CHAPTER 3

ANIMATING TABLE VIEW CELLS

ANIMATING TABLE VIEW CELL

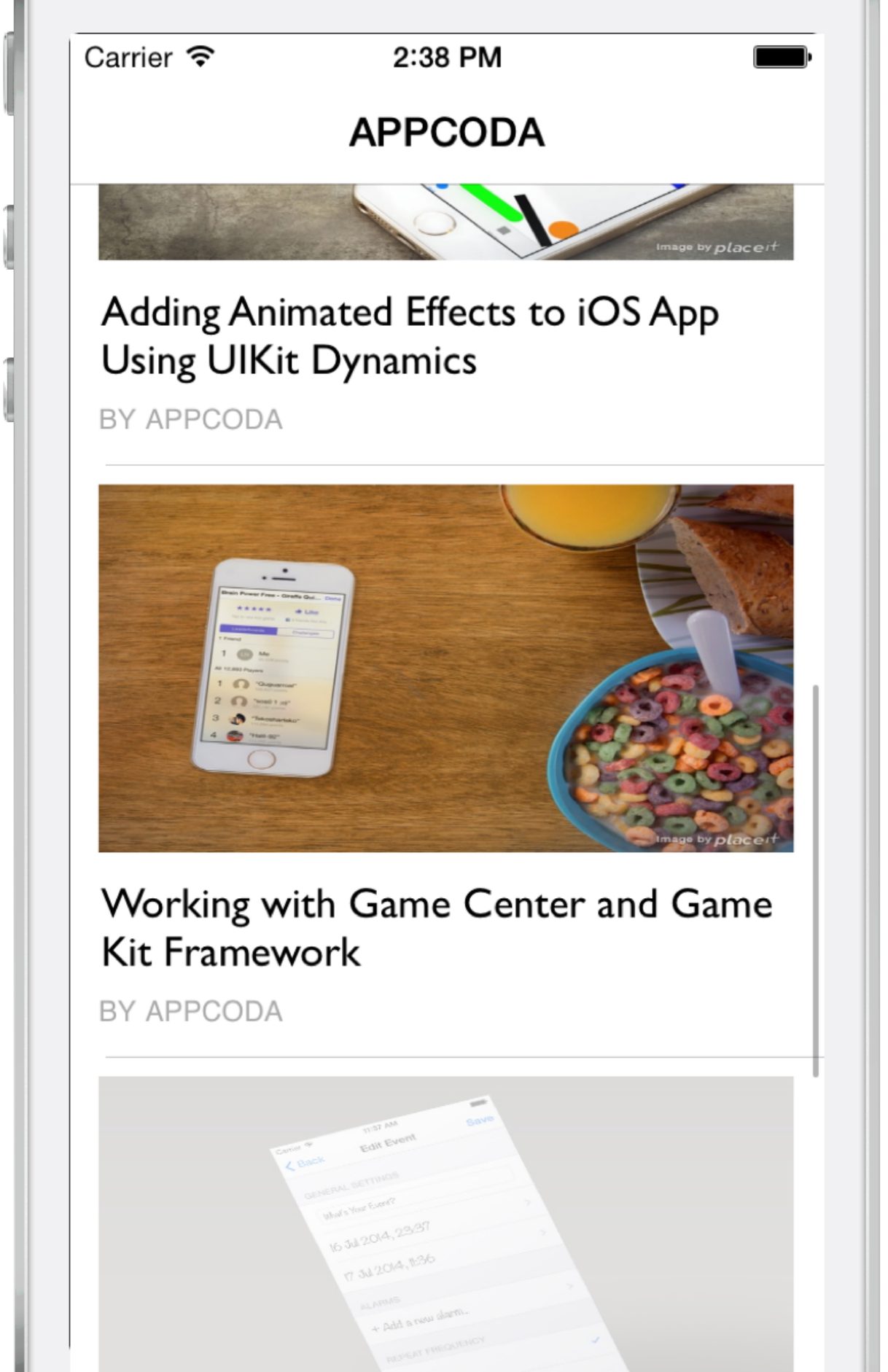
When you read this chapter, I believe you should already know how to create a basic UITableView to present your data. If not, go back and read the Beginning iOS 8 Programming with Swift book.

The UITableView provides a powerful way to present information in table form, and it is one of the most commonly used components in iOS apps. Whether you're building a productivity app, to-do app, or social app, you would make use of a table view in one form or another. The default implementation of UITableView is preliminary and only suitable for basic apps. To differentiate one's app from the rest, you usually customize the table view and table cell so as to make the app stand out. In this chapter, we'll show you another powerful way to liven up your app by adding subtle animation.

The Google+ app is a great example of table view animation. If you've used the app, every table row (or card) is animated as you scroll through the table. The card seems to slide in from the side when it first appears. This subtle animation would greatly enhance the user experience of your app.

It is very easy to animate a table view cell. Again, to demonstrate how the animation is done, we'll tweak an existing table-based app and add subtle animation.

To start with, first download this project template from https://www.dropbox.com/s/qnxqfwppra62mg8/TableCellAnimationTemplate.zip?dl=0. After downloading, compile the app and make sure you can run it properly. It's just a very simple app displaying a list of articles.



CREATING SIMPLE FADE-IN ANIMATION FOR TABLE VIEW CELL

Let's start tweaking the table-based app with a simple fade-in effect. So how can we add this subtle animation when the table row appears? If you look into the documentation of UITableViewDelegate protocol, you should find a method called *tableView(_:willDisplayCell:forRowAtIndexPath:)*:

```
optional func tableView(_ tableView: UITableView, willDisplayCell cell: UITableViewCell, forRowAtIndexPath indexPath: NSIndexPath)
```

The method will be called right before the row is drawn. Apparently, you can customize the cell object before it is displayed. Thus, to add our own animation, all we need to do is write the animation code in the method. Here is what you need to implement the fade-in effect. Insert the code snippet in this ArticleTableViewController.swift:

```
override func tableView(tableView: UITableView, willDisplayCell cell: UITableViewCell, forRowAtIndexPath indexPath: NSIndexPath) {

    // Define the initial state (Before the animation)
    cell.alpha = 0

    // Define the final state (After the animation)
    UIView.animateWithDuration(1.0, animations: { cell.alpha = 1 })
}
```

Core Animation provides iOS developers with an easy way to create animation. What we need to do is define the initial and final state of the visual element. Core Animation will automatically figure out the animation between these two states.

We first set the initial alpha value of the cell to 0, which represents total transparency. Then we begin the animation; set the duration to 1 second and define the final state of the cell, which is completely opaque. This will automatically create a fade-in effect when the table cell appears.

You can now compile and run the app. Scroll through the table view and enjoy the fade-in animation.

CREATING ROTATION EFFECT USING CATRANSFORM3D

Easy, right? With a few lines of code, your app looks a bit different than a standard table-based app. The *tableView(_:willDisplayCell:forRowAtIndexPath:)* method is the key to table view cell animation. You can implement whichever type of animation in the method. The fade-in animation is very simple. Now let's try to implement another animation using CATransform3D. Don't worry, you just need a few lines of code.

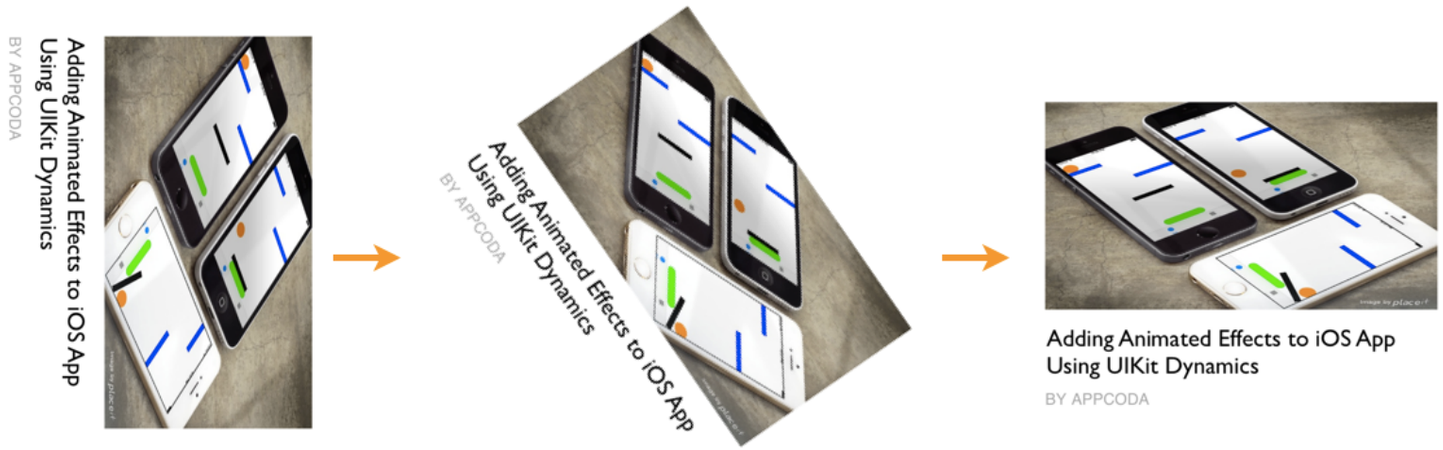
To add a rotation effect to the table cell, here is the code you need:

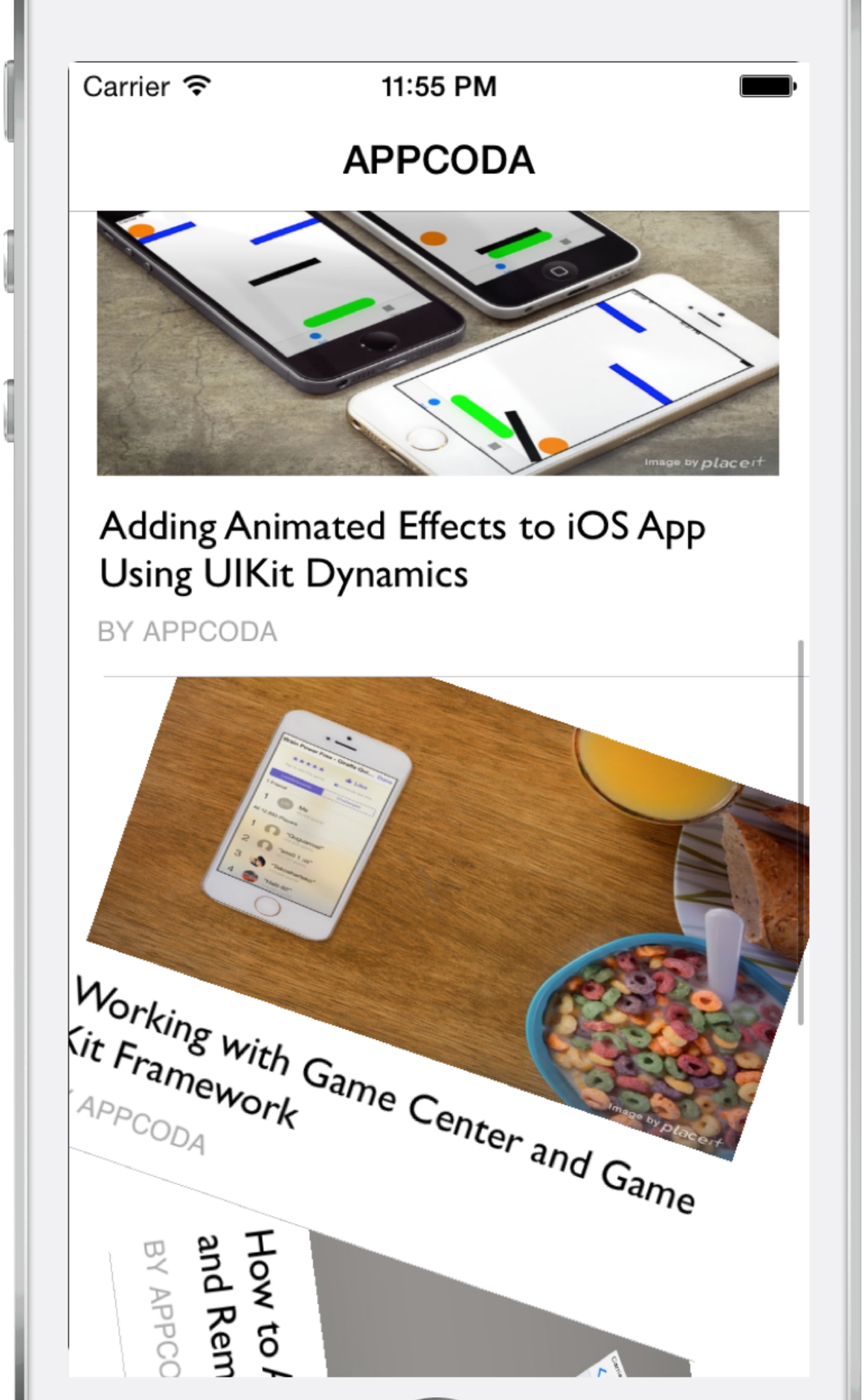
```
override func tableView(tableView: UITableView, willDisplayCell cell: UITableViewCell, forRowAtIndexPath indexPath: NSIndexPath) {

    // Define the initial state (Before the animation)
    let rotationAngleInRadians = 90.0 * CGFloat(M_PI/180.0);
    let rotationTransform = CATransform3DMakeRotation(rotationAngleInRadians, 0, 0, 1);
    cell.layer.transform = rotationTransform;

    // Define the final state (After the animation)
    UIView.animateWithDuration(1.0, animations: { cell.layer.transform = CATransform3DIdentity })

}
```

Same as before, we define the initial and final state of the transformation. The general idea is that we first rotate the cell by 90 degrees clockwise and then bring it back to the normal orientation which is the final state.

Okay, but how can we rotate a table cell by 90 degrees clockwise?

The key is to use the *CATransform3DMakeRotation* function to create the rotation transform. The function takes four parameters:





- Angle in radians - this is the angle of rotation. As the angle is in radian, we first need to convert the degrees into radians.
- X axis - this is the axis that goes from the left of the screen to the right of the screen.
- Y axis - this is the axis that goes from the top of the screen to the bottom of the screen.
- Z axis - this is the axis that points directly out of the screen.

Since the rotation is around the Z axis, we set the value of this parameter to 1, while leaving the value of the X axis and Y axis at zero. Once we create the transform, it is assigned to the cell's layer.

Next, we start the animation with the duration set to 1 second. The final state of the cell is set to *CATransform3DIdentity*. This will reset the cell to the original position.

Okay, hit Run to test the app!

Quick Tip

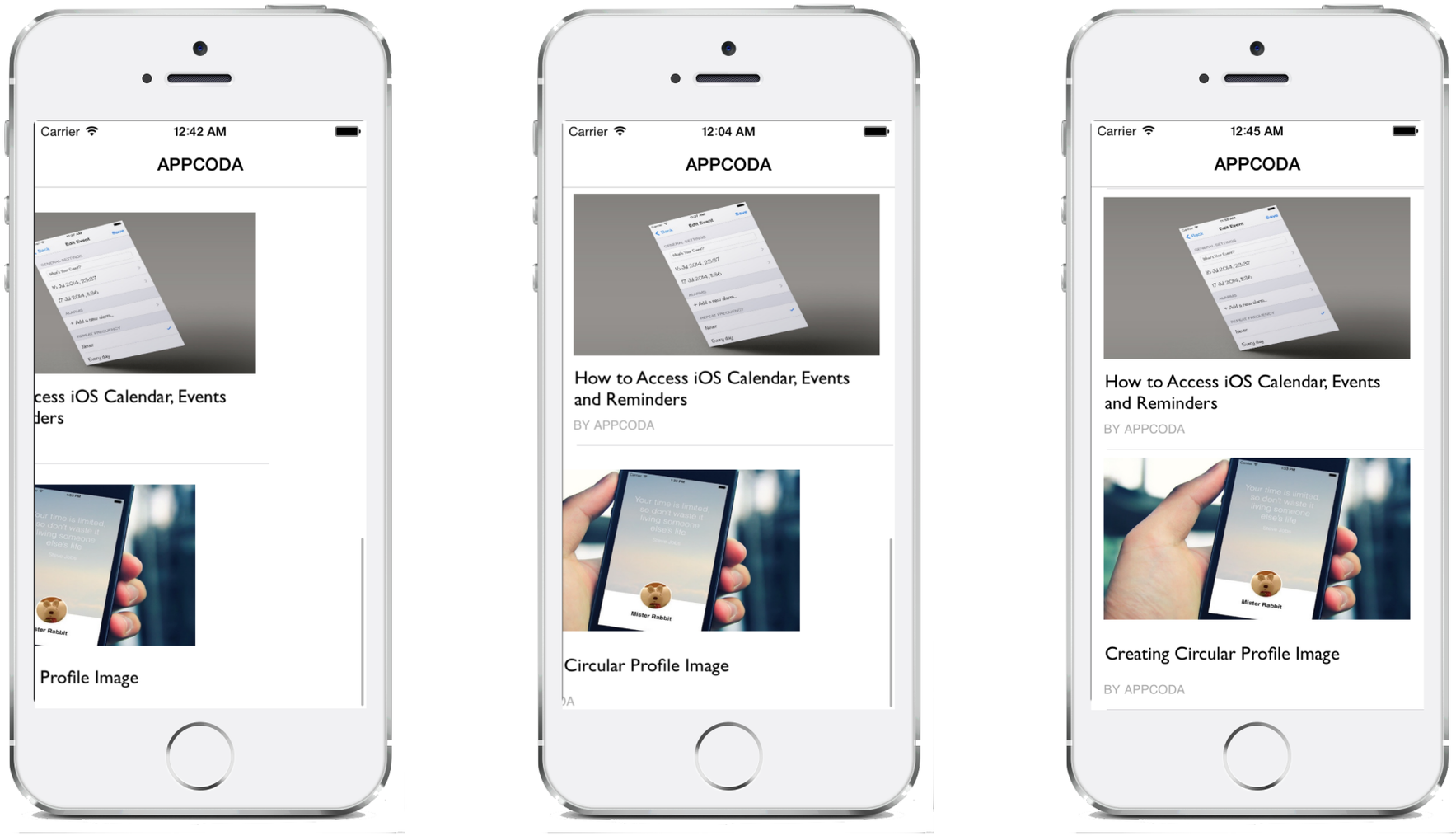
You may wonder what CATransform3D is. It is actually a structure representing a matrix. Performing transformation in 3D space such as rotation, involves some matrices calculation. I'll not go into the details of matrices calculation. If you want to learn more, you can check out http://www.matrix44.net/cms/notes/opengl-3d-graphics/

CREATING FLY-IN EFFECT USING CATRANSFORM3DTRANSLATE

Does the rotation effect look cool? You can further tweak the animation to make it even better. Try to change the *tableView(_:willDisplayCell:forRowAtIndexPath:)* method and replace the initialization of rotationTransform with the following line of code:

```
let rotationTransform = CATransform3DTranslate(CATransform3DIdentity, -500, 100, 0)
```

The line of code simply translates or shifts the position of the cell. It indicates the cell is shifted to the left (negative value) by 500 points and down (positive value) by 100 points. There is no change in the Z axis.



Now you're ready to test the app again. Hit the Run button and play around with the fly-in effect.

YOUR EXERCISE

For now, the cell animation is shown every time you scroll through the table, whether you're scrolling down or up the table view. Though the animation is nice, your user will find it annoying if the animation is displayed too frequently. You may want to display the animation only when the cell first appears. Try to modify the existing project and add that restriction.

SUMMARY

In this chapter, I just showed you the basics of table cell animation. Try to change the values of the transform and see what effects you get.

For your reference, you can download the complete Xcode project from https://www.dropbox.com/s/98s9wiwe38l81sc/TableCellAnimation.zip?dl=0. The solution for the exercise is included in the project.