# Các khái niệm cơ bản:

**Frame** là 1 property của View, frame thuộc kiểu CGRect. **Frame** sẽ quyết định vị trí và size của chính View đó.

**Layer** thuộc **Core** **Animation**, layer có nhiệm vụ quản lý cách hiển thị của View. Layer được sử dụng để tạo **animation** và áp dụng hiệu ứng hình ảnh vào View. Khi thay đổi **position** của **layer,** **position của view cũng thay đổi theo**. Bởi vì khi ta thay đổi vị trí của layer, sự thay đổi đó sẽ thông báo tới thuộc tính **frame của View**. Vị trí của frame cũng update theo sự thay đổi đó.

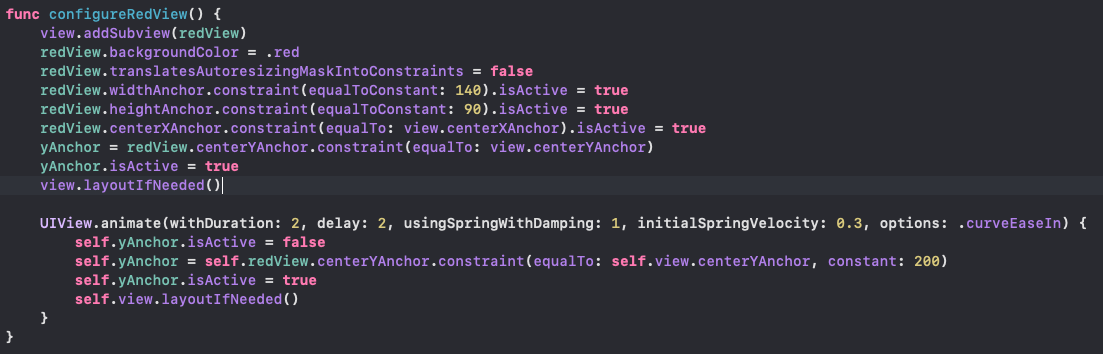
* Trong IOS, có rất nhiều thuộc tính có thể được animated bằng cách sử dụng các APIs của UIKit hoặc của Core Animation. Các thuộc tính đó là:
  + Position và Size
  + Opacity
  + Color
  + Transform: Thuộc tính transform của layer, view có thể được animated bằng cách thay đổi thuộc tính **transform** thông qua việc sử dụng **CGAffineTransform** hoặc **CATransform3D**
  + Text, Font
  + Shape và Path: Shape và Path của layer có thể được animated bằng cách thay đổi thuộc tính **path** bằng cách sử dụng **UIBezierPath hoặc CGPath**
  + Constraints

## Animation with UIKit

Đối với UIKit, nó cung cấp cho ta 1 API để có thể animate các thuộc tính kia bằng cách sử dụng UIView.animate(), những sự chuyển đổi liên quan đến position, size, color, opacity, and transform,.. đều sẽ được **animated.**

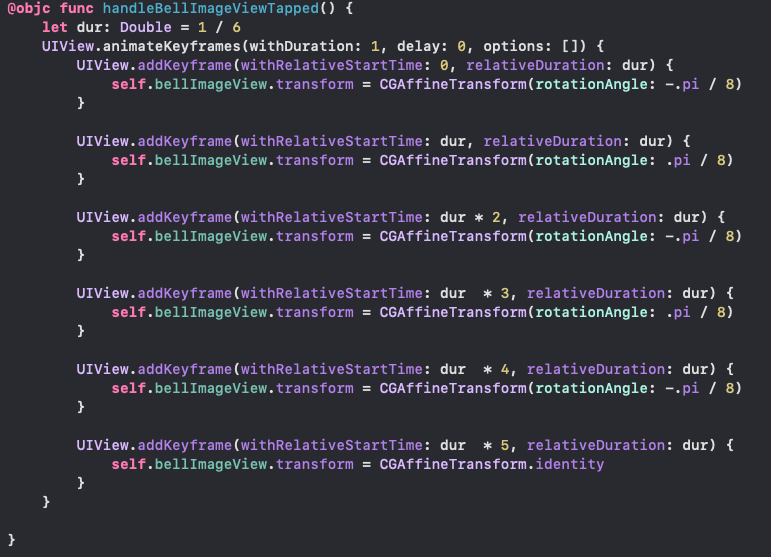
* **UIView.animate(withDuration:, animations):** Là phương thức được sử dụng để animate bình thường 😊))
* **UIView.animate(withDuration, delay, usingSpringWithDamping, initialSpringVelocity, option: animation):** cũng là phương thức để animate, nhưng nó cung cấp cho chúng ta các **spring animation**, **spring animation** là những animation mô phỏng hoạt động của lò xo, kiểu dãn nở. kiểu khi ta thay đổi view từ vị trí sát **top xuống sát bottom,** thì sẽ có hiệu ứng rơi, thì ta có thể thêm các hiệu ứng nảy nún xuống, rơi xuống xong nảy lên 1 tí như quả bóng 😊)).
  + **usingSpringWithDamping:** là thuộc tính chỉ **giá trị giảm chấn,** nghĩa là nó là 0.0 thì giảm chấn rất bé, nên nó sẽ lún xuống rất nhiều, dẫn đến view nảy rất cao, **nếu là 1.0 nghĩa là giảm chất là lớn nhất, sẽ ko nảy tí gì.**
  + **initialSpringVelocity:** là giá trị chỉ vận tốc của view ta xét
  + **Option:** Các option có thể là:
    - **.** **curveEaseInOut:** đâu tiên animation sẽ chậm, sau đó tăng tốc, sau đó giảm tốc lần nữa để hoàn thành(Như quả bóng khi rơi xuống mặt đất ấy)
    - **.curlinear:** Tốc độ không đổi từ lúc thực thi tới khi kết thúc
    - **.autoreverse:** This option causes the animation to play in reverse after completing its forwards animation. This means that the view will animate to its end state, and then animate back to its starting state. This can be useful for creating repeating animations.
    - **.repeat**
    - **.allowUserInteraction:** cho phép người dùng tương tác với view trong lúc animation. By default, user ko thể tương tác được
    - **…..**

**Chú ý: Khi thay đổi layout 1 view như frame, bounds, constrait, UIKit sẽ không apply những thay đổi đó ngay lập tức mà** UIKit sẽ lên lịch để layout vào 1 thời gian nhất thích hợp. Nhưng đôi khi ta muốn apply layout đó ngay lập tức, ta sử dụng **layoutIfNeeded().**



Đây là ví dụ animate, 1 view từ giữa màn, sẽ rơi xuống dưới bottom, **ta phải sử dụng view.layoutIfNeed() để layout ngay lập tức chỗ đó.** Bởi vì nếu ko, nó sẽ ko layout luôn view ban đầu, dẫn đến ta ko thấy **redView ở giữa view, dẫn đến animation bị sai(vì ko ở giữa view thì xuống dưới bottom sẽ sai).**

* **UIView.animateKeyFrames():** Với thằng cha **.animate()** bên trên, ta thấy nó **animate đồng thời các thay đổi,** vậy để animate lần lượt các animation theo thời gian, thì ta sử dụng thằng này, hoặc bên dưới có thằng **CAKeyframeAnimation cũng làm tương tự**

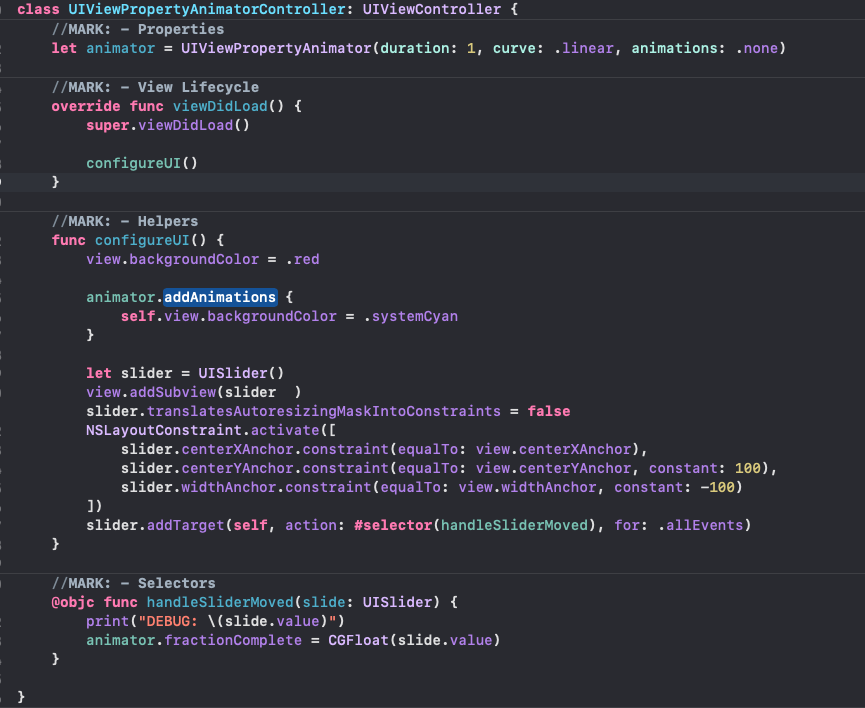


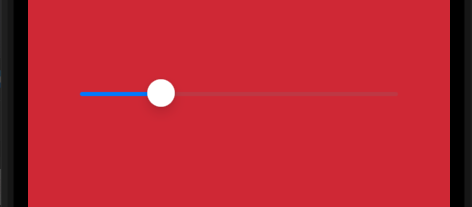
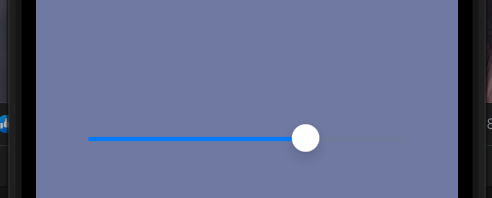
Đoạn code này rung chuông 😊), nó sẽ rotate trái, rồi phải với các thông số

* **withRelativeStartTime**: Được dùng để xác định thời gian bắt đầu animation đó: **startTime = withRelativeStartTime \* withDuration**
* **relativeDuration**: Chỉ thời gian thực hiện: **excuteTime** = **relativeDuration** \* **withDuration**

## UIViewPropertyAnimator:

Đây là 1 class của UIKit, cho phép chúng ta tương tác với animation. Ví dụ với các animation bên trên, ta thấy nó sẽ thực thi từ đầu đến cuối animation mà người dùng ko tác động được gì, nhưng với class này, ta có thể chỉ cho animation đang được thực hiện ở đâu



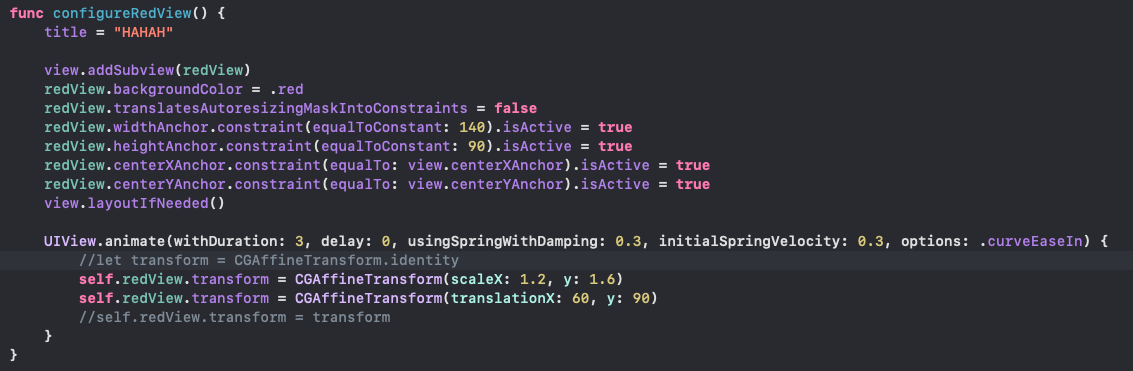
Ở đoạn code này, ta có 1 UIViewPropertyAnimator, trong đây ta addAnimation **self.view.backgroundColor = .systemCyan.** Nghĩa là background view lúc này sẽ thay đổi từ red sang systemCycan. Ta sử dụng **slider** để gán % thay đổi.

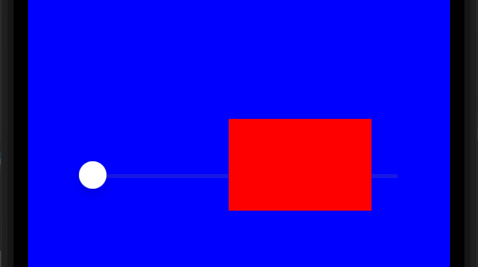
Ta thấy rang lúc slider mới chỉ kéo đc 1 tí thì màu vẫn còn đỏ đỏ, sau đó kéo thêm thì màu đã được sang màu cyan dần. Ta đã tự mình kiểm soát được quá trình animation thông qua **UIViewPropertyAnimator.**

Khi gọi: **animator.startAnimation()** thì việc animation sẽ được thập thi ngay lập tức, tuy nhiên ta có thể dừng việc thực thi nó, sau đó lại tiếp tục

## 1.3 CGAffineTransform

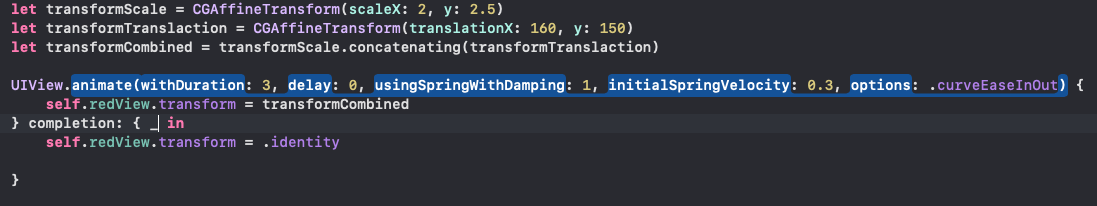
Ta sử dụng CGATransform để có thể thay đổi kích thước, xoay,… layout của 1 view.





Thông qua CGAffineTransform, ta đã View di chuyển theo trục x y là 60 và 90 pixel, nhưng size của view không to ra. This is because each time a new transform is set on a view, it replaces the existing transform.

**Chú ý:** This is because each time a new transform is set on a view, it replaces the existing transform. Vì vậy để cả 2 transform. Vì vậy để có thể sử dụng 2 transform cùng 1 lúc ta làm như sau:



Ta thấy rằng sau khi **animated xong**, vị trí của view, size của view đã được thay đổi, vậy để sau khi animated xong view trở về original, ta sử dụng thuộc tính **CGAffineTransform.identity.**

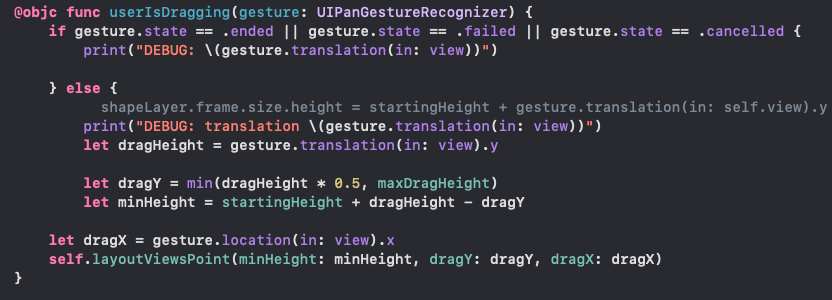
## 1.4 UIPanGestureRecognizer

Để có thể thu nhận các hành động như taps, rotations, swipes, pans,.. là rất khó khăn khi UIGestureRecognizers chưa ra mắt, ta phải register các notifications như **touchBegan, touchMoves, touchesEnds** trong tất cả các view. Nhưng kể từ IOS 3.0, Apple giới thiệu **UIGestureRecognizers**. Nó cung cấp phương thức để dễ dàng thu thập, xử lý các hành động của user như taps, rotations, swipes, pans,.. Để làm việc với **UIGestureRecognizers, ta làm theo các bước sau:**

* Tạo 1 gesture(cử chỉ) recognizer: Ta cần gán target và action cho gesture.
* Add gesture vừa tạo đó vào view: Ta sẽ đính mỗi gesture recognizer vào 1 view, và chỉ 1 mà thôi. Mỗi khi ta chạm vào view được gán với gesture recognizer đó, nó sẽ thực thi **action.**

Chú ý rằng: khi add gesture cho container View, thì mọi hành động như chạm, kéo,… với các subView bên trong container đều trigger action.



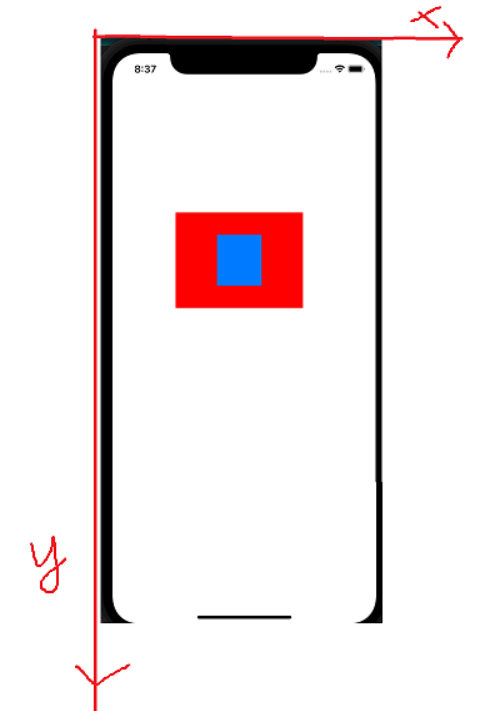
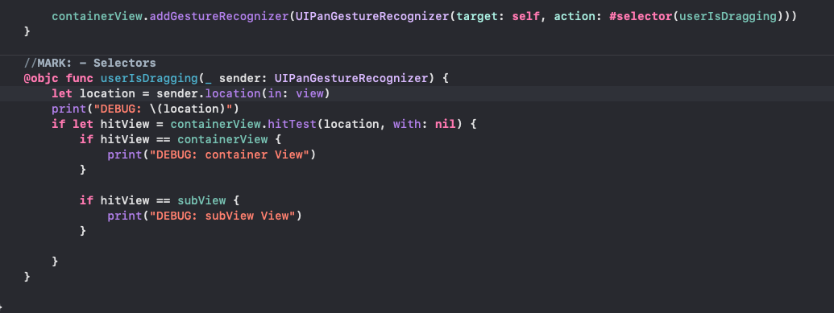


Ta có các phương thức đính kèm với gesture recognizer state:

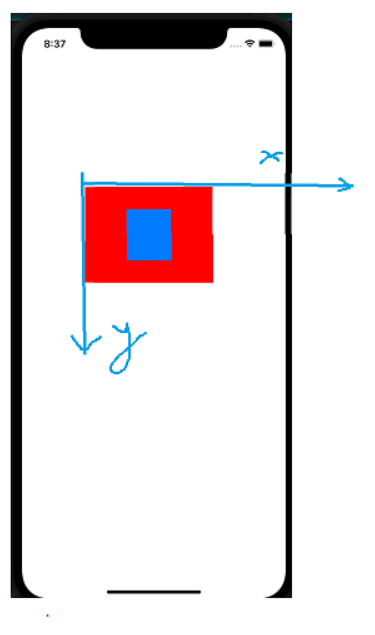
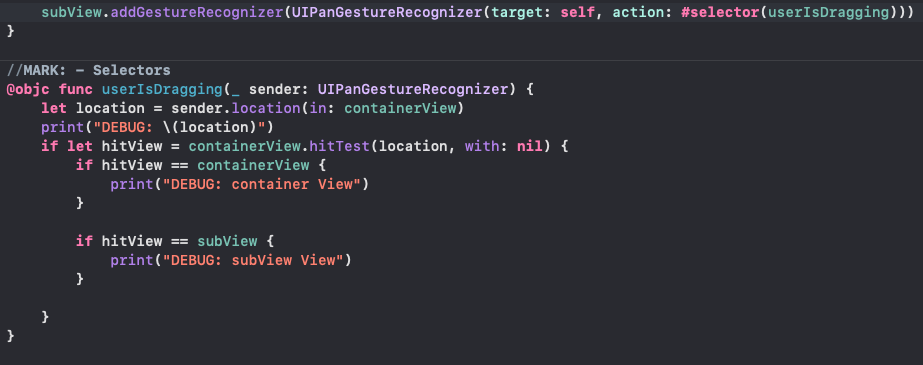
* Began: The gesture has started. For example, when a user starts touching the screen or moving their finger, the gesture recognizer's state is changed to began.
* Changed: The gesture is changing. For example, when a user is swiping their finger across the screen, the gesture recognizer's state is constantly changing as the finger moves.
* Ended: The gesture has ended. For example, when a user lifts their finger off the screen, the gesture recognizer's state is changed to ended.
* Cancelled: The gesture has been cancelled. For example, if a user touches the screen but then swipes too far and accidentally cancels the gesture, the gesture recognizer's state is changed to cancelled.

Bênh cạnh đó ta còn có các phương thức khác đính kèm như:

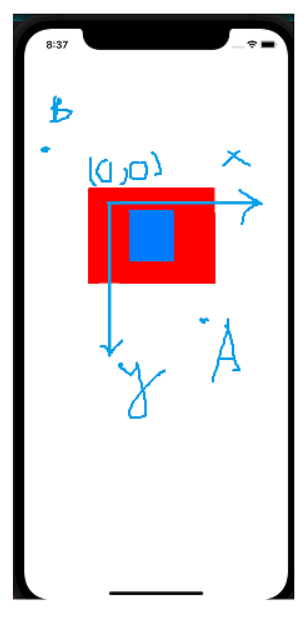
* Gesture.location(in: view1): Là vị trí điểm tay ta chạm screen so với view1.

VD1: với trường hợp containerView là màu red, chứa subView là blue, containerView được add gesture và location là :view. Thì vị trí tay của ta sẽ được tính theo trục này

VD2: subView được add gesture và location là :container. Thì vị trí tay của ta sẽ được tính theo trục này

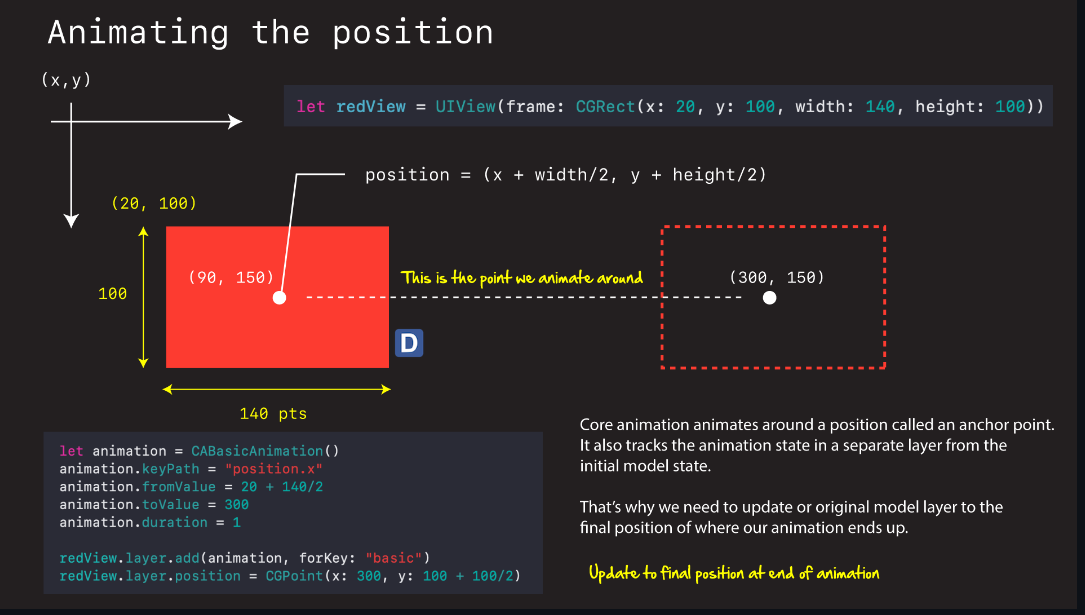
* translation(in: view):

Với ví dụ này**, containerView sẽ được add gesture, điểm đầu tiên ta chạm trong containerView chính là điểm gốc có tọa độ (0, 0**). Các điểm mà ta di xung quanh màn hình sẽ được xác định theo gốc (0, 0) đó và các trục x, y. Như điểm A sẽ có tọa độ x y là 2 số dương, số B sẽ có tọa độ là 2 số âm.



# Core Animation

## 2.1 Animating the position



Tâm hình chữ nhật là điểm ta sẽ animates xung quanh.

In Core Animation everything animates around the views position or anchor point. To animate a rectangle from left-to-right along it's x-axis we first need to create a **CABasicAnimation().**

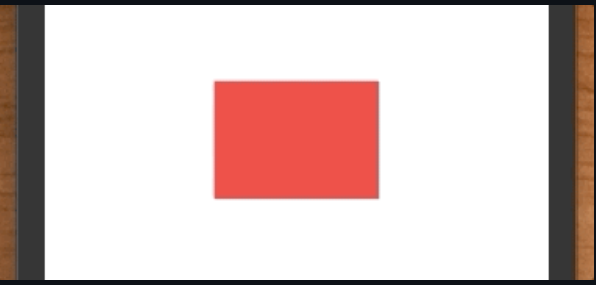
Với **CABasicAnimation()** ta có thể nội suy 2 giữa 2 điểm bắt đầu và kết thúc theo **duration time,** ta cũng có thể xét thêm số lần lặp lại thông qua **thuộc tính .repeatCount.** Sau đó ta có thể add animation đó cho layer thông qua phương thức **.add(\_:forkey:), animation đó sẽ được applied cho đến khi nó hoàn thành xong hoặc bị xóa bởi removeAnimation(forKey:)**

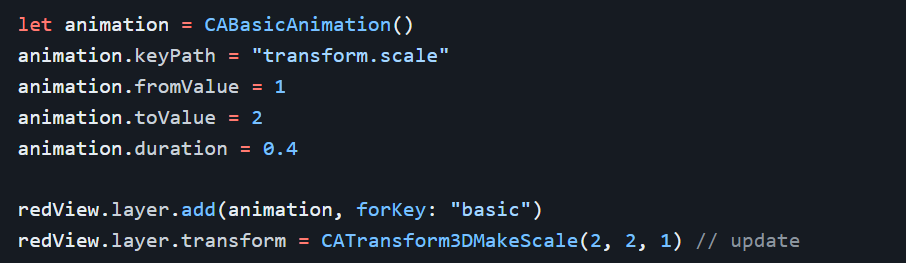
**Chú ý:** Ta phải thông báo cho **animation** biết chính xác ta muốn animate cái gì, bằng cách setting **keypath: animation.keypath = “position.x”**

Ta biết rằng, **CA** có 2 layer, đầu tiên là **modal layer(current state) và presentation layer(animation state).** Nên khi kết thúc animation ta phải update model state. Lệnh update này thông qua dòng code: **redView.layer.position = …**

## Scaling

## 

Chúng ta sử dụng đoạn code:

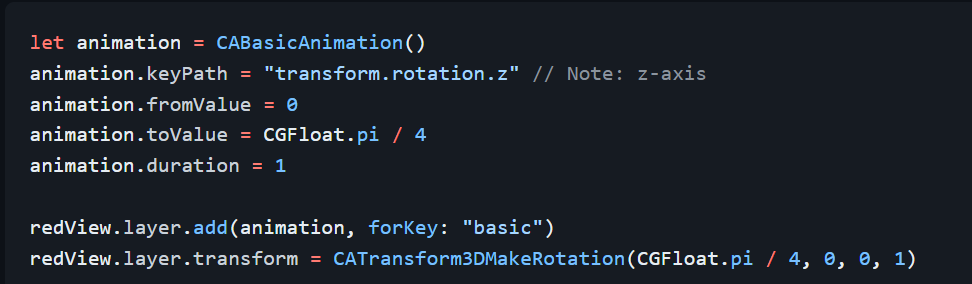


Ta xet **keypath** là **transform.scale** để chỉ rằng **animated** ở đây ta muốn là scale layer. **Với .fromValue = 1** thì size ban đầu giữ nguyên, và .**toValue là ta muỗn x2 chiều dài với chiều rộng của view ta muốn.**

Đoạn code: **redView.layer.transform = CATransform3DMakeScale(2, 2, 1),** sẽ scale kích thước của layer lên. Sẽ x2 width, và x2 height, và x1 z

## Rotating

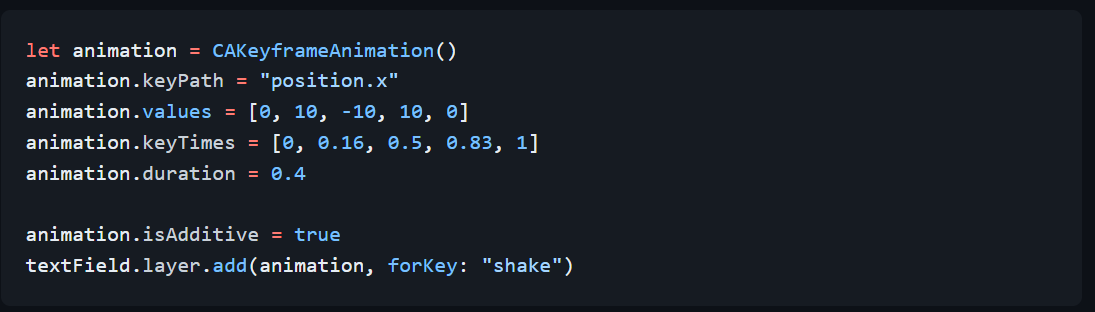
Khi rotate(quay), chúng ta sẽ xoay xung quanh trục z, và quay theo đơn vị **radians.**



Ta đang muốn quay 45 độ nên là pi / 4

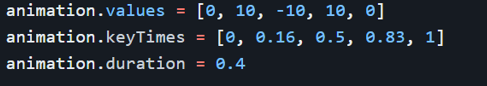
## Shaking

Shaking là kiểu làm cái view rung rung như khi sai mật khẩu.



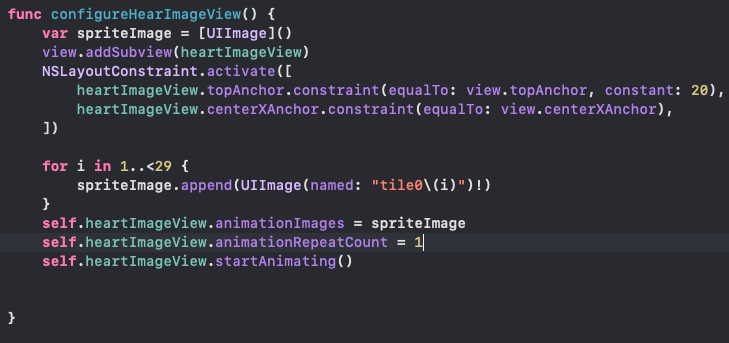
**CAKeyframeAnimation** is a subclass of **CAPropertyAnimation** that allows you to animate an object's property with a **series of values instead of just a single starting and ending value.**

With **CAKeyframeAnimation**, you can specify **a list of values** for the property you want to animate, and the animation will **interpolate(nội suy, nội suy nghĩa là đi từ điểm này đến điểm kia)** between those values. You can also specify the timing function used to interpolate between the values, as well as the duration of the animation.



* The values property is an array that specifies the list of values to interpolate between
* The keyTimes property is also an array, but it specifies the relative timing for each value in the values array. In other words, it specifies the percentage(phần trăm) of the total animation duration at which each value should be reached.
  + At the start of the animation (time 0), the value is 0.
  + At 16% of the animation duration (time 0.064 seconds in this case), the value is 10.
  + At 50% of the animation duration (time 0.2 second in this case), the value is -10.
  + At 83% of the animation duration (time 0.332 seconds in this case), the value is 10.
  + At the end of the animation (time 0.4 seconds), the value is 0.

1. Animation Images

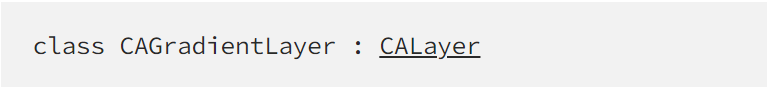


4. CAGradientLayer

Color gradient là 1 loạt các màu phụ thuộc vào vị trí. Có 2 cách để tạo ra một color gradient:

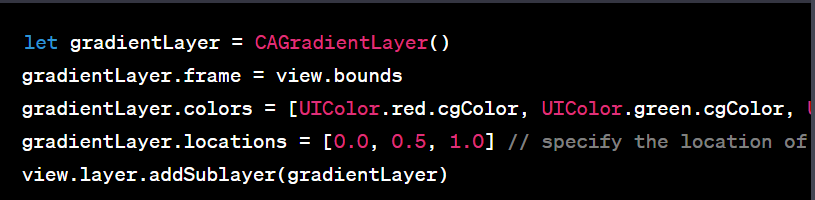
* Core Graphics
* Core Animation

Ta có



Với các thuộc tính như sau:

* .**colors**: Là 1 mảng các đối tương CGColorRef mà xác định các màu của mỗi điểm dừng chuyển sắc
* .**locations**: Là 1 mảng các đối tượng NSNumber mà xác định vị trí của từng màu trong gradient. Mảng localionsArray phải có số lượng phần tử bằng với số màu **.colors,** mỗi phần tử trong locationArray có giá trị từ 0.0 đến 1.0

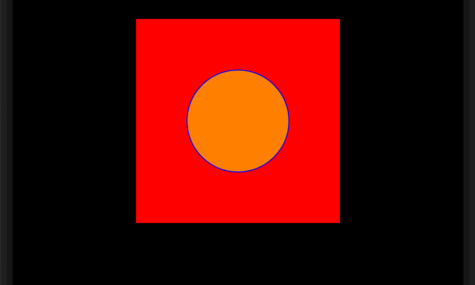
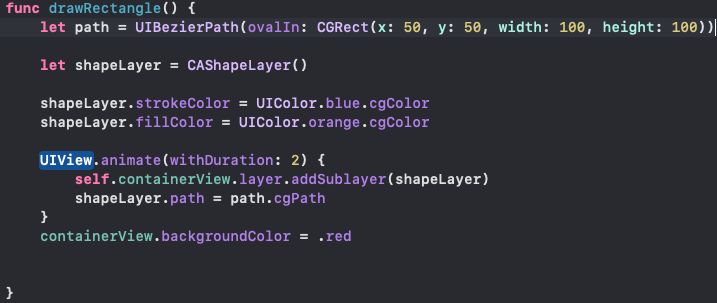


* Trong ví dụ này, red color sẽ bắt đầu từ vị trí 0.5, green color sẽ bắt đầy từ vị trí 0.5 của frame, và màu xanh ở vị trí cuối cùng
* **endpoint**:

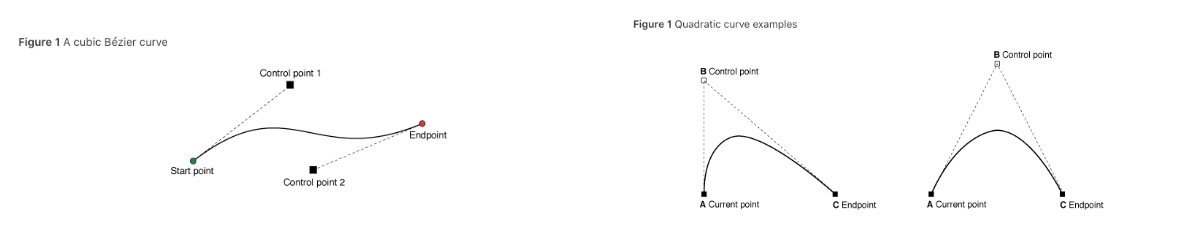
## 5. Draw using CAShapeLayer

Ta có thể dễ dàng tạo ra các hình chữ nhật, hình tròn. Vậy làm thế nào để tạo ra các hình phưc tạp hơn, như hình tam giác, hình nửa đường tròn,… Để làm việc đó ta sẽ sử dụng **CAShapeLayer.**

5.1 Drawing a Rectangle

Để vẽ đường cong thì apple cung cấp 2 phương thức **addCurve() và addQuadCurve().**

.**fillcolor:** Là vẽ color hình shape, màu của shape sẽ ghi đè lên View.

****

Nó đều ra vẽ kiểu tiếp tuyến

