iWangKe.me

Oct 6 2014 iWangKe.me

基于Core Animation的KTV歌词视图的平滑实现

KTV歌词视图，只要去过KTV的的朋友一定不会陌生。我们先来看一下最终的效果，再一步步说明唱吧歌词视图的演进。想把事件事情说得清清楚楚的确很难，有很多tricky的地方；另外毕竟不是open source的，只能给大家挑重点分享一下实现的过程和思路。

歌词视图剖析

一个体验良好的歌词视图，由以下方面组成，这也是我们的设计目标：

- 有倒计时功能，歌者可以提前作演唱的准备
- 根据场景的不同，支持多行或者双行显示，为歌者提供演唱的上下文
- 歌者清晰的了解当前在唱哪一句歌词，我称之为焦点行
- 焦点行需要染色，并需要精准地作逐字渲染
- 两句之前使用适当的动画换行过渡
- 歌词动画平滑不突兀，适应不同节奏的歌曲
- 根据产品和设计师的要求，灵活地对歌词视图进行字体、颜色调整（1/3/5是绿色，2/4/6是红色，阴历节日是黄色，I'm serious and it's safe to forget Sunday! Cheers!）

此外我们还需要了解一下歌词信息的结构，大致如下：

```
1   @interface Line : NSObject
2   @property (nonatomic, strong) NSArray *words;
3   @property (nonatomic, strong) NSString *text;
4   @property CGFloat start;
5   @property CGFloat length;
6   @end
7
8   @interface Word : NSObject
9   @property (nonatomic, strong) NSString *text;
10  @property CGFloat start;
11  @property CGFloat length;
12  @end
```

- 一首歌的歌词我们称之为Lyrics
- Lyrics包含多行，每行我们称之为Line; Line有它的start及length，分别代表时间戳以及长度
- Line包含多个字，每个字我们称之为Word; Word也有它的start以及length，分别代表时间戳以及长度

了解完这些我们看看如何来渲染焦点行歌词，先看简单直接的方式。

基于Core Graphics的实现

我们知道歌曲的开始时间，也有歌词数据提供时间支持，那么就可以计算出当前歌词视图的状态。对于歌词的焦点行，有两部分状态：

- 歌者已经演唱的部分，渲染成绿色
- 歌者待演唱的部分，渲染成白色

我们省略计算的过程，假设已经得出绿色、白色歌词的rect及point，就可以直接渲染了：

```
@implementation LyricsView
- (void)drawRect:(CGRect)rect {
    // Assume we calcuated them before
    CGRect greenRect;
    CGRect whiteRect
    CGPoint greenPoint;
    CGPoint whitePoint;

    // We have the focus line and font
    Line *line;
    UIFont *font;

    // Render focus line text
    CGContextRef context = UIGraphicsGetCurrentContext();

    CGContextSaveGState(context);
    CGContextClipToRect(context, greenRect);
    [[UIColor greenColor] set];
    [line.text drawAtPoint:greenPoint withFont:font];
    CGContextRestoreGState(context);

    CGContextSaveGState(context);
    CGContextClipToRect(context, whiteRect);
    [[UIColor whiteColor] set];
    [line.text drawAtPoint:whitePoint withFont:font];
    CGContextRestoreGState(context);
}
@end
```

本质上我们使用了NSString的UIStringDrawing Category搞定了这个事情。既然我们解决了任一时间点的状态，那么把它动起来也很容易：

- 将这段code snippet放到LyricsView的drawRect中
- 以60 FPS的频率调用[lyricsView setNeedsDisplay]

一切看起来很直观，但问题来了，这个歌词视图根本跑不到60 FPS（我保证这个效果看起来像癫痫一样儿，v4.9之前就是一直这么癫过来的），即使在目前性能最强的iPhone 5S上。我们来分析一下原因：

- Core Graphics使用CPU作渲染
- 这个界面是CPU intensive，需要播放伴奏，还需要录制歌者的声音，甚至需要给声音加“滤镜”
- 还有对歌者进行实时打分的task及动画
- 回望过去5年iPhone的硬件发展，GPU的提升也远高于CPU，不能指望短期设备升级解决这个问题

5S上毕竟还可以跑到50FPS，但低端设备的FPS对我来讲是实在是没法接受的。唱吧是线上KTV

的应用的用户体验标准，不解决这个问题是说不过去的。既然CPU不给力，那么我们让GPU来做这件事情。

基于Core Animation的实现

14年初的时候，Facebook open source了惊艳的Shimmer。由于跟我设想的实现机制是相同的，直接拖了几百个shimmer view作了一下profile，在4S上都可以达到完美的60FPS。

让我们先理一下思路，看看基于Core Animation的焦点行的视图结构：

```
1   - GreenLineLabel: UILabel
2   - WhiteLineLabel: UILabel
```

没错，就是简单的把绿色的UILabel置于白色的之上，剩下的问题就是如何控制绿色的UILabel按我们的时间控制进行部分渲染。

部分渲染就是加一个mask，我们来看一下CALayer的mask property:

```
1   @interface CALayer : NSObject <NSCoding, CAMediaTiming>
2   /* A layer whose alpha channel is used as a mask to select between the
3    * layer's background and the result of compositing the layer's
4    * contents with its filtered background. Defaults to nil. When used as
5    * a mask the layer's `compositingFilter' and `backgroundFilters'
6    * properties are ignored. When setting the mask to a new layer, the
7    * new layer must have a nil superlayer, otherwise the behavior is
8    * undefined. Nested masks (mask layers with their own masks) are
9    * unsupported. */
10  @property(strong) CALayer *mask;
11  @end
```

我们可以知道，mask layer的alpha用来与CALayer的content进行alpha blending，如果alpha为1则content显示，反之不显示。受Shimmer的启发，我们可以对mask作动画，让它从左到右移动到绿色歌词的layer上，并最终与之重合。

```
1   @interface GreenLineLabel: UILabel
2   @end
3   @implementation GreenLineLabel {
4       CALayer *_maskLayer;
5   }
6   - (instance)initWithFrame:(CGRect)frame {
7       self = [super initWithFrame:frame];
8       if (self) {
9           _maskLayer = [CALayer layer];
10          _maskLayer.backgroundColor = [[UIColor whiteColor] CGColor];
11          _maskLayer.anchorPoint = CGPointZero;
12          _maskLayer.frame = CGRectOffset(self.frame, -CGRectGetWidth(sel
13          self.layer.mask = _maskLayer;
14          self.backgroundColor = [UIColor clearColor];
```

```
    }
    return self;
}
```

上面这段代码我们将_maskLayer的anchorPoint设置为CGPointZero，便于后面的动画计算坐标。

下面我们对_maskLayer的position作CAKeyframeAnimation动画，根据歌词数据我们可以算出每个字渲染的时间(keyTimes)和动画总时长(duration)。假设每个字是等宽的，我们可以算出_maskLayer在每一个keyTime的position，也就是values。

```
- (void)startAnimation {
    // Assume we calculated keyTimes and values
    NSMutableArray *keyTimes;
    NSMutableArray *values;
    CGFloat duration;
    CAKeyframeAnimation *animation = [CAKeyframeAnimation animationWith
    animation.keyTimes = keyTimes;
    animation.values = values;
    animation.duration = duration;
    animation.calculationMode = kCAAnimationLinear;
    animation.fillMode = kCAFillModeForwards;
    animation.removedOnCompletion = NO;
    [_maskLayer addAnimation:animation forKey:@"MaskAnimation"];
}
```

至此我们完成了基于Core Animation的歌词焦点行染色动画。

写在后面

很抱歉我提供的code snippet不是production ready，歌词动画是一个非常复杂的系统，很难单独抽离出来介绍给大家，所以只能管窥一豹地介绍下。

附小广告一则：唱吧iOS团队诚招iOS工程师，推荐成功即奖励6000元现金或iPhone 6一部，详见这篇blog。

#ChangBa #iOS

NEWER
为什么唱吧iOS 6.0选择了Mantle

OLDER
唱吧诚聘iOS开发工程师

传人 尚 • 7个月前

使用drawRect肯定慢，每次View的backing Layer都要分配View大小的backing store，用Core Graphics生成位图写入backing store，然后再上传到GPU形成纹理显示。分配缓存，上传GPU都慢，而且字体的渲染很慢的，1/60s(60FPS)很难达到。即使通过setNeedsDisplayInRect指定dirty区域也是慢。看看需求，其实在动画过程中视图的内容并没有变，变化的是颜色，而颜色、阴影之类的效果GPU是支持的，这种情况下用Core Animation是合适的。使用mask虽然会导致离屏渲染，但是两个Label只需要上传到GPU一次，剩下的动画交给GPU去完成，GPU对mask合成跟时间函数是天生优化的，所以会很流畅。对于动画效果，我记得苹果的文档说过最好不要重绘+自己设定Timer实现，这样卡到爆，而且苹果也建议尽量少用drawRect。当然，很多文档建议：在做优化之前一定要用Instrument跑一跑，找出性能瓶颈后再有针对的优化，靠猜测不靠谱，所以文章中的做法是对的。关于Core Animation有本《iOS Core Animation: Advanced Techniques》，看完功力绝对大涨。PS. 写了这么多，只想问一句：还招人不?

1 • 回复 • 分享 ›

Ke 管理员 传人 尚 • 7个月前

招！邮件联系我，吃顿饭先

• 回复 • 分享 ›

lesvio • 7个月前

有代码demo可以看看吗?有的话发我邮箱,951974984@qq.com,谢谢了

• 回复 • 分享 ›

lovesunstar • 9个月前

使用UIView没有那么低的，你写的有问题，并且不是每次调用setNeedsDisplay都会调用drawRect。 使用UIView也可以两个Label分别两层(定为顶层和底层)，上一层为绿色，下一层为白色，这样可以做到文字可以渲染0.5宽度，而你所做的只需要修改顶层Label的width，以及设置clipsToBounds而已。

• 回复 • 分享 ›

Ke 管理员 lovesunstar • 9个月前

能跑到多少FPS? 可以假设CPU不做任何其他事情？理想条件下的Demo没意义。或者给个benchmark?

Best Regards

• 回复 • 分享 ›

lovesunstar Ke • 9个月前

你理解错了，不管是哪种方式，CPU在处理其他事情，只是看CPU做这的这些事情足不足以导致帧数变少或者帧数变卡，就算是使用CALayer，也需要用到CPU，只是一种解决方案，不代表唯一的解决方案，以前做过那种音频的实时滤镜以及可视化效果，使用OpenGL或者其他方式的GPU处理固然没有问题，但是使用UIView.frame的解决方式并没有你所写的那么差

• 回复 • 分享 ›

Ke 管理员 ➚ lovesunstar • 9个月前

单纯就一行歌词的染色来讲，你的方案没问题。我们当时重写是比较激进的，因为以前的drawRect实现问题很多，不仅限于性能方面:)

• 回复 • 分享 ›

Ke 管理员 ➚ lovesunstar • 9个月前

能跟我邮件联系下么？ewangke # gmail.com

Best Regards

• 回复 • 分享 ›

wu • 9个月前

大神，你qq号多少啊，请教一些问题，不便的话加我747709993，谢谢

• 回复 • 分享 ›

Ke 管理员 ➚ wu • 9个月前

ewangke#gmail.com, 邮件吧

• 回复 • 分享 ›

Roger Ren • 9个月前

请问怎么可以知道在5S上还可以跑到50FPS，这个帧数的值是用什么方法测出来的啊?

• 回复 • 分享 ›

Ke 管理员 ➚ Roger Ren • 9个月前

Instruments

Best Regards

1 • 回复 • 分享 ›

terryso • 9个月前

很赞。。。之前用渐变层实现过一个KTV的效果，但不是很理想，准备根据你这篇文章改进一下。

• 回复 • 分享 ›

嘿嘿 ➚ terryso • 9个月前

大神，求您的demo

回复 • 分享 ›

Ke 管理员 → terryso • 9个月前

Gradient layer?

Best Regards

回复 • 分享 ›

terryso → Ke • 8个月前

是的。。。

回复 • 分享 ›

answerhuang • 10个月前

受教。

回复 • 分享 ›

BeyondVincent(破船) • 10个月前

多年前我在Symbian上面写了一个http://www.devdiv.com/mp_-thre...

回复 • 分享 ›

Ke 管理员 → BeyondVincent(破船) • 10个月前

ORZ..,不过打不开啊，跳到首页了

回复 • 分享 ›

June → Ke • 9个月前

他给的是这个链接 http://www.devdiv.com/mp_-thre...

回复 • 分享 ›

Coneboy • 10个月前

去唱吧了！你不再是一个人战斗了！:D

回复 • 分享 ›

Ke 管理员 → Coneboy • 10个月前

是啊，长期宅在家里太难受了

回复 • 分享 ›

大明湖畔小小猪 • 10个月前

学到了裁切图片的方法，受教了。不过我更想知道如何使显示文字与歌曲精确匹配。

回复 • 分享 ›

Ke 管理员 → 大明湖畔小小猪 • 10个月前

根据歌词lrc数据作动画即可:)

回复 • 分享 ›

xiaoming guo • 10个月前

赞! 但是有必要达到60FPS的刷新频率么?

回复 • 分享 ›

Ke 管理员 ➚ xiaoming guo • 10个月前

有必要，差异是肉眼可见的

回复 • 分享 ›

zoe • 10个月前

赞!

回复 • 分享 ›

在 **IWANGKE.ME** 上还有.....

这是什么?

Search

IWANGKE.ME

Modify from Apollo theme, designed by SANOGRAPHIX.NET

Powered by Hexo