ATLS 4120/5120: Mobile Application Development
Week 7: Animations

We're going to create a simple animation using the NSTimer class.

Create a new Single-view app called animation for the iPad.
Add tennis_ball.png into your Images folder and name it ball.

Go into MainStoryboard and in the File Inspector uncheck Use Auto Layout and Use Size Classes (choose to keep the iPad size class)

Add an image view that is just the size of tennis_ball.png and add that as its image.
Select the view outside the image view and change the background to black so the background of the image blends in.
Add 2 labels and a slider.
One label should say Interval and the other one will be blank. It will hold the slider's value.
The slider should have a min of 0 and a max of 1. Set the initial value to .1
Now create outlet connections for the slider called slider, for its label called sliderLabel, and for the image called imageView.
Also create an action for the slider called sliderMoved for when the slider is moved.
You can run it and see the lay out but it doesn't do anything yet.

ViewController.swift
We will also need some variables in our class.

```
    var delta = CGPointMake(12, 4) //initialize the delta to move 12 pixels
horizontally, 4 pixels vertically
    var ballRadius = CGFloat() //radius of the ball image
    var timer = NSTimer() //animation timer
```

We need a method that changes the position of the image view.

```
    //changes the position of the image view
    func moveImage() {
        imageView.center=CGPointMake(imageView.center.x + delta.x,
imageView.center.y + delta.y)
        if imageView.center.x > view.bounds.size.width-ballRadius ||
imageView.center.x < ballRadius{
            delta.x = -delta.x
        }
        if imageView.center.y > view.bounds.size.height - ballRadius ||
imageView.center.y < ballRadius {
            delta.y = -delta.y
        }
    }
```

Now lets create a method that's going to create a timer based on the chosen interval.

```
//updates the timer and label with the current slider value
    @IBAction func changeSliderValue() {
        sliderLabel.text=String(format: "%.2f", slider.value)
        timer = NSTimer.scheduledTimerWithTimeInterval(Double(slider.value),
target: self, selector: "moveImage", userInfo: nil, repeats: true)
    }
```

Now we implement the method that will be called when the user moves the slider.

```
    @IBAction func sliderMoved(sender: UISlider) {
        timer.invalidate()
        changeSliderValue()
    }
```

Let's do some set up in viewDidLoad

```
    override func viewDidLoad() {
        //ball radius is half the width of the image
        ballRadius=imageView.frame.size.width/2
        changeSliderValue()
        super.viewDidLoad()
    }
```

Your ball should now animate.
Snapshot: animation basic

Cool, but at the slower speeds it's pretty choppy. Let's make the animation smoother using UIView animations to move the image view.

Let's add the following to the beginning of the `moveImage` method

```
        let duration=Double(slider.value)
        UIView.beginAnimations("tennis_ball", context: nil)
        UIView.animateWithDuration(duration, animations:
{self.imageView.center=CGPointMake(self.imageView.center.x + self.delta.x,
self.imageView.center.y + self.delta.y)})
        UIView.commitAnimations()
```

Remove

```
imageView.center=CGPointMake(imageView.center.x + delta.x,
imageView.center.y + delta.y)
```

Now try it. At the faster speeds it's smoother.
Snapshot: uiview animation block

Now let's use transforms in Core Graphics to do translation, rotation, and scaling.

Translation
In translation we'll use the transform property instead of center.

In ViewController.swift add

```
var translation = CGPointMake(0.0, 0.0) //specifies how many pixels the
image will move
```

Then edit moveImage and remove in the animation block

```
self.imageView.center=CGPointMake(self.imageView.center.x + self.delta.x,
self.imageView.center.y + self.delta.y)
```

and change it to:

```
animations: {
self.imageView.transform=CGAffineTransformMakeTranslation(self.translation.x
, self.translation.y)
        self.translation.x += self.delta.x
        self.translation.y += self.delta.y
        //sets transform to CGAffineTransform
})
```

Modify to use translation

```
        if imageView.center.x + translation.x > self.view.bounds.size.width-
ballRadius || imageView.center.x + translation.x < ballRadius{
            delta.x = -delta.x
        }
        if imageView.center.y + translation.y > self.view.bounds.size.height
- ballRadius || imageView.center.y + translation.y < ballRadius {
            delta.y = -delta.y
        }
```

Now run it using translation.
Snapshot: animation translation

<u>Rotation</u>
The rotation transformation enables you to rotate a view using the angle you specify.

ViewController.swift

```
var angle: CGFloat=0.0 //angle for rotation
```

Edit moveImage
Change the animation block to implement rotation

```
animations: {
self.imageView.transform=CGAffineTransformMakeRotation(self.angle)
        self.imageView.center=CGPointMake(self.imageView.center.x +
self.delta.x, self.imageView.center.y + self.delta.y)
}
```

Remove

```
        self.translation.x += self.delta.x
        self.translation.y += self.delta.y
```

Add

```
        angle += 0.02 //amount by which you increment the angle
        //if it's a full rotation reset the angle
        if angle > CGFloat(2*M_PI) {
            angle=0
        }
```

Modify(remove translation piece)
```
        if imageView.center.x > self.view.bounds.size.width-ballRadius ||
imageView.center.x < ballRadius{
            delta.x = -delta.x
        }
        if imageView.center.y  > self.view.bounds.size.height-ballRadius ||
imageView.center.y  < ballRadius {
            delta.y = -delta.y
        }
```

Snapshot: animation rotation 2

Scaling

You can change the transform to scale by using angle as the factor to scale the x and y axis
Change the animations block to call CGAffineTransformMakeScale to scale the image.

```
self.imageView.transform=CGAffineTransformMakeScale(self.angle, self.angle)
```

Add app icons and launchscreen

Snapshot: final