跑马灯动画

1. 简述：跑马灯在 app 中一般用来展示警示、提示性语句，循环地跑动指定的文字来给予用户提醒。跑马灯的实现在安卓中系统有相应的封装，只要简单调用即可。Ios 则需要开发者自己去封装了。

2. 原理：跑马灯的原理是什么呢？动画效果上来讲，是一行文本的滚动，在没结束但快结束前，文本又重头接了进来。那么，从实现上来说呢？没错，我们需要两个一样的文本，在一个文本快滚动结束时让第二个文本跑起来，第一个文本跑动结束后复位并等待第二个文本滚动结束，这样无限循环下去就是所谓的跑马灯动画。

3. 思路：知道了跑马灯的原理后就可以开始探究实现方案了。当然，方案没有唯一性，肯定有各种各样的实现。以下就简单介绍下本人想到的几种思路。

   (1) 思路 1:两个文本，启用两个定时器，定义走一个屏幕文字所需时间得出时长，从而得出一秒定时器触发移动多少距离；在第一个定时器即将完成时开启第二个定时器，让第二个文本跑起来；在第一个定时器完成后第一个文本及定时器复位；待第二个定时器即将完成时开启第一个定时器，让第一个文本跑起来；循环下去。

   (2) 思路 2:两个文本，不用定时器，定义走一个屏幕文字所需时间得出时长，的出整个文本走完所需总时长；定义动画块，通过指定动画时间内改变两个文本位置结合动画块的时间延迟参数来实现两个文本交替进行动画；循环采用递归调用，也就是两个文本动画结束，调用自身让自己延迟指定 s 后继续做动画；

   (3) 思路 3:一个文本，不用定时器，定义走一个屏幕文字所需时间得出时长，的出整个文本走完所需总时长；定义 layer 层 position 动画；通过指定动画时间结束时新增文本图层动画结合图层动画的时间延迟参数来实现前后两个图层动画交替进行；

4. 实现：以下就 3 中第(2)个思路做具体实现，有兴趣的可以尝试下另外两种思路。

```
#import <UIKit/UIKit.h>

//----------------------------------RunHorseLampViewOptions---------------------------------

@interface RunHorseLampViewOptions : NSObject

@property (assign, nonatomic) CGFloat duration_per_width;
@property (strong, nonatomic) UIColor *textColor;
@property (strong, nonatomic) UIFont *font;
@property (strong, nonatomic) UIColor *backgroundColor;

/**
* 初始化方法
*
* @ param duration    可选配置项，iphone5下文本移动一个 跑马灯宽度所需的时间，默认为3s，其它设备会自动适配出合适时间
* @ param textColor   可选配置项，跑马灯文本的颜色，默认为白色
* @ param font    可选配置项，跑马灯文本的字体，默认为15号字体
* @ param backgroundColor     可选配置项，跑马灯背景颜色，默认为透明度百分80的黑色
*/

+ (instancetype)optionsWithDuration:(CGFloat)duration textColor:(UIColor *)textColor textFont:(UIFont *)font backgroundColor:(UIColor *)backgroundColor;

@end
```

```
//----------------------------------RunHorseLampViewOptions---------------------------------

@interface RunHorseLampViewOptions ()

@property (assign, nonatomic) BOOL runLabelIsRun;//记录runLabel的动画状态
@property (assign, nonatomic) BOOL runLabelV2IsRun;//记录runLabelV2的动画状态

@property (assign, nonatomic) CGFloat timeInset;//两个文本间隔时间，即后面的文本在前面文本跑完前几秒接上去跑，决定文本距离
@property (assign, nonatomic) CGFloat duration;//整个text滚动需要的时间

@property (assign, nonatomic) BOOL shouldStop;//记录用户停止操作(YES)，默认为NO，仅当runLabelIsRun和runLabelV2IsRun均为NO时复位

@end

@implementation RunHorseLampViewOptions

+ (instancetype)optionsWithDuration:(CGFloat)duration textColor:(UIColor *)textColor textFont:(UIFont *)font backgroundColor:(UIColor *)backgroundColor{
    RunHorseLampViewOptions *options = [[RunHorseLampViewOptions alloc] init];

    options.duration_per_width = duration <= 0 ? ShiPei(3) : ShiPei(duration);
    options.textColor = textColor == nil ? [UIColor whiteColor] : textColor;
    options.font = font == nil ? [UIFont systemFontOfSize:15] : font;
    options.backgroundColor = backgroundColor == nil ? [UIColor colorWithRed:0 green:0 blue:0 alpha:0.8] : backgroundColor;

    options.timeInset = options.duration_per_width * (5.0 / 6);//跑马灯label宽度为文本宽加一个屏幕的宽度
    return options;
}

@end
```

```
//----------------------------------RunHorseLampView---------------------------------

typedef void (^RuningStateChangedBlock) (BOOL isRuning);//跑马灯状态跟踪回调，仅在stopType为Normally时有效

typedef NS_ENUM(NSUInteger, RunHorseLampStopType) {
    RunHorseLampStopNormally,//正常模式，stopRuning调用后跑马灯不会立即停止，而是继续跑完
    RunHorseLampStopImediately//立即模式，stopRuning调用后跑马灯会立即停止
};

@interface RunHorseLampView : UIView

@property (strong, nonatomic) id text;
@property (assign, nonatomic) RunHorseLampStopType stopType;
@property (copy, nonatomic) RuningStateChangedBlock runingStateChangedBlock;
```

```
/**
 * 初始化方法
 *
 * @ param frame   跑马灯的位置和大小
 * @ param text    初始化文本
 * @ param options    跑马灯相关可选配置项
 */
- (instancetype)initWithFrame:(CGRect)frame text:(id)text options:(RunHorseLampViewOptions *)options;//初始化

/**
 * 开始动画
 */
- (void)startRuning;

/**
 * 结束动画
 *
 * @ note   立即停止或者等当前文本跑完后停止，取决于stopType
 */
- (void)stopRuning;

@end
```

```
//----------------------------------RunHorseLampView---------------------------------

@interface RunHorseLampView ()

@property (strong, nonatomic) UILabel *runLabel;
@property (strong, nonatomic) NSLayoutConstraint *runLabelWidth;
@property (strong, nonatomic) NSLayoutConstraint *runLabelLeft;

@property (strong, nonatomic) UILabel *runLabelV2;
@property (strong, nonatomic) NSLayoutConstraint *runLabelV2Width;
@property (strong, nonatomic) NSLayoutConstraint *runLabelV2Left;

@property (strong, nonatomic) RunHorseLampViewOptions *options;//偏好设置

@end

@implementation RunHorseLampView
```

```
- (instancetype)initWithFrame:(CGRect)frame text:(id)text options:(RunHorseLampViewOptions *)options{
    if(self = [super initWithFrame:frame]){
        self.layer.masksToBounds = YES;
        self.options = options;
        self.backgroundColor = options.backgroundColor;

        [self addSubview:self.runLabel];
        [self addSubview:self.runLabelV2];

        _runLabelLeft = [_runLabel setLayoutLeft:self multiplier:1 constant:0];
        [_runLabel setLayoutTopFromSuperViewWithConstant:0];
        [_runLabel setLayoutBottomFromSuperViewWithConstant:0];
        _runLabelWidth = [_runLabel setLayoutWidth:0];

        _runLabelV2Left = [_runLabelV2 setLayoutLeft:self multiplier:1 constant:0];
        [_runLabelV2 setLayoutTopFromSuperViewWithConstant:0];
        [_runLabelV2 setLayoutBottomFromSuperViewWithConstant:0];
        _runLabelV2Width = [_runLabelV2 setLayoutWidth:0];

        _text = text;
        [self configWithText:text];

        [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(applicationWillResignActive) name:UIApplicationWillResignActiveNotification object:nil];
        [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(applicationDidBecomeActive) name:UIApplicationDidBecomeActiveNotification object:nil];
    }
    return self;
}
```

```
- (void)configWithText:(id)text{
    __block NSString *str = @"";
    if([text isKindOfClass:[NSString class]]){
        if([text length] == 0)
            return;
        str = text;
    }else if([text isKindOfClass:[NSArray class]]){
        if([text count] == 0)
            return;
        [text enumerateObjectsUsingBlock:^(NSString *obj, NSUInteger idx, BOOL * _Nonnull stop) {
            str = [str stringByAppendingFormat:@"%@                ", obj];
        }];
    }
    _text = text;

    [_runLabel setText:str];
    [_runLabelV2 setText:str];

    CGRect rect = [str boundingRectWithSize:CGSizeMake(CGFLOAT_MAX, self.frame.size.height) options:NSStringDrawingUsesLineFragmentOrigin attributes:
        @{NSFontAttributeName:[UIFont systemFontOfSize:15]} context:nil];
    _runLabelWidth.constant = rect.size.width + CGRectGetWidth(self.bounds);
    _runLabelV2Width.constant =  _runLabelWidth.constant;
    [self layoutIfNeeded];

    _options.duration = (_options.duration_per_width * _runLabelWidth.constant / CGRectGetWidth(self.bounds));
}
```

```
#pragma mark public methods

- (void)startRuning{
    if(_options.runLabelIsRun || _options.runLabelV2IsRun)//跑马灯还在跑
        return;

    if(([self.text isKindOfClass:[NSArray class]] && [self.text count] > 0) ||( [self.text isKindOfClass:[NSString class]] && [self.text length] > 0)){
        [self configWithText:_text];
    }else{//文本非法
        [self stopRuning];
        return;
    }

    [self runLabelStartRun:0];
    [self runLabelV2StartRun:_options.duration - _options.timeInset];
}
```

```
- (void)stopRuning{
    switch (_stopType) {
        case RunHorseLampStopNormally:{//只记录状态，在跑马灯一个循环结束后停止
            if(_options.shouldStop == NO){
                _options.shouldStop = YES;
            }
        }
            break;
        case RunHorseLampStopImediately:{//立即停止
            [self applicationWillResignActive];
        }
            break;
        default:
            break;
    }
}
```

```
- (void)runLabelStartRun:(CGFloat)delay{
    _runLabelLeft.constant = -_runLabelWidth.constant;

    _options.runLabelIsRun = YES;
    [self responseBlock:YES];

    __weak typeof(self) weakSelf = self;
    [UIView animateWithDuration:_options.duration delay:delay options:UIViewAnimationOptionCurveLinear animations:^{
        [weakSelf layoutIfNeeded];
    } completion:^(BOOL finished) {
        if(finished == NO)//应用挂起或强制移除动画后finished = NO
            return ;

        weakSelf.runLabelLeft.constant = 0;
        [weakSelf layoutIfNeeded];

        //递归调用临界条件
        if(weakSelf.options.shouldStop == YES){//(1)当动画停止类型为RunHorseLampStopNormally时用户请求停止动画
            weakSelf.options.runLabelIsRun = NO;
            if(weakSelf.options.runLabelV2IsRun == NO){//(1)条件下，两个label动画均结束，通知外界“可以更换文本，重新start了”
                weakSelf.options.shouldStop = NO;
                [weakSelf responseBlock:NO];
            }
            return ;
        }

        [weakSelf runLabelStartRun:weakSelf.options.duration - weakSelf.options.timeInset * 2];//递归调用。参数：确保能在前面文本跑完前kTimeInset秒出发
    }];
}
```

```
- (void)runLabelV2StartRun:(CGFloat)delay{
    _runLabelV2Left.constant = -_runLabelV2Width.constant;

    _options.runLabelV2IsRun = YES;
    [self responseBlock:YES];

    __weak typeof(self) weakSelf = self;
    [UIView animateWithDuration:self.options.duration delay:delay options:UIViewAnimationOptionCurveLinear animations:^{
        [weakSelf layoutIfNeeded];
    } completion:^(BOOL finished) {
        if(finished == NO) //应用挂起或强制移除动画后finished = NO
            return ;

        weakSelf.runLabelV2Left.constant = 0;
        [weakSelf layoutIfNeeded];

        //递归调用临界条件
        if(weakSelf.options.shouldStop == YES){// (1)当动画停止类型为RunHorseLampStopNormally时用户请求停止动画
            weakSelf.options.runLabelV2IsRun = NO;
            if(weakSelf.options.runLabelIsRun == NO){//(1)条件下，两个label动画均结束，通知外界“可以更换文本，重新start了”
                weakSelf.options.shouldStop = NO;
                [weakSelf responseBlock:NO];
            }
            return ;
        }

        [weakSelf runLabelV2StartRun:weakSelf.options.duration - weakSelf.options.timeInset * 2];//递归调用。参数：确保能在前面文本跑完前kTimeInset秒出
            发
    }];
```

```
- (void)responseBlock:(BOOL)isRuning{
    if(_stopType == RunHorseLampStopImediately)//动画停止类型为：立即停止，所以不需要反馈动画状态；
        return;

    if(_runingStateChangedBlock){
        _runingStateChangedBlock(isRuning);
    }
}
```

```
#pragma mark 通知事件处理

- (void)applicationWillResignActive{ //跑马灯复位操作、中止正在执行的动画
    _runLabelV2Left.constant = 0;
    _runLabelLeft.constant = 0;
    [self layoutIfNeeded];

    [_runLabel.layer removeAllAnimations];//用户强制remove动画，会走动画块的completion回调告知动画未完成
    [_runLabelV2.layer removeAllAnimations];//用户强制remove动画，会走动画块的completion回调告知动画未完成
    _options.runLabelIsRun = NO;
    _options.runLabelV2IsRun = NO;
}

- (void)applicationDidBecomeActive{//恢复动画
    [self startRuning];
}

#pragma mark 控件get方法

- (UILabel *)runLabel{
    if(_runLabel ==  nil){
        _runLabel = [[UILabel alloc] init];
        _runLabel.textColor = _options.textColor;
        [_runLabel setFont:_options.font];
        [_runLabel setTranslatesAutoresizingMaskIntoConstraints:NO];
    }
    return _runLabel;
}
```

```
- (UILabel *)runLabelV2{
    if(_runLabelV2 ==  nil){
        _runLabelV2 = [[UILabel alloc] init];
        _runLabelV2.textColor = _options.textColor;
        [_runLabelV2 setFont:_options.font];
        [_runLabelV2 setTranslatesAutoresizingMaskIntoConstraints:NO];
    }
    return _runLabelV2;
}

@end
```