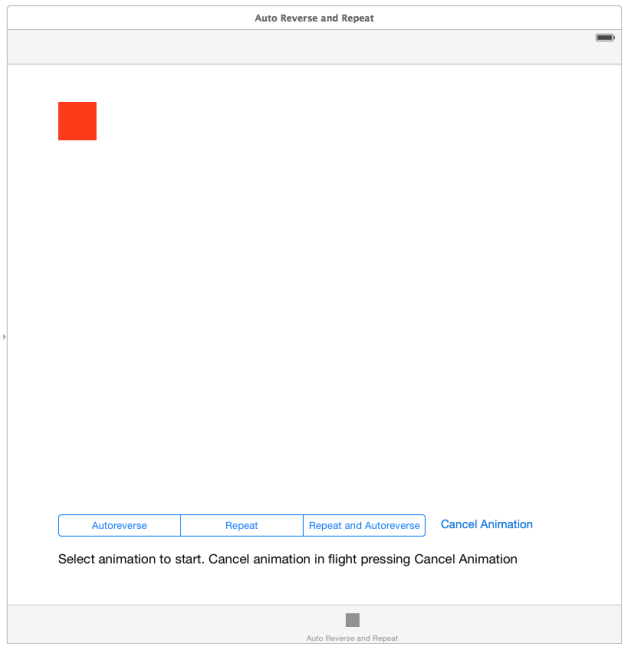
The animation moves myView by 600 points downwards by using the option assigned to its only parameter. Selecting a segment of the UISegment control chooses the option. The action method for this control is

```
@IBAction func changeOption(sender: AnyObject) {
    //Optional variable for the options parameter
    var opt:UIViewAnimationOptions!
    let segment = sender as UISegmentedControl
    // Reset myView to its original state
    myView.center = self.originalPosition

    switch segment.selectedSegmentIndex{
    case 0:
        opt = UIViewAnimationOptions.CurveEaseInOut
    case 1:
        opt = UIViewAnimationOptions.CurveEaseIn
    case 2:
        opt = UIViewAnimationOptions.CurveEaseOut
    case 3:
        opt = UIViewAnimationOptions.CurveLinear
    default:
        println("Should not be here")
    }
    animate(opt) //Call animate function with chosen option
}
```

When a segment is pressed, the animate function is called with the corresponding option and myView animates. The delay parameter has been set at 0.5 seconds; the animation will start with a delay of 0.5 seconds.

The second tab is used for animation that repeat, reverse, or do both. A screen shot is shown below:



This has the controls on the interface:

- A UIView to be used in the animation
- A UISegment control with three segments for the three options: Autoreverse, Repeat, and Autoreverse and Repeat combined
- A UILabel control

The outlets and variable declarations needed are:

```
@IBOutlet weak var myView: UIView!
@IBOutlet weak var cancelButton: UIButton!
@IBOutlet weak var segmentControl: UISegmentedControl!

//variables for original state of myView
var originalPosition:CGPoint!
var originalColor: UIColor!
```

The initialization is performed in the viewDidAppear method:

```
override func  viewDidAppear(animated: Bool) {
    //Stores myView's initial position
    originalPosition = myView.center
    //Stores myView's initial background color
    originalColor = myView.backgroundColor
    cancelButton.enabled = false //Disables cancelButton
}
```

The animation is performed by calling the animate(option:UIViewAnimationOptions) function:

```
func animate(option:UIViewAnimationOptions){
    cancelButton.enabled = true ////Enables cancelButton

    UIView.animateWithDuration(2.0,delay: 0,
                               options:option,animations: {
        var center = self.myView.center
        center.y += 600
        //Move myView center by 600 vertically down
        self.myView.center = center
        //Change myView background color to green
        self.myView.backgroundColor = UIColor.greenColor()
        }, completion:nil)
}
```

The animation includes moving the view down by 600 points and changing its background color to green.

The choice of option and calling the animate method is in the action method hooked to the segment control:

```
@IBAction func chooseOption(sender: AnyObject) {
    //Optional variable for the options parameter
    var opt:UIViewAnimationOptions!
    //sender downcasted to UISegmentedControl type
    let segment = sender as UISegmentedControl
    // Reset myView to its original state
    myView.center = self.originalPosition
    myView.backgroundColor = self.originalColor

    switch segment.selectedSegmentIndex{

       case 0:
           opt = UIViewAnimationOptions.Autoreverse
       case 1:
           opt = UIViewAnimationOptions.Repeat
       case 2:
          //Combination of Repeat and Autoreverse
          opt = UIViewAnimationOptions.Autoreverse
                    | UIViewAnimationOptions.Repeat
    default:
        println("Should not be here")
    }
    //Call animate function with chosen option
    animate(opt)
}
```

Note that Autoreverse and Repeat options are combined using bitwise or operator.

The animation may be cancelled by pressing the Cancel Animation button hooked to the following method:

```
@IBAction func cancelAnimation(sender: AnyObject) {
    //Animation to restore the myView initial state
    UIView.animateWithDuration(0.5,delay: 0,
        options:UIViewAnimationOptions.BeginFromCurrentState,
                                                animations: {

        self.myView.center = self.originalPosition
        self.myView.backgroundColor = self.originalColor
        }, completion:nil)

    //Unselects segment selected
    segmentControl.selectedSegmentIndex = -1
    cancelButton.enabled = false //Disables cancelButton
}
```

This calls an animation of short duration to take the view back to its original state.