// Custom Timing Functions // cubic-bezier.com

/*

Timing Functions / Timing Curves / Easing Functions & all three same thing

↳ pacing of the animation

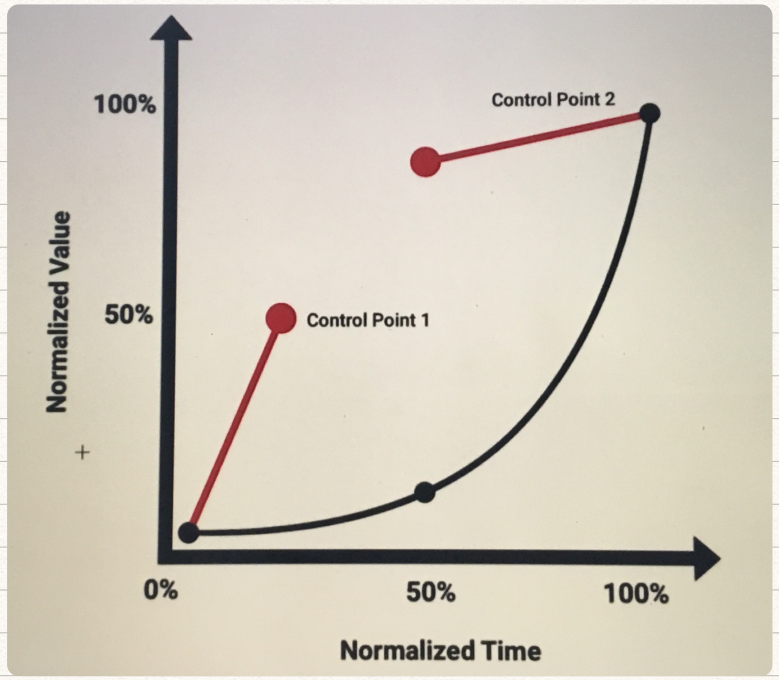
CABasicAnimation

UIView Property Animator

UIView Animate With Duration

CAKeyframe Animation

*/




```
import UIKit

class ViewController: UIView (controller{

    override func viewDidLoad(){
        super.viewDidLoad()


    }
```

```
    func exampleBasicAnimation(){
        let timingFunction = CAMediaTimingFunction(controlPoints: 0.3, 0.85, 0.8, 0.15)
        let basicAnim = CABasicAnimation(keyPath: "position.x")
        basicAnim.timingFunction = timingFunction
    }

    func ExampleProperyAnimation(){
        //method 1
        let cubicTimingParameters: UITimingCurveProvider = UICubicTimingParameters(controlPoint1: CGPoint(x:0.3, y: 0.85),
            controlPoint2: CGPoint(x:0.8, y: 0.15))
        let animator = UIViewPropertyAnimator(duration: 1, timingParameters: cubicTimingParameters)
        animator.addAnimations{
            // animations
        }
        animator.startAnimation()
        //method 2
        let propertyAnimator = UIViewPropertyAnimator(duration:1, controlPoint: CGPoint(x:0.3, y: 0.85), controlPoint2: CGPoint(x:0.8, y:0.15)){
            //animations
        }
        propertyAnimator.startAnimation()
    }
    func exampleViewAnimation(){
        let timingFunction = CAMediaTimingFunction(controlPoints: 0.3, 0.85, 0.8, 0.15)
        CATransaction.begin()
        CATransaction.setAnimationTimingFunction(timingFunction)
        UIView.animate(withDuration: 1){
            //animations
```

```
        }
        CATransaction.commit()
    }

    func exampleKeyFrameAnimation(){
        let timingFunction = CAMediaTimingFunction(controlPoints: 0.3, 0.85, 0.8, 0.15)
        let keyFrameAnim = CAKeyFrameAnimation(keyPath: "position.x")
        keyFrameAnim.values = [0, 20, -20, 0]
        keyFrameAnim.keyTimes = [0, 0.4, 0.8, 1]
        keyFrameAnim.duration = 3
        keyFrameAnim.timingFunctions = [timingFunction, timingFunction, timingFunction]
    }
```

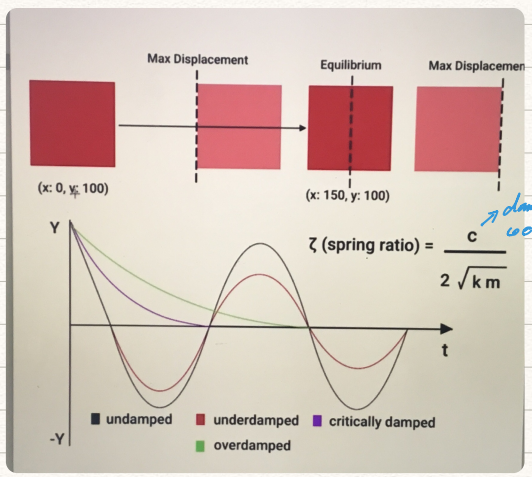
## Path Animation

```
import UIKit

class ViewController: UIViewController {
    let sunView = UIView()
    let planetView = UIView()
    let planetSize: CGFloat = 80
    let orbitMargin: CGFloat = 10
    var halfHeight: CGFloat {
        return view.frame.height/2
    }

    override func viewDidLoad(){
        super.viewDidLoad()
        view.addSubview(planetView)
        // planetView.backgroundColor = UIColor.blue
        planetView.frame = CGRect(x: view.frame.width - orbitMargin - planetSize, y: halfHeight - (planetSize/2),
                width: planetSize, height: planetSize)
        // planetView.layer.cornerRadius = planetView.frame.width/2

        planetView.layer.contents = UIImage(named: "earth")?.cgImage
        view.addSubview(sunView)
        // sunView.backgroundColor = UIColor.yellow
        sunView.frame = CGRect(x: view.frame.midX - 55, y: view.frame.midY - 55, width: 110, height: 110)
        // sunView.layer.cornerRadius = sunView.frame.width/2
        sunView.layer.contents = UIImage(named: "sun")?.cgImage
    }

    override func viewDidAppear(_ animated: Bool){
        super.viewDidAppear(animated)
        startOrbit()
    }
```

```
func startOrbit(){
    let orbitWidth : CGFloat = view.frame.width - ( 2*orbitMargin ) - CGFloat(planetSize)
    let orbitPath = UIBezierPath(ovalIn : CGRect(x: -orbitWidth, y: -(orbitWidth/2), width: orbitWidth, height: orbitWidth))
    let orbitAnim = CAKeyFrameAnimation(keyPath:"position")
    orbitAnim = 3
    orbitAnim.repeatCount = .greatestFiniteMagnitude
    orbitAnim.isAdditive = true
    orbitAnim.calculationMode = kCAAnimationPaced
    //orbitAnim.rotationMode = kCAAnimationRotateAuto
    orbitAnim.path = orbitPath.cgPath
    planetView.layer.add(orbitAnim, forKey: nil)
}
```

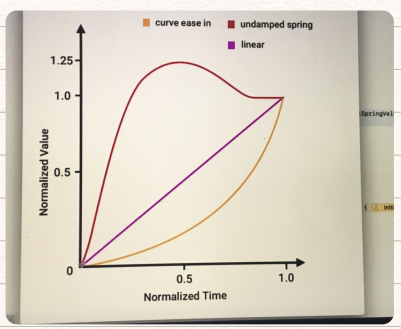
//Fluid spring animations

facebook pop API

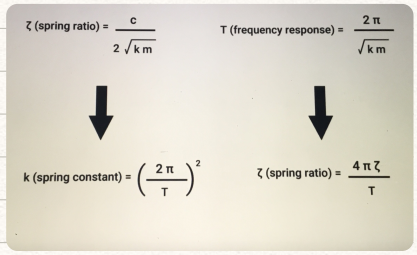



```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad(){
        super.viewDidLoad()

    }

    func viewAnimation(){
        UIView.animate( withDuration: 1, delay: 0, usingSpringWithDamping: 0.4, initialSpringVelocity: 2, options: .curveEaseIn,
            animations: {


        }, completion: nil)
    }

    func propertyAnimation(){
        let propertyAnimator = UIViewPropertyAnimator(duration: 1, dampingRatio: 0.4){


        }
    }
}
```

cubic and quadardic



```
    func springPropertyAnimation(){
        let springTiming = UISpringTimingParameters(damping: 0.4, response: 2)
        let propertyAnimator = UIViewPropertyAnimator(duration: 1, timingParameters: springTiming)

    }

}

extension UISpringTimingParameters{
    convenience init(damping: CGFloat, response: CGFloat, initialVelocity: CGVector = .zero){
        let stiffness = pow(2*.pi/response, 2)
        let damp = 4*.pi*damping/response
        self.init(mass: 1, stiffness, damping: damp, initialVelocity: initialVelocity: initialVelocity)
    }
}
```




## CADisplayLink

```
import UIKit

class ViewController: UIViewController{
        @IBOutlet weak var progressView: ProgressView!
        @IBOutlet weak var startButton: UIButton!
        private var displayLink: CADisplayLink?
        var startTime: Double = 0.0
        let animLength: Double = 4.0

        override func viewDidLoad(){
          super.viewDidLoad()



    }
    @IBAction func startAnim(_ sender: Any){
        animWithDisplayLink()
    }

    func animWithDisplayLink(){
        startButton.isEnabled = false
        startTime = CACurrentMediaTime()
        displayLink = CADisplayLink(target: self, selector: #selector(updateProgressBar))
        displayLink?.add(to: .main, forMode: .defaultRunLoopMode)

    }
```

```
@objc func updateProgressBar(){
    var elapsed = CACurrentMediaTime() - startTime
    let currentProgress = elapsed / animLength
    progressView.currentProgress = currentProgress

    if elapsed > animLength{
        stopDisplayLink()
        elapsed = animLength
    }
}

func stopDisplayLink(){
    displayLink?.invalidate()
    displayLink = nil
}
```

//ProgressView.swift

```
import UIKit

class ProgressView: UIView{

    var currentProgress: Double = 0.0{
        didSet{
            if currentProgress > 1.0{
                currentProgress = 1.0
            }
            if currentProgress < 0.0{
                currentProgress = 0.0
            }
            setNeedsDisplay()
        }
    }

    let progressBarWidth: CGFloat = 300.0
    let progressBarHeight: CGFloat = 50.0

    override func draw(_ rect: CGRect){
        let currentProgressBarWidth: CGFloat = progressBarWidth * CGFloat(currentProgress)
        let progressBar = UIBezierPath(rect: CGRect(x: (frame.width/2) - (progressBarWidth/2)
            , y: (frame.height/2) - (progressBarHeight/2), width: currentProgressBarWidth, height: progressBarHeight))
        UIColor.red.setFill()
        progressBar.fill()
    }

}
```