WatchKit Animation Tutorial

One of the most surprising things about WatchKit is that we no longer get to use our favorite UIKit components. Instead, we're dealing with WKInterfaces, which do not have a layer property for us to animate with. Instead, to animate on the Watch, we have to essentially flip through many images.

So let's get started!

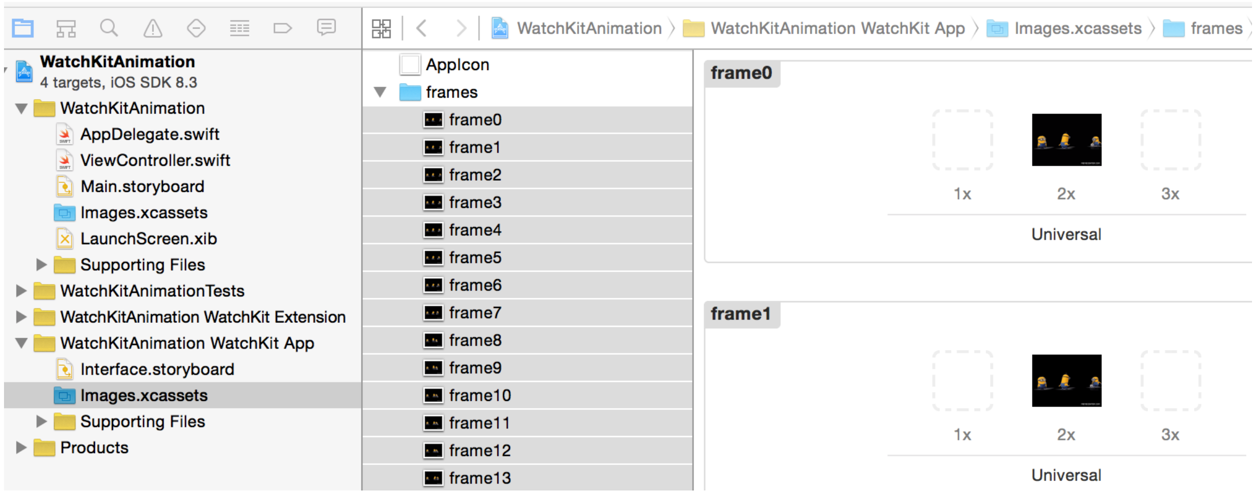
Start a New WatchKit Project

Follow the **Hello, WatchKit Tutorial** to set up a new WatchKit project. Make sure your Watch App runs properly.

Import Animation Images

For this tutorial, I took a fun gif of Minions playing soccer, and used a random Gif Splitter I found online to extract the individual frames from the gif. I then renamed the results to be all of the same name, but with the frame number at the end, ending with @2x.pngs. You can see the resulting frame images in the **AnimationAssets** folder of this tutorial.

Import these images into the **Images.xcassets** folder in your WatchKit App (NOT the WatchKit Extension! This will make sure that the asset is readily available on the Watch instead of having to get the image from the iPhone via Bluetooth).



Create the Image Interface

1. In **Interface.storyboard**, add an **Image** from the Object Library to your Interface Controller

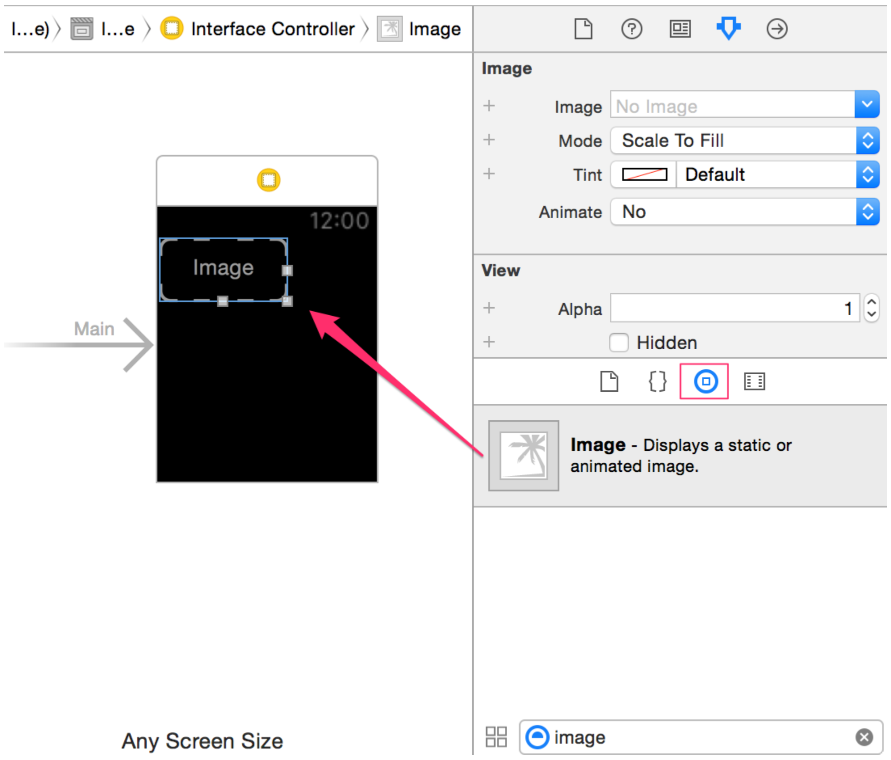


Image
Image: No Image
Mode: Scale To Fill
Tint: Default
Animate: No

View
Alpha: 1
Hidden: ☐

Image - Displays a static or animated image.

2. Adjust the **Image Position** and **Size**

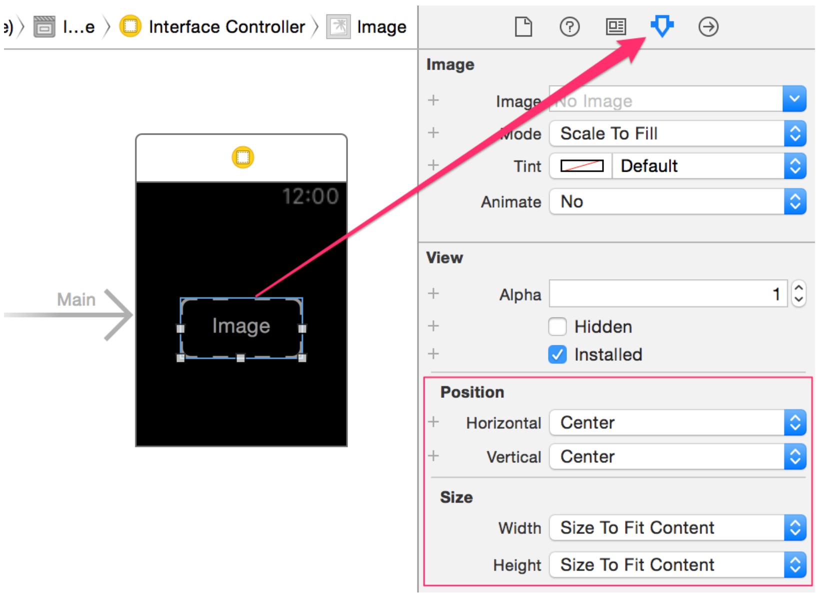


Image
Image: No Image
Mode: Scale To Fill
Tint: Default
Animate: No

View
Alpha: 1
Hidden: ☐
Installed: ✓

Position
Horizontal: Center
Vertical: Center

Size
Width: Size To Fit Content
Height: Size To Fit Content

Create Animation

In Storyboard

1. In **Interface.storyboard**, set the Image to the core animation image name (drop the numbers at the end!) - in our case **frame**. You will not be able to see the image.

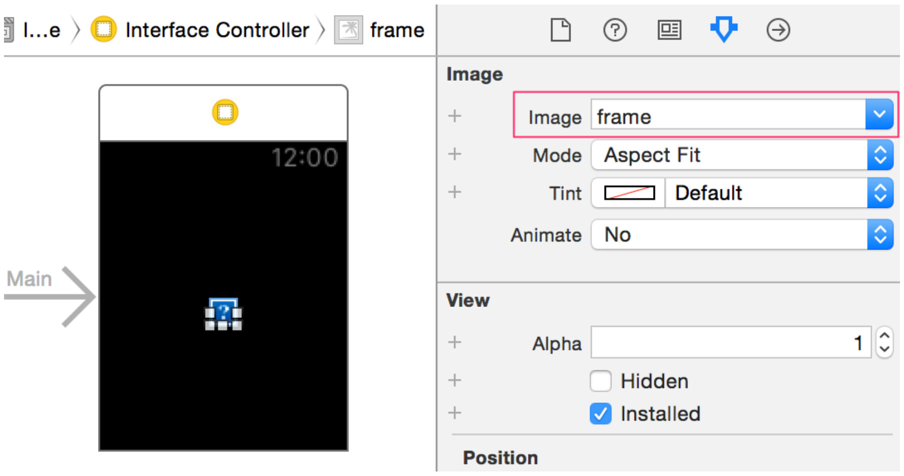


Image
Image: frame
Mode: Aspect Fit
Tint: Default
Animate: No

View
Alpha: 1
☐ Hidden
☑ Installed

Position

2. Change the **Animate** option to **Yes** and set the **Duration to 5 seconds**.

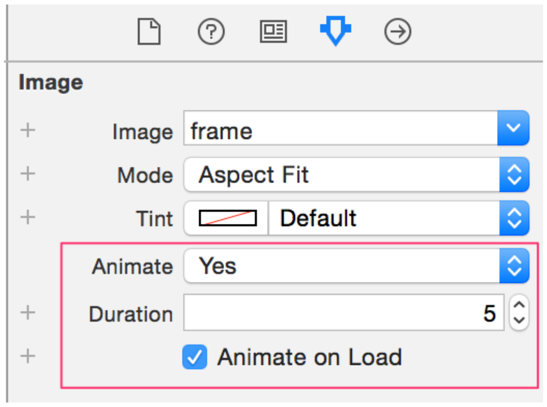
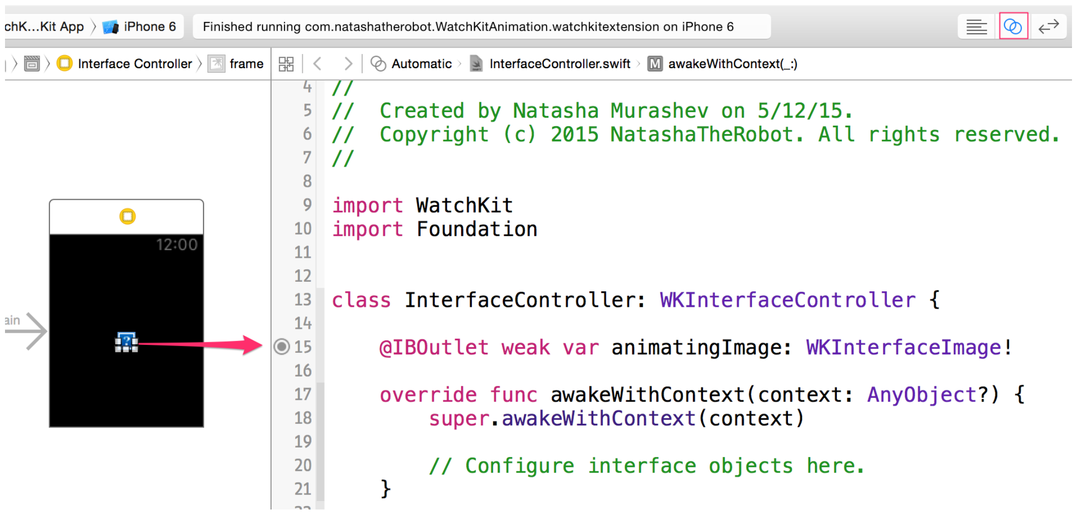


Image
Image: frame
Mode: Aspect Fit
Tint: Default
Animate: Yes
Duration: 5
☑ Animate on Load

3. That’s it! Now just run the app to see the animation.

In Code

1. Set the **Animate** option for the Image back to **No** - we'll animate in code this time!
2. Create an **@IBOutlet** for your **Image**


chK...Kit App 〉 iPhone 6
Finished running com.natashatherobot.WatchKitAnimation.watchkitextension on iPhone 6
Interface Controller 〉 frame
Automatic 〉 InterfaceController.swift 〉 awakeWithContext(_:)
12:00
ain

```swift
//
//  Created by Natasha Murashev on 5/12/15.
//  Copyright (c) 2015 NatashaTheRobot. All rights reserved.
//

import WatchKit
import Foundation


class InterfaceController: WKInterfaceController {

    @IBOutlet weak var animatingImage: WKInterfaceImage!

    override func awakeWithContext(context: AnyObject?) {
        super.awakeWithContext(context)

        // Configure interface objects here.
    }
```

3. In your **InterfaceController**, **Control + Click** on **WKInterfaceImage** to bring up the Image documentation. Notice this is NOT a UIImage, it is very limited. Notice the three animation methods:

```swift
@availability(iOS, introduced=8.2)
class WKInterfaceImage : WKInterfaceObject {

    func setImage(image: UIImage?)
    func setImageData(imageData: NSData?)
    func setImageNamed(imageName: String?)

    func setTintColor(tintColor: UIColor?)

    func startAnimating() // play all images repeatedly using duration
        specified in interface description
    func startAnimatingWithImagesInRange(imageRange: NSRange,
        duration: NSTimeInterval, repeatCount: Int) // play subset of
        images for a certain number of times. 0 means repeat until
        stop
    func stopAnimating()
}
```

4. Try out the **startAnimating()** method.

```swift
class InterfaceController: WKInterfaceController {

    @IBOutlet weak var animatingImage: WKInterfaceImage!

    override func awakeWithContext(context: AnyObject?) {
        super.awakeWithContext(context)

        animatingImage.startAnimating()
    }
```

5. Run the app to see what the default animation looks like.

6. Try out the **startAnimatingWithImagesInRange:duration:repeatCount:**

```
class InterfaceController: WKInterfaceController {

    @IBOutlet weak var animatingImage: WKInterfaceImage!

    override func awakeWithContext(context: AnyObject?) {
        super.awakeWithContext(context)

        animatingImage.startAnimatingWithImagesInRange(
            NSRange(location: 0, length: 132),
            duration: 5,
            repeatCount: Int.max)
    }
```

7. Run the app to see what the animation option looks like with your custom parameters.
8. Adjust the parameter input and re-run the app.