

(译) 在 **cocos2d** 里面如何拖拽精灵

整理：Taiyangmobile (泰然论坛管理组)

**著作权声明：**本文由 子龙山人 翻译，欢迎转载分享。请尊重作者劳动，转载时保留该声明和作者博客链接，谢谢！首发于[泰然论坛](#)

免责声明（必读！）：本博客提供的所有教程的翻译原稿均来自于互联网，仅供学习交流之用，切勿进行商业传播。同时，转载时不要移除本申明。如产生任何纠纷，均与本博客所有人、发表该翻译稿之人无任何关系。谢谢合作！

原文出处：

<http://www.raywenderlich.com/2343/how-to-drag-and-drop-sprites-with-cocos2d>

PS:非常感谢 aom7610 给我拟的这份免责声明，以后我会继续努力翻译，把这件事情坚持下去。

程序截图：



我收到许多读者来信说，能不能写一个教程，关于如何在 **cocos2d** 里面使用 **touch** 事件来拖拽精灵（**sprite**）。既然你们这么要求，我就满足你们啦！

在这个教程中，你将学到下列内容：

- 使用 **touch** 事件拖拽精灵的基本方法
- 如何通过 **touch** 事件来滚动视图本身
- 如何方便地计算坐标
- 如何通过识别手势来实现一些更 **cool** 的效果

为了使事件变得有趣，你将要移动一些非常可爱的动画图片，它是[我可爱的妻子](#)创作的，背景则是由 [gwebstock](#) 创建。

这个教程假设你已经有一些基本的 **cocos2d** 的知识，同时已经安装了一份 **cocos2d** 的版本。如果你对 **cocos2d** 还不熟悉，你可能需要先学习一下译者翻译的[《如何使用 cocos2d 来制作一个简单的 iphone 游戏：第一部分》](#)

好了，不多说，准备好键盘，开始吧！

## Getting Started

在实现 touch 事件之前，首先你需要创建一个基本的 cocos2d 场景来显示背景和这些动物精灵。

打开 XCode，点击 File\New Project，选择 User Templates\cocos2d X.X.X\cocos2d Application，再点击“Choose...”。把工程命名为“DragDrop”并点击 Save。

接下来，继续，下载你需要的[图片](#)。下载完后，解压，然后把这些图片拖到 Resources 分组下面。确保“Copy items into destination group's folder (if needed)”被选中，然后点击 Add。

在你把图片导入到工程之后，在 Xcode 中展开 Classes 分组，然后选择 HelloWorld.h。在@interface 申明处，像下面所示，申明 3 个实例变量：

```
CCSprite * background;
CCSprite * selSprite;
NSMutableArray * movableSprites;
```

你将使用这些变量来追踪你的背景图片、当前选中的精灵以及一个在处理 touch 事件时需要移动的精灵的数组。

现在，回到 HelloWorldScene.m，找到 init 方法，把它替换成下面的代码：

```
-(id) init {
    if((self = [super init])) {
        CGSize winSize = [CCDirector sharedDirector].winSize;

        [CCTexture2D setDefaultAlphaPixelFormat:kCCTexture2DPixelFormat_RGB565];
        background = [CCSprite spriteWithFile:@"blue-shooting-stars.png"];
        background.anchorPoint = ccp(0,0);
        [self addChild:background];
        [CCTexture2D setDefaultAlphaPixelFormat:kCCTexture2DPixelFormat_Default];

        movableSprites = [[NSMutableArray alloc] init];
        NSArray *images = [NSArray arrayWithObjects:@"bird.png", @"cat.png", @"dog.png", @"turtle.png", nil];
        for(int i = 0; i < images.count; ++i) {
            NSString *image = [images objectAtIndex:i];
            CCSprite *sprite = [CCSprite spriteWithFile:image];
            float offsetFraction = ((float)(i+1))/(images.count+1);
            sprite.position = ccp(winSize.width*offsetFraction, winSize.height/2);
            [self addChild:sprite];
            [movableSprites addObject:sprite];
        }
    }
    return self;
}
```

这里有一些新的知识点需要引入，让我们一步步来学习吧！

## 加载背景

这个方法的第一部分加载了一张本场景的背景图片（blue-shooting-stars.png）。注意，这里把图片的锚点（anchor point）设置成图片的左下角（0，0）点。

在 cocos2d 里面，当你设置一个精灵的位置的时候，实际上，你设置的是这个精灵的锚点的位置。默认情况下，图片的锚点就是图片的中点。因此，通过把精灵锚点设置成左下角，当你设置精灵位置的时候，实际上你就是指定了精灵的中心位置在左下角。

这个方法并没有设置背景的位置，因此背景的位置默认情况下是（0，0）。因此，图片的实际位置就是在（0，0）。（因此设置精灵位置是相对于锚点来的，锚点在左下角，因此图片的左下角就位于屏幕的左下角）。因此，这个图片有 800 个像素宽，那么超过的部分就在屏幕之外了。

另外需要注意的一点就是，在加载图片之前，转换了一下像素格式。在默认情况下，cocos2d 里面加载图片，它们是作为 32 位的图片加载进来的。这意味着每个像素占 4 个字节的内存空间。当你需要非常高质量的显示效果时，非常好！但是，有时候需要折中一下，因为以前的设备内存很有限，如果全部使用 32 的像素格式来加载图片的话，会造成内存消耗过多。

当你加载大的图片的时候（比如背景图片），最佳实践是使用 16 位的像素格式来加载--也就是牺牲一点质量来减少内存开销。cocos2d 里面有[很多不同的像素格式](#)--这个教程中，我们选择 16 位的像素格式，RGB565，因为背景一般不需要透明效果。（少了 Alpha 通道，RGBA 就是有 Alpha 通道）

## 加载图片

init 方法的另外一部分，就是循环遍历一个图片数组，然后创建精灵并且计算精灵放置的坐标。这些精灵会一字排开，显示在屏幕上。同时把这些精灵的引用保存在 movableSprites 数组里面，这个数组后面会使用到。

最后，我们需要一些清理内存的操作。找到 dealloc 方法，然后添加下列代码：

```
[movableSprites release];  
movableSprites = nil;
```

就这么多！编译并运行，你将看到一排非常可爱的小动物，在等待你 touch 呢！



## 基于 touch 事件选取精灵

现在，我们将编写一些代码基于用户的 touch 事件来决定哪一个精灵被选到了。

第一步，就是激活你的 HelloWorldLayer 层，让它能够接收 touch 事件。在 init 方法的最后添加下列代码：

```
[[CCTouchDispatcher sharedDispatcher] addTargetedDelegate:self priority:0 swallowsTouches:YES];
```

注意，这是一种新的方式来激活层的 touch 事件--老的方式就是，设置层的 isTouchEnabled 属性为 Yes，然后实现 ccTouchesBegan 方法。如果你非常关心，这个新的方法和旧的方法的优缺点的话，可以参考译者翻译的[《如何使用 cocos2d 制作基于 tiled 地图的游戏教程》](#)。

接下来，在 HelloWorldScene.m 的底部添加一些新的方法：

```
- (void)selectSpriteForTouch:(CGPoint)touchLocation {
    CCSprite * newSprite = nil;
    for (CCSprite *sprite in movableSprites) {
        if (CGRectContainsPoint(sprite.boundingBox, touchLocation)) {
            newSprite = sprite;
            break;
        }
    }
    if (newSprite != selfSprite) {
        [selfSprite stopAllActions];
        [selfSprite runAction:[CCRotateTo actionWithDuration:0.1 angle:0]];
        CCRotateTo * rotLeft = [CCRotateBy actionWithDuration:0.1 angle:-4.0];
```

```
CCRotateTo * rotCenter = [CCRotateBy actionWithDuration:0.1 angle:0.0];
CCRotateTo * rotRight = [CCRotateBy actionWithDuration:0.1 angle:4.0];
CCSequence * rotSeq = [CCSequence actions:rotLeft, rotCenter, rotRight, rotCenter, nil];
[newSprite runAction:[CCRepeatForever actionWithAction:rotSeq]];
self.sprite = newSprite;
}
}

- (BOOL)ccTouchBegan:(UITouch *)touch withEvent:(UIEvent *)event {
    CGPoint touchLocation = [self convertTouchToNodeSpace:touch];
    [self selectSpriteForTouch:touchLocation];
    return TRUE;
}
```

第一个方法（`selectSpriteForTouch`）是一个帮助方法，这个方法遍历 `movableSprites` 数组中的所有精灵，查找第一个精灵位置与 `touch` 点位置相交的精灵。

注意，`CCNode` 有一个辅助属性叫做 `boundingBox`，它返回精灵的边界矩形。这比你手动计算精灵的边界矩形要好多了。因为，第一，它更简单；第二，它考虑了精灵的位置坐标变换。（比如锚点变了，要执行相应的矩阵变换，具体可以参考源代码，这些就不再细说了。）

如果找到一个匹配的精灵，那么就让这个精灵执行一些动画，这样用户就知道哪个精灵被选中了。如果动画还没执行完，又选中另一个精灵了，那么就中断前一个精灵的动画。这里的动画效果，使用了一系列的 `CCAction` 来实现的。

最后，`ccTouchBegan` 方法基于用户的 `touch` 事件调用上面的方法。注意，这里把 `touch` 坐标点从 `UIView` 的坐标系转换成了结点坐标系。为了实现这个目的，通过调用 `CCNode` 的一个辅助函数，`convertTouchToNodeSpace`。这个方法做了以下三件事：

- 计算 `touch` 视图（也就是屏幕）的 `touch` 点位置（使用 `locationInView` 方法）
- 转换 `touch` 坐标点为 `OpenGL` 坐标点（使用 `convertToGL` 方法）
- 转换 `OpenGL` 坐标系为指定结点的坐标系（使用 `convertToNodeSpace` 方法）

这是一个非常常用的转换过程，所以提供这个方法可以节约很多时间。

编译并运行代码，并且用手触摸这些动物。当你点中一个精灵的时候，它就会以一种非常可爱的方式旋转，表明它被你选中啦！



## 基于 **touch** 事件移动精灵和背景层

是时候让小动物移动了！基本的思想就是实现 `ccTouchMoved` 回调函数，然后计算本次 `touch` 点到上一次 `touch` 点之间的距离。如果一个动物被选中，将按照计算出来的 `touch` 偏移量来移动它。如果动物没有被选中，那就移动整个层，因此用户可以从左至右滚动层。

在编写代码之前，让我们花点时间来讨论一下，如何在 `cocos2d` 里面滚动一个层。

首先，看到下面两张图片：





(0, 0)



(-100, 0)

如你所见，你设置背景锚点在左下角（0，0），前景其它部分就在屏幕之外。黑色框框表示当前可见的区域，也就是屏幕范围。

因此，如果你将图片往右边滚动 100 个像素，你可以把整个 cocos2d 的层往左移动 100 个像素，如第二张图所示。

同时，你要确保不会移得太多。比如，你不能够把层往右移动，因为那样左边就是空白的了。

现在，你了解了一些背景信息，让我们看看代码怎么写吧！在文件的最后添加下列新的方法：

```
- (CGPoint)boundLayerPos:(CGPoint)newPos {
    CGSize winSize = [CCDirector sharedDirector].winSize;
    CGPoint retval = newPos;
    retval.x = MIN(retval.x, 0);
    retval.x = MAX(retval.x, -background.contentSize.width+winSize.width);
    retval.y = self.position.y;
    return retval;
}

- (void)panForTranslation:(CGPoint)translation {
```

```
if (selSprite) {
    CGPoint newPos = ccpAdd(selSprite.position, translation);
    selSprite.position = newPos;
} else {
    CGPoint newPos = ccpAdd(self.position, translation);
    self.position = [self boundLayerPos:newPos];
}
}

- (void)ccTouchMoved:(UITouch *)touch withEvent:(UIEvent *)event {
    CGPoint touchLocation = [self convertTouchToNodeSpace:touch];
    CGPoint oldTouchLocation = [touch previousLocationInView:touch.view];
    oldTouchLocation = [[CCDirector sharedDirector] convertToGL:oldTouchLocation];
    oldTouchLocation = [self convertToNodeSpace:oldTouchLocation];
    CGPoint translation = ccpSub(touchLocation, oldTouchLocation);
    [self panForTranslation:translation];
}
```

第一个方法（**boundLayerPos**），用来确保你在滚动层的时候不会超出背景图片的边界。你传递一个目标点坐标，然后相应地修改 **x** 值，保证不会超出边界。如果你不是很理解的话，可以拿出纸和笔，结合上面给出的图片，自己动手画一画。

接来的方法（**panForTranslation**）基于传入的目标点位置移动精灵（如果有精灵被选中就移动之，否则移动整个层）。具体的做法就是设置精灵或者层的位置。

最后一个方法（**ccTouchMoved**）是一个回调函数，它在你拖动屏幕上的手指时调用。像之前一样，把 **touch** 坐标转换成局部 **node** 坐标。因为没有辅助方法可以把前一个层的 **touch** 坐标转换成 **node** 坐标，因此，我们需要手工地调用那 3 个方法来执行这个操作。

然后，计算 **touch** 偏移量，通过把当前的点坐标减去上一个点坐标，然后调用 **panForTranslation** 方法。

编译并运行，现在，你可以通过拖拽来移动精灵和层啦！





## 如何在 **cocos2d** 里面识别手势

这里还有另一种方式来实现刚刚的 **touch** 处理--通过使用手势识别！

手势识别是新版本的 **ios sdk** 增加的内容（**iOS SDK 3.2** 引入）。说实话，这个功能太棒了！

首先，你不再需写一大堆代码来检测“点击（**tap**）、双击（**double tap**）、滑过（**swipe**）、平移（**pan**），挤压（**pinch**）”的区别了。你只需要创建一个手势识别对象，然后把它加到 **view** 里面。当有相应的手势发生的时候，会给你一个回调通知。

它非常容易使用，在 **cocos2d** 里面使用也非常方便。让我们看看它是如何工作的。

首先，回到 **init** 方法，把注册 **touch** 的调用注释掉，因为你将使用一种新的方法：

```
//[[CCTouchDispatcher sharedDispatcher] addTargetedDelegate:self priority:0 s
wallowsTouches:YES];
```

然后，回到 **DragDropAppDelegate.m**，找到 **applicationDidFinishLaunching** 方法，把最后一行替换掉：

```
CCScene *scene = [HelloWorld scene];
HelloWorld *layer = (HelloWorld *) [scene.children objectAtIndex:0];
UIPanGestureRecognizer *gestureRecognizer = [[[UIPanGestureRecognizer alloc]
initWithTarget:layer action:@selector(handlePanFrom:)] autorelease];
[viewController.view addGestureRecognizer:gestureRecognizer];

[[CCDirector sharedDirector] runWithScene:scene];
```

这段代码获得 **HelloWorld** 层的引用（它是 **HelloWorldScene** 的唯一孩子节点），然后，创建一个 **UIPanGestureRecognizer** 对象。注意，为了创建手势对象，你必须初始化它，并且指定回调函数--在本例中就是 **handlePanFrom** 方法。

创建完手势识别对象后，你需要把它加入到 **OpenGL** 视图中（**viewController.view**）。

接下来，在 **HelloWorldScene.m** 底部加入下面的代码：

```
- (void)handlePanFrom:(UIPanGestureRecognizer *)recognizer {

if (recognizer.state == UIGestureRecognizerStateBegan) {

CGPoint touchLocation = [recognizer locationInView:recognizer.view];
touchLocation = [[CCDirector sharedDirector] convertToGL:touchLocation];
touchLocation = [self convertToNodeSpace:touchLocation];
[self selectSpriteForTouch:touchLocation];

} else if (recognizer.state == UIGestureRecognizerStateChanged) {

CGPoint translation = [recognizer translationInView:recognizer.view];
translation = ccp(translation.x, -translation.y);
[self panForTranslation:translation];
```

```
[recognizer setTranslation:CGPointZero inView:recognizer.view];

} else if (recognizer.state == UIGestureRecognizerStateEnded) {

if (!self.sprite) {
float scrollDuration = 0.2;
CGPoint velocity = [recognizer velocityInView:recognizer.view];
CGPoint newPos = ccpAdd(self.position, ccpMult(velocity, scrollDuration));
newPos = [self boundLayerPos:newPos];

[self stopAllActions];
CCMoveTo *moveTo = [CCMoveTo actionWithDuration:scrollDuration position:newPos];
[self runAction:[CCEaseOut actionWithAction:moveTo rate:1]];
}

}
}
```

这个回调函数在 **pan** 手势开始、变更（用户连续拖拽）和结束的时候被触发。通过判别不同的状态来处理不同的行为。

当手势开始的时候，把 **touch** 坐标转换成 **node** 坐标，然后调用之前写好的 **selectSpriteForTouch** 方法。

当手势变更的时候，需要计算出手势移动偏移量。这里使用手势对象的方法 **translationInView** 来获得连续的偏移量。注意，这里需要把 **y** 值设置成负值。因为 **UIKit** 的坐标和 **OpenGL** 坐标系统不一样。（**UIKit** 左上角是（0，0），而 **OpenGL** 左下角的坐标是（0，0）。

之后把 **recognizer** 设置为 0，因为这个移动是连续的，而我们只需要得到离散的偏移量。

当手势结束的时候，这里有一些新的代码。另一个非常 **cool** 的事情就是，你可以通过 **UIPanGestureRecognizer** 对象可以得到 **Pan** 移动的速度。你可以使用这个速度来滑动层，这个效果就像你使用 **tableview** 一样。

因此，这个部分包含一些代码，基于移动的速度来计算移动点。然后运行 **CCMoveTo action**（使用 **CCEaseOut** 效果会更好）。

编译并运行代码，你可以通过手势识别来滑动层啦！



## 何去何从？

这里有本教程的[完整源代码](#)。

到目前为止，你应该知道如何在 cocos2d 里面使用 touch 事件来移动精灵了。同时，你也了解了在 cocos2d 里面如何使用手势识别对象。

现在，你可以尝试扩展本范例，可以使用不同的手势识别对象，比如挤压（pinch）和旋转（rotate）手势识别对象。

如果大家看教程有什么不明白的地方，都可以拿出来，大家一起讨论，共同提高！

**著作权声明：**本文由 <http://www.cnblogs.com/andyque> 翻译，欢迎转载分享。请尊重作者劳动，转载时保留该声明和作者博客链接，谢谢！