跑酷教程3

**[教程内所有素材来自网络，如有侵权，请联系删除]**

在第三篇教程这里，我们主要是为了添加障碍物，以及金币，并能让地图按照若干种定式不断的生成出来，为了简化问题，我们这里只分成两种物体，一种是障碍物——当游戏主角与其相撞时就会触发Crash事件，第二种是金币，金币在地图里会不断地旋转，若与游戏主角相碰的话，那么他就会触发GetMoney事件。

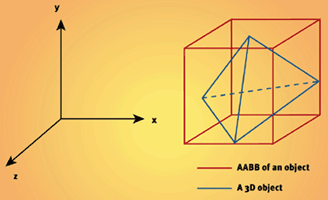
如何检测两个物体是否相撞呢？我们先从极端的办法开始考虑，有两个极端：

第一个办法，计算两个物体中心点之间的距离，如果这个距离小于等于一个定值，那么就判定两物体已经碰撞，如果大于则判定两物体相离，这个方法相当于将两个物体看作了两个球体，对于游戏中的一些情形而言，这样的方法已经够了，但是对于一些游戏而言，模型并没有那么的“圆”，所以很可能在模型与模型之间明显没有碰到的时候，就已经触发碰撞了。

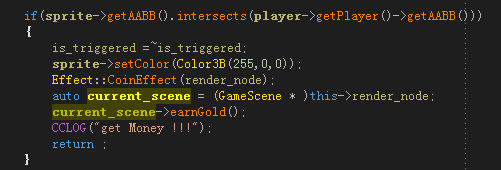
第二个办法，是针对把两个模型都看成复杂的多边形集合，然后逐一从中取出多边形进行多边形相交检测，这种方法最准确，可以说是达到“像素级”的检测，但是随着，游戏领域的不断蓬勃发展，模型的精度越来越高，单个模型包含的多边形总数也是爆炸性的增长。用这种方式去检测模型，效果应当是非常的不错，但是如果一个游戏场景中，包含了多个这样的物体，那么在移动端可能性能会大幅的退化。

现在我们已经得到两个极端的解法，在实际项目中，这两种做法，都不是特别好的实践，我们现在需要的是一种这样的做法——1、它要尽可能的逼近一个模型的多边形特征，2、它的检测算法要足够的简单快速，接下来我就要介绍在Cocos2dx 3d部分的AABB以及 OBB 类，他们就提供了一个快速的算法用于简单的碰撞检测。

所谓的AABB，就是轴对齐包围盒，所谓的包围盒大家可以想象一下有一个盒子——它恰好能将模型给装起来，所谓的轴对齐就是说这个包围盒是与坐标系的x,y,z的轴对齐的，AABB的好处就是把对模型和模型的之间是否相交的问题转换成了两个简单的包围盒(长方体)是否相交的问题 ，同时因为包围盒是与轴对齐的，所以判断两个包围盒是否碰撞，只需要简单的把包围盒的最大最小坐标对相互比较一下大小就好了。

  
在cocos2dX中，AABB包围盒可以通过如下方式获得：

C:\Users\ziwen.tang\AppData\Roaming\Tencent\Users\576419632\QQ\WinTemp\RichOle\G@OSG6P(7EGCPX_]NB5{G@A.png，AABB类提供一个intersects方法,所以，计算两物体的碰撞就非常简单了。请看下图代码



我们直接将当前的金币以及游戏主角的AABB拉取出来，然后直接检测，就可以得到是否相撞，在我们这个demo中，金币被撞上了之后，会变一个颜色，所以我们使用一个成员变量is\_triggered来确保一个金币只会触发一次。

刚才，如大家所见，AABB确实非常的有用，但是现在我们来想一个另外的问题，如果，一个模型，比方说一艘战舰：



它的宽和长的差距很大，那么用AABB来做碰撞检测就可能会遇到不太合适的地方了，比如说一艘战舰在偏航角方向进行了一个转向，可是AABB的包围盒是与坐标轴对齐的，这样就会产生非常大的误差——原本不应该碰到的地方碰了，该碰的地方没碰。

遇到这种情况，只要让包围盒跟着物体的旋转而旋转就好了，在此引入一个新的包围盒的概念OBB，OBB可以看作是带方向的AABB，OBB通过AABB创建而来。

我们需要对创建出来的OBB乘上对应精灵的transform，让旋转生效。