Core Animation:

Simplified Animation Techniques for Mac and iPhone Development

第一部分

核心动画开篇

第一章

什么是核心动画

版本 1.0

翻译时间：2012-09-17

DevDiv 翻译：animeng
DevDiv 校对：symbian_love BeyondVincent(破船)
DevDiv 编辑：BeyondVincent(破船)

写在前面

目前，移动开发被广大的开发者们看好，并大量的加入移动领域的开发。

鉴于以下原因：

- 国内的相关中文资料缺乏
- 许多开发者对 E 文很是感冒
- 电子版的文档利于技术传播和交流

DevDiv.com 移动开发论坛特此成立了翻译组，翻译组成员具有丰富的移动开发经验和英语翻译水平。组员们利用业余时间，把一些好的相关英文资料翻译成中文，为广大移动开发者尽一点绵薄之力，希望能对读者有些许作用，在此也感谢组员们的辛勤付出。

关于 DevDiv

DevDiv 已成长为国内最具人气的综合性移动开发社区

更多相关信息请访问 DevDiv 移动开发论坛。

技术支持

首先 DevDiv 翻译组对您能够阅读本文以及关注 DevDiv 表示由衷的感谢。

在您学习和开发过程中，或多或少会遇到一些问题。DevDiv 论坛集结了一流的移动专家，我们很乐意与您一起探讨移动开发。如果您有什么问题和技术需要支持的话，请访问网站 www.devdiv.com 或者发送邮件到 BeyondVincent@DevDiv.com，我们将尽力所能及的帮助你。

关于本文的翻译

感谢 animeng 对本文的翻译，同时非常感谢 symbian_love 和 BeyondVincent(破船)在百忙中抽出时间对翻译初稿的认真校验。才使本文与读者尽快见面。由于书稿内容多，我们的知识有限，尽管我们进行了细心的检查，但是还是会存在错误，这里恳请广大读者批评指正，并发送邮件至 BeyondVincent@devdiv.com，在此我们表示衷心的感谢。

本书翻译贴汇总

Core Animation 中文翻译各章节汇总

第一部分 核心动画开篇

第一章 什么是核心动画

推荐资源

iOS

iOS 5 Programming Cookbook 中文翻译各章节汇总

iOS6 新特征：参考资料和示例汇总

Android

DEVDIV 原创 ANDROID 学习系列教程实例

Windows Phone

Windows Phone 8 新特征讲义与示例汇总

Windows 8

Building Windows 8 apps with XAML and C#中文翻译全部汇总

Building Windows 8 apps with HTML5 and JavaScript 中文翻译汇总

Windows 8 Metro 开发书籍汇总

Windows 8 Metro App 开发 Step by Step

其它

DevDiv 出版作品汇总

目录

第 1 章 什么是核心动画

核心动画是你能够为不同类型的应用程序开发复杂的动画。你可以在窗口中简单的活动下一个视图，或者你可以在一个击杀类的游戏中，通过层创建 1000 个精灵展现到屏幕上。在这一章里，我们介绍核心动画背后的一些基本概念。我们会一瞥视图和层，确保你理解动画在层和视图上面做了什么。开始，我们会讨论动画和坐标系，之后会认识核心动画的基层即 CALayer 这个类，以及这些类的协议。

1.1. 动画和笛卡尔坐标

当你熟悉了核心动画，在程序中开始使用它时，你会发现不管是在 mac os 中，还是在 iPhone 中，动画是多么的容易。事实上动画不总是那么容易的。在有核心动画之前，如果你要做一个动画，你需要理解一些像双缓冲这样非常复杂的概念，如果你想给你的动画运用在更深的场景中，还需要了解平面几何的知识。核心动画抽象了所有这些方式。以前的方法，如果你想让一个精灵穿过屏幕，你需要创建一个离屏幕的图形上下文，画在你的精灵在上下文中，把离屏幕的图形上下文和目前屏幕显示的内容交换，然后再移动精灵的位置，并且再交换离屏幕的图形上下文，之后重复这个工作。

虽然这个过程不是十分的可怕，但是对开发者来说也是一个挑战，然而核心动画就完全消除了这些麻烦。核心动画同样去创建一个包含精灵的层，只需要简单的调用 setPosition 就可以了。在这一章你将了解到，一个层的位置就是这个层的中心点在父坐标空间中的位置。然后那又是什么意思呢？

核心动画使用了三角几何学中的标准笛卡尔级。X 轴的值是横坐标，Y 轴的值是纵坐标。当命名一个点时，X 总是第一个值，因此一个点的值为 25,35 意思就是 x=25 和 y=35。不同于标准笛卡尔级的是，你仅仅使用坐标系中右上部分正数的点。这就意味着屏幕的左下角是 0,0 屏幕的右上角是高和宽，这里高和宽依赖屏幕的分辨率。在 15 英寸的 macbook pro 上，这个值是 1440,900，例如你创建了一个精灵，并且想要从坐标 0,0 点移动。你可以在创建的时候设置这个精灵的位置 0,0，然后调用 setPostion（CGPointMake（1440,900））， 那么精灵层将会运动到屏幕的右上角。当然这需要精灵在一个覆盖在整个屏幕的视图上面，因为如果没有视图做依托，层是没有办法展现的。

不同的坐标系

你应该记住，当你要在视图中放置一个层时，不同的坐标系统方法就不同。如果在 windows 平台上，你有在坐标系中绘制的经验的话，你可能就会感到混乱。因为 windows 上，坐标原点 0,0 在屏幕的左上角，而不是左下角。

1.2. 你的免费午餐

都说没有免费的午餐，但是当你使用核心动画时，你不仅仅能得到午餐，而且还能得到饮料和一些应得的东西。例如，所有的绘制动作都是利用 OpenGL 在场景的后台发生。然而，你不需要知道任何关于 OpenGL 如何利用显卡进行加速处理的。这些都被核心动画抽象过了。如果你想了解更多 OpenGL 的具体细节，在 OS X 上，你可以使用 CAOpenGLLayer。截至这本书出来之前，在 IPhone 上使用 OpenGL 是不容易的，但是不久也会出来相应的东西的。层和层后面的视图也会给你很多性能的提升。

在核心动画中，另一个免费的是所有的动画都发生在后台的线程中。这意味这，用户界面不会因为动画而被阻塞。为了使用核心动画，你可不必成为多线程的专家！其实，就是你从来没有使用过多线程，你也不用注意太多。你仅仅需要在层或者视图上，设置 animatable 的属性，然后观看动画的发生就可以了。

1.3. 什么是层

CALayer 类和它的子类代表层，对于基于核心动画的应用程序，它是最基础的一块。所有的核心动画层在视图上都提供了一个轻量级的层。这里是苹果 API 文档描述的 CALayer 类：

对于层树对象来说，CALayer 是一个模型类。它封装了位置，尺寸和定义你坐标系中的几何转换属性。它也封装了持续时间和步调，并且它的动画实现了 CAMediaTiming 协议，这个协议定义了一个层的时间空间。

这里要注意到说的“模型对象”。如果你熟悉模型-视图-控制器（MVC）设计模式的话，这里你可能会困

惑，核心动画是拥有视图特性的，应该属于 MVC 中的视图部分才更有意义。事实上，视图是 OS X 中的 NSView 和 iPhone 中的 UIView 类，在这些视图中，层是存储需要绘制的东西。CALayer 简单的包含了一些数据模型，这些数据一般是和层相关联的属性，例如背景颜色，边框宽度，位置，尺寸，转换属性等等。

基于你应用程序的需要，核心动画提供了大量的不同层，这些层可以帮助你完成不同的功能。例如，如果你想播放一个电影，你不想手动的获取电影所有的帧，展示在层的内容区域中，因为这样你将耗费大量的开支，并且你的电影也不能流畅的播放。相反，你应该自然的想到使用 QTMovieLayer，这个层很好的抽象了电影回放展示。你需要做的全部就是给 QTMovieLayer 提供影像文件在磁盘的路径，然后这个层就控制剩下的东西了。图 1-1 展示了核心动画层的继承树和它们属于哪个框架。

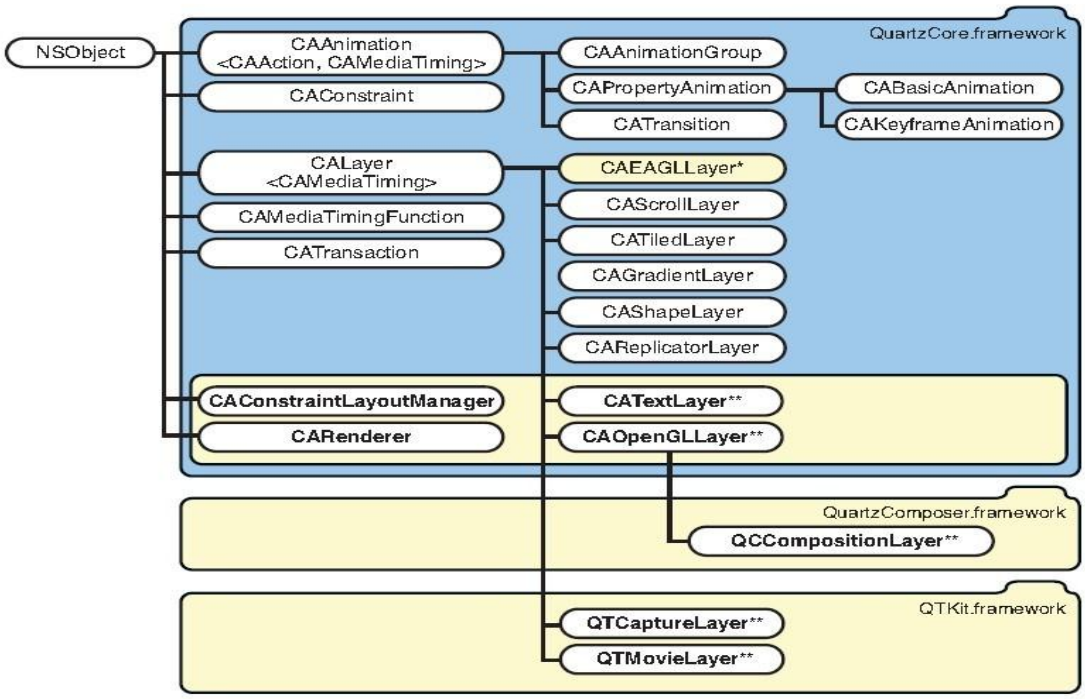

图 1-1 核心动画的层

图 1-1 显示，在核心动画的工具包中有很多有用的层，这里有每个层简单的解释：

CALayer

这是所有核心动画层继承的基类。

CATextLayer

这个层提供了一个显示字符串的便捷方法。正如其他层一样，你不仅可以指定所有的布局信息，你也可以指定文本，字体，字体大小和背景颜色属性，这个层使用这些属性绘制文本。如果你指定了 NSAtrributedString，这些其他属性字段都会被忽略。这个层仅仅在 OS X 上可用。

CAScrollLayer

这个层能够使你滚动到一个视图可视范围外的区域。

CATitledLayer

这个层能够缩放和平移一个视图可视范围外的区域。

CAOpenGLLayer

这个层提供了用 OpenGL 渲染的快捷方式。我们会在第八章深入讲解。“OpenGL Layer”这个层仅仅在 OS X 上可用。

CAEAGLLAyer

这个层提供了一个在 IPhone 上用 OpenGL 渲染的方法。这个层仅仅在 IPhone 上可用。

QTMovieLayer

这个层能够使你在基于核心动画的应用程序中，方便的播放多媒体影像文件。因为它是一个核心动画层，所以你可以给这个层增加其他的层到它的子层属性中，这样就能在播放影像视频时，组合其他的画面。我们会在第七章深入讲解“QUickTime Layers"。这个层在 OS X 上可用。

QTCaptureLayer

这个层是利用连接你电脑的相机，达到捕获视频和即时回放的目的。你可以使用 iSight(可视摄聊天像头)

和多功能视频，例如 DV，来捕获视频。真正的视频捕获是通过 QuickTime 类实现的，但是 QTCaptureLayer 能够使你实时的看到你连接的相机目前的帧。我们会在第七章深入讲解这个层。这个层仅仅在 OS X 上可用。

QCCompositionLayer

这个层能够使你回放一个 Quartz（苹果下一种绘图引擎）组件来作为你的核心动画应用程序的一部分。它能够让你使用键值编码（Key-value coding）来控制 Quartz 组件。我们在第九章深入讲解此层。这个层仅仅在 OS X 上可用。

从 Mac OS X10.6 雪豹和 iPhone OS 3.0，这些增加的层都是可用的：

CAShapeLayer

这个层让你能够根据你定义的路径，创建任意的形状。我们将在第十章深入了解这个层“Other Useful Layers”

CAGradientLayer

这个层提供了一个便捷的显示渐近梯度的方法。你可以定义多种颜色和断点，在断点处，你可以转换成你指定的不同颜色。我们将在第十章详细讲解此层。

CAReplicatorLayer

根据你指定的参数，这个层会复制你增加到层上的子层。我们将在第十章详细讲解。

1.4. 层是用来干什么的？

层为创建复杂的动画提供了一个详细周到的组件，或者说是构件。动画有两种基本的类别：

视图动画，当你想给用户界面一个视觉提示时，主要使用这个动画

层动画，这个动画在应用内容和功能性上普遍使用。

有时在用户界面接口上，你想要使用层。然而，这种方式不会普遍使用的，因为层不能够接受像点击、按下的事件。如果你想要在层上处理一些信息，你需要在背后的视图层捕获这些事件，然后传递给这个层。

想象下，一个 iPhone 下的太空入侵游戏。当你倾斜设备的一边时，你创建的飞船精灵会在视图的底部移动。这个精灵在它自己的层上被绘制。在背后的视图上，当一个加速事件是被接受时，你传递这个事件到该层上，然后调用 setPosition 方法来移动到它应该到达的位置。你也可能要接收一个轻拍的事件，以便于发射光电雷打击入侵者。还有一些精灵存在于他们自己的层上。

你会看到，在一个游戏里积聚很多数量的层是多么的快！在这种应用程序里，层是一个恰当的选择，因为他们是轻量级的，并且执行效率高。你当然可以用视图替换层，也能很好的执行。然而，当这些东西（层或者视图）在屏幕上达到一定程度时，在视图上使用层，你会获得更好的性能。当然 OS X 会比在 IPhone 上有更少的问题，但是保证你的应用程序轻量、高效总是一个好的方案。

1.5. 动画和层

在核心动画中，时间无处不在。当你需要一个层有动画属性时，你需要考虑动画的时长，如何动，是否要返回到初始的值，以及在动画其间属性应该有多少个不同的值（基础动画与关键帧动画）。通过设定动画属性的值，可以让你指定动画的不同属性。然而，这些值都不能被层本身指派，而是在 CAAnimation 这个对象或者它的子类中，CABasicAnimation、CAKeyframeAnimation 里指派的。

再次重申，在 MVC 设计模式中，层是一个模型对象，而不是一个视图对象。层包含了如下的属性：位置，轮廓，颜色等等。简单的说，尽管一个动画只要描述动画本身，但是你设定的动画属性也关系到你正在做动画的层。你可以通过调用 addAnimation:forKeyPath 这个方法，来增加动画对象到层中，这样就能触发动画了。同样的，你可以通过调用 removeAnimation:forKeyPath 来停止层中的动画，不过在默认情况下，动画运行完毕就回自动的执行以上行为。

1.6. 动画步调

像前面提到的，核心动画给你了大量的免费函数。假如你不用核心动画做动画，你需要使用一个循环来迭代属性的值。通常都是用线性的改变来循环迭代的。也就是说动画是恒速的。动画的每一步，就是走了总增量的一个子增量。在核心动画中，默认的动画是渐进渐出的。这里的意思是，动画开始的时候慢，然后在中间加速，之后当靠近目的地的时候再减速。这个渐进的概念让动画更接近自然平滑，然而一个线性的动画更多的是一个静态的感觉，因为这些属性的变化都是恒定的。

当你提到动画步调时，会发现核心动画设定是多么的快捷。核心动画的步调功能，满足了很多的应用的需求。这些步调功能包括：

线性，是用来给你一个相对静态的感觉。
渐进，你的动画缓慢进入，然后加速离开。
渐出，动画全速进入，然后减速的到达目的地。
渐进渐出，你的动画缓慢的进入，中间加速，然后减速的到达目的地。这个是默认的动画行为。
你只需简单的指定这些步调函数，所有的计算都会自动的做出。然而，如果你想到得到更多的步调控制，你通过使用关键帧动画，指定一个时间函数和次数的数组。关于动画的步调。在第四章中可以了解更多。

1.7. 数学？我将告诉你这里没有数学

当你第一次接触核心动画时，你可能会想需要使用一些三角函数进行一些复杂的缩放和旋转变换。幸运的是这些都不需要。缩放和旋转一些很多其他属性，都只需一点代码就能搞定。你只需要指定这些属性的开始值和结束值，然后剩下的就可以交给核心动画了。
在我的核心动画的博客里，我解释过怎么去缩放一簇层。我也提及了仪表盘的效果。我最近实现了这个仪表盘的效果。当增加一个小工具到仪表盘上时，人们往往好奇这些链锁反应的效果。我的意思就是，当你使用一个键或者点击，调出仪表盘时，你看到的那个效果，不过这个也依赖于你 Mac 操作系统的配置。当你打开仪表盘时，小工具似乎是渐变的飞向了你屏幕的四周。当你关闭时，这些小工具会渐渐的飞离。
起初，我打算为每一个层配置一个缩放的方程，之后我发现核心动画层使用了一个继承树，并且逐层的影响它的子层。这意味着你仅仅把一簇子层增加到一个更大的父层上，然后简单的缩放父层，所有的子层都回自动的缩放。你可以通过表 1-1 的代码完成这个效果

表 1-1 缩放图层

```
(void)doItIn:(float)duration
{
// Set the animation duration
[[NSAnimationContext currentContext] setDuration:duration];
if( scaleUp )
{
// Scale everything (x, y, and z) by a factor of
// factor (static variable) and reduce the opacity
CATransform3D transform = CATransform3DMakeScale(factor,
factor,
factor);
[mainLayer setTransform:transform];
mainLayer.opacity = 0.0f;
scaleUp = NO;
}
else
{
// Scale the rect back down to the original and bring up
// the opacity
CATransform3D transform = CATransform3DMakeScale(1.0f,
1.0f,
1.0f);
[mainLayer setTransform:transform];
mainLayer.opacity = 1.0f;
scaleUp = YES;
}
}
```

这个表展示的仅仅是父层的转换。增加子层也是非常的简单，我已经在博客中提供了这章代码的资源，你可以看下完整的代码如何实现这个效果。
对于许多人来说，使用复杂的数学是头疼的，但是作为苹果免费的午餐，使用核心动画你不需要知道很多数学方面的知识。你仅仅需要知道基本的四则运算。它就是这么简单。

1.8. 核心动画和 iPhone

你读这本书，可能因为想要做 iPhone 应用程序下的动画。我特地写了一个章节在 iPhone 上使用核心动画，以帮助你理解和桌面应用的一些不同，这些不同很小，你通常都可以在 iPhone 上使用同样的代码。下面的几章中，你会了解到更多核心动画的特性，之后再讨论 iPhone 上和桌面上不同才更有意义。不过在某些特殊层的章节，我们就不讨论 iPhone 了，因为这些层在 iPhone 上不可用。

1.9. 示例工程的注意事项

这本书中所有的示例工程都有一个主应用程序的代理类。在 OS X 上，我们使用名字 AppDelegate 创建这个类。在 IPhone 工程中，工程模板帮我们自动创建了这个应用程序代理，名字是<project_name>AppDelegate，其中<project_name>就是你项目的名字。

在 OS X 上代码主要是放在 AppDelegate 这个类里。而在 IPhone 里，代码主要放在了视图控制器类中（view Contrller），这个控制器主要是用来控制视图开发的，在应用程序启动时，应用程序的代理会安装这些类。尽管都很重要，但是还是有点微小的差别。

1.10. 在 OS X 上安装 Xcode 工程

当你在 Xcode3.1 版本或者更早的版本上创建工程，工程模板不会为你自动的创建应用程序的代理类，在 Xcode 3.2 或者之后的版本都会自动创建的。尽管增加一个代理类并不困难，但是当你不利用任何模板代码时，你要知道如何从最开始创建一个工程。你可以使用下面的步骤去创建一个应用程序代理和增加一个 QuartZCore 框架，这个框架提供了核心动画的类。

1. 在 Xcode 中，按 shift-command-N，然后在工程模板对话框中选择 Cocoa Application。
2. 命名工程，然后点击保存。
3. 展开框架组，按住 Control 建点击框架的子组，然后选择 Add -> New File.
4. 在结果对话框中，导航到/System/Library/Frameworks 和选择 QuartzCore.framework。点击增加和当提示时再次增加
5. 按住 Control 点击 Class group，选择 add->New File.
6. 在新文件模板窗口中，在 Cocoa 组选择 Objective-c 类点击 Next。
7. 命名文件为 AppDelegate.m 然后确保也创建了 AppDelegate.h。点击完成。
8. 选择 Appdelegate.h 用代码编辑器打开，导入 QuartzCore 框架并且在你的 XIB 文件上，为窗口创建一个插座：

```
#import <QuartzCore/QuartzCore.h>
@interface AppDelegate : NSObject
{
IBOutlet NSWindow *window;
}
```

9. 选择 AppDelegate.m，然后打开，增加下面的代码，给你的窗口视图开启层：

```
@implementation AppDelegate
(void)awakeFromNib;
{
[[window contentView] setWantsLayer:YES];
}
@end
```

10. 在 Resource group 下，双击 MainMenu.xib 打开。
11. 从 Library 模板中，拖一个 NSObject 对象到 MainMenu.xib 中，然后重命名为 AppDelegate。你需要在图标视图中重命名这个对象。点击一次是选择这个对象。继续再点击一次就可以编辑了。然后就重命名这个对象
12. 确保你的 AppDelegate 对象是被选择。在对象检查器中，点击定义的 tab 并且改变类的区域到 AppDelegate

13. 在 MainMenu.xib 中，在 File's Owner 上左键，然后拖动链接到 AppDelegate 对象上，选择代理。
14. 在 MainMenu.xib 中，左键 AppDelegate 和拖动链接到 window 对象。选择 window。
15. 保存 XIB 文件，返回 Xcode。

在 OS X 上，这些是基础的去创建工程的步骤。利用这个模板，你可以在 AppDelegate 类中增加 action 和 outlets，然后链接到 XIB 里。我们整本书都会基于这种工程。

1.11. 总结

在 Mac OS 上，核心动画是一个巨大的革命。其他的操作系统一直在模仿，但是从来未超越。核心动画免费的提供给我们是一件令人兴奋的事。需要一个好的想法，仅仅一点努力，你就能做出一个令人惊讶的应用程序。核心动画给予的超出了你的期望，并且我们也希望在接下来的版本中，看到更加令人兴奋的特性。