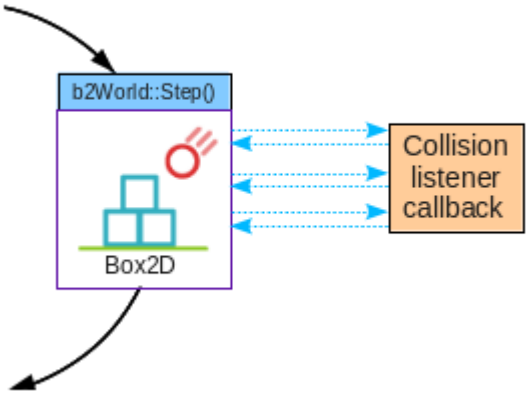
Box2D C++ 教程-安全地移除物体

Dec 12th, 2012 | Comments

声明：本文翻译自Box2D C++ tutorial-Removing bodies safely，仅供学习参考。

安全地移除物体

除非你正在完成一个最基本地游戏，否则很有可能你会在游戏进行的某一时刻删除一些物体。比如说当玩家杀死一个敌人后，所发出的子弹需要被清除掉，还比如说两个物体发生碰撞其中一个完全毁坏，等等很多此类的例子。在代码中移除一个物体简单的令人发指 - 只要调用world->DestroyBody(b2Body*)，如果进行删除操作期间没有对此物体发生调用，那么整个删除操作就完成了。问题经常出现在删除物体的同一个时间步长内发生对此物体的调用，通常是一个碰撞回调。如果我们回想一下之前碰撞回调的方法，我们会发现不能在碰撞回调内部删除物体，因为此时world正好运行在同一个物理步长中，此时被删除的物体还处于运行期间，这么做不是一个好的办法。

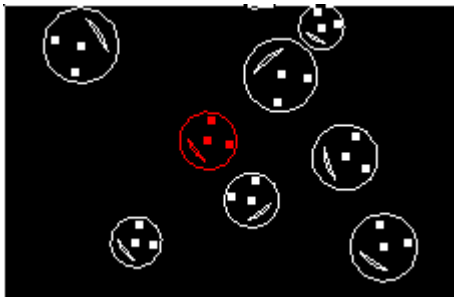


如果你使用线程或者计时器（timers）来移除物体的话，会经常碰到类似的情况，比如说你想在游戏中某个事件发生后的几秒钟删除一个物体。如果计时器执行删除的时候world当中的执行步长还在运行当中，也会引起类似的问题。

解决方案其实很简单。当你获取一个需要移除某个物体的碰撞回调的时候，只要把物体添加到一个自定义的物体删除链表当中，在此时间步长执行完之后整体进行删除就行了。

例子

作为一个例子，让我们做一个示范，根据场景中碰撞的物体来进行移除。至于场景的搭建，我们可以借鉴碰撞回调话题中的场景。我们可以从碰撞回调话题中的末尾处找到相关部分。简单回顾一下，此场景中有一个红色小球，当这个红色小球碰到另一个小球的时候就把红色进行状态传递给另一个小球，就好像在玩儿标签游戏一样，红色就代表这个标签：



这一次，让我们把红色的小球想象成携带有可致命的病毒，但是它把病毒感染给其它小球之前是不会死亡的。一旦它通过碰撞把病毒感染给另一个小球，之前感染病毒的那个小球就会从world中移除。

完成这个例子我们只需要修改很少的代码。我们需要用一个链表来存储时间步长执行完之后需要从world中删除的物体，然后通过碰撞回调方法把已经感染的小球放到链表里。要真正删除一个小球可能我们需要在Ball类内部添加一个析构函数，接下来我们会把物理物体从Box2D世界中移除出去。

```
1  //at global scope
2  std::set<Ball*> ballsScheduledForRemoval;
3
4  //revised handleContact function
5  void handleContact( Ball* b1, Ball* b2 ) {
6      if ( ! outbreak )
7          return;
8      if ( b1->m_imIt ) {
9          ballsScheduledForRemoval.push_back(b1);
10         b2->m_imIt = true;
11     }
12     else if ( b2->m_imIt ) {
13         ballsScheduledForRemoval.push_back(b2);
14         b1->m_imIt = true;
15     }
16 }
17
18 //implement Ball class destructor
19 Ball::~Ball()
20 {
21     m_body->GetWorld()->DestroyBody( m_body );
22 }
```

‘outbreak’ 变量是可选的，此变量设置成全局可见并初始化为false，为的是我能够用键盘控制传染什么开始 - 如果一开始不控制一下的话红色小球会在眨眼间就会感染很多其它小球然后瞬间干掉很多小球，反而显得有点儿无趣。当然了根据我自己的猜测你可以在一开始的时候把红色小球放置到一个比较空的地方，这取决于你。

现在让我们转向Step()方法，当调用Test::Step()方法后并在渲染场景之前，我们需要处理将要删除的物理实体。

```
1  //process list for deletion
2  std::set<Ball*>::iterator it = ballsScheduledForRemoval.begin();
```

```
3  std::set<Ball*>::iterator end = ballsScheduledForRemoval.end();
4  for (; it!=end; ++it) {
5      Ball* dyingBall = *it;
6
7      //delete ball... physics body is destroyed here
8      delete dyingBall;
9
10     //... and remove it from main list of balls
11     std::vector<Ball*>::iterator it = std::find(balls.begin(), balls.end(), dyingBall);
12     if ( it != balls.end() )
13         balls.erase( it );
14 }
15
16 //clear this list for next time
17 ballsScheduledForRemoval.clear();
```

现在场景中你应该能够看到只有一个红色小球，不断的通过碰撞感染其它小球以及不断的有小球从场景中被移除。

使用线程或者计时进行移除物体方法也是类似的，只要把所有需要移除的物体放到一个链表中，当你确定world的执行步长运行结束的时候就可以整体处理它们。一个简单的方法也可以在游戏循环（例如，60fps速率下2秒等于120次的步长调用）中进行’定时移除’，在实体中设置一个值，然后每一帧减少一直到变为零，然后移除它。

顺便说一下，虽然这个话题专门说了下移除物体，但是这个地方经常引起一些问题，信物体的创建，添加和移除定制器必须用这种方法进行处理。

Posted by Snow Dec 12th, 2012 box2d-tutorials

« Box2D C++ 教程-查询 World Box2D C++ 教程-跳跃问题 »

Comments

About Me