cocos2d之Box2D详解 删除物理世界对象

DionysosLai 2014-5-8

物理世界对象，有创造，就有销毁。

通常，我们在更新函数中会这样写到:

void HelloWorld::update(float dt)

{

/// 第一件事情就是调用world对象的step方法，这样它就可以进行物理仿真了。

/// 这里的两个参数分别是“速度迭代次数”和“位置迭代次数”--你应该设置他们的范围在8-10之间。（译者：这里的数字越小，

/// 精度越小，但是效率更高。数字越大，仿真越精确，但同时耗时更多。8一般是个折中，如果学过数值分析，应该知道迭代步数的具体作用）。

world->Step(dt, 10, 10);

/// 使我们的精灵匹配物理仿真。

for (b2Body \*b = world->GetBodyList(); b; b = b->GetNext())

{

/// 遍历world对象里面的所有body，然后看body的user data属性是否为空，如果不为空，就可以强制转换成精灵对象。接下来，就可以根据body的位置来更新精灵的位置了。

if (b->GetUserData() != NULL)

{

CCSprite\* ballData = (CCSprite\*)b->GetUserData();

ballData->setPosition(ccp(b->GetPosition().x\*PTM\_RATIO,

b->GetPosition().y\*PTM\_RATIO));

/// 弧度值转化为角度值 cocos2d里面使用的是角度值，而box2d里面使用的是弧度值。单位不统一，所以需要转化。为什么前面\*-1呢？

/// 因为cocos2d的角度是顺时针，而box2d是逆时针。

ballData->setRotation(-1\*CC\_RADIANS\_TO\_DEGREES(b->GetAngle()));

}

}

}

在这里，我们做几件事，一是：遍历物理世界中的对象；将物理世界对象数据，转换成我们的数据对象，然后更新对象的位置和旋转度。

如果我们要销毁其中的某个物理世界对象。通常我们会赋予我们的对象一个属性，然后根据这个属性来判断是否要销毁该对象。

如下面代码所示：

m\_world->Step(delta, 10, 10);

/// 使我们的精灵匹配物理仿真。

for (b2Body \*b = m\_world->GetBodyList(); b; ->GetNext())

{

if (b->GetUserData() != NULL)

{

boxSprite\* ballData = (boxSprite\*) b ->GetUserData();

if (ballData->getDead())

{

m\_world->DestroyBody(b);

}

else

{

ballData->setPosition(ccp(b ->GetPosition().x\*PTM\_RATIO,

b ->GetPosition().y\*PTM\_RATIO));

ballData->setRotation(-1\*CC\_RADIANS\_TO\_DEGREES(b ->GetAngle()));

}

}

}

Ps：ballData->getDead()，这里的getDead函数，就是我们对对象赋予的一个属性，来说判断我们是否要销毁对象。

这段代码有个很严重的问题，就是当我们销毁对象时，这个对象的所有信息都将被消除，那么我们将会去不到去下一个节点信息，正确的做法就是要提前获取下一个节点，同时b->GetUserData() == NULL 情况下，我们也要获取下一个节点，修改代码如下：

m\_world->Step(delta, 10, 10);

/// 使我们的精灵匹配物理仿真。

for (b2Body \*b = m\_world->GetBodyList(); b; )

{

if (b->GetUserData() != NULL)

{

b2Body\* b2node = b;

b = b2node->GetNext();

boxSprite\* ballData = (boxSprite\*)b2node->GetUserData();

if (ballData->getDead())

{

m\_world->DestroyBody(b2node);

}

else

{

ballData->setPosition(ccp(b2node->GetPosition().x\*PTM\_RATIO,

b2node->GetPosition().y\*PTM\_RATIO));

ballData->setRotation(-1\*CC\_RADIANS\_TO\_DEGREES(b2node->GetAngle()));

}

}

else

{

b = b->GetNext();

}

}

当然这是一个方法，如果不考虑效率的话，我们通过不修正我们对象的位置信息，我们也可以看起来，就像是将对象销毁掉，即在删除这段代码即可：

ballData->setPosition(ccp(b2node->GetPosition().x\*PTM\_RATIO,b2node->GetPosition().y\*PTM\_RATIO));

ballData->setRotation(-1\*CC\_RADIANS\_TO\_DEGREES(b2node->GetAngle()));

这是删除对象一个方法，还有一个方法就是，我们可以将要删除对象保存在一个临时容器中，等物体链表遍历完成之后，就可以删除临时容器中的物体了。