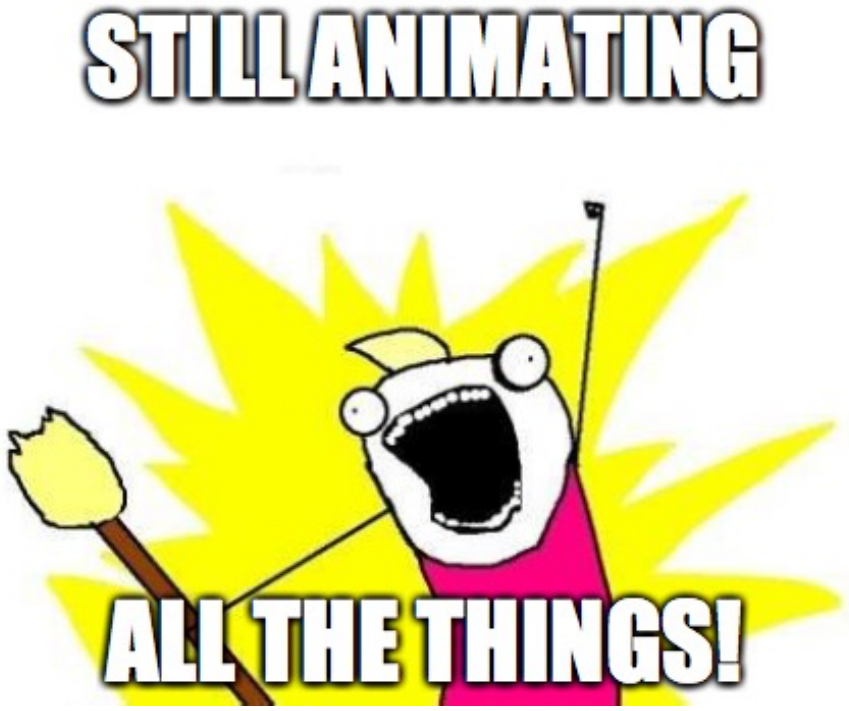
Collection Views

Hands-On Challenges

Collection Views
Hands-On Challenges

Challenge E: Shrinking Cells

So you can now delete cells, great job! But just as with the previous challenge, your work here is not yet finished. Wouldn't deleting cells look so much better if, instead of fading, they shrank down in-place, and then simply popped out-of-existence? Of course it would :]



Your challenge this time is to update your UICollectionViewFlowLayout subclass so that the cells being deleted don't move or fade, but simply shrink down until they've disappeared completely.

Again, this isn't as straightforward as it may first appear, so here are some hints to help you along your way:

1. You'll need a way to inform the layout subclass of the index paths of the items being deleted; you already have this information in MasterViewController. Remember though, you're dealing with multiple items this time around.
2. In your layout subclass you'll need to override the method that deals with the final layout attributes for disappearing items.
3. For this animation to work, you'll want to manipulate the `alpha`, `transform`, and `zIndex` properties of the layout attributes.

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

Solution

Open **PapersFlowLayout.swift** and add the following property just below the existing ones:

```
var disappearingItemsIndexPaths: [NSIndexPath]?
```

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

Now you need to override the method that provides the layout with the final layout attributes for items being removed from the collection view. Add the following to the class:

```
override func
finalLayoutAttributesForDisappearingItemAtIndexPath(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 for these. Add the following to the top of the method:

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

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

```
// 1
if let indexPaths = disappearingItemsIndexPaths {
  // 2
  if let attributes = attributes {
    // 3
    if contains(indexPaths, itemIndexPath) {
      // 4
      attributes.alpha = 1.0
      attributes.transform = CGAffineTransformMakeScale(0.1, 0.1)
      // 5
      attributes.zIndex = -1
    }
  }
}
return attributes
```

Here's what's happening:

1. Since the `disappearingItemsIndexPaths` 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 contained in the array of index paths that are being deleted
4. If it is, update the layout attributes accordingly
5. It's important the `zIndex` is set to something really low so the cell will appear below all the other cells, which is crucial to this animation

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

With the layout subclass updated, the final step is to update MasterViewController.

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

```
let layout = collectionViewLayout as PapersFlowLayout
layout.disappearingItemsIndexPaths = indexPaths
```

Here you get a reference to the collection view layout, casting it to the `PapersFlowLayout` class. You then set the `disappearingItemsIndexPaths` property to the index paths of the items being deleted.

Finally, replace this line:

```
collectionView!.deleteItemsAtIndexPaths(indexPaths)
```

With the following:

```
UIView.animateWithDuration(0.65, delay: 0.0, options:
    .CurveEaseInOut, animations: { () -> Void in
  self.collectionView!.deleteItemsAtIndexPaths(indexPaths)
}) { (finished: Bool) -> Void in
  layout.disappearingItemsIndexPaths = nil
}
```

Here you delete the items from collection view using an animation block, and then take advantage of the completion block to reset the `disappearingItemsIndexPaths` property to `nil`.

Build and run. Tap the **Edit** button and select a couple of wallpapers. Now tap the **Trash** icon and you'll see the selected wallpapers shrink down to nothing, and any around them will move over them into their new locations.