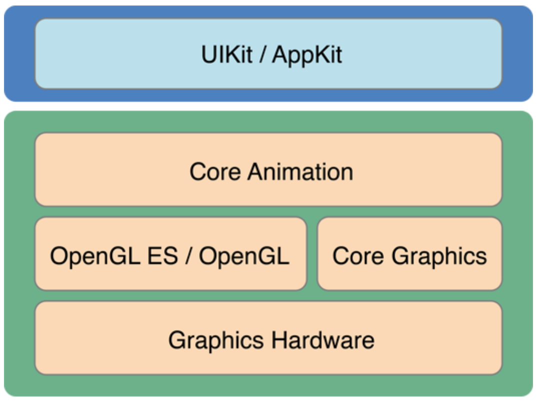
About Core Animation

Core Animation is a graphics rendering and animation infrastructure available on both iOS and OS X that you use to animate the views and other visual elements of your app. With Core Animation, most of the work required to draw each frame of an animation is done for you. All you have to do is configure a few animation parameters (such as the start and end points) and tell Core Animation to start. Core Animation does the rest, handing most of the actual drawing work off to the onboard graphics hardware to accelerate the rendering. This automatic graphics acceleration results in high frame rates and smooth animations without burdening the CPU and slowing down your app.

If you are writing iOS apps, you are using Core Animation whether you know it or not. And if you are writing OS X apps, you can take advantage of Core Animation with extremely little effort. Core Animation sits beneath AppKit and UIKit and is integrated tightly into the view workflows of Cocoa and Cocoa Touch. Of course, Core Animation also has interfaces that extend the capabilities exposed by your app's views and give you more fine-grained control over your app's animations.



At a Glance

You may never need to use Core Animation directly, but when you do you should understand the role that Core Animation plays as part of your app's infrastructure.

Core Animation Manages Your App's Content

Core Animation is not a drawing system itself. It is an infrastructure for compositing and manipulating your app's content in hardware. At the heart of this infrastructure are *layer objects*, which you use to manage and manipulate your content. A layer captures your content into a bitmap that can be manipulated easily by the graphics hardware. In most apps, layers are used as a way to manage the content of views but you can also create standalone layers depending on your needs.

Relevant Chapter: Core Animation Basics, Setting Up Layer Objects,

Layer Modifications Trigger Animations

Most of the animations you create using Core Animation involve the modification of the layer's properties. Like views, layer objects have a bounds rectangle, a position onscreen, an opacity, a transform, and many other visually-oriented properties that can be modified. For most of these properties, changing the property's value results in the creation of an implicit animation whereby the layer animates from the old value to the new value. You can also explicitly animate these properties in cases where you want more control over the resulting animation behavior.

Relevant Chapters: , Animating Layer Content, Advanced Animation Tricks, Layer Style Property Animations, Animatable Properties

Layers Can Be Organized into Hierarchies

Layers can be arranged hierarchically to create parent-child relationships. The arrangement of layers affects the visual content that they manage in a way that is similar to views. The hierarchy of a set of layers that are attached to views mirrors the corresponding view hierarchy. You can also add standalone layers into a layer hierarchy to extend the visual content of your app beyond just your views.

Relevant Chapters: Building a Layer Hierarchy

Actions Let You Change a Layer's Default Behavior

Implicit layer animations are achieved using *action objects*, which are generic objects that implement a predefined interface. Core Animation uses action objects to implement the default set of animations normally associated with layers. You can create your own action objects to implement custom animations or use them to implement other types of behaviors too. You then assign your action object to one of the layer's properties. When that property changes, Core Animation retrieves your action object and tells it to perform its action.

Relevant Chapters: Changing a Layer's Default Behavior

How to Use This Document

This document is intended for developers who need more control over their app's animations or who want to use layers to improve the drawing performance of their apps. This document also provides information about the integration between layers and views for both iOS and OS X. The integration between layers and views is different on iOS and OS X and understanding those differences is important to being able to create efficient animations.

Prerequisites

You should already understand the view architecture of your target platform and be familiar with how to create view-based animations. If not, you should read one of the following documents:

- For iOS apps, you should understand the view architecture described in *View Programming Guide for iOS*.
- For OS X apps, you should understand the view architecture described in *View Programming Guide*.

See Also

For examples of how to implement specific types of animations using Core Animation, see *Core Animation Cookbook*.