CUSTOM TRANSITIONS

WHAT WE'LL COVER

- Animated Transition Controller
  - `UIViewControllerAnimatedTransitioning`
- Transition Context
  - `UIViewControllerContextTransitioning`
- Transition Delegate
  - `UIViewControllerTransitioningDelegate`

TRANSITION CONTROLLER

TRANSITION CONTROLLER

- `UIViewControllerAnimatedTransition` Protocol
  - Defines your animator object
    - Animations for transitioning a view controller on/off screen
    - Fixed Time
  - Two required methods
    - `- transitionDuration(using:) -> NSTimeInterval`
    - `- animateTransition(using:)`

TRANSITIONING CONTEXT

TRANSITIONING CONTEXT

- Animator objects involved receive a fully configured context object
  - `UIViewControllerContextTransitioning`
- Encapsulates information about:
  - The views and view controllers involved
    - `containerView: UIView`
    - `- view(forKey:)`
    - `- viewController(forKey:)`
  - Details about how to execute the transition

IMPLEMENTING ANIMATOR OBJECTS

1. Get animation parameters

2. Create animations using `CoreAnimation` or `UIKit` methods

3. Clean up and complete the transition

ANIMATOR OBJECT CODE

```
import UIKit

class FlipPresentAnimatedTransitionController: NSObject, UIViewControllerAnimatedTransitioning {

    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
         TimeInterval { return 0.75 }

    func animateTransition(using transitionContext: UIViewControllerContextTransitioning){...}
}
```

ANIMATOR OBJECT CODE (CONT'D)

```
guard let collectionVC = transitionContext.viewController(forKey: .from) as? FlipCollectionVC,
    let detailVC = transitionContext.viewController(forKey: .to) as? DetailVC,
    ...
    let cell = collectionView.cellForItem(at: selectedIndexPath),
    let detailViewSnapshot = detailViewController.view.snapshotView(afterScreenUpdates: true)
else { return }

    let cellFrame = collectionView.convert(cell.frame, to: collectionViewController.view)
    detailViewSnapshot.frame = cellFrame

    let containerView = transitionContext.containerView
    containerView.addSubview(detailViewController.view)
    containerView.addSubview(detailViewSnapshot)
    containerView.backgroundColor = collectionView.backgroundColor
    detailViewController.view.isHidden = true

    detailViewSnapshot.layer.transform = CATransform3D.makeYRotation(90.0)

    let duration = transitionDuration(using: transitionContext)
})
```

ANIMATOR OBJECT CODE (CONT'D)

```
let keyframeAnimation0 = {
collectionViewController.view.layer.transform = CATransform3D.makeYRotation(-90.0)
}
let keyframeAnimation1 = {
detailViewSnapshot.layer.transform = CATransform3D.makeYRotation(0.0)
}
let keyframeAnimation2 = {
detailViewSnapshot.frame = transitionContext.finalFrame(for: detailViewController)
}

animateTransitionEvenly(with: duration,
          keyframeAnimations: [keyframeAnimation0,
                               keyframeAnimation1,
                               keyframeAnimation2],
                  completion: { (complete: Bool) in
                      detailViewController.view.isHidden = false
                      detailViewSnapshot.removeFromSuperview()
                      transitionContext.completeTransition(complete) })
```

TRANSITION DELEGATE

TRANSITIONING DELEGATE

- `UIViewControllerTransitioningDelegate`
- Creates and returns your custom animator objects
  - Based on presenting or dismissing
  - `- animationController(forPresented:presenting:source:) -> UIViewControllerAnimatedTransitioning?`
  - `- animationController(forDismissed:) -> UIViewControllerAnimatedTransitioning?`

TRANSITIONING DELEGATE CODE

```
class DetailViewController: UIViewController, UIViewControllerTransitioningDelegate {

    …

    func animationController(forPresented presented: UIViewController,
                                        presenting: UIViewController,
                                            source: UIViewController)
                                                 -> UIViewControllerAnimatedTransitioning? {
        return FlipPresentAnimatedTransitionController()
    }

    func animationController(forDismissed dismissed: UIViewController)
                                                 -> UIViewControllerAnimatedTransitioning? {
        return FlipDismissAnimatedTransitionController()
    }

    @IBAction func tapGestureRecognizerActivated() {
        dismiss(animated: true, completion: nil)
    }
}
```

SETTING UP SEGUE

- Set up a modal style segue
- Set the `transitionDelegate` property of your destination view controller
- Set up rest of segue as you've previously done

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "toDetail" {
        guard let destinationVC = segue.destination as? DetailViewController,
            let collectionView = collectionView,
            let selectedIndexPaths = collectionView.indexPathsForSelectedItems,
            let selectedIndexPath = selectedIndexPaths.first else { return }

        destinationVC.color = colors[selectedIndexPath.item]
        destinationVC.transitioningDelegate = destinationVC
    }
}
```

QUESTIONS?