// assign a new transition style to a view controller using the modalTransitionStyle property

// An interaction transition is an animated view controller transition whose progress can be controlled by the user, usually by panning

Transitioning Delegate → Animator
Transitioning Delegate → Interactive Animator
Transitioning Delegate → Presentation Controller

// Custom Modal Transitions

// ViewController.swift

```
import UIKit

class ViewController: UIViewController{

    override func viewDidLoad(){
        super.viewDidLoad()
    }
    @IBAction func showNewVC(_ sender:Any){
        performSegue(withIdentifier: "ShowNewVC", sender: nil)
    }

    override func prepare(for segue: UIStoryboardSegue, sender:Any?){
        if let destinationVC = segue.destination as? NewViewController{
            destinationVC.transitioningDelegate = self.

        }
    }

}

extension ViewController: UIViewControllerTransitioningDelegate{
    func animationController(forPresented presented: UIViewController, presenting: UIViewController, source: UIViewController) ->
        UIViewControllerAnimatedTransitioning?{
        return PresentAnimator()
    }

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

}
```

```
// PresentAnimator.swift
import UIKit

class PresentAnimator: NSObject, UIViewControllerAnimatedTransitioning {
    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) -> TimeInterval {
        return 0.6
    }
    func animateTransition(using transitionContext: UIViewControllerContextTransitioning) {
        guard let fromView = transitionContext.view(forKey: .from) else { return }
        guard let toView = transitionContext.view(forKey: .to) else { return }
        toView.transform = CGAffineTransform(scaleX: 0, y: 0)
        toView.alpha = 0
        transitionContext.containerView.addSubview(fromView)
        transitionContext.containerView.addSubview(toView)
        UIView.animate(withDuration: transitionDuration(using: transitionContext), animations: {
            toView.transform = CGAffineTransform.identity
            toView.alpha = 1.0
        }) { (sucess) in
            transitionContext.completeTransition(true)
        }
    }
}


// DismissAnimator.swift
import UIKit

class DismissAnimator: NSObject, UIViewControllerAnimatedTransitioning {
    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) -> TimeInterval {
        return 0.6
    }
    func animateTransition(using transitionContext: UIViewControllerContextTransitioning) {
        guard let fromView = transitionContext.view(forKey: .from) else { return }
        guard let toView = transitionContext.view(forKey: .to) else { return }

        transitionContext.containerView.addSubview(toView)
        transitionContext.containerView.bringSubviewToFront(fromView)
        UIView.animate(withDuration: transitionDuration(using: transitionContext), animations: {
            fromView.transform = CGAffineTransform(scaleX: 0.1, y: 0.1)
            fromView.alpha = 0
        }) { (success) in
            transitionContext.completeTransition(true)
        }
    }
}

// NewViewController.swift  => create an IBAction to dismiss the view
```

11 Circular Modal Transition

// ViewController.swift

```
import UIKit

class ViewController: UIViewController{

    @IBOutlet weak var startButton: UIButton!

    override func viewDidLoad(){
        super.viewDidLoad()
    }

    override func viewDidLayoutSubviews(){
        super.viewDidLayoutSubviews()
        startButton.layer.cornerRadius = startButton.frame.width / CGFloat(2)
    }

    @IBAction func showNewVC(_ sender: Any){
        performSegue(withIdentifier: "ShowNewVC", sender: nil)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?){
        if let destinationVC = segue.destination as? NewViewController{
            destinationVC.transitioningDelegate = self
        }
    }
}

extension ViewController: UIViewControllerTransitioningDelegate{
    func animationController(forPresented presented: UIViewController, presenting: UIViewController, source: UIViewController) ->
    UIViewControllerAnimatedTransitioning?{
        let animator = PresentAnimator(buttonFrame: startButton.frame)
        return animator
    }

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

}
```

```
// PresentAnimator.swift
import UIKit
class PresentAnimator: NSObject, UIViewControllerAnimatedTransitioning {
    var buttonFrame: CGRect
    //var transitionContext: UIViewControllerContextTransitioning?
    init(buttonFrame: CGRect) {
        self.buttonFrame = buttonFrame
    }
    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) -> TimeInterval {
        return 0.6
    }
    func animateTransition(using transitionContext: UIViewControllerContextTransitioning) {
        // use guard in real apps
        let fromViewController: ViewController = transitionContext.viewController(forKey: .from) as! ViewController
        let destinationController = transitionContext.viewController(forKey: .to)!
        let destinationView = destinationController.view!
        transitionContext.containerView.addSubview(fromViewController.view)
        //transitionContext.containerView.addSubview(destinationView)
        self.transitionContext = transitionContext
        let maskPath = UIBezierPath(ovalIn: buttonFrame)
        let maskLayer = CAShapeLayer()
        maskLayer.frame = destinationView.frame
        maskLayer.path = maskPath.cgPath
        destinationController.view.layer.mask = maskLayer
        let screenHeight = UIScreen.main.bounds.height
        let screenYHeight = screenHeight - buttonFrame.midX
        let buttonYHeight = buttonFrame.height/2
        let scaleFactor = screenYHeight / buttonYHeight
        let endFrameWidth = scaleFactor * buttonFrame.width
        let endFrameHeight = endFrameWidth
        let endFrameXPos = destinationController.view.frame.midx - (endFrameWidth/2)
        let endFrameYPos = CGFloat(0)
        let endFrame = CGRect(x: endFrameXPos, y: endFrameYPos, width: endFrameWidth, height: endFrameHeight)
        let bigCirclePath = UIBezierPath(ovalIn: endFrame)
        let pathAnimation = CABasicAnimation(keyPath: "path")
        pathAnimation.setValue(transitionContext, forKey: "transitionContext")
        pathAnimation.setValue(destinationController, forKey: "destinationVC")

        pathAnimation.delegate = self
        pathAnimation.fromValue = maskPath.cgPath
        pathAnimation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionEaseIn)
        pathAnimation.toValue = bigCirclePath.cgPath
        maskLayer.path = bigCirclePath.cgPath

        maskLayer.add(pathAnimation, forKey: nil)
    }
}
```

```
extension PresentAnimator : CAAnimationDelegate{
    func animationDidStop(_ anim: CAAnimation, finished flag: Bool){
        // if let _transitionContext = transitionContext, flag {
        //     _transitionContext.completeTransition(!_transitionContext.transitionWasCancelled)
        // }
        if let transitionContext = anime.value(forKey: "transitionContext") as? UIViewControllerContextTransitioning,
            let destinationVC = anim.value(forKey: "destinationVC") as? NewViewController, flag{
            transitionContext.completeTransition(!transitionContext.transitionWasCancelled)
            destination.view.layer.mask?.removeFromSuperlayer()
        }

    }

}
```

Custom Navigation Transition

embeded in Navigation Controller and show

```
// ViewController.swift
import UIKit

class ViewController: UIViewController{

    override func viewDidLoad(){
        super.viewDidLoad()
        navigationController?.delegate = self

    }

    @IBAction func pushVC(_ sender: Any){
        performSegue(withIdentifier: "NewVCSegue", sender: nil)
    }
}

extension ViewController: UINavigationControllerDelegate{
    func navigationController(_ navigationController: UINavigationController, animationControllerFor operation:
        UINavigationController.Operation, from fromVC: UIViewController, to toVC: UIViewController) ->
        UIViewControllerAnimatedTransitioning?{
        switch operation{
            case .push:
                return PresentAnimator()
            case .pop:
                return nil
            default:
                return nil
        }
    }
}
```

// PresentAnimator.swift

```
import UIKit

class PresentAnimator: NSObject, UIViewControllerAnimatedTransitioning{
    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) -> TimeInterval{
        return 0.4
    }

    func animateTransition(using transitionContext: UIViewControllerContextTransitioning){
        guard let toViewController = transitionContext.viewController(forKey: .to) else {return}
        transitionContext.containerView.addSubview(toViewController.view)
        let currentPos = toViewController.view.frame.origin
        toViewController.view.frame.origin = CGPoint(x: currentPos.x + toViewController.view.frame.width, y: currentPos.y)
        let duration = self.transitionDuration(using: transitionContext)
        UIView.animate(withDuration: duration, delay: 0, usingSpringWithDamping: 0.4, initialSpringVelocity: 2, options: [.curveEaseOut], animations: {
            toViewController.view.frame.origin = CGPoint(x: currentPos.x, y: currentPos.y)

        }) { (success) in
            transitionContext.completeTransition(!transitionContext.transitionWasCancelled)
        }
    }
}
```

// Photo Album Transition

embed in navigation controller, setup segue, collection view (do not select constrain to margins)

// ViewController.swift

```
import UIKit

class ViewController: UIViewController, UICollectionViewDelegate, UICollectionViewDataSource, UICollectionViewDelegateFlowLayout{
    @IBOutlet weak var collectionView: UICollectionView!

    let dataCount = 12
    var selectedCell = UICollectionViewCell()
    var offset: CGFloat = 0

    override func viewDidLoad(){
        super.viewDidLoad()
        collectionView.delegate = self
        collectionView.dataSource = self
        self.navigationController?.delegate = self

        offset = self.navigationController!.navigationBar.frame.height + UIApplication.shared.statusBarFrame.size.height

    }
```

```
func numberOfSections(in collectionView: UICollectionView) -> Int {
    return 1
}

func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {
    let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "imageCell", for: indexPath) as! ImageCollectionViewCell
    return cell
}

func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath) {
    selectedCell = collectionView.cellForItem(at: indexPath)!
    performSegue(withIdentifier: "NewVCSegue", sender: nil)
}

func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
    return dataCount
}

func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt
    indexPath: IndexPath) -> CGSize {
    let itemsPerRow: CGFloat = 3
    let hardcodedPadding: CGFloat = 2
    let itemWidth = (collectionView.bounds.width / itemsPerRow) - hardcodedPadding
    let itemHeight = itemWidth
    return CGSize(width: itemWidth, height: itemHeight)
}
}
```

check section insets all zeros

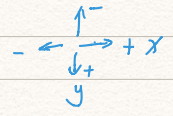
```
extension ViewController: UINavigationControllerDelegate {
    func navigationController(_ navigationController: UINavigationController, animationControllerFor operation:
    UINavigationController.Operation, from fromVC: UIViewController, to toVC: UIViewController) ->
    UIViewControllerAnimatedTransitioning? {
        switch operation {
        case .pop:
            let animator = DismissAnimator()
            animator.originFrame = selectedCell.frame
            animator.offset = offset
            return animator

        case .push:
            let animator = PresentAnimator()
            animator.originFrame = selectedCell.frame
            animator.offset = offset
            return animator
        default:

        }
    }
}
```

```
// PresentAnimator.swift

import UIKit

class PresentAnimator: NSObject, UIViewControllerAnimatedTransitioning {
    var originFrame: CGRect = CGRect.zero
    var offset: CGFloat = 0
    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) -> TimeInterval {
        return 0.4
    }
    func animateTransition(using transitionContext: UIViewControllerContextTransitioning) {
        let containerView = transitionContext.containerView
        guard let detailView = transitionContext.viewController(forKey: .to) else { return }
        let finalFrame = detailView.frame
        var initialFrame = originFrame
        let detailAspectRatio = finalFrame.width / finalFrame.height
        initialFrame.size = CGSize(width: initialFrame.width, height: initialFrame.width * detailAspectRatio)
        let scaleFactor = finalFrame.width / initialFrame.width
        let growScaleFactor = scaleFactor
        let shrinkScaleFactor = 1/growScaleFactor
        detailView.transform = CGAffineTransform(scaleX: shrinkScaleFactor, y: shrinkScaleFactor)
        detailView.center = CGPoint(x: originFrame.midX, y: originFrame.midY + offset)
        detailView.clipsToBounds = true
        containerView.addSubview(detailView)
        UIView.animate(withDuration: transitionDuration(using: transitionContext), delay: 0, usingSpringWithDamping: 0.7,
            initialSpringVelocity: 1, options: [.curveEaseIn], animations: {
            detailView.transform = CGAffineTransform.identity
            detailView.center = CGPoint(x: finalFrame.midX, y: finalFrame.midY)
        }) { (success) in
            transitionContext.completeTransition(!transitionContext.transitionWasCancelled)
        }
    }
}

// DismissAnimator.swift
import UIKit

class PresentAnimator: NSObject, UIViewControllerAnimatedTransitioning {
    var originFrame: CGRect = CGRect.zero
    var offset: CGFloat = 0
    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) -> TimeInterval {
        return 0.2
    }
    func animateTransition(using transitionContext: UIViewControllerContextTransitioning) {
        let containerView = transitionContext.containerView
        guard let detailView = transitionContext.viewController(forKey: .from) else { return }
        guard let thumbnailView = transitionContext.viewController(forKey: .to) else { return }
        let finalFrame = originFrame
```

```
        let initialFrame = detailView.frame
        let scaleFactor = initialFrame.width / finalFrame.width
        let growScaleFactor = scaleFactor
        let shrinkScaleFactor = 1/growScaleFactor
        detailView.clipsToBounds = true
        containerView.addSubview(thumbnailView)
        containerView.bringSubviewToFront(detailView)
        detailView.backgroundColor = UIColor.clear
        UIView.animate(withDuration: transitionDuration(using: transitionContext), animations: {
            detailView.transform = CGAffineTransform(scaleX: shrinkScaleFactor, y: shrinkScaleFactor)
            detailView.center = CGPoint(x: finalFrame.midX, y: finalFrame.midY + self.offset)

        }){ (sucess) in
            transitionContext.completeTransition(!transitionContext.transitionWasCancelled)
        }
    }
}
```

Pan Gesture Recognizers

```
import UIKit

class ViewController: UIViewController{
    @IBOutlet var panRecognizer: UIPanGestureRecognizer!

    override func viewDidLoad(){
        super.viewDidLoad()

    }

    @IBAction func panTouched(_ sender: UIPanGestureRecognizer){
        switch sender.state{
        case .began:

        case .changed:
            let translation = panRecognizer.translation(in: view)
            if translation.x > 0{

            }
            let velocity = panRecognizer.velocity(in: view)
            if velocity.x > 300{

            }

        case .ended:

        default:

        }
    }
}
```



vector ⇒ scale and direction

velocity

```
import UIKit

class ViewController : UIViewController{

    lazy var practiceView : UIView = {
        let _practiceView = UIView()
        _practiceView.backgroundColor = UIColor.orange
        return _practiceView
    }()

    override func viewDidLoad(){
        super.viewDidLoad()
        view.addSubView(practiceView)
        practiceView.frame = CGRect(x: view.frame.midX-60, y: view.frame.midY-60, width:120, height:120)
        let gesture = UIPanGestureRecognizer(target: self, action: #selector(panTouched(recognizer:)))
        practiceView.addGestureRecognizer(gesture)
        practiceView.isUserInteractionEnabled = true
    }

    @objc func panTouched(recognizer: UIPanGestureRecognizer){
        let translation = recognizer.translation(in: view)
        practiceView.center = CGPoint(x: practiceView.center.x + translation.x, y: practiceView.center.y + translation.y)
        recognizer.setTranslation(CGPoint.zero, in: view)
    }

}
```