

23 | 如何构造酷炫的物理效果和过场动画效果?

戴铭 2019-05-02



00:00

讲述：冯永吉

大小：8.94M

09:45

你好，我是戴铭。今天，我要和你分享的是如何为你 App 添加酷炫的动画效果。

不论是 iOS 开发，还是 Android 开发，现在的动画库差不多都需要手动去编写动画代码。这样的话，iOS 和 Android 开发者就需要分别去编写适合自己系统的代码。而且，手动编写动画的代码也非常复杂，不容易维护，很多动画细节的调整还需要和动画设计师不断沟通打磨，尤其是千行以上的动画代码编写、维护、沟通的成本巨大。

手动编写动画代码，除了会影响到开发者外，动画设计师也难以幸免。一款产品适配的平台越多，动画设计师设计走查的周期就越长，相应的动画成本就越高。同时，动画设计师很兴奋地设计出一套炫酷地动画效果后，在要通过开发者实现出来时，却因为工时评估过长而一再被简化，甚至被直接取消。试想一下，以后他还会动力十足地去设计酷炫的动画效果吗？

所以，你会发现现在有酷炫的动画效果的 App 非常少，而且多是出自个人开发者之手。那么，这就提高了对个人开发者的要求，不但要求他代码写得好，还要能够设计出好的动画效果。但是，这样的人才也是不可多得。

那，到底有没有什么办法能够把动画制作和 App 开发隔离开，专人做专事，而且还能使得多个平台的动画效果保持一致呢？

办法总比困难多。接下来，我们就一起来看看如何实现的问题吧。

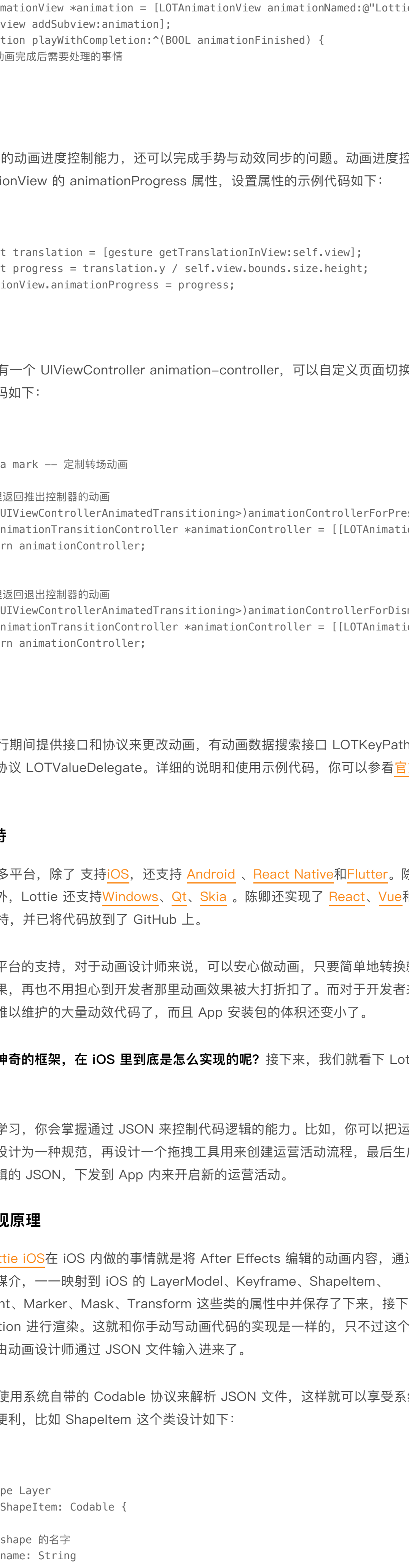
Lottie

[Lottie 框架](#)就很好地解决了动画制作与开发隔离，以及多平台统一的问题。

Lottie 是 Airbnb 开源的一个动画框架。Lottie 这个名字来自于一名德国导演洛特·赖尼格（Lotte Reiniger），她最著名的电影叫作“阿赫迈德王子历险记（The Adventures of Prince Achmed）”。这个框架和其他的动画框架不太一样，动画的编写和维护将由动画设计师完成，完全无需开发者操心。

动画设计师做好动画以后，可以使用[After Effects](#)将动画导出成 JSON 文件，然后由 Lottie 加载和渲染这个 JSON 文件，并转换成对应的动画代码。由于是 JSON 格式，文件也会很小，可以减少 App 包大小。运行时还可以通过代码控制更改动画，比如更改颜色、位置以及任何关键值。另外，Lottie 还支持页面切换的过场动画（UIViewController Transitions）。

下面的两张动画，就是使用 Lottie 做出来的效果。



上面这些动画，就是由动画设计师使用 After Effects 创作，然后使用 [Bodymovin](#)进行导出的，开发者完全不用做什么额外的代码工作，就能够使用原生方式将其渲染出来。

Bodymovin 是 Hernan Torrisi 做的一个 After Effects 的插件，起初导出的 JSON 文件只是通过 JavaScript 在网页中进行动画的播放，后来才将 JSON 文件的解析渲染应用到了其他平台上。

那么，如何使用 Bodymovin 呢？

Bodymovin

你需要先到[Adobe 官网](#)下载 Bodymovin 插件，并在 After Effects 中安装。使用 After Effects 制作完动画后，选择 Windows 菜单，找到 Extensions 的 Bodymovin 项，在菜单中选择 Render 按钮就可以输出 JSON 文件了。

[LottieFiles 网站](#)还是一个动画设计师分享作品的平台，每个动画效果的 JSON 文件都可下载使用。所以，如果你现在没有动画设计师配合的话，可以到这个网站去查找并下载一个 Bodymovin 生成的 JSON 文件，然后运用到工程中去试试效果。

在 iOS 中使用 Lottie

在 iOS 开发中使用 Lottie 也很简单，只要集成 Lottie 框架，然后在程序中通过 Lottie 的接口控制 After Effects 生成的动画 JSON 就行了。

首先，你可以通过 CocoaPods 集成 Lottie 框架到你工程中。Lottie iOS 框架的 GitHub 地址是<https://github.com/airbnb/lottie-ios/>，官方也提供了[可供学习的示例](#)。

然后，快速读取一个由 Bodymovin 生成的 JSON 文件进行播放。具体代码如下所示：

```
1 LottieAnimationView *animation = [LottieAnimationView animationNamed:@"Lottie"];
2 [self.view addSubview:animation];
3 [animation playWithCompletion:^(BOOL animationFinished) {
4     // 动画完成后需要处理的事情
5 }];
6
```

利用 Lottie 的动画进度控制能力，还可以完成手势与动效同步的问题。动画进度控制是 LottieAnimationView 的 animationProgress 属性，设置属性的示例代码如下：

```
1 CGPoint translation = [gesture getTranslationInView:self.view];
2 CGFloat progress = translation.y / self.view.bounds.size.height;
3 animationView.animationProgress = progress;
4
```

Lottie 还带有一个 UIViewController animation-controller，可以自定义页面切换的过场动画，示例代码如下：

```
1 #pragma mark -- 定制转场动画
2
3 // 代理返回推出控制器的动画
4 - (id<UIViewControllerAnimatedTransitioning>)animationControllerForPresentedCont
5     LottieAnimationTransitionController *animationController = [[LottieAnimationTransiti
6     return animationController;
7 }
8
9 // 代理返回退出控制器的动画
10 - (id<UIViewControllerAnimatedTransitioning>)animationControllerForDismissedCont
11     LottieAnimationTransitionController *animationController = [[LottieAnimationTransiti
12     return animationController;
13 }
14
```

Lottie 在运行期间提供接口和协议来更改动画，有动画数据搜索接口 LOTKeyPath，以及设置动画数据的协议 LOTValueDelegate。详细的说明和使用示例代码，你可以参看[官方 iOS 教程](#)。

多平台支持

Lottie 支持多平台，除了支持iOS，还支持 [Android](#)、[React Native](#)和[Flutter](#)。除了官方维护的这些平台外，Lottie 还支持[Windows](#)、[Qt](#)、[Skia](#)。陈卿还实现了 [React](#)、[Vue](#)和[Angular](#)对 Lottie 的支持，并已将代码放到了 GitHub 上。

有了这么多平台的支持，对于动画设计师来说，可以安心做动画，只要简单地转换就可以完美展现动画效果，再也不用担心到开发者那里动画效果被大打折扣了。而对于开发者来说，再也不用写那些难以维护的大量动效代码了，而且 App 安装包的体积还变小了。

那么，**这么神奇的框架，在 iOS 里到底是怎么实现的呢？**接下来，我们就看下 Lottie 的实现原理吧。

通过原理的学习，你会掌握通过 JSON 来控制代码逻辑的能力。比如，你可以把运营活动流程的代码逻辑设计为一种规范，再设计一个拖拽工具用来创建运营活动流程，最后生成一份表示运营活动逻辑的 JSON，下发到 App 内来开启新的运营活动。

Lottie 实现原理

实际上，[Lottie iOS](#)在 iOS 内做的事情就是将 After Effects 编辑的动画内容，通过 JSON 文件这个中间媒介，——映射到 iOS 的 LayerModel、Keyframe、Shapeltem、DashElement、Marker、Mask、Transform 这些类的属性中并保存了下来，接下来再通过 CoreAnimation 进行渲染。这就和你手动写动画代码的实现是一样的，只不过这个过程的精准描述，全部由动画设计师通过 JSON 文件输入进来了。

Lottie iOS 使用系统自带的 Codable 协议来解析 JSON 文件，这样就可以享受系统升级带来性能提升的便利，比如 Shapeltem 这个类设计如下：

```
1 // Shape Layer
2 class ShapeItem: Codable {
3
4     // shape 的名字
5     let name: String
6
7     // shape 的类型
8     let type: ShapeType
9
10    // 和 json 中字符映射
11    private enum CodingKeys : String, CodingKey {
12        case name = "nm"
13        case type = "ty"
14    }
15    // 初始化
16    required init(from decoder: Decoder) throws {
17        let container = try decoder.container(keyedBy: Shapeltem.CodingKeys.self)
18        self.name = try container.decodeIfPresent(String.self, forKey: .name) ?? ""
19        self.type = try container.decode(ShapeType.self, forKey: .type)
20    }
21
22 }
23
24
```

通过上面代码可以看出，Shapeltem 有两个属性，映射到 JSON 的字符串键值是 nm 和 ty，分别代表 shape 的名字和类型。下面，我们再一起看一段 Bodymovin 生成的 JSON 代码：

```
1 {"ty":"st","fillEnabled":true,"c":{"k":[{"i":{"x":0.833},"y":0.833},"o":{"x":
2
3
```

在这段 JSON 代码中，nm 键对应的值是 Stroke 1，ty 键对应的值是 st。那我们再来看看，**st 是什么类型**。

我们知道，ShapeType 是个枚举类型，它的定义如下：

```
1 enum ShapeType: String, Codable {
2     case ellipse = "el"
3     case fill = "fl"
4     case gradientFill = "gf"
5     case group = "gr"
6     case gradientStroke = "gs"
7     case merge = "mm"
8     case rectangle = "rc"
9     case repeater = "rp"
10    case round = "rd"
11    case shape = "sh"
12    case star = "sr"
13    case stroke = "st"
14    case trim = "tm"
15    case transform = "tr"
16 }
17
```

通过上面的枚举定义，可以看到 st 对应的是 stroke 类型。

Lottie 就是通过这种方式，定义了一系列的类结构，可以将 JSON 数据全部映射过来。所有映射用的类都放在 Lottie 的 Model 目录下。使用 CoreAnimation 渲染的相关代码都在 NodeRenderSystem 目录下，比如前面举例的 Stoke。

在渲染前会生成一个节点，实现在 StrokeNode.swift 里，然后对 StokeNode 这个节点渲染的逻辑在 StrokeRenderer.swift 里。核心代码如下：

```
1 // 设置 Context
2 func setupForStroke(_ inContext: CGContext) {
3     inContext.setLineWidth(width) // 行宽
4     inContext.setMiterLimit(miterLimit)
5     inContext.setLineCap(LineCap.cgLineCap) // 行间隔
6     inContext.setLineJoin(lineJoin.cgLineJoin)
7     // 设置线条样式
8     if let dashPhase = dashPhase, let lengths = dashLengths {
9         inContext.setLineDash(phase: dashPhase, lengths: lengths)
10    } else {
11        inContext.setLineDash(phase: 0, lengths: [])
12    }
13 }
14
15 // 渲染
16 func render(_ inContext: CGContext) {
17     guard inContext.path != nil && inContext.path!.isEmpty == false else {
18         return
19     }
20     guard let color = color else { return }
21     hasUpdate = false
22     setupForStroke(inContext)
23     inContext.setAlpha(opacity) // 设置透明度
24     inContext.setStrokeColor(color) // 设置颜色
25     inContext.strokePath()
26 }
27
```

这段代码看起来是不是就很熟悉了？

如果是手写动画，这些代码就需要不断重复地写。使用第三方库去写动画的话，也无非就是多封装了一层，而属性的设置、动画时间的设置等，还是需要手动添加很多代码来完成。

但是，使用 Lottie 后，你就完全不用去管这些代码了，只需要在 After Effects 那设置属性、控制动画时间就好了。

小结

今天这篇文章，我分享了一个制作动画的利器 Lottie，并和你说了如何在 iOS 中使用，以及它的实现原理。听到这，你一定感到奇怪，iOS 开发中还有很多优秀的动画框架，比如 Pop，但是为什么我只跟你说了 Lottie 呢？

因为在我看来，Lottie 这样的工作流程或许就是未来的趋势，就像 iOS 现在的发展趋势一样，越来越多的业务逻辑不再需要全部使用 Objective-C 或 Swift 来实现了，而是使用 JavaScript 语言或者 DSL 甚至是工具来描述业务，然后将描述业务的代码转换成一种中间代码，比如 JSON，不同平台再对相同的中间代码进行解析处理，以执行中间代码描述的业务逻辑。

这样做不仅可以减轻 App 包的大小，实现多端逻辑的统一处理，还可以让团队分工更加明确，一部分人专门开发业务代码，另一部分人负责端内稳定性、质量把控、性能提升工作的建设。

课后作业

相信你看到这，一定已经忍不住想小试身手了，那么就请你到 [LottieFiles](#)网站下载一个 JSON 文件，做一个 Lottie Demo 感受一下吧。

感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎把它分享给更多的朋友一起阅读。

iOS 开发高手课

从原理到实战，带你解决 80% 的开发难题

戴铭
前滴滴出行技术专家

新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金奖励**。