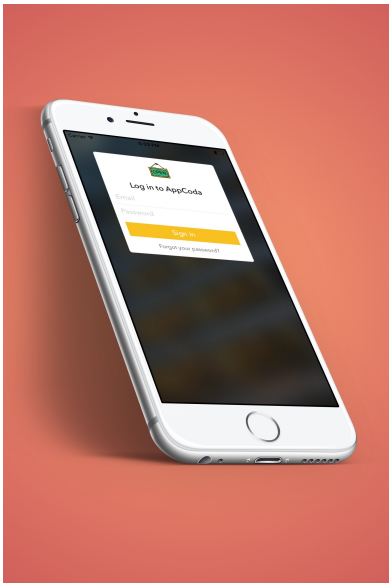
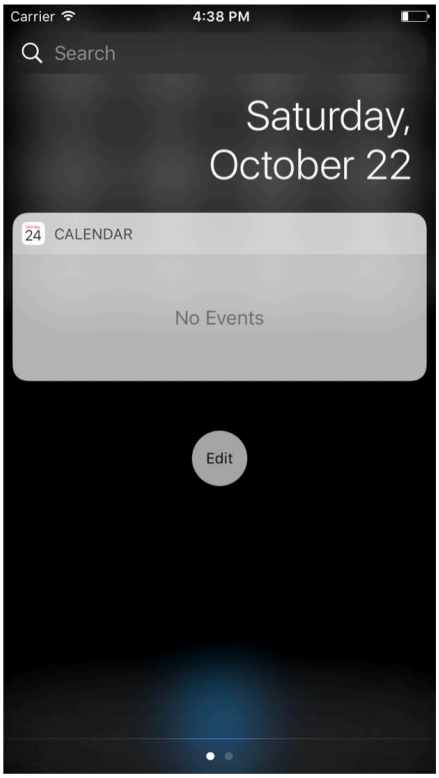
Blur App

Applying a Blurred Background Using UIVisualEffect



Jonathan Ive described the user interface redesign of iOS 7 more than 3 years ago. Other than "Flat" design, the mobile operating system introduced new types of depth.

One of the ways to achieve depth is to use layering and translucency in the view hierarchy. The use of blurred backgrounds could be found throughout the mobile operating system. For instance, when you swipe up the Control Center, its background is blurred. Moreover, the blurring effect is dynamic and keeps changing with respect to the background image. At that time, Apple did not provide APIs for developers to create such stunning visual effects. To replicate the effects, developers were required to create their own solutions or make use of the third-party libraries.



From iOS 8 and onwards, Apple opened the APIs and provided a new method which makes it very easy to create translucent and blurring-style effects. It introduced a new visual effect API that lets developers apply visual effects to a view. You can now easily add blurring effect to an existing image.

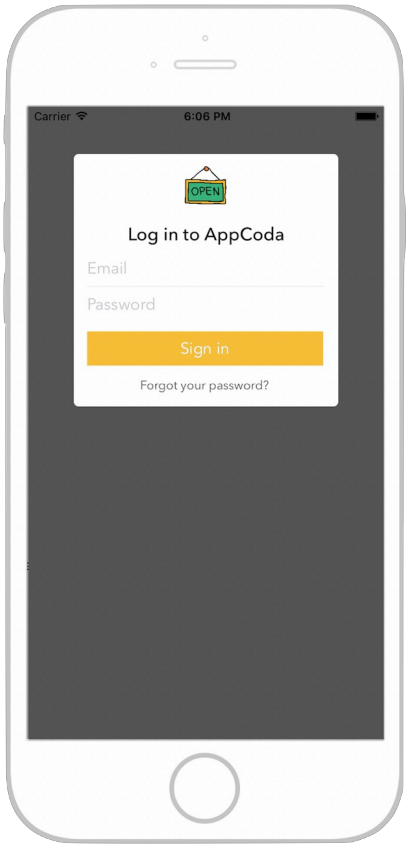
In this practice, we will go through the API by building a simple app to see how to apply the blurring effect.

The Demo App

The demo app is not fully functional and primarily used for demonstrating the blurring effect. It will work like this: when launched, it randomly picks an image from its image set. The selected image, with blurring effect applied, is used as a background image for the login screen.

We will focus on learning the UIVisualEffect API. Firstly, create the project with name BlurApp. It should look look like the screenshot shown below. It now only displays a background view in

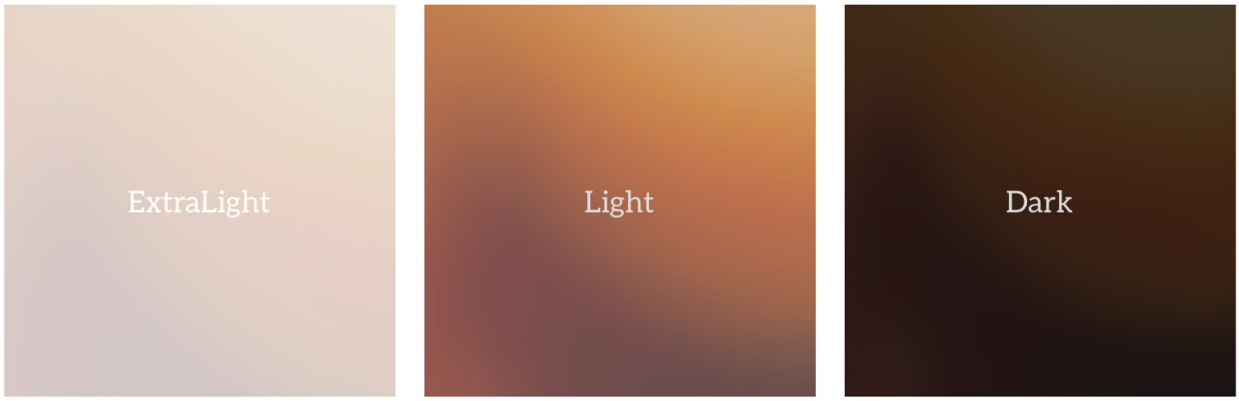
gray. Next up, we will change it to an image background with a live blurring effect.



Understanding UIVisualEffect and UIVisualEffectView

The iOS SDK provides two UIKit classes for applying visual effects:

- UIVisualEffect - There are only two kinds of visual effects including blur and vibrant. The UIBlurEffect class is used to apply a blurring effect to the content layered behind UIVisualEffectView . A blur effect comes with three types of style: ExtraLight, Light and Dark. The UIVibrantEffect class is designed for adjusting the color of the content, such that the element (e.g. label) inside a blurred view looks sharper.



- UIVisualEffectView - This is the view that actually applies the visual effect. The class takes in a UIVisualEffect object as a parameter. Depending on the parameter passed, it performs a blur or vibrant effect to the existing view.

To apply a blurring effect, you first create a UIBlurEffect object like this: `let blurEffect = UIBlurEffect(style: UIBlurEffectStyle.light)`

Here we create a blurring effect with a Light style. Once you have created the visual effect, you initialize a UIVisualEffectView object like this:

```
let blurEffectView = UIVisualEffectView(effect: blurEffect)
```

Suppose you have a UIImageView object serving as a background view; you can simply add the blurEffectView to the view hierarchy using the addSubView method and the background view will be blurred:

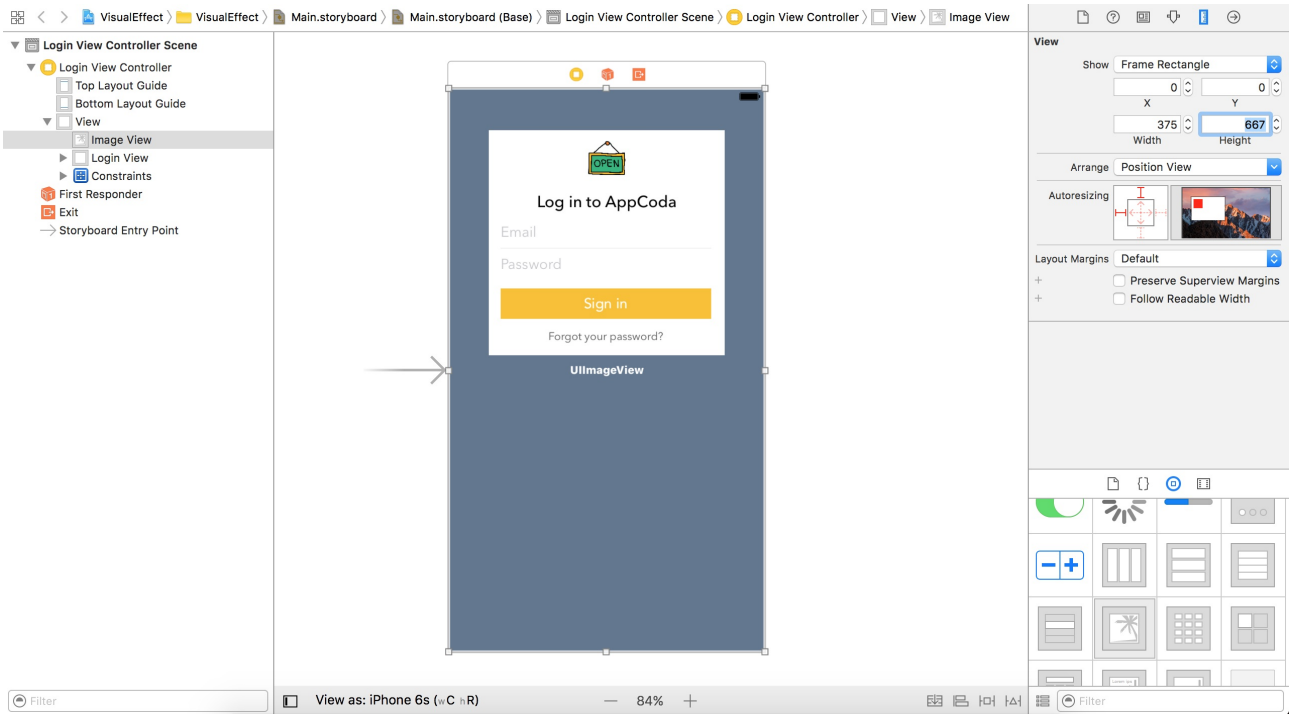
```
backgroundImageView.addSubview(blurEffectView)
```

Now that you have some ideas about the visual effect API, let's continue to work on the demo app.

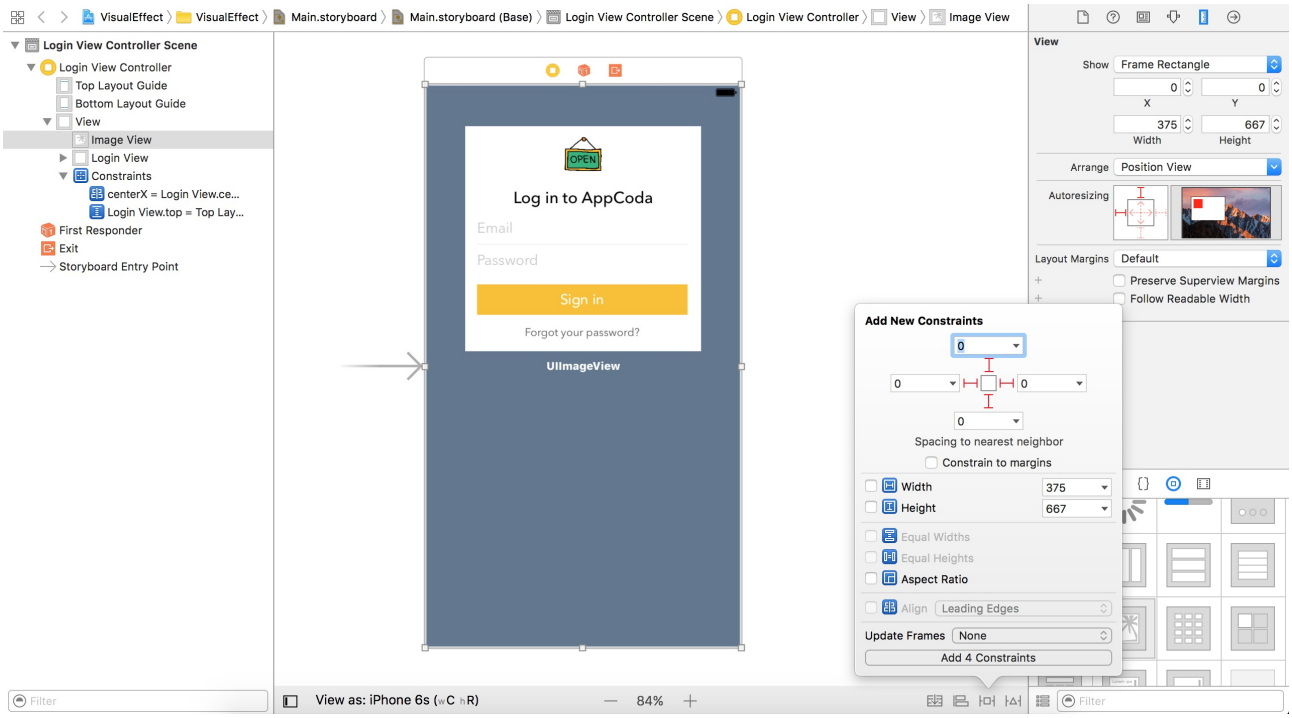
Adding a Background Image View

First, open Main.storyboard and make sure you use iPhone 6s as the device. Drag an image view to the view controller. As this image

view is used as a background image, make sure you put the image view beneath the login view (refer to the document outline). In the Size inspector, set the value of x and y to 0 , width to 375 and height to 667 .



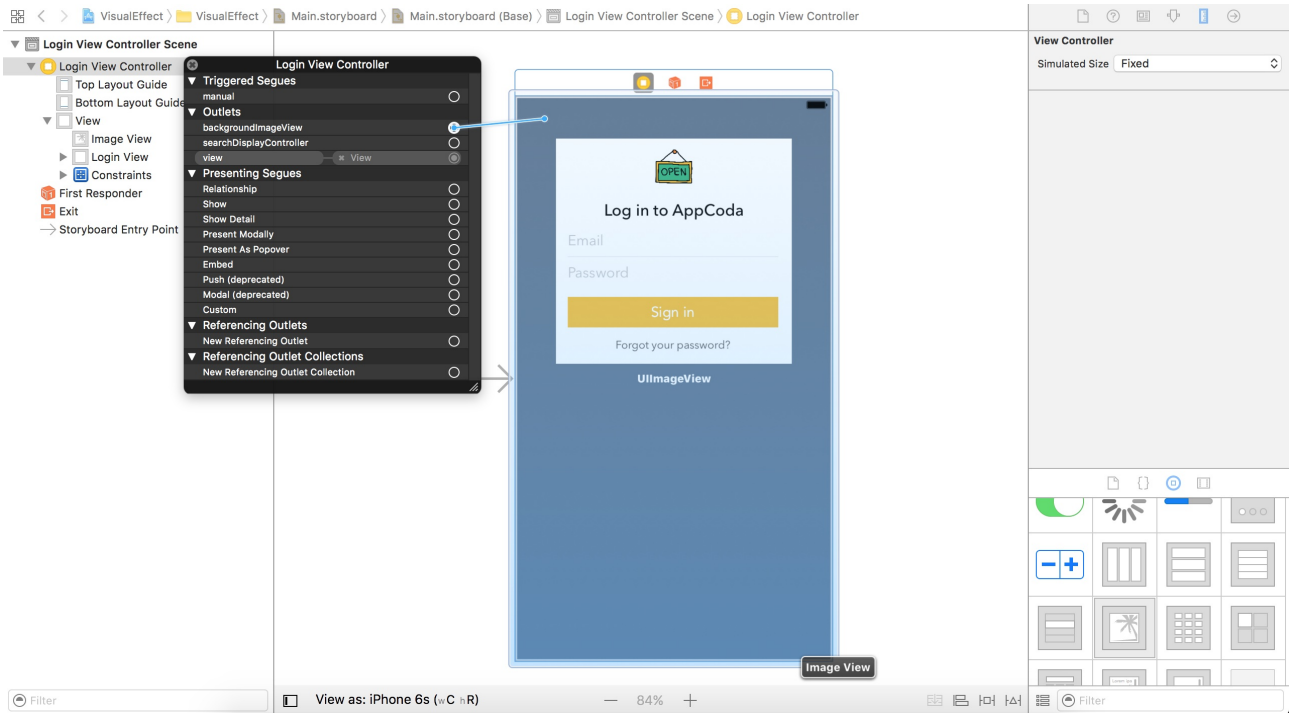
Once you have added the image view, select it and add the auto layout constraints. Click the pin button, and set the space value for each side to zero. Make sure you uncheck Constrain to margins and then click Add 4 constraints.



Next, go to the LoginViewController.swift file and add an outlet variable for the background image view:

```
@IBOutlet var backgroundImageView:UIImageView!
```

Now go back to the storyboard and establish a connection between the outlet variable and the image view.



Applying a Blurring Effect

Add to the project five background images. Every time the app is loaded, it will randomly pick one and use it as the background image. Now declare the following array in

LoginViewController.swift :

```
let imageSet = ["cloud", "coffee", "food", "pmq", "temple"]
```

Also declare a variable to hold the UIVisualEffectView object:

```
var blurEffectView: UIVisualEffectView?
```

Next, update the viewDidLoad method with the following code:

```
override func viewDidLoad() {

    super.viewDidLoad()
    // Randomly pick an image
    let selectedImageIndex = Int(arc4random_uniform(5))
    // Apply blurring effect
    backgroundImageView.image = UIImage(named: imageSet[selectedImageIndex])

    let blurEffect = UIBlurEffect(style: UIBlurEffectStyle.light)
    blurEffectView = UIVisualEffectView(effect: blurEffect)
    blurEffectView?.frame = view.bounds
    backgroundImageView.addSubview(blurEffectView!)
}
```

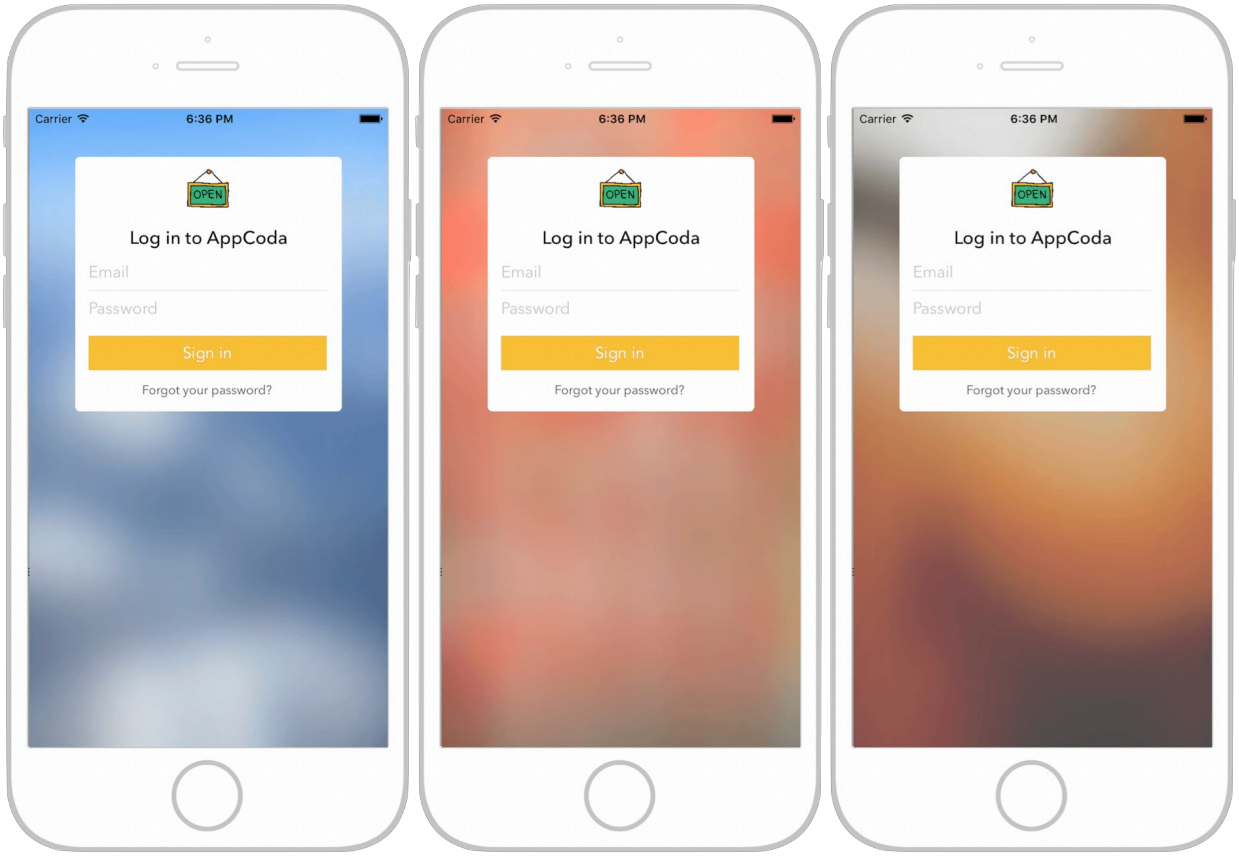
To pick an image randomly, we use the arc4random_uniform() function to generate a random number. By specifying 5 as the parameter, the function returns a value between 0 and 4. The rest of the code is similar to what we have discussed in the previous section. You are free to configure other blurring styles such as Dark.

To ensure the blur effect works in landscape mode, we have to update the frame property when the device's orientation changes. Insert the following method in the class:

```
override func traitCollectionDidChange(previousTraitCollection: UITraitCollection?) {
```

```
    blurEffectView?.frame = view.bounds
}
```

When the orientation is altered, the traitCollectionDidChange method will be called. We then update the frame property accordingly. Now run the project and see what you get. If you followed everything correctly, your app will display a blurred background.



***