UIImageView and UIImage Animation

Setting animation using UIImageView and UIImage is as simple as setting some properties and calling some methods. With UIImageView we can create a slide show:

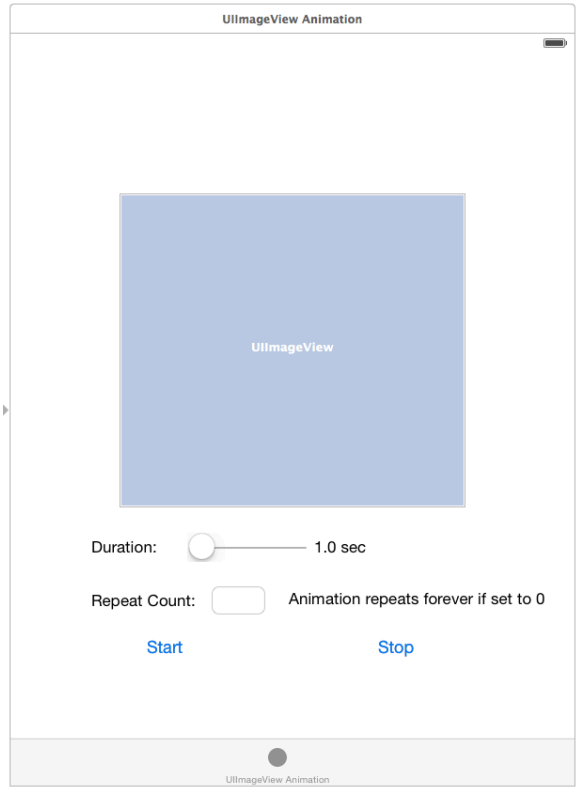
- Create an array of UIImages
- Assign the array to the animatedImages property of an instance of UIImageView
- Assign a value (NSTimeInterval) to the animationDuration property of the UIViewImage instance
- Assign a value to the animationRepeatCount property of the UIViewImage instance
- Call the instance method startAnimation() to start the animation
- Call the instance method stopAnimation() to stop the animation

UIImage can be animated by calling one of the class methods for UIImage:

- animatedImageNamed(name:, duration:)
- animatedImageWithImages(images:, duration)

Our primary objective is animation, so details of creating the project interface using Xcode will not be addressed.

We need a tabbed application with the first tab displaying UIImageView animation and the second tab displaying the UIImage animation. We will first discuss the UIImageView animation. The first tab is shown below:



The controls are:

- UIImageView: To display animation images
- UISlider: To change the value of the animationDuration property of UIImageView
- UITextField: To enter the value of animationRepeatCount property of the UIImageView
- UILabel (with displayed text: 1.0 sec): To display current value of animationDuration set by the slider control.
- Two buttons: Start button and Stop button to start and stop animation

We need to create outlets for the UIImageView, the UILabel to display the value of animationDuration, and the UITextField, so that they can be referenced in code.

```
@IBOutlet weak var imageView: UIImageView!
@IBOutlet weak var duration: UILabel!
@IBOutlet weak var repeatCount: UITextField!
```

Next we introduce the variables: currentDuration - a float type variable to store the current value of animationDuration, repeat – an Int type to store the value of animationRepeatCount, and imageArray of type UIImage to store an array of images

```
var currentDuration:Float = 1.0  //Current value of duration
var repeat = 1 //Initial value of repeat
var imageArray  = [UIImage]()//Empty array for images
```

We have sixteen images, conveniently named alaska1 … alaska16 in the Images folder of the project. These images are added to the imageArray calling the following function"

```
//Function to fill image array
func fillImageArray(){
    for i in 1...16{
        let imageName = "alaska\(i)"//Image name
        let image: UIImage = UIImage(named: imageName)
        imageArray.append(image)
    }
}
```

The overridden function viewDidLoad() is used to initialize FirstViewController view:

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Call function to fill image array
    fillImageArray()
    //Display repeat value in text field
    repeatCount.text = "\(repeat)"
    //Initial image in image view
    imageView.image = imageArray[0]
}
```

The animation is set up and started in the method hooked to the Start button:

```
@IBAction func startAnimation(sender: AnyObject) {
    if repeatCount.text.toInt() == nil{
        repeat = 0
        repeatCount.text = "\(repeat)"
    }
    else{
        repeat = repeatCount.text.toInt()!
    }
    //Set imageArray as animationImages
    imageView.animationImages = imageArray
    //Set imageView animationDuration. It is necessary
    // to convert currentDuration to NSTimeInterval
    imageView.animationDuration =
                        NSTimeInterval(currentDuration)
    //Set animationRepeatCount with repeat
    imageView.animationRepeatCount = repeat
    imageView.startAnimating() //Startanimation
}
```

In this function we first read the text of the text field by using the toInt() method applied to the text property of the text field. This returns nil if the text is not convertible to Int, otherwise it returns the integer value of the text. If the toInt() function returns nil, we set the variable repeat to be 0 and reset the text of the text field to be 0. Next step in setting up the animation is to assign the imageArray to the animationImages property of the image view. The animationDuration property expects a NSTimeInterval, so we need to perform a conversion from float to NSTimeInterval. Then the repeat variable is assigned to the animationRepeatCount property. Finally, the class method startAnimation() is called to start the animation on press of Start button.

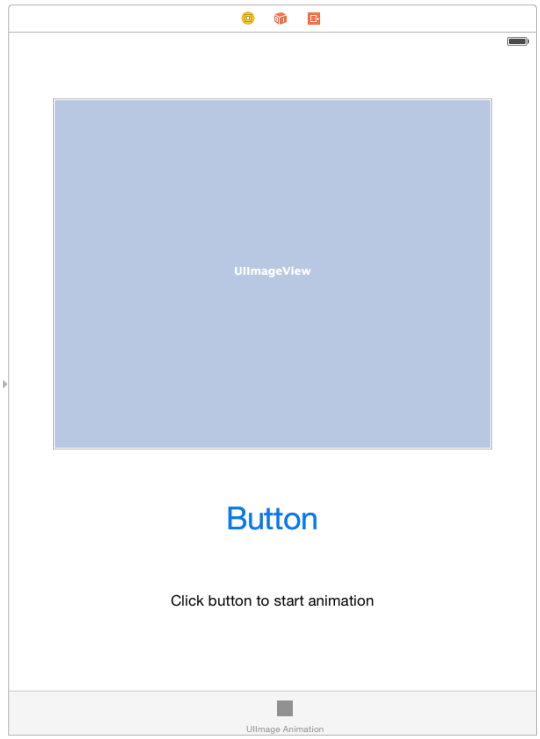
The following function is hooked to the Stop button to stop the animation when the button is pressed

```
@IBAction func stopAnimation(sender: AnyObject) {
    imageView.stopAnimating()//Stop animation
}
```

The following method is hooked to the slider to change the variable currentDuration when the slider is moved:

```
@IBAction func durationChanged(sender: AnyObject) {
    // AnyObject sender downcasted to UISlider
    let slider = sender as UISlider
    //currentDuration is set at slider value
    currentDuration = slider.value
    //duration label text updated with new currentDuration
    duration.text = String(format: "%0.1f sec", currentDuration)
}
```

The second tab (SecondViewController) is used for animation of UIImage. The controls placed on the interface are shown below:



The controls are:

- UIImageView: To display animation images
- UIButton: To display animation of images created in code.
- UILabel: To display the message: “Click button to start animation.

Now we need two outlets for the image view and the button:

```
@IBOutlet weak var imageView: UIImageView!
@IBOutlet weak var button: UIButton!
```

We also need an array variable to store the images drawn in code.

```
//Array to hold images created in code
var imageArray = [UIImage]()
```

The following function creates six filled circles of different radii using CGContext:

```
func createImageArray(){
    let w:CGFloat = 140
    for (var i = 0; i < 6; i++) {
        let j = CGFloat(i)
        UIGraphicsBeginImageContextWithOptions(CGSizeMake(w, w), false, 0)
        let con:CGContextRef = UIGraphicsGetCurrentContext()
        CGContextSetFillColorWithColor(con, UIColor.blueColor().CGColor)
        //Circles of gradually decreasing radius are created
        CGContextAddEllipseInRect(con, CGRectMake(0,0,w-j*20,w-j*20))
        CGContextFillPath(con)
        //UIImage created with content of UIGraphicsContext
        let im:UIImage = UIGraphicsGetImageFromCurrentImageContext()
        UIGraphicsEndImageContext();
        imageArray.append(im)//The image is appended to the array
    }
}
```

The circles have radii 140, 120, 100, 80, 60, and 20.

We initialize by calling the function createImageArray() and setting the button background green in the override func viewDidLoad() method.

```
override func viewDidLoad() {
    super.viewDidLoad()
    createImageArray() //Call createImageArray
    //Initial image in image view
    imageView.image = UIImage(named: "alaska1")
    button.backgroundColor = UIColor.greenColor()
}
```

There are two ways we can make a UIImage into an animated image.

The first method is using the UIImage class method: animatedImageNamed(name:, duration:). The name parameter is provided with a string such that appending it with an integer will give the name of an image in a folder in the project. The method gets the first image name by appending “0” (or if that fails “1”) and then adds the image to an array. This is continued by adding larger integers until either we run out of images or we reach “1024”. The resulting array constitutes an animated image. We can add this image to the image view to animate.

In the second method, we supply an array of UIImages as the first parameter of the class method: animatedImageWithImages(images:, duration). In our example, we use the array of images created in code. Then we provide the animated image to the setImage instance method of the button.

```
@IBAction func startAnimation(sender: AnyObject) {
    //Animating UIImageView by loading an animated image
    let animatedImage:UIImage = UIImage.animatedImageNamed("alaska",
        duration: 5.0)
    imageView.image = animatedImage

    //Animate image from the array using a UIImage class method
    let im:UIImage = UIImage.animatedImageWithImages(imageArray,
                                                     duration: 3.0)
    //Set the button image with the animated image im
    button.setImage(im, forState: UIControlState.Normal)
}
```

Note that the value of the second parameter duration in both the class methods determines the duration of the complete animation. However, the animation will continue forever until the application shuts down.