

CAlayer

view.layer.backgroundColor = UIColor.blue.cgColor
 \nearrow core graph colors
 view.backgroundColor = UIColor.blue \nearrow equivalent each other
 view.layer.cornerRadius = CGFloat(10.0)
 view.layer.borderWidth = CGFloat(15.0)
 view.layer.borderColor = UIColor.orange.cgColor
 view.layer.shadowOpacity = 0.7
 view.layer.shadowRadius = CGFloat(15.0)
 view.layer.shadowOffset = CGSize(width: CGFloat(-15), height: CGFloat(15))
 view.layer.shadowOffset = CGSize.zero
 view.layer.contents = UIImage(named: "cat")?.cgImage
 view.layer.masksToBounds = true // will also clip the shadow
 view.layer.contentsGravity = CAlayerContentsGravity.center // center the image

// stand alone layer

```

lazy var newlayer: CALayer = {
  return CALayer()
}
  
```

viewDidLayout

:

func setup

view.layer.addSublayer(newlayer)

```

override func viewDidLayoutSubviews() {
  super.viewDidLayoutSubviews()
  newlayer.frame = view.bounds
}
  
```

3

// Gradients with CA Gradient Layer

import UIKit

class ViewController: UIViewController {

```
    lay var gradientLayer: CAGradientLayer = {}  
    return CAGradientLayer()  
}
```

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

```
override func viewDidLayoutSubviews() {  
    super.viewDidLayoutSubviews()  
    gradientLayer.frame = view.bounds  
}
```

```
func setGradientLayer() {  
    gradientLayer.colors = [UIColor.red.cgColor, UIColor.green.cgColor]  
    gradientLayer.startPoint = CGPoint(x: 0.0, y: 0.0)  
}
```

```
gradientLayer.endPoint = CGPoint(x: 1.0, y: 1.0)
```

```
view.layer.addSublayer(gradientLayer)
```

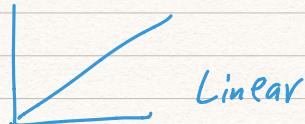
}

→ 100% not target.cgColor ←

// Animation

time + pacing + Spring

Timing functions → CAMediaTiming function



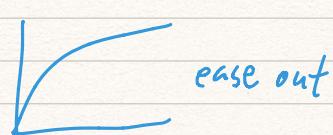
Linear



ease in



ease In Out



ease out

UI View Animation

import UIKit

```
class ViewController : UIViewController {
    @IBOutlet weak var block: UIView!
    override func viewDidLoad() {
        super.viewDidLoad()
        let tap = UITapGestureRecognizer(target: self, action: #selector(blockTapped))
        block.addGestureRecognizer(tap)
        block.isUserInteractionEnabled = true
    }
}
```

```
@objc func blockTapped() {
    UIView.animate(withDuration: 1.2) {
        let sizeMultiplier: CGFloat = 1.5
        //let currentBlockFrame = self.block.frame
        //self.block.frame.size = CGSize(width: currentBlockFrame.width * sizeMultiplier,
        //                                height: currentBlockFrame.height * sizeMultiplier)
        let newWidth = self.block.frame.width * sizeMultiplier
        let newHeight = self.block.frame.height * sizeMultiplier
        let newX = self.block.frame.origin.x - (newWidth - self.block.frame.size.width)/2
        let newY = self.block.frame.origin.y - (newHeight - self.block.frame.size.height)/2
        self.block.frame = CGRect(x: newX, y: newY, width: newWidth, height: newHeight)
    }
}
```

How to chain animations

import UIKit

```
class ViewController : UIViewController {
    @IBOutlet weak var block: UIView!
    override func viewDidLoad() {
        super.viewDidLoad()
    }
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        DispatchQueue.main.asyncAfter(deadline: .now() + .seconds(2)) {
            self.upMove()
        }
    }
}
```

```
func upMove() { → by default ⇒ ease in, ease out
    UIView.animate(withDuration: 1, animations: {
        self.block.frame.origin = CGPoint(x: self.view.center.x - (self.block.frame.width/2), y: 0)
    }) { (success) in
        self.rightMove
    }
}
```

func rightMove()\{

3

11 thinning function

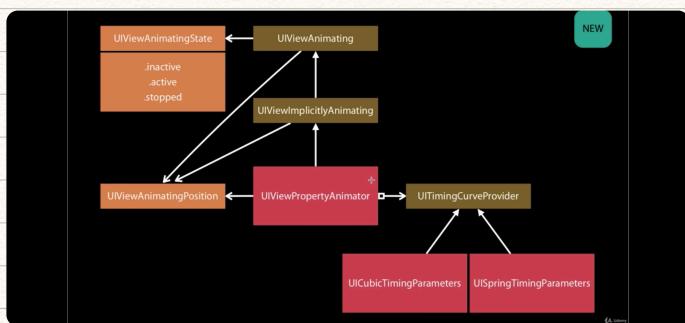
```
U2View.animateWithDuration:2,delay:2,options:[.curveEaseIn, animations:{[...]
```

11 Spring

ω_1 over shoot most

U2 View.animate(withDuration: 2, delay: 2, usingSpringWithDamping: 0.5, initialSpringVelocity: 4, options: .curveEaseIn, animations: { [unowned self] in self.startAnimation() }, completion: { _ in })

Property Animation



Advantages:

- Dynamic Animations
 - Interruptible
 - Scrubbable
 - Reversible
 - More Timing Functions

```
import UIKit  
class ViewController : UIViewController {  
    var animator : UIViewPropertyAnimator!  
  
    @IBOutlet weak var block: UIView  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        setupAnimation()  
    }  
}
```

```
func setupAnimation() {
    self.block.transform = CGAffineTransform.identity // make sure
    self.block.frame.origin.x = CGFloat(0)
    animator = UIViewPropertyAnimator(duration: 1.2, curve: UIView.AnimationCurve.linear, animations: {
        let screenWidth = self.view.frame.size.width
        let blockWidth = self.view.frame.size.width
        self.block.frame.origin.x = screenWidth - blockWidth
        self.block.transform = CGAffineTransform(rotationAngle: CGFloat.pi)
    })
}
```

```
animator.addCompletion { (position) in
    self.block.backgroundColor = UIColor.red
}
```

```
@IBAction func startAnimation(_ sender: Any) {
    if animator.isRunning {
        return
    }
    switch animator.state {
        case .active:
            animator.isReversed = true // will still call the completion handler
            animator.addAnimations {
                self.block.transform = CGAffineTransform(scaleX: 1.7, y: 1.7)
            }
    }
}
```

```
animator.startAnimation()
case .inactive:
    setupAnimation()
    animator.startAnimation()
case .stopped:
    animator.finishAnimation(at: UIViewAnimatingPosition.current)
    break
}
```

```
@IBAction func stopAnimation(_ sender: Any) {
    animator.stopAnimation(false) // without finishing, if true → will become inactive
}
```

```
@IBAction func pauseAnimation(_ sender: Any) {
    if animator.state == UIViewAnimatingState.active {
        animator.pauseAnimation()
    }
}
```

```
@IBAction func sliderDidChange(_ sender: Any) {
    animator.fractionComplete = CGFloat(slider.value)
}
```

// animate the login screen

import UIKit

```
class ViewController : UIViewController {  
    lazy var logoImage : UIImageView = {  
        let image = UIImage(named: "cat")!  
        let imageView = UIImageView(image: image)  
        imageView.layer.masksToBounds = true  
        return imageView  
    }()
```

gray color \Rightarrow d8d8d8

space black \Rightarrow 4A4A4A

* make sure all views embeded properly

```
@IBOutlet weak var usernameTextField : UITextField!  
@IBOutlet weak var passwordTextField : UITextField!  
@IBOutlet weak var loginButton : UIButton!  
@IBOutlet weak var loginView : UIView!
```

Var moveLogoAnimator : UIViewPropertyAnimator!

override func viewDidLoad() {

super.viewDidLoad()

↗ // important

logoImage.translatesAutoresizingMaskIntoConstraints = false

loginView.addSubview(logoImage)

NSLayoutConstraint.activate([

logoImage.centerXAnchor.constraint(equalTo: loginView.centerXAnchor),

logoImage.centerYAnchor.constraint(equalTo: loginView.centerYAnchor),

logoImage.heightAnchor.constraint(equalToConstant: CGFloat(100)),

logoImage.widthAnchor.constraint(equalToConstant: CGFloat(100))

])

loginView.transform = CGAffineTransform(scaleX: 0, y: 0)

usernameTextField.alpha = CGFloat(0)

passwordTextField.alpha = CGFloat(0)

loginButton.alpha = CGFloat(0)

loginView.backgroundColor = UIColor(red: 0.29, green: 0.29, blue: 0.29, alpha: 1.0)

)

override func viewDidLayoutSubviews() {

super.viewDidLoad()

loginView.layer.cornerRadius = CGFloat(7.0)

loginButton.layer.cornerRadius = CGFloat(5.0)

logoImage.layer.cornerRadius = CGFloat(50.0)

)

override func viewDidAppear(_ animated: Bool) {

super.viewDidAppear(animated)

↗ if no delay, sometimes we do not see the whole animation

UIView.animate(withDuration: 0.8, delay: 1, usingSpringWithDamping: 0.5, initialSpringVelocity: 2, options: .curveEaseOut){

animation {

self.loginView.transform = CGAffineTransform(scaleX: 1.0, y: 1.0)

) { (success) in

self.setupMoveLogoAnimation()

self.moveLogoAnimator.startAnimation()

3 -05:04

```
func setupMoveLogoAnimation() {
    moveLogoAnimator = UIViewPropertyAnimator(duration: 2.0, curve: .easeIn, animations: nil)

    moveLogoAnimator.addAnimations({
        self.logoImage.frame.origin.y = CGFloat(20.0)
        self.loginView.backgroundColor = UIColor.white
    }, delayFactor: 0.2)

    moveLogoAnimator.addAnimations({
        self.usernameTextField.alpha = 1.0
    }, delayFactor: 0.6)

    moveLogoAnimator.addAnimations({
        self.passwordTextField.alpha = 1.0
    }, delayFactor: 0.7)

    moveLogoAnimator.addAnimations({
        self.loginButton.alpha = 1.0
    }, delayFactor: 0.8)
}
```

II Constraint animations

```
import UIKit
class ViewController: UIViewController {
    @IBOutlet
    var isMenuOpen: Bool = false
    override func viewDidLoad() {
        super.viewDidLoad()
        let statusBarHeight = UIApplication.shared.statusBarFrame.height
        topMenuConstraint.constant = -menuView.frame.size.height + CGFloat(40) + statusBarHeight
        menuView.layer.shadowOpacity = 0.4
        menuView.layer.shadowColor = UIColor.black.cgColor
        menuView.layer.shadowOffset = CGSize.zero
        menuView.layer.shadowRadius = CGFloat(15.0)
    }
}
```

```
override func viewDidLayoutSubviews() {
    super.viewDidLayoutSubviews()
    redButton.layer.cornerRadius = redButton.frame.size.width/2
    // similar for other buttons
}
```

```
② IBAction func redButtonDidTouch(_ sender: Any){  
    view.backgroundColor = UIColor.red
```

}

↓ other buttons

```
③ IBAction func toggleButtonDidTouch (_ sender: UIButton){
```

```
if isMenuOpen
```

```
    sender.setTitle("open", for: .normal)
```

```
    isMenuOpen = false
```

```
    closeMenu()
```

```
} else {
```

```
    sender.setTitle("close", for: .normal)
```

```
    isMenuOpen = true
```

```
    openMenu()
```

}

```
func closeMenu(){
```

```
let statusBarHeight = UIApplication.shared.statusBarFrame.height
```

```
self.view.layoutIfNeeded()
```

```
UIView.animate(withDuration: 0.7, delay: 0, options: .curveEaseOut, animations: {
```

```
    self.topMenuConstraint.constant = -menuView.frame.size.height + CGFloat(140) + statusBarHeight
```

```
    self.view.layoutIfNeeded()
```

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

}

```
func openMenu(){
```

```
self.view.layoutIfNeeded()
```

```
UIView.animate(withDuration: 0.7, delay: 0, options: .curveEaseOut, animations: {
```

```
    self.topMenuConstraint.constant = CGFloat(0)
```

```
    self.view.layoutIfNeeded()
```

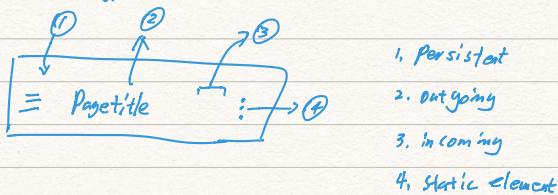
```
}, completion: nil)
```

}

II Motion Design

II Transition

broadly two types of transitions : modal and navigation



II micro interactions

Trigger → rules → feedback → loops & Modes

PDF converted to PNG by Xcode

II AppDelegate.swift

```
import UIKit
```

```
let tabBarDelegate = TabBarDelegate()
```

② UI Application Main

```
class AppDelegate : UIResponder, UIApplicationDelegate {  
    var window: UIWindow?
```

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {  
    let tabController = UITabBarController()  
    let mainStoryboard = UIStoryboard(name: "Main", bundle: nil)  
    let firstVC = mainStoryboard.instantiateViewController(withIdentifier: "FirstVC") as! ViewController  
    firstVC.view.backgroundColor = UIColor.red  
    let secondVC = mainStoryboard.instantiateViewController(withIdentifier: "SecondVC") as! SecondViewController  
    secondVC.view.backgroundColor = UIColor.yellow  
    let vcData: [(UIViewController, UIImage, UIImage)] = [  
        (firstVC, UIImage(named: " ")!, UIImage(named: " ")!),  
        (secondVC, UIImage(named: " ")!, UIImage(named: " ")!)]
```

```
let vcs = vcData.map { (vc, defaultImage, selectedImage) -> UIViewController in  
    vc.tabBarItem.image = defaultImage  
    vc.tabBarItem.selectedImage = selectedImage  
    return vc}
```

```
}
```

```
tabController.viewControllers = vcs  
tabController.tabBar.isTranslucent = false  
tabController.delegate = tabBarDelegate  
window?.rootViewController = tabController  
return true
```

```
3
```

//Tab Bar Delegate.swift

import UIKit

class TabBarDelegate : NSObject, UITabBarControllerDelegate {

```
func tabBarController(_ tabBarController: UITabBarController, didSelect viewController: UIViewController) {  
    guard let controllerIndex = tabBarController.viewControllers?.firstIndex(of: viewController) else { return }  
    guard let tabBarView = tabBarController.tabBar.items![controllerIndex].value(forKey: "view") as? UIView  
    else { return }
```

let imageView = tabBarView.subviews.compactMap{ \$0 as? UIImageView}.first

↓ chain animations

↓ imageView?.transform = CGAffineTransform(scaleX: 1.2, y: 1.2) // 0.25s

0.9 0.9 // 0.1s

1.2 1.2 // 0.25s

imageView?.transform = CGAffineTransform.identity

}