

为者常成，行者常至

思从深而行从简 高调做事 低调做人

个人资料



陌苏湮雪

[关注](#) [发私信](#)

访问：405210次

积分：4497

等级：

8LOG5

排名：第5144名

原创：115篇

转载：65篇

译文：0篇

评论：44条

文章搜索

文章分类

iOS开发- - Objective-C (9)

iOS开发- - UI&网络 (9)

iOS开发- - 提高篇 (31)

iOS开发- - 其他 (11)

算法&设计模式 (6)

提高篇——书籍 (2)

web开发- - 前端框架 (28)

web开发- - 前端 (11)

web开发- - 数据库 (5)

基础篇- - java (22)

杂七杂八篇- - 转载 (23)

杂七杂八篇- - 工具 (11)

文章存档

2016年04月 (1)

目录视图

摘要

[程序员12月书讯](#) [【系列直播】算法与游戏实战技术](#) [“我的2016”主题征文活动](#)

iOS collectionViewLayout布局和自定义

2015-11-18 15:58

9503人阅读

评论(1)

分类：

iOS开发- - UI&网络 (8) ▼

目录(?)

[+]

UICollectionView的结构回顾

首先回顾一下Collection View的构成，我们能看到的有三个部分：

- Cells
- Supplementary Views 追加视图 （类似Header或者Footer）
- Decoration Views 装饰视图 （用作背景展示）

而在表面下，由两个方面对UICollectionView进行支持。其中之一和tableView一样，即提供数据的UICollectionView及处理用户交互的UICollectionViewDelegate。另一方面，对于cell的样式和组织方式，由于collectionView比table多，因此没有按照类似于tableView的style的方式来定义，而是专门使用了一个类来对collectionView的布局和行为进行控制，即UICollectionViewLayout。

这次的笔记将把重点放在UICollectionViewLayout上，因为这不仅是collectionView和tableView的最重要区别，UICollectionView的精髓所在。

如果对UICollectionView的基本构成要素和使用方法还不清楚的话，可以移步到我之前的一篇笔记：[Session笔记——Introducing Collection Views](#)中进行一些了解。

UICollectionViewLayoutAttributes

UICollectionViewLayoutAttributes是一个非常重要的类，先来看看property列表：

- @property (nonatomic) CGRect frame
- @property (nonatomic) CGPoint center
- @property (nonatomic) CGSize size
- @property (nonatomic) CATransform3D transform3D
- @property (nonatomic) CGFloat alpha
- @property (nonatomic) NSInteger zIndex
- @property (nonatomic, getter=isVisible) BOOL hidden

2016年03月 (16)

2016年02月 (16)

2016年01月 (26)

2015年12月 (4)

展开

阅读排行	
java、八大经典书籍，你看过...	(43977)
4.1Bootstrap学习js插件篇之模...	(38592)
4.6Bootstrap学习js插件篇之弹...	(17898)
3.1Bootstrap学习组件篇之下拉...	(16432)
Bootstrap学习js插件篇之轮播	(16046)
程序员的出路在哪里？一个33...	(14162)
3.5Bootstrap组件篇之导航条	(13653)
2.4Bootstrap表单	(12139)
iOS collectionViewLayout布局...	(9486)
Bootstrap学习js插件篇之提示框	(9334)

评论排行

4.1Bootstrap学习js插件篇之模...	(10)
程序员的出路在哪里？ 一个33...	(5)
4.6Bootstrap学习js插件篇之弹...	(4)
Emmet：HTML/CSS代码快速...	(4)
iOS category内部实现原理	(4)
java、八大经典书籍，你看过...	(3)
4.3Bootstrap学习js插件篇之标...	(3)
Bootstrap学习js插件篇之轮播	(2)
4.2Bootstrap学习js插件篇之下...	(2)
4.5Bootstrap学习js插件篇之工...	(2)



可以看到，`UICollectionViewLayoutAttributes`的实例中包含了诸如边框，中心点，大小，形状，透明度，层次关系。和`DataSource`的行为十分类似，当`UICollectionView`在获取布局时将针对每一个`indexPath`的部件（包括cell，视图），向其上的`UICollectionViewLayout`实例询问该部件的布局信息（在这个层面上说的话，实现一个`UICollectionView`候，其实很像是zap一个`delegate`，之后的例子中会很明显地看出），这个布局信息，就以`UICollectionViewLayout`的方式给出。

自定义的UICollectionViewLayout

UICollectionViewLayout的功能是为UICollectionView提供布局信息，不仅包括cell的布局信息，也包括追加视图和信息。实现一个自定义layout的常规做法是继承UICollectionViewLayout类，然后重载下列方法：

- `-(CGSize)collectionViewContentSize`
 - 返回collectionView的内容的尺寸
- `-(NSArray *)layoutAttributesForElementsInRect:(CGRect)rect`
 - 返回rect中的所有的元素的布局属性
 - 返回的是包含UICollectionViewLayoutAttributes的NSArray
 - UICollectionViewLayoutAttributes可以是cell，追加视图或装饰视图的信息，通过不同的UICollectionViewLayoutAttributes初始化方法可以得到不同类型的UICollectionViewLayoutAttributes：
 - `layoutAttributesForCellWithIndexPath:`
 - `layoutAttributesForSupplementaryViewOfKind:withIndexPath:`
 - `layoutAttributesForDecorationViewOfKind:withIndexPath:`
- `-(UICollectionViewLayoutAttributes *)layoutAttributesForItemAtIndexPath:(NSIndexPath *)indexPath`
 - 返回对应于indexPath的位置的cell的布局属性
- `-(UICollectionViewLayoutAttributes *)layoutAttributesForSupplementaryViewOfKind:(NSString *)kind (NSIndexPath *)indexPath`
 - 返回对应于indexPath的位置的追加视图的布局属性，如果没有追加视图可不重载
- `-(UICollectionViewLayoutAttributes *)layoutAttributesForDecorationViewOfKind:(NSString *)decorativeKind (NSIndexPath *)indexPath`
 - 返回对应于indexPath的位置的装饰视图的布局属性，如果没有装饰视图可不重载
- `-(BOOL)shouldInvalidateLayoutForBoundsChange:(CGRect)newBounds`
 - 当边界发生改变时，是否应该刷新布局。如果YES则在边界变化（一般是scroll到其他地方）时，将重新计算信息。

另外需要了解的是，在初始化一个UICollectionViewLayout实例后，会有一系列准备方法被自动调用，以保证layout

首先，`-(void)prepareLayout`将被调用，默认下该方法什么没做，但是在自己的子类实现中，一般在该方法中设定一结构和初始需要的参数等。

之后，`-(CGSize) collectionViewContentSize`将被调用，以确定collection应该占据的尺寸。注意这里的尺寸不是尺寸，而应该是所有内容所占的尺寸。`collectionView`的本质是一个`scrollView`，因此需要这个尺寸来配置滚动行为。

接下来-(NSArray *)layoutAttributesForElementsInRect:(CGRect)rect被调用，这个没什么值得多说的。初始的lay方法返回的UICollectionViewLayoutAttributes来决定。

另外，在需要更新layout时，需要给当前layout发送 `-invalidateLayout`，该消息会立即返回，并且预约在下一个layout，这一点和UIView的`setNeedsLayout`方法十分类似。在`-invalidateLayout`后的下一个collectionView的刷新`prepareLayout`开始，依次再调用`-collectionViewContentSize`和`-layoutAttributesForElementsInRect`来生成更新

Demo

说了那么多, 其实还是Demo最能解决问题。Apple官方给了一个flow layout和一个circle layout的例子, 都很经典, 从[这里](#)下载。

LinearLayout——对于个别UICollectionViewLayoutAttributes的调整

先看LinearLayout, 它继承了UICollectionViewFlowLayout这个Apple提供的基本的布局。它主要实现了单行布局, 自及当前网格cell放大三个特性。如图:



先看LinearLayout的init方法:

```
-(id)init
{
    self = [super init];
    if (self) {
        self.itemSize = CGSizeMake(ITEM_SIZE, ITEM_SIZE);
        self.scrollDirection = UICollectionViewScrollDirectionHorizontal;
        self.sectionInset = UIEdgeInsetsMake(200, 0.0, 200, 0.0);
        self.minimumLineSpacing = 50.0;
    }
    return self;
}
```

self.sectionInset = UIEdgeInsetsMake(200, 0.0, 200, 0.0); 确定了缩进, 此处为上方和下方各缩进200个point。经定义为200x200, 因此屏幕上在缩进后就只有一排item的空间了。

self.minimumLineSpacing = 50.0; 这个定义了每个item在水平方向上的最小间距。

UICollectionViewFlowLayout是Apple为我们准备的开袋即食的现成布局, 因此之前提到的几个必须重载的方法中需少, 即使完全不重载它们, 现在也可以得到一个不错的线状一行的gridview了。而我们的LinearLayout通过重载父类方一些新特性, 比如这里的动对齐到网格以及当前网格cell放大。

自动对齐到网格

```
- (CGPoint)targetContentOffsetForProposedContentOffset: (CGPoint)proposedContentOffset
withScrollingVelocity:(CGPoint)velocity
{
    //proposedContentOffset是没有对齐到网格时本来应该停下的位置
    CGFloat offsetAdjustment = MAXFLOAT;
    CGFloat horizontalCenter = proposedContentOffset.x +
    (CGRectGetWidth(self.collectionView.bounds) / 2.0);
    CGRect targetRect = CGRectMake(proposedContentOffset.x, 0.0,
    self.collectionView.bounds.size.width, self.collectionView.bounds.size.height);
    NSArray* array = [super layoutAttributesForElementsInRect:targetRect];

    //对当前屏幕中的UICollectionViewLayoutAttributes逐个与屏幕中心进行比较, 找出最接近中心的一个
    for (UICollectionViewLayoutAttributes* layoutAttributes in array) {
        CGFloat itemHorizontalCenter = layoutAttributes.center.x;
        if (ABS(itemHorizontalCenter - horizontalCenter) < ABS(offsetAdjustment)) {
            offsetAdjustment = itemHorizontalCenter - horizontalCenter;
        }
    }
    return CGPointMake(proposedContentOffset.x + offsetAdjustment,
    proposedContentOffset.y);
}
```

当前item放大

```
-(NSArray *)layoutAttributesForElementsInRect:(CGRect)rect
{
    NSArray *array = [super layoutAttributesForElementsInRect:rect];
    CGRect visibleRect;
    visibleRect.origin = self.collectionView.contentOffset;
    visibleRect.size = self.collectionView.bounds.size;

    for (UICollectionViewLayoutAttributes* attributes in array) {
        if (CGRectIntersectsRect(attributes.frame, rect)) {
```

```

        CGFloat distance = CGRectGetMidX(visibleRect) - attributes.center.x;
        CGFloat normalizedDistance = distance / ACTIVE_DISTANCE;
        if (ABS(distance) < ACTIVE_DISTANCE) {
            CGFloat zoom = 1 + ZOOM_FACTOR*(1 - ABS(normalizedDistance));
            attributes.transform3D = CATransform3DMakeScale(zoom, zoom, 1.0);
            attributes.zIndex = 1;
        }
    }
}
return array;
}

```

对于个别UICollectionViewLayoutAttributes进行调整，以达到满足设计需求是UICollectionView使用中的一种思路。不同layout属性的时候，需要记得让-shouldInvalidateLayoutForBoundsChange:返回YES，这样当边界改变的时候invalidateLayout会自动被发送，才能让layout得到刷新。

CircleLayout——完全自定义的Layout，添加删除item，以及手势识别

CircleLayout的例子稍微复杂一些，cell分布在圆周上，点击cell的话会将其从collectionView中移出，点击空白处会入和移出都有动画效果。

这放在以前的话估计够写一阵子了，而得益于UICollectionView，基本只需要100来行代码就可以搞定这一切，非常。CircleLayout的实现，可以完整地看到自定义的layout的编写流程，非常具有学习和借鉴的意义。



首先，布局准备中定义了一些之后计算所需要用到的参数。

```

-(void)prepareLayout
{
    //和init相似，必须call super的prepareLayout以保证初始化正确
    [super prepareLayout];

    CGSize size = self.collectionView.frame.size;
    _cellCount = [[self collectionView] numberOfItemsInSection:0];
    _center = CGPointMake(size.width / 2.0, size.height / 2.0);
    _radius = MIN(size.width, size.height) / 2.5;
}

```

其实对于一个size不变的collectionView来说，除了_cellCount之外的中心和半径的定义也可以扔到init里去做，但是prepareLayout里做的话具有更大的灵活性。因为每次重新给出layout时都会调用prepareLayout，这样在以后如果大小变化的需求时也可以自动适应变化。

然后，按照UICollectionViewLayout子类的要求，重载了所需要的方法：

```

//整个collectionView的内容大小就是collectionView的大小（没有滚动）
-(CGSize)collectionViewContentSize
{
    return [self collectionView].frame.size;
}

//通过所在的indexPath确定位置。
- (UICollectionViewLayoutAttributes *)layoutAttributesForItemAtIndexPath:(NSIndexPath *)path
{
    UICollectionViewLayoutAttributes* attributes = [UICollectionViewLayoutAttributes
    layoutAttributesForCellWithIndexPath:path]; //生成空白的attributes对象，其中只记录了类型是cell以及对应的位置是indexPath
    //配置attributes到圆周上
    attributes.size = CGSizeMake(ITEM_SIZE, ITEM_SIZE);
    attributes.center = CGPointMake(_center.x + _radius * cosf(2 * path.item * M_PI /
    _cellCount), _center.y + _radius * sinf(2 * path.item * M_PI / _cellCount));
    return attributes;
}

//用来一开始给出一套UICollectionViewLayoutAttributes
-(NSArray*)layoutAttributesForElementsInRect:(CGRect)rect
{
    NSMutableArray* attributes = [NSMutableArray array];
    for (NSInteger i=0 ; i < self.cellCount; i++) {
        //这里利用了-layoutAttributesForItemAtIndexPath:来获取attributes
        NSIndexPath* indexPath = [NSIndexPath indexPathForItem:i inSection:0];
        [attributes addObject:[self layoutAttributesForItemAtIndexPath:indexPath]];
    }
}

```

```
    }  
    return attributes;  
}
```

现在已经得到了一个circle layout。为了实现cell的添加和删除，需要为collectionView加上手势识别，这个很简单，ViewController中：

```
UITapGestureRecognizer* tapRecognizer = [[UITapGestureRecognizer alloc] initWithTarget:self  
action:@selector(handleTapGesture:)];  
[self.collectionView addGestureRecognizer:tapRecognizer];
```

对应的处理方法handleTapGesture:为

```
- (void)handleTapGesture:(UITapGestureRecognizer *)sender {  
    if (sender.state == UIGestureRecognizerStateEnded) {  
        CGPoint initialPinchPoint = [sender locationInView:self.collectionView];  
        NSIndexPath* tappedCellPath = [self.collectionView  
indexPathForItemAtPoint:initialPinchPoint]; //获取点击处的cell的indexPath  
        if (tappedCellPath!=nil) { //点击处没有cell  
            self.cellCount = self.cellCount - 1;  
            [self.collectionView performBatchUpdates:^(  
                [self.collectionView deleteItemsAtIndexPaths:[NSArray  
arrayWithObject:tappedCellPath]];  
            } completion:nil];  
        } else {  
            self.cellCount = self.cellCount + 1;  
            [self.collectionView performBatchUpdates:^(  
                [self.collectionView insertItemsAtIndexPaths:[NSArray arrayWithObject:  
[NSIndexPath indexPathForItem:0 inSection:0]]];  
            } completion:nil];  
        }  
    }  
}
```

performBatchUpdates:completion: 再次展示了block的强大的一面..这个方法可以用来对collectionView中的元素删除，移动等操作，同时将触发collectionView所对应的layout的对应的动画。相应的动画由layout中的下列四个方

- initialLayoutAttributesForAppearingItemAtIndexPath:
- initialLayoutAttributesForAppearingDecorationElementOfKind:atIndexPath:
- finalLayoutAttributesForDisappearingItemAtIndexPath:
- finalLayoutAttributesForDisappearingDecorationElementOfKind:atIndexPath:

更正：正式版中API发生了变化（而且不止一次变化 initialLayoutAttributesForInsertedItemAtIndexPath:在正删除。现在在insert或者delete之前，prepareForCollectionViewUpdates:会被调用，可以使用这个方法来完成添局。关于更多这方面的内容以及新的示例demo，可以参看[这篇博文](#)（需要翻墙）。新的示例demo在Github上也

在CircleLayout中，实现了cell的动画。

```
//插入前，cell在圆心位置，全透明  
- (UICollectionViewLayoutAttributes *)initialLayoutAttributesForInsertedItemAtIndexPath:  
(NSIndexPath *)indexPath  
{  
    UICollectionViewLayoutAttributes* attributes = [self  
layoutAttributesForItemAtIndexPath:indexPath];  
    attributes.alpha = 0.0;  
    attributes.center = CGPointMake(_center.x, _center.y);  
    return attributes;  
}  
  
//删除时，cell在圆心位置，全透明，且只有原来的1/10大  
- (UICollectionViewLayoutAttributes *)finalLayoutAttributesForDeletedItemAtIndexPath:  
(NSIndexPath *)indexPath  
{  
    UICollectionViewLayoutAttributes* attributes = [self  
layoutAttributesForItemAtIndexPath:indexPath];  
    attributes.alpha = 0.0;  
    attributes.center = CGPointMake(_center.x, _center.y);  
    attributes.transform3D = CATransform3DMakeScale(0.1, 0.1, 1.0);  
}
```

```
        return attributes;
    }
```

在插入或删除时，将分别以插入前和删除后的attributes和普通状态下的attributes为基准，进行UIView的动画过渡。很多代码要写，几乎是free的，感谢苹果...

布局之间的切换

有时候可能需要不同的布局，Apple也提供了方便的布局间切换的方法。直接更改collectionView的collectionViewLayout即切换布局。而如果通过setCollectionViewLayout:animated:，则可以在切换布局的同时，使用动画来过渡。对于4有对应的UIView动画进行对应，又是一个接近free的特性。

对于我自己来说，UICollectionView可能是我转向iOS 6 SDK的最具有吸引力的特性之一，因为UIKit团队的努力和C++成熟，使得创建一个漂亮优雅的UI变的越来越简单了。可以断言说UICollectionView在今后的iOS开发中，一定会成为一样的强大和最常用的类之一。在iOS 6还未正式上市前，先对其特性进行一些学习，以期尽快能使用新特性来简化开发是非常值得的。

顶 1
踩 0

- [上一篇](#) [UICollectionView详解](#)
- [下一篇](#) [技术练级攻略](#)

我的同类文章

iOS开发——UI&网络（8）

- | | | | | | |
|---------------------------------------|------------|--------|--------------------------------|------------|----|
| • 【提高】 沙盒目录文件解析 | 2016-02-19 | 阅读 236 | • 【网络】 iOS上传下载 | 2016-02-19 | 阅读 |
| • 【网络】 多线程--NSThread、GCD、NS... | 2016-02-19 | 阅读 118 | • 【UI篇】 一、UIApplication | 2016-01-20 | 阅读 |
| • 【iOS开发】 XML解析--GData语法 | 2016-01-14 | 阅读 369 | • iOS开发篇——UITextField | 2015-12-24 | 阅读 |
| • iOS NSURLConnection GET和POST | 2015-11-30 | 阅读 793 | • UICollectionView详解 | 2015-11-18 | 阅读 |



广告

极光统计 JAnalytics

一站式集成 专业化解读

参考知识库

iOS

iOS知识库

3576 关注 | 1443 收录



Swift知识库

3533 关注 | 1164 收录


猜你在找

iOS8开发技术（Swift版）：屏幕...
iOS8开发视频教程Swift语言版-Pa...
iOS8开发视频教程Swift语言版-Pa...
小波说iOS8 智能视图 完整版
老郭全套iOS开发课程【UI技术】

UICollectionView
iOS-UICollectionViewLayout 自定...
IOS 集合视图指南7自定义布局范例
iOS中自定义View实现layoutSubvi...
iOS中自定义View实现layoutSubvi...

广告

查看评论

lovechris00

1楼 2016-07-

可惜官方下载demo的地址无法访问

您还没有登录,请[登录](#)或[注册](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题

Hadoop

AWS

移动游戏

Java

Android

iOS

Swift

智能硬件

Docker

OpenStack

VPN

Spark

Eclipse

CRM

JavaScript

数据库

Ubuntu

NFC

WAP

jQuery

BI

HTML5

Spring

Apache

.NET

API

SDK

IIS

Fedora

XML

LBS

Unity

Splashtop

UML

components

Windows Mobile

Rails

QEMU

KDE

CloudStack

FTC

coremail

OPhone

CouchBase

云计算

iOS6

Rackspace

Web App

SpringSide

Maemo

Compuv

aptech

Perl

Tornado

Ruby

Hibernate

ThinkPHP

HBase

Pure

Solr

Angular

Cloud Foundry

Redis

Scala

Bootstrap