# Stalker App

## Project Overview

### Reading Touches Directly

UIView, the principal display component of the UIViewController has many built-in overrides available to subclasses to assist in modifying or enhancing its functionality. Of these are a group of overrides that are called whenever touches are detected on the UIView. They are:

1. `func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?)`
2. `func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?)`
3. `func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?)`
4. `func touchesCancelled(_ touches: Set<UITouch>, with event: UIEvent?)`

You override these built-in functions by preceding the function declaration with the keyword `override` and following the declaration with a pair of open and closed braces. For the purposes of this course, the various code enclosing characters will be referred to by the following names:

- ( )  Parenthesis
- [ ]  Brackets
- { }  Braces

Each of the overrides functions above are called by the iOS and passed the same two parameters:

1. `touches: Set<UITouch>`
   This is a Set (a collection of non-indexed, non-repeatable objects representing UITouch objects). Each UITouch object provides properties regarding the location in the view where the touch occurred as x, y coordinates. It also provides the x,y coordinates of the previous touch so you can compute direction, speed, distance, etc.
2. `event: UIEvent?`
   The optional UIEvent if provided will give information about any event that might be associated with the touch. This is primarily useful of a touch is part of a higher-level gesture being handled by a UIGestureRecognizer.

### Animating UIView Property Changes

Many properties of a UIView can be animated, including size, position, and transform (rotation).  To animate changes to these properties, use the built-in class functions of UIView to animate changes to any UIView's properties as follows:

```
UIView.animate(withDuration: 1.5, animations: {
    // Enter the property changes here you want to animate.
})
```

Where 1.5 is the number of seconds for the animation. You can use any number you like.

# Stalker App

## Project Setup

We've now had several of these written exercises, so we should be pretty adept at performing the required operations without a lot of *step-by-step* instructions. So you'll find these instructions to be at a somewhat "higher-level" than previous exercises.

### *Create a new project*

1. Download "flash.png" and "redball.png" images from eCampus for this project.
2. Create a new Xcode project for a "Single View" application using Swift and Universal, and save it as "Stalker" to your Desktop. Then drag your downloaded images to this project by first dragging them into your "Stalker" folder, then into your project.
3. Create two new UIImageViews in your UIViewController's main view in Storyboard, and initialize one with the "flash.png" image, and the other with the "redball.png" image. Set the flash.png image view to 'hidden'.
4. Connect the "flash.png" image view to your "ViewController.swift" source as "flash", and connect the "redball.png" image view as "redball".

### *Add UIView Touch Overrides*

5. Add override functions for all four touch events (i.e. touchesBegan, touchesMoved, touchesEnded, and touchesCancelled).
6. In touchesBegan, access the center point of the first touch and assign it to the flash image view center property. Also set the flash image view isHidden property to false so it shows up visible on the screen. Use UIView animation to animate the setting of the redball image view center property to the same center as the flash image view center, using the following code fragment:

   ```
   UIView.animate(withDuration: 1.5, animations: {
     self.redBall.center = touches.first!.location(in: self.view)
   })
   ```

   This will start the center adjustment animation for the redball to the initial location of the flash center where the initial touch occurred.

7. In touchesMoved, set both the flash center and the redball center to the touch's current location. Re-animation is not necessary as the current animation will auto-adjust to the new destination.
8. In touchesEnded and touchesCancelled, set the flash image view isHidden property to true. Either of these overrides indicate that the touch has ended, so the flash should disappear.
9. Run your app and check for bugs.