Core Animation

A more advanced (complicated) way of animating

LIGHTHOUSE LABS

Animation under the hood

Core Animation is the framework that UIKit uses to perform animations. It’s got a bunch of classes and protocols that work together, but the real important ones are CALayer and CAAnimation.

CALayers: behind views

Layers are the objects that are actually drawn on screen. They're 2D rectangular services in a 3D space. Layers don't really manage their own appearance, they just store information about a bitmap. When you were drawing a custom view, that drawing code made a bitmap which was then put onto a layer for display. Although this seems odd, think of layers not as views in MVC, but as models. They model graphical information.

LIGHTHOUSE LABS

The CALayer Monster Manual

CAEmitterLayer: throw a bunch of particles onscreen
CAGradientLayer: show a nice gradient, with animation.
CAReplicatorLayer: make copies with transforms
CAScrollLayer: like UIScrollView, but a layer
CAShapeLayer: draw a shape path, can animate changes
CATextLayer: basic text drawing
CATiledLayer: how we draw those tiles on a map
CATransformLayer: view sublayers in 3D, not flattened
And more…

LIGHTHOUSE LABS

CAAnimation

CAAnimation is an object that conforms to several protocols. Combined, they give the information needed for animations. CAAnimation has the timing function & delegate. NSCopying allows animations to be copied between layer ‘versions’. CAAction is a one-method protocol that tells the animation to run. CAMediaTiming gives more information about the duration and speed of an animation. Subclasses of CAAnimation are useful in particular cases.

LIGHTHOUSE LABS

CAAnimation Subclasses

- CABasicAnimation
- CAKeyframeAnimation
- CAAnimationGroup
- CATransition

A CABasicAnimation. Very basic.

To define a basic animation, we need to create a CABasicAnimation object, set a few properties, and add it to a layer.

```
CABasicAnimation* fadeAnim = [CABasicAnimation
animationWithKeyPath:@"opacity"];
fadeAnim.fromValue = [NSNumber numberWithFloat:1.0];
fadeAnim.toValue = [NSNumber numberWithFloat:0.0];
fadeAnim.duration = 1.0;
[theLayer addAnimation:fadeAnim forKey:@"opacityAnim"];
```

LIGHTHOUSE LABS

Presentation vs model

There are two ‘versions’ of a layer: The presentation layer and the model layer.

The presentation layer is the one actually shown on screen, and the model layer is the one with the current values of the layer. That means that if you ask a layer for a value mid-animation, it will return the value of the model, not the presentation, and it won’t match what’s on screen. You can ask a layer for either its model version or its presentation version.

LIGHTHOUSE LABS

What can you animate?

anchorPoint, backgroundColor, borderColor, borderWidth, bounds, cornerRadius, frame, mask, opacity, position, shadow, transform, zPosition…

Animations work on key paths, a string of period-delimited property names.

Timing is everything

Animations have durations, but they also have speed. Speed is a multiplier for how fast an animation progresses. You can set it to 2 to be half as long, 0.5 to be twice as long, and to 0 for not changing at all. You can also set the offset of the animation, so that it starts displaying a value that’s partway through the animation. Layers also have a speed, so you can change how quickly animations happen to a layer.

Keyframe animations

CAKeyframeAnimation is how to set up keyframe animations. Give it an array of values, an array of corresponding times from 0.0 to 1.0, and it calculates a whole series of interpolations between each value. It has a number of fine controls over the timing functions between each value.

LIGHTHOUSE LABS

Animating along a path

You can create complex animations using a path for a property. For example, you could create a path of a circle and animate the center of a layer to move along the path of that circle. CAKeyframeAnimation is the class to use, but use its path property rather than its value property. To have the layer rotate along the path, set the rotationMode property to kCAAnimationRotateAuto or AutoReverse.

Animation delegate

Animations have delegates too, and you can use them to get callbacks when an animation starts and stops. It’s an informal delegate. You can also set up blocks to run when a whole animation transaction is complete.

Animating custom properties

If we have a layer with a custom property, we can set up a custom animation for it. We do this by overriding the needsDisplayForKey to return YES if it’s the key for that property, and overriding actionForKey to return an animation for the change. Then either layer or the view will be responsible for drawing the frames of the animation, given the values that will be provided over the duration of the animation.

LIGHTHOUSE LABS