UIKit Dynamics学习

> *在Core Animation章节中，我们学习如何使核心动画实现对CALayer的动画操作，在学习CAAnimationGroup时，我们用CAKeyFrameAnimation实现了一个抛物线的功能，但最终重力效果并不是十分理想，没有强烈的重力加速度感受。我们通过今天的学习，可以非常简单地实现该功能，并且效果更加逼真。这就是iOS 7中更新的UIKit Dynamic（UIKit 动力学）功能。*

章节一 UIKit Dynamics简介

UIKit Dynamics是UIKit的一套动画和交互体系。我们现在进行UI动画基本都是使用CoreAnimation或者UIView animations。而UIKit动力学最大的特点是将现实世界动力驱动的动画引入了UIKit，比如重力，铰链连接，碰撞，悬挂等效果。一言蔽之，即是，将2D物理引擎引入了人UIKit。需要注意，UIKit动力学的引入，**并不是以替代CA或者UIView动画为目的的**，在绝大多数情况下CA或者UIView动画仍然是最优方案，只有在需要引入逼真的交互设计的时候，才需要使用UIKit动力学它是作为现有交互设计和实现的一种补充而存在的。

使用UIKit Dynamics的目的当然是更加自然和炫目的UI动画效果，比如模拟现实的拖拽和弹性效果，放在以前如果单用iOS SDK的动画实现起来还是相当困难的，而在UIKit Dynamics的帮助下，复杂的动画效果可能也只需要很短的代码。配合UI交互设计，以前很多不敢想和不敢写的效果实现起来会非常方便，动画效果也会上升一个档次。

UIKit Dynamics策略结构

使用UIKit Dynamics需要以下四种参与者：

- UIDynamicItem 协议对象
- UIDynamicBehavior 对象
- UIDynamicAnimator 对象
- ReferenceView 对象

1. UIDynamicItem 协议

给对象添加物理行为的前提，需要所在类实现该协议。UIView实现了该协议，因此所有UIView和UIView的子类都可以作为UIDynamicItem

2. UIDynamicBehavior 行为描述

物理行为描述，比如碰撞，重力等等，需要和相关的item关联，默认支持的Behavior有以下几种：

1. UIAttachmentBehavior 连接行为（可以设置有无弹性）
2. UICollisionBehavior 碰撞行为

3. UIGravityBehavior 重力行为
4. UIPushBehavior 推力行为
5. UISnapBehavior 吸附行为

3. UIDynamicAnimator 行为容器

需要将Behavior添加到Animator中，行为描述才能够起作用。

4. ReferenceView 物理参考坐标系

需要给Animator的ReferenceView赋值，给Animator中的所有动力计算提供坐标系。

章节二 UIKit Dynamics基本行为演示

本章将逐一介绍几种默认Behavior的使用。首先我们先要准备好其他的参与者，UIDynamicItem 即`BallView`的实例对象，Animator 在类`BaseView`中创建，并将Animator的referenceView设置为BaseView，Behavior在BaseView的子类中创建，并且和Item关联，然后加入到Animator中。

创建类BaseView ，添加并初始化Animator

```
theAnimator = [[UIDynamicAnimator alloc] initWithReferenceView:self];
```

创建类BallView，在BaseView中添加一个BallView实例`_ballView`作为UIDynamicItem

BallView.m

```
#import "BallView.h"

@implementation BallView {

}

- (id)init {
    self = [super init];
    if (self) {
        [self setFrame:CGRectMake(0, 0, 40, 40)];
        self.backgroundColor = [UIColor lightGrayColor];
        self.layer.cornerRadius = 20;
        self.layer.borderColor = [UIColor grayColor].CGColor;
        self.layer.borderWidth = 2;
    }
    return self;
}

@end
```

BaseView.h

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

@interface BaseViewWithBall : UIView{
    BallView *ballView;
    UIDynamicAnimator *theAnimator;
}
@end
```

BaseView.m

```
#import "BaseView.h"
#import "BallView.h"

@implementation BaseViewWithBall {

}

- (id)init {
    self = [super init];
    if (self) {
        [self buildBall];
        //初始化Animator
        theAnimator = [[UIDynamicAnimator alloc]
initWithReferenceView:self];
    }
    return self;
}
//初始化ballView
- (void)buildBall {
    ballView = [[BallView alloc] init];
    ballView.center = CGPointMake(100, 100);
    [self addSubview:ballView];
}
@end
```

1. UIGravityBehavior 重力

单位： 1000p/s² 对应现实世界中：9.8m/s² 【p是指点 points】

```
UIGravityBehavior *gravityBehaviour = [[UIGravityBehavior alloc]
initWithItems:@[ballView]];
[theAnimator addBehavior:gravityBehaviour];
```

2. UIDynamicItemBehavior 物体描述

- elasticity 弹性 （0-1）
- friction 摩擦力 （0 表示无摩擦力）
- density 密度 （默认1）
- resistance 阻尼 （0表示无阻尼，速度不会衰减）

```
  UIDynamicItemBehavior *itemBehavior = [[UIDynamicItemBehavior alloc]
initWithItems:@[ballView]];
  [itemBehavior setElasticity:0.6];
  [theAnimator addBehavior:itemBehavior];
  [collisionBehavior setCollisionDelegate:self];
```

3. UICollisionBehavior 碰撞

setTranslatesReferenceBoundsIntoBoundary 设置边界碰撞作用

```
UICollisionBehavior *collisionBehavior = [[UICollisionBehavior alloc]
initWithItems:@[ballView]];
[collisionBehavior setTranslatesReferenceBoundsIntoBoundary:YES];
[theAnimator addBehavior:collisionBehavior];
```

4. UIAttachmentBehavior 连接行为

连接到锚点

```
UIAttachmentBehavior *attachmentBehavior = [[UIAttachmentBehavior alloc]
initWithItem:ballView attachedToAnchor:CGPointMake(100, 0)];
// 长度 单位p
[attachmentBehavior setLength:300];
//衰减 1表示临界值
[attachmentBehavior setDamping:0.1];
//frequency 频率 单位：赫兹
[attachmentBehavior setFrequency:5];
[theAnimator addBehavior:attachmentBehavior];
```

连接到其他item

```
UIAttachmentBehavior *attachmentBehavior = [[UIAttachmentBehavior alloc]
initWithItem:ballViews[i] attachedToItem:ballViews[i + 1]];
```

5. UISnapBehavior 捕捉（吸附）行为

```
        UISnapBehavior *snapBehavior = [[UISnapBehavior alloc]
initWithItem:ballView

snapToPoint:CGPointMake(10，10)];
```

6. UIPushBehavior 捕捉（吸附）行为

```
UIPushBehavior *pushBehavior = [[UIPushBehavior alloc]
initWithItems:@[barView] mode:UIPushBehaviorModeInstantaneous];
        [pushBehavior setPushDirection:CGVectorMake([gestureRecognizer
velocityInView:self].x /5000.f, 0)];
        [theAnimator addBehavior:pushBehavior];
```

章节三 修改垃圾投掷动画

请参照具体代码

拓展：实现牛顿摆