Collection Views

Hands-On Challenges

Collection Views
Hands-On Challenges

Challenge D: Flying Cells

With the functionality to insert cells now complete, it's time to turn your attention to the stock insert animation. Currently cells simply appear, with the others around them moving to make way. This is nice, but you can do even better!

Your challenge this time is to create a subclass of UICollectionViewFlowLayout that animates the cells in from just underneath the Add button in navigation bar. They should start off the same size as the button, and grow to their final size as part of the animation.

Since this is quite challenging, here are some hints to help you on you way:

1. You'll need a way to inform the layout subclass of the index path of the item being inserted; you already have this information in MasterViewController.
2. In your layout subclass you'll need to override the method that deals with the initial layout attributes for appearing items.
3. Once you have the layout attributes of the item that's being inserted, you'll want to manipulate the `alpha`, `center`, `transform`, and `zIndex` properties to get the animation just right.

Before you turn the page for our solution, be sure to give it a try for yourself first!

Solution

Right-click on the **Layouts** group in the Project Navigator and select **New File...**. In the dialog box choose **iOS\Source\Swift File** and click Next. Name the file **PapersFlowLayout.swift** and click Create.

Open PapersFlowLayout.swift and replace the existing `import` statement with the following:

```
import UIKit
```

Below that, add the following class definition:

```
class PapersFlowLayout: UICollectionViewFlowLayout {

}
```

Next, you need to add property to hold the index path of the item that's being inserted. Add the following to the top of the class definition:

```
var appearingIndexPath: NSIndexPath?
```

This will be used to make sure you're adjusting the layout attributes of the correct item.

Now you need to override the method that provides the layout with the initial layout attributes for items being inserted into the collection view. Add the following to the class:

```
override func
initialLayoutAttributesForAppearingItemAtIndexPath(itemIndexPath:
NSIndexPath) -> UICollectionViewLayoutAttributes? {

}
```

Next, you need a set of layout attributes for the item at the given index path; you can ask the super implementation to provide them to you. Add the following to the top of the method:

```
let attributes =
    super.initialLayoutAttributesForAppearingItemAtIndexPath(
        itemIndexPath)
```

This will return the set of layout attributes that would have been passed to the collection view by the default UICollectionViewFlowLayout class. You're going to manipulate them if the index path matches the index path of the item being inserted.

Add the following just below the call to `super`:

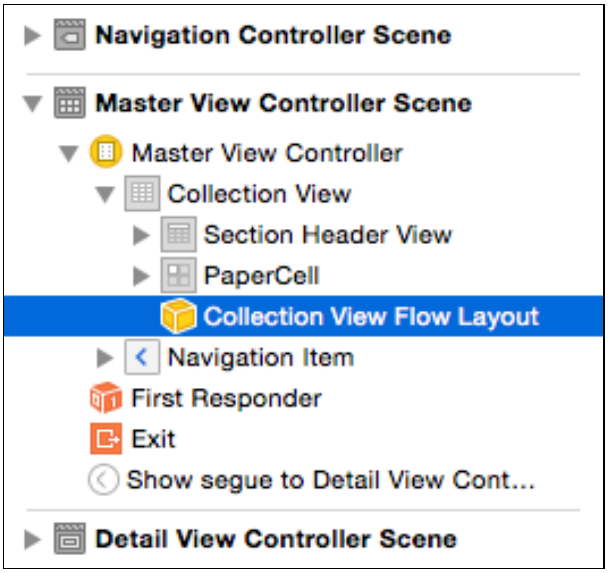
```
// 1
if let indexPath = appearingIndexPath {
  // 2
  if let attributes = attributes {
    // 3
    if indexPath == itemIndexPath {
      // 4
      attributes.alpha = 1.0
      attributes.center = CGPoint(x:
          CGRectGetWidth(collectionView!.frame) - 23.5, y: -24.5)
      attributes.transform = CGAffineTransformMakeScale(0.15,
          0.15)
      // 5
      attributes.zIndex = 99
    }
  }
}
return attributes
```

Here's what's happening:

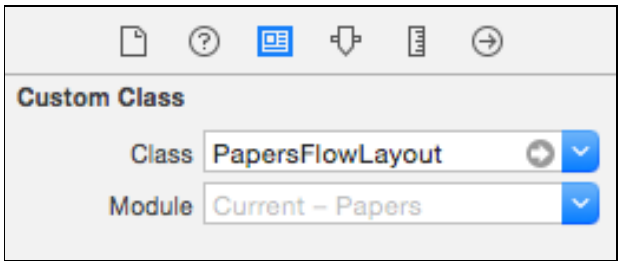
1. Since the `appearingIndexPath` property is an optional, you need to unwrap it
2. The call to `super` to get the initial layout attributes also returns an optional, so that too needs to be unwrapped
3. Check to see if the current index path is the same as the one being inserted
4. If it is, update the layout attributes so the cell is positioned and scaled accordingly
5. It's important the `zIndex` is set to something high so the cell will appear above all the other cells, which is key to this animation

Finally you return the layout attributes so they can be used by the collection view to place the cells accordingly. Remember, this animation will only be applied to items being inserted.

Now you need to tell the collection view to use your custom layout. Open **Main.storyboard**, and in the Document Outline for the **Master View Controller Scene** select **Collection View Flow Layout**:



Then in the Identity Inspector, set the **Class** in the Custom Class section to **PapersFlowLayout**:



You should notice the name of the object in the Document Outline change to **Papers Flow Layout**.

With the subclass created and the collection view updated to use it, the final step is to update MasterViewController.

Open **MasterViewController.swift** and find `addButtonTapped(_:)`. Add the following directly below the line where you create the `indexPath` constant:

```
let layout = collectionViewLayout as PapersFlowLayout
layout.appearingIndexPath = indexPath
```

Here you get a reference to the collection view layout, casting it to the flow layout subclass you just created. You then set the `appearingIndexPath` property to the index path of the item being inserted.

Finally, you need to reset the `appearingIndexPath` property once the animation is complete. Replace this line:

```
}, completion: nil)
```

With the following:

```
}, completion: { (finished: Bool) -> Void in
  layout.appearingIndexPath = nil
})
```

Here you're simply using the animation's completion block to reset the property.

Build and run. Tap the **Add** button and you'll see new papers fly down from underneath the Add button, growing in size until they find their final resting place.