ByteDance 字节跳动

20秋

MOSAD

现代操作系统应用开发

#7 动画编程

Content

- iOS动画基础
- UIView动画
- CoreAnimation 核心动画
- UIDynamic - UIKit 动力学
- Lottie - 跨平台复杂动画之道
- 转场动画

iOS动画基础

什么是动画？

动画英文“animation”，是由拉丁文“anima”引申而来的。

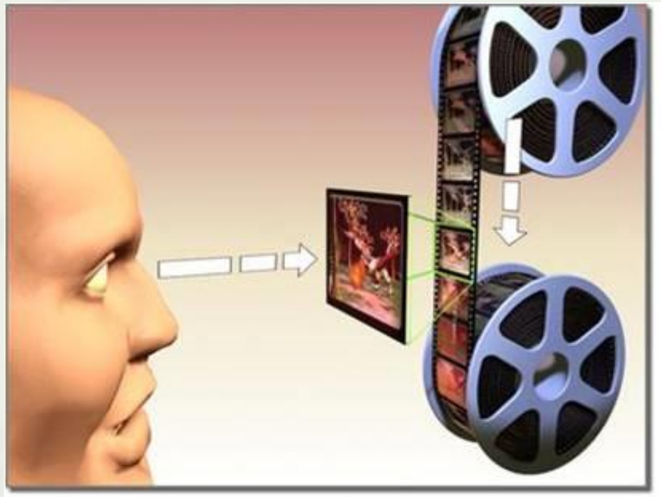
“anima”在拉丁文中为“灵魂”的意思；

“animate”是“使……活起来”的意思；

“animator”则是“赋予生命”的意思。

视觉残留现象

人的眼睛对图像有短暂的记忆效应，当眼睛看到多张图片连续快速的切换时，会认为是一段连续活动的影像。



UIView动画

UIView动画

- UIView 动画允许你提供一个视图的最终状态以及动画时间，由系统自动计算变化规律并重绘

  View hierarchy (adding and removing subviews)

  hidden

  frame/bounds/center

  transform (translation, rotation and scale)

  alpha (opacity)

  backgroundColor

  contentStretch

- 基于UIView类方法和blocks实现

  The class method takes animation parameters and an animation block as arguments.

  The animation block contains the code that makes the changes to the UIView(s).

  Most also have a "completion block" to be executed when the animation is done.

  The changes inside the block are made immediately (even though they will appear "over time").

- Built on top of underlying Core Animation framework

UIView动画

- Animation class method in UIView

```
+ (void)animateWithDuration:(NSTimeInterval)duration
                      delay:(NSTimeInterval)delay
                    options:(UIViewAnimationOptions)options
                 animations:(void (^)(void))animations
                 completion:(void (^)(BOOL finished))completion;
```

- Example

```
[UIView animateWithDuration:3.0
                      delay:0.0
                    options:UIViewAnimationOptionBeginFromCurrentState
                 animations:^{ myView.alpha = 0.0; }
                 completion:^(BOOL fin) { if (fin) [myView removeFromSuperview]; }];
```

This would cause myView to "fade" out over 3 seconds (starting immediately).

Then it would remove myView from the view hierarchy (but only if the fade completed).

If, within the 3 seconds, someone animated the alpha to non-zero, the removal would not happen.

UIView动画

- Another example

```
if (myView.alpha == 1.0) {
  [UIView animateWithDuration: 3.0
                        delay: 2.0
                      options: UIViewAnimationOptionBeginFromCurrentState
                   animations: ^{ myView.alpha = 0.0; }
                   completion: nil];
   NSLog(@"alpha is %f.", myView.alpha);
}
```

myView "fade" out over 3 seconds (starting in 2 seconds).

The NSLog() would happen immediately (i.e. not after 3 or 5 seconds) and print "alpha is 0."

也就是说，动画block立刻执行修改，但透明度变化的动画将在2秒后开始，持续3秒！

UIView动画属性

- UIViewAnimationOptions

```
BeginFromCurrentState       // interrupt other, in-progress animations of these properties
AllowUserInteraction        // allow gestures to get processed while animation is in progress
LayoutSubviews              // animate the relayout of subviews along with a parent's animation
Repeat                      // repeat indefinitely
Autoreverse                 // play animation forwards, then backwards
OverrideInheritedDuration   // if not set, use duration of any in-progress animation
OverrideInheritedCurve      // if not set, use curve (e.g. ease-in/out) of in-progress animation
AllowAnimatedContent        // if not set, just interpolate between current and end state image
CurveEaseInEaseOut          // slower at the beginning, normal throughout, then slow at end
CurveEaseIn                 // slower at the beginning, but then constant through the rest
CurveLinear                 // same speed throughout
TransitionFlipFromLeft/Right      // only for hiding/removing views from the view hierarchy
TransitionCurlUp/Down       // only for hiding/removing views from the view hierarchy
```

UIView动画

- Animating changes to the view hierarchy is slightly different

```
+ (void)transitionFromView:(UIView *)fromView
                    toView:(UIView *)toView
                  duration:(NSTimeInterval)duration
                   options:(UIViewAnimationOptions)options
                completion:(void (^)(BOOL finished))completion;
```

Include UIViewAnimationOptionShowHideTransitionViews if you want hidden property to be set.

Otherwise it will actually remove fromView from the view hierarchy and add toView.

Or you can do the removing/adding/hiding yourself in a block with ...

```
+ (void)transitionWithView:(UIView *)view
                  duration:(NSTimeInterval)duration
                   options:(UIViewAnimationOptions)options
                animations:(void (^)(void))animations
                completion:(void (^)(BOOL finished))completion;
```

UIView动画控制

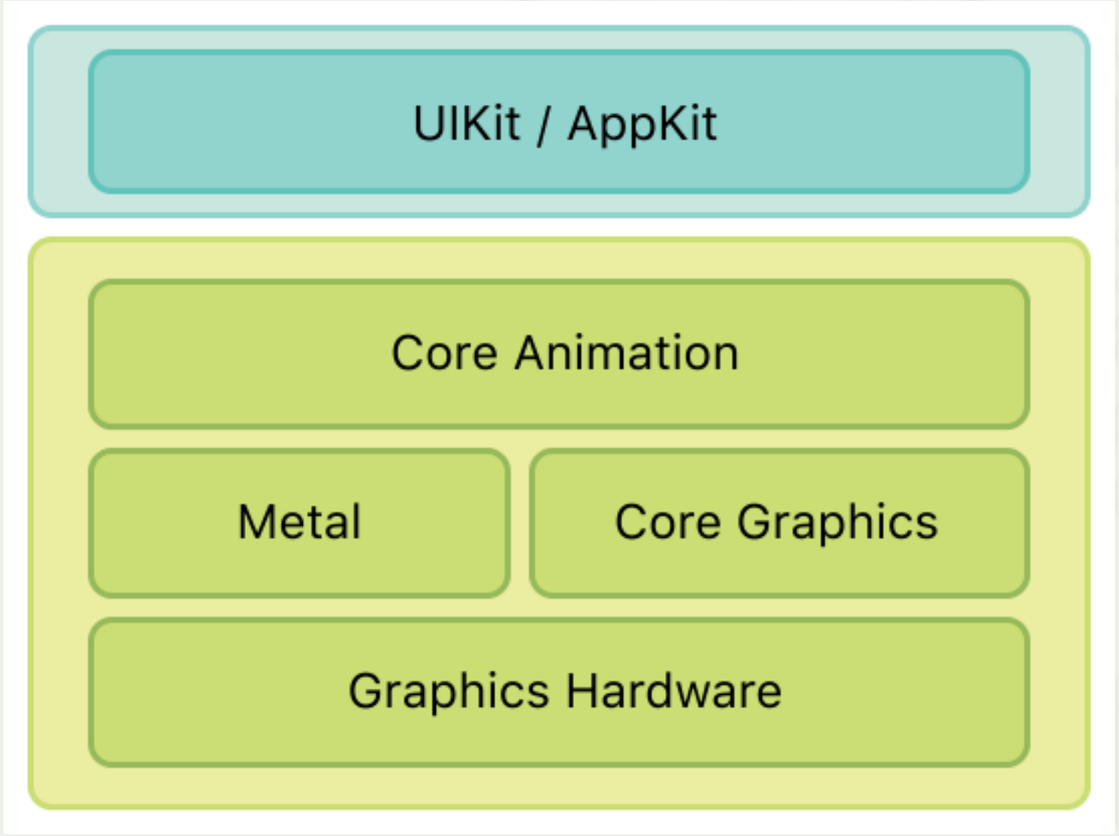
- 通过UIView动画委托控制动画执行

```objc
- (void)startAnimation {
  [UIView beginAnimations:nil context:nil];
  [UIView setAnimationDelay:0.1];
  [UIView setAnimationDuration:0.3];
  [UIView setAnimationDelegate:self];
  [UIView setAnimationWillStartSelector:@selector(animationWillStart:context:)];
  [UIView setAnimationDidStopSelector:@selector(animationDidStop:finished:context:)];
  [UIView setAnimationCurve:UIViewAnimationCurveEaseInOut];

  // Do Animation...

  [UIView commitAnimations];
}

- (void)animationWillStart:(NSString *)animationID context:(void *)context {
    NSLog(@"animationWillStart");
}

- (void)animationDidStop:(NSString *)animationID finished:(NSNumber *)finished context:(void *)context {
    NSLog(@"animationDidStop");
}
```

Core Animation核心动画

Core Animation

- UIKit动画基于Core Animation，而Core Animation基于CALayer层，UIView动画本质上是对其layer完成。



Core Animation

- 一个简单的渐显 CA 动画

```
CABasicAnimation *anim =
        [CABasicAnimation animationWithKeyPath:@"alpha"];
anim.fromValue = @(0.0);
anim.toValue = @(1.0);
anim.duration = 0.3;
[self.view.layer addAnimation:anim forKey:@"alphaAnimation"];
```

CALayer

- 每一个 UIView 都包含一个CALayer，负责在屏幕上绘制一个矩形区域
- 除了基础的 CALayer，Core Animation也提供了许多专用图层。
- 例如：CAGradientLayer

```
CAGradientLayer *gradientLayer = [CAGradientLayer layer];
gradientLayer.frame = self.view.bounds;
[self.view.layer addSublayer:gradientLayer];
gradientLayer.colors = @[[UIColor redColor].CGColor,
                         [UIColor blueColor].CGColor];
gradientLayer.startPoint = CGPointMake(0, 0);
gradientLayer.endPoint = CGPointMake(1, 1);
```

CATransaction - 动画事务管理

- 隐式动画：基于Core Animation，屏幕上的任何东西都可以（可能）做成动画，当CALayer 的可动画属性改变后，CA都会默认以duration 0.25秒、插值生成平滑动画实现
- CA使用CATransaction 来管理动画事务，对于隐式动画，系统自动创建了一个隐式 CATransaction 来管理。
- 我们也可以用CATransaction来扩展设置一些动画参数。例如：调用接口实现动画，而接口并没有提供 completionBlock参数，可以用CATransaction来添加 completionBlock而不需修改别人的接口。
- 也可关闭隐式动画：[CATransaction setDisableActions:YES];

```
[CATransaction begin];
[CATransaction setDisableActions:YES];
gradientLayer.frame = CGRectMake(0, 0, 20.0f, 20.0f);
[CATransaction commit];
```

CAAnimation

- CABasicAnimation - 基础动画
- CAKeyframeAnimation - 关键帧动画
- CAAnimationGroup - 动画组
- CATransition - 转场动画
- CASpringAnimation - 弹簧动画

CAAnimation - 常用的动画属性

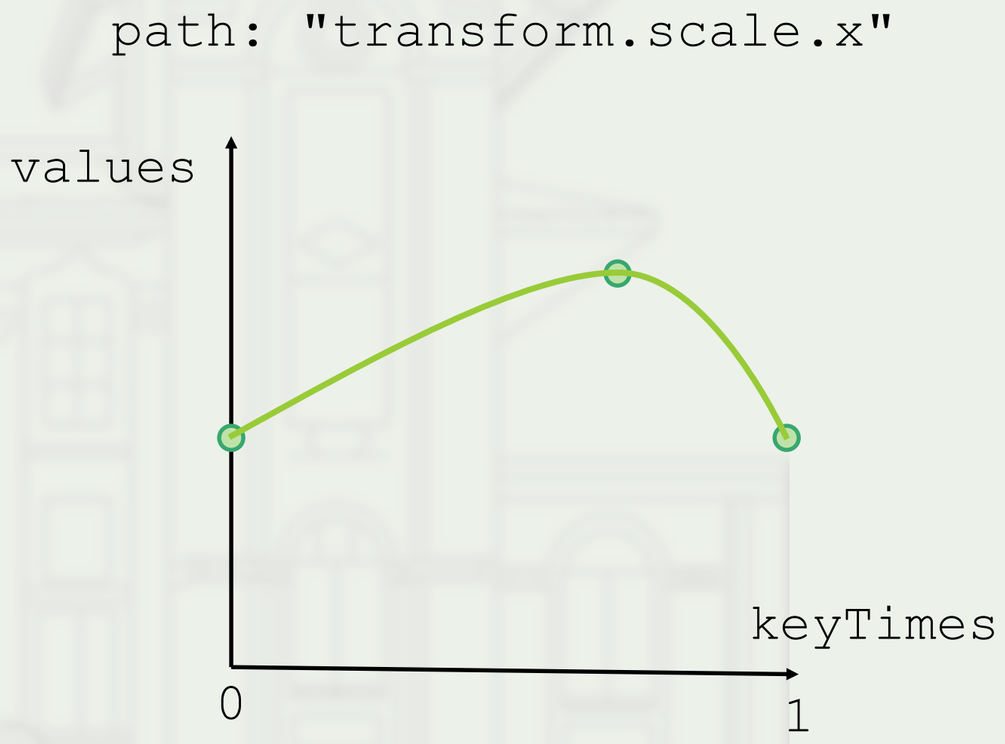
属性	说明
transform.scale	整体比例
transform.scale.x	宽的比例
transform.scale.y	高的比例
transform.rotation.x	围绕x轴旋转
transform.rotation.y	围绕y轴旋转
transform.rotation.z	围绕z轴旋转
cornerRadius	圆角的设置
backgroundColor	背景颜色的变化
bounds	大小的变化，中心不变
position	位置(中心点的改变)
opacity	透明度
contentsRect	内容拉伸缩放

CAAnimation - 常见的动画参数

属性	说明
autoreverses	动画结束时是否执行逆动画
duration	动画执行的时间
removedOnCompletion	动画执行完毕后，图层会保持显示动画执行后的状态； 图层动画属性被移除，图层的属性值还是动画执行前的初始值，并没有真正被改变。
fillMode	kCAFillModeForwards 当动画结束后保持结束时状态 kCAFillModeBackwards 当动画开始时保持动画初始状态 kCAFillModeBoth 上面两者 kCAFillModeRemoved 当动画结束后删除
speed	动画执行速度
repeatCount	动画重复次数
repeatDuration	动画重复执行的时间段
fromValue	动画执行的初始值
toValue	动画执行的效果值
byValue	所改变属性相同起始值的改变量
timingFunction	控制动画的显示节奏： 1.kCAMediaTimingFunctionLinear 线性动画 2.kCAMediaTimingFunctionEaseIn 先慢后快 3.kCAMediaTimingFunctionEaseOut 先快后慢 4.kCAMediaTimingFunctionEaseInEaseOut 先慢后快再慢 5.kCAMediaTimingFunctionDefault 默认，也属于中间比较快
beginTime	可以用来设置动画延迟执行时间，若想延迟1s，就设置为CACurrentMediaTime()+1，CACurrentMediaTime()为图层的当前时间

关键帧动画

- values：关键帧属性值序列数组，每个元素为一个关键属性值，由path指定的属性将在keyTimes设置的时间点依次设置为该值，中间时间点的值插值生成。
- path：动画属性路径，指定的属性将根据时间变化，形成动画。
- keyTimes：设置关键帧对应的时间点，范围：0-1。如果没有设置该属性，则每一帧的时间平分。



关键帧动画

- 例子：基于关键帧的位移动画

```
CAKeyframeAnimation *ani= CAKeyframeAnimation animationWithKeyPath:@"position"];
ani.duration = 4.0;
ani.removedOnCompletion = NO;
ani.fillMode = kCAFillModeForwards;
ani.timingFunction = [CAMediaTimingFunction
                      functionWithName:kCAMediaTimingFunctionEaseInEaseOut];

NSValue *value1 = [NSValue valueWithCGPoint:CGPointMake(150, 200)];
NSValue *value2 = [NSValue valueWithCGPoint:CGPointMake(250, 200)];
NSValue *value3 = [NSValue valueWithCGPoint:CGPointMake(250, 300)];
NSValue *value4 = [NSValue valueWithCGPoint:CGPointMake(150, 300)];
NSValue *value5 = [NSValue valueWithCGPoint:CGPointMake(150, 200)];

ani.values = @[value1, value2, value3, value4, value5];

[self.view.layer addAnimation:ani forKey:@"positionAnimation"];
```

关键帧动画

- 例子：点击放大缩小的动画组CAAnimationGroup

```
CAKeyframeAnimation *scaleXAni =
    [CAKeyframeAnimation animationWithKeyPath:@"transform.scale.x"];
scaleXAni.values = @[@1, @0.9, @1];
scaleXAni.keyTimes = @[@0, @0.8, @1];

CAKeyframeAnimation *scaleYAni =
    [CAKeyframeAnimation animationWithKeyPath:@"transform.scale.y"];
scaleXAni.values = @[@1, @1.1, @1];
scaleXAni.keyTimes = @[@0, @0.8, @1];

CAAnimationGroup *aniGroup = [CAAnimationGroup animation];
aniGroup.animations = @[scaleXAni, scaleYAni];
aniGroup.duration = 0.15;
aniGroup.beginTime = CACurrentMediaTime();
[self.view.layer addAnimation:aniGroup forKey:@"scale"];
```

UIDynamic动力学动画

UIKit动力学——UIDynamic

- UIDynamic是从iOS 7开始引入的一种技术，隶属于UIKit框架，让UIView能够模拟和仿真现实生活中的物理现象，例如重力、弹簧和吸附等效果
- 使用步骤

1. 创建一个仿真者UIDynamicAnimator，用来仿真所有的物理行为
2. 创建物理仿真行为，如重力UIGravity用来模拟重力的行为
3. 将物理仿真行为添加给仿真者实现仿真效果

UIDynamic基本元素

- UIDynamicAnimator：动画者，为动力学元素提供物理学相关的能力及动画，同时为这些元素提供上下文，是动力学元素与iOS底层物理引擎之间的中介，将Behavior对象添加到Animator即可实现动力仿真。
- UIDynamicItem：动力学元素，是任何遵守UIDynamic协议的对象，如果自定义对象实现了该协议，即可通过Dynamic Animator实现物理仿真。
- UIDynamicBehavior：仿真行为，是动力学行为的父类，基本的动力学行为类UIGravityBehavior、UICollisionBehavior、UIAttachmentBehavior、UISnapBehavior、UIPushbehavior以及UIDynamicItemBehavior均继承自该类。

```
UIDynamicAnimator *animator =
        [[UIDynamicAnimator alloc] initWithReferenceView:self];
UIGravityBehavior *gravity = [[UIGravityBehavior alloc] init];
[animator addBehavior:gravity];
```

UIDynamic动力行为

- UIGravityBehavior - 重力
- UICollisionBehavior - 碰撞
- UIAttachmentBehavior - 吸附
- UISnapBehavior - 振动
- UIPushBehavior - 推力

UIDynamic例子

- 红包雨

```
UIDynamicAnimator *flyRedPacketAnimator = [[UIDynamicAnimator alloc] initWithReferenceView:self];
UIDynamicItemBehavior *itemBehavior = [[UIDynamicItemBehavior alloc] initWithItems:@[self.view]];
itemBehavior.elasticity = 1; // 弹性
itemBehavior.resistance = 1; // 阻力
[flyRedPacketAnimator addBehavior:itemBehavior];

// 吸力
CGPoint anchor = CGPointMake(MIN(self.frame.size.width, self.frame.size.height) / 2,
self.view.frame.size.height / 2);
UIAttachmentBehavior *attachBehavior =
        [[UIAttachmentBehavior alloc] initWithItem:self.view attachedToAnchor:anchor];
attachBehavior.length = 0;
attachBehavior.damping = 0.4;
attachBehavior.frequency = 0.3;
[flyRedPacketAnimator addBehavior:attachBehavior];

// 拉力
UIPushBehavior *pullBehavior =
        [[UIPushBehavior alloc] initWithItems:@[self.view] mode:UIPushBehaviorModeInstantaneous];
pullBehavior.active = NO;
[flyRedPacketAnimator addBehavior:pullBehavior];
```

Lottie复杂动画

Lottie

- 什么是 Lottie

  Lottie 动画是 Airbnb 开源的一套动画解决方案，支持 iOS、Android、Web、React Native；

  通过bodymovin插件解析Adobe After Effects动画并导出为Json文件；

  通过Json文件将动画引入App中，可节省复杂的动画绘制编码工作

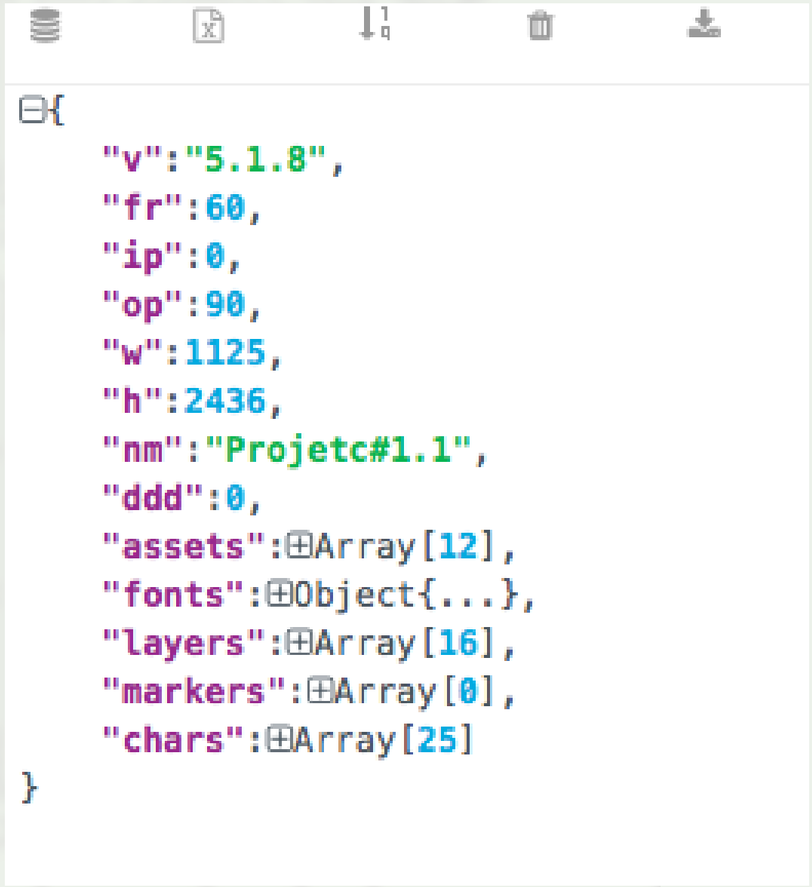
- 最常用的方式是用UIView子类LOTAnimationView来使用Lottie动画

```
LOTAnimationView *lottieAV = [LOTAnimationView animationNamed:@"Lottie"];
[self addSubview:lottieAV];
[lottieAV playWithCompletion:^(BOOL animationFinished) {
    // Do something
}];
```

Lottie动画创建

- 将AE动画导出为json文件
- 导出的 Json 文件格式化后类型如图：
- 部分参数定义：

v：版本号
ip：原大小
op：目标大小
w：宽度
h：高度
nm：文件名称
assets：图片文件
fonts：字体
layers：动画效果
markers：
chars：文字效果



Lottie动画导入

- Lottie提供的创建方法（Json、file path、URL等）

```
/// Load animation by name from the default bundle, Images are also loaded from the bundle
+ (nonnull instancetype)animationNamed:(nonnull NSString *)animationName NS_SWIFT_NAME(init(name:));

/// Loads animation by name from specified bundle, Images are also loaded from the bundle
+ (nonnull instancetype)animationNamed:(nonnull NSString *)animationName inBundle:(nonnull NSBundle *)bundle
NS_SWIFT_NAME(init(name:bundle:));

/// Creates an animation from the deserialized JSON Dictionary
+ (nonnull instancetype)animationFromJSON:(nonnull NSDictionary *)animationJSON NS_SWIFT_NAME(init(json:));

/// Loads an animation from a specific file path. WARNING Do not use a web URL for file path.
+ (nonnull instancetype)animationWithFilePath:(nonnull NSString *)filePath NS_SWIFT_NAME(init(filePath:));

/// Creates an animation from the deserialized JSON Dictionary, images are loaded from the specified bundle
+ (nonnull instancetype)animationFromJSON:(nullable NSDictionary *)animationJSON inBundle:(nullable NSBundle
*)bundle NS_SWIFT_NAME(init(json:bundle:));

/// Creates an animation from the LOTComposition, images are loaded from the specified bundle
- (nonnull instancetype)initWithModel:(nullable LOTComposition *)model inBundle:(nullable NSBundle *)bundle;

/// Loads animation asynchronously from the specified URL
- (nonnull instancetype)initWithContentsOfURL:(nonnull NSURL *)url;
```

Lottie动画播放

- Lottie 提供的使用方法（可指定进度、帧播放）

```
- (void)playToProgress:(CGFloat)toProgress
        withCompletion:(nullable LOTAnimationCompletionBlock)completion;

- (void)playFromProgress:(CGFloat)fromStartProgress
              toProgress:(CGFloat)toEndProgress
          withCompletion:(nullable LOTAnimationCompletionBlock)completion;

- (void)playToFrame:(nonnull NSNumber *)toFrame
     withCompletion:(nullable LOTAnimationCompletionBlock)completion;

- (void)playFromFrame:(nonnull NSNumber *)fromStartFrame
              toFrame:(nonnull NSNumber *)toEndFrame
       withCompletion:(nullable LOTAnimationCompletionBlock)completion;

- (void)playWithCompletion:(nullable LOTAnimationCompletionBlock)completion;

- (void)play;

- (void)pause;

- (void)stop;
```

Lottie使用例子

- 通过NSURL加载动画

```
LOTAnimationView *animation =
        [[LOTAnimationView alloc] initWithContentsOfURL:[NSURL URLWithString:URL]];
[self.view addSubview:animation];
```

- 控制动画进度

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

- 用任意视图来给Lottie View中的动画图层做遮罩，需要知道After Effects动画中对应的图层的名字

```
UIView *snapshot = [self.view snapshotViewAfterScreenUpdates:YES];
[lottieAnimation addSubview:snapshot toLayerNamed:@"AfterEffectsLayerName"];
```

Transition转场动画

UIViewController转场动画

- 系统已自带转场动画，也就是页面切换的动画

  导航切换：push动画/pop动画

  模式弹出：present动画/dismiss动画

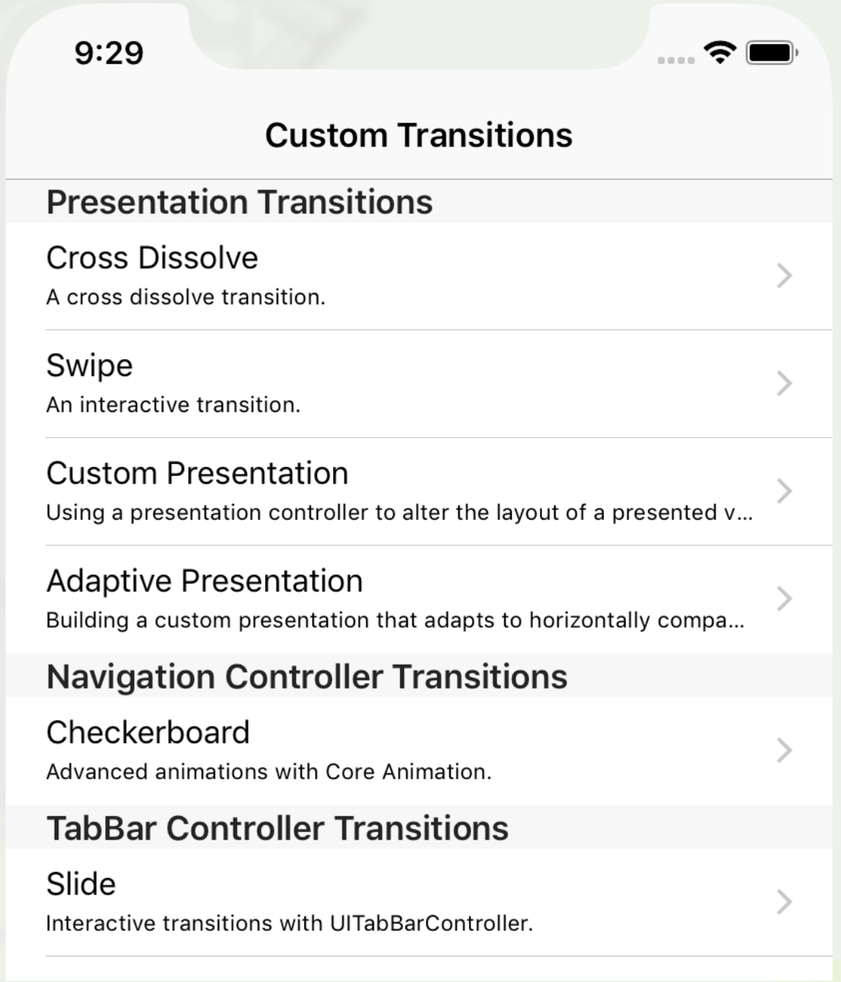
  TabBar切换动画

- 可自定义代理实现自定义转场动画（present/dismiss为例）

  UIViewControllerTransitioningDelegate

  UIViewControllerAnimatedTransitioning

  Official sample code for custom transitions



自定义UIViewController转场动画代理

- 返回负责转场动画的代理对象（非交互的显隐、交互的显隐以及iOS8的）

```
@protocol UIViewControllerTransitioningDelegate <NSObject>
@optional
- (nullable id <UIViewControllerAnimatedTransitioning>)
        animationControllerForPresentedController:(UIViewController *)presented
                                presentingController:(UIViewController *)presenting
                                    sourceController:(UIViewController *)source;
- (nullable id <UIViewControllerAnimatedTransitioning>)
        animationControllerForDismissedController:(UIViewController *)dismissed;
- (nullable id <UIViewControllerInteractiveTransitioning>)
        interactionControllerForPresentation:(id <UIViewControllerAnimatedTransitioning>)animator;
- (nullable id <UIViewControllerInteractiveTransitioning>)
        interactionControllerForDismissal:(id <UIViewControllerAnimatedTransitioning>)animator;
- (nullable UIPresentationController *)
        presentationControllerForPresentedViewController:(UIViewController *)presented
                                presentingViewController:(nullable UIViewController *)presenting
                                    sourceViewController:(UIViewController *)source API_AVAILABLE(ios(8.0));
@end
```

自定义UIViewController转场动画

- 具体负责转场动画代理，返回持续时间及实现转场动画等

```
@protocol UIViewControllerAnimatedTransitioning <NSObject>
/// This is used for percent driven interactive transitions, as well as for container controllers that have
/// companion animations that might need to synchronize with the main animation.
- (NSTimeInterval)transitionDuration:(nullable id <UIViewControllerContextTransitioning>)transitionContext;
/// This method can only  be a nop if the transition is interactive and not a percentDriven interactive transition.
- (void)animateTransition:(id <UIViewControllerContextTransitioning>)transitionContext;
@optional
/// A conforming object implements this method if the transition it creates can be interrupted. For example,
/// it could return an instance of a UIViewPropertyAnimator. It is expected that this method will return the same
/// instance for the life of a transition.
- (id <UIViewImplicitlyAnimating>)
        interruptibleAnimatorForTransition:(id <UIViewControllerContextTransitioning>)transitionContext
        API_AVAILABLE(ios(10.0));
/// This is a convenience and if implemented will be invoked by the system when the transition context's completeTransition: method is invoked.
- (void)animationEnded:(BOOL) transitionCompleted;
@end
```

dismiss消失转场实现的例子

```
@implementation CustomDismissAnimation

- (NSTimeInterval)transitionDuration:(id<UIViewControllerContextTransitioning>)transitionContext {
    return 0.3; // 返回转场动画时间0.3秒
}

- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext {
    UIViewController *fromVC = [transitionContext viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [transitionContext viewControllerForKey:UITransitionContextFromViewControllerKey];
    CGRect bounds = [[UIScreen mainScreen] bounds];
    UIView *containerView = [transitionContext containerView];
    [containerView addSubview:fromVC.view];
    [containerView addSubview:toVC.view];
    fromVC.view.frame = bounds;
    toVC.view.frame = bounds;
    [UIView animateWithDuration:0.3 animations:^{
        fromVC.view.frame = CGRectMake(0, bounds.size.height, bounds.size.width, bounds.size.height);
    } completion:^(BOOL finished) {
        if ([transitionContext transitionWasCancelled]) {
            [transitionContext completeTransition:NO];
        } else {
            [fromVC.view removeFromSuperview];
            [transitionContext completeTransition:YES];
        }
    }];
}
@end
```

把fromVC移动到屏幕底部，实现消失

总结

- 对于复杂动画的实现，先分解成简单动画的组合
- 一个动画效果有多种实现方式，选择合理的方案很重要

- 课后思考题：如何实现两个小球的碰撞爆炸动画效果?

Thanks