（译）使用 cocos2d 和 box2d 制作滚动背景

整理：qq963922432 和 apiter（泰然论坛管理组）

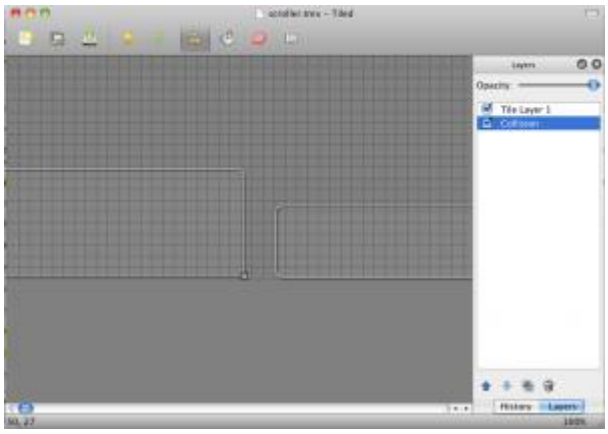
原文链接地址：http://www.uchidacoonga.com/2011/01/side-scrolling-the-background-in-box2d/

前言：这次我要翻译的两篇教程，其实和大家比较关心的“超级玛利”有关，就是如何使用 cocos2d 和 box2d 制作一个带有滚动背景的 platform 游戏。但是，这两篇教程并没有教大家如何制作超级玛利。但是，一些关键技术点还是有涉及到。相信看完这两篇文章，应该能对大家有所启发。如果哪位朋友有空，能利用本教程制作一个超级玛利游戏，并且分享出来的话，那就太好了。如果能再写一篇教程，那就再好不过了。我在此再给出一些超级玛利的图片资源链接。希望有兴趣的朋友可以尝试一下，有问题大家可以一起探讨一下。

译文：

因为我实在是找不到一个这样的教程，它教您如何使用 box2d 来制作一个滚动背景，同时把视角固定在玩家身上。所以，我自己来写一个这样的教程。在游戏画面中，玩家几乎总是固定在屏幕的某个地方，但是，背景在移动。所以，你看起来好像就是玩家在游戏世界里面以第一视角在走一样。

如果您对 box2d 的一些基础知识还不了解的话，建议您先看我翻译的 box2d 基础教程。



对于这个简单的教程，我将使用 Tile Map 编辑器来做（因为超级玛利的关卡也是使用 TileMap 制作的）。如果你对于如何下载和使用 TiledMap 编辑器不熟悉的话，建议你先看我翻译的关于 TiledMap 的教程。上图中可以看出，我要制作的地图有2个层。一个是地图层，另外一个是对象层，对象层里面定义了一些对象，用来处理碰撞检测。在这个教程中，我只是简单地画了两个矩形盒子。而本游戏中的地图层完全是空的，没有任何地图信息，大家可以用我提供的超级玛利的图片资源来画一关。然后，在代码里面，我将把对象层里面的对象的位置信息读出来，然后创建相应的 box2d 对象。下面是添加 tiledMap 的代码：

```
- (void) addScrollingBackgroundWithTileMap {
    tileMapNode= [CCTMXTiledMap tiledMapWithTMXFile:@"scroller.tmx"];
    tileMapNode.anchorPoint= ccp(0,0);
```

```
    [self addChild:tileMapNode];
}
```

上面的代码加载“scroller.tmx”，然后把它加载到当前层中，注意要把“scroller.tmx”添加到 resource 目录下去。如果你去查看 CCTMXTiledMap 的源码的话，你会看到，那些 tiles 实际上都被创建成了 CCSprite 对象。但是，话说回来，我在这个例子中没有使用任何 tileset 来画地图层，所以你在屏幕上看不到任何东西。接下来的代码是根据对象层中的对象来创建 box2d 的对象。

```
- (void) drawCollisionTiles {
    CCTMXObjectGroup*objects= [tileMapNode objectGroupNamed:@"Collision"];
    NSMutableDictionary* objPoint;

    float x, y, w, h;
    for (objPointin [objects objects]) {
        x= [[objPoint valueForKey:@"x"] floatValue];
        y= [[objPoint valueForKey:@"y"]floatValue];
        w= [[objPoint valueForKey:@"width"]floatValue];
        h= [[objPoint valueForKey:@"height"]floatValue];

         CGPoint _point=ccp(x+w/2.0f,y+h/2.0f);
        CGPoint _size=ccp(w,h);

         [self makeBox2dObjAt:_point
                    withSize:_size
                     dynamic:false
                    rotation:0
                    friction:0.0f
                     density:0.0f
                 restitution:0
                       boxId:-1];
    }
}
```

这里的 drawCollisionTiles 方法把 tileMap 中的碰撞矩形读出来，然后位置和长宽信息创建相应的 box2d 对象。这里需要注意的是，读取坐标点使用的数据类型一定要是 float 型，否则会出现你在 tileMap 编辑器中画出来的矩形区域与你程序生成的矩形区域大小不一致的问题!!! 大家一定要记住！还有，这里我把 friction 设置成了0。这会使我们的主角（其它就是个圆）沿着平台滑动。如果你把这个值设置成非0值，比如0.4，那么球会在平台上面滚动。

下面我们看看 makeBox2dObjAt 方法：

```
- (void) makeBox2dObjAt:(CGPoint)p
              withSize:(CGPoint)size
               dynamic:(BOOL)d
              rotation:(long)r
              friction:(long)f
```

```
                density:(long)dens
            restitution:(long)rest
                 boxId:(int)boxId {

    // Define the dynamic body.
   //Set up a 1m squared box in the physics world
   b2BodyDef bodyDef;
//    bodyDef.angle = r;

    bodyDef.position.Set(p.x/PTM_RATIO, p.y/PTM_RATIO);
    bodyDef.userData= NULL;
   //bodyDef.userData = sprite;

    b2Body*body= world->CreateBody(&bodyDef);

    // Define another box shape for our dynamic body.
   b2PolygonShape dynamicBox;
    dynamicBox.SetAsBox(size.x/2/PTM_RATIO, size.y/2/PTM_RATIO);

    // Define the dynamic body fixture.
   b2FixtureDef fixtureDef;
    fixtureDef.shape= &dynamicBox;
     fixtureDef.density= dens;
    fixtureDef.friction= f;
    fixtureDef.restitution= rest;
    body->CreateFixture(&fixtureDef);
}
```

上面这个方法负责创建 box2d 的对象。这是标准的创建 box2d 对象的方式。

Let's Continue...

现在，把所有的代码都拿出来吧。。。

```
-(id) init {
   if( (self=[super init])) {
       b2Vec2 gravity= b2Vec2(0.0f,-9.8f);
      bool doSleep= true;
       world= new b2World(gravity, doSleep);

        m_debugDraw= new GLESDebugDraw(PTM_RATIO);
       world->SetDebugDraw(m_debugDraw);
       uint32 flags= 0;
       flags+= b2DebugDraw::e_shapeBit;
```

```
        m_debugDraw->SetFlags(flags);


        [self addScrollingBackgroundWithTileMap];
        [self drawCollisionTiles];

        CCSprite*sprite= [CCSprite spriteWithFile:@"Icon-Small.png"];
        sprite.position= ccp(100.0f,180.0f);

        [self addChild:sprite];

        b2BodyDef playerBodyDef;
        playerBodyDef.type= b2_dynamicBody;
        playerBodyDef.position.Set(sprite.position.x/PTM_RATIO,
sprite.position.y/PTM_RATIO);
        playerBodyDef.userData= sprite;

        playerBody= world->CreateBody(&playerBodyDef);

        b2CircleShape circleShape;
        circleShape.m_radius= 1.0f;

        b2FixtureDef fixtureDef;
        fixtureDef.shape= &circleShape;
        fixtureDef.density= 1.0f;
        fixtureDef.friction= 0.0f;
        fixtureDef.restitution= 0.0f;
        playerBody->CreateFixture(&fixtureDef);

        b2Vec2 impulse= b2Vec2(10,0);
        playerBody->ApplyLinearImpulse(impulse, playerBody->GetWorldCenter());

        [self scheduleUpdate];
    }
   return self;
}
```

首先，在上面的代码中，创建了一个 box2d world，然后设置了 debug draw。为什么要设置 debug draw？因为我们的游戏世界里面什么都没有，如果不设置 debug draw，你将什么都看不见！而且 debug draw 可以帮助我们更好地调试 box2d。然后调用 addScrollingBackgroundWithTileMap 和 drawCollisionTiles，用来加载 tiledMap 及创建相应的 box2d 对象。我们使用 cocos2d 的 log 来作为我们的主角精灵。前面已经提到了，它只是一个圆形的 body。接下来，我们给它一个冲力。你可以把这个力改大一点，那么主角就会走得更快。

```
- (void) update:(ccTime)dt {
   //It is recommended that a fixed time step is used with Box2D for stability
   //of the simulation, however, we are using a variable time step here.
   //You need to make an informed choice, the following URL is useful
   //http://gafferongames.com/game-physics/fix-your-timestep/

    int32 velocityIterations= 8;
    int32 positionIterations= 1;

    // Instruct the world to perform a single step of simulation. It is
   // generally best to keep the time step and iterations fixed.
    world->Step(dt, velocityIterations, positionIterations);


    //Iterate over the bodies in the physics world
   for (b2Body* b= world->GetBodyList(); b; b= b->GetNext()) {
      if (b->GetUserData()!= NULL) {
         //Synchronize the AtlasSprites position and rotation with the
corresponding body
          CCSprite*myActor= (CCSprite*)b->GetUserData();
          myActor.position= CGPointMake( b->GetPosition().x* PTM_RATIO,
                    b->GetPosition().y* PTM_RATIO);
          myActor.rotation= -1 * CC_RADIANS_TO_DEGREES(b->GetAngle());
       }
    }

    b2Vec2 pos= playerBody->GetPosition();
   CGPoint newPos= ccp(-1 * pos.x* PTM_RATIO+ 50, self.position.y* PTM_RATIO);
    [self setPosition:newPos];
}
```

接下来，上面的代码是标准的 box2d 代码，它负责处理 box2d 世界的仿真。根据 body 的位置来更新 sprite 的位置。实际给背景添加滚动的代码，只有最后3行。（说实话，本教程基本上都是知识的回顾。只有这3句话是新的，呵呵）

首先，我们获得玩家在 box2d 世界中的位置，然后我们需要把它转换成 cocos2d 的像素值，通过乘以 PRM_RATIO。y 值还是不变，因为，我们只想让背景在 x 轴方向变化，就是水平滚动。而屏幕是从右边滚动到左边，所以要乘以-1.而50在这里只是让玩家每次都远离屏幕左边一些。如果你不添加这个50的话，那么玩家每次都会在屏幕的左边。而且是只有半边球会露出来。