物理引擎与图形渲染引擎绑定的研究与实现

欧阳慧琴， 陈福民
(同济大学 计算机中心，上海 200092)

摘 要：近年来，物理引擎在计算机仿真模拟和游戏应用等领域扮演着越来越重要的角色。一款专业的图形渲染引擎与物理引擎相融合，能够更好地体现出虚拟现实场景的逼真性。在对国外专业的面向对象图形渲染引擎和牛顿物理引擎的基本内容和设计原理进行研究的基础上，分析了如何有效地将物理引擎与图形渲染引擎相互绑定，并给出了绑定系统的类库结构及实现方法。最后在实现的绑定平台下构建出了两个体现物体物理特性的虚拟场景。

关键词：物理引擎；面向对象图形渲染引擎；牛顿物理引擎；绑定；碰撞检测
中图法分类号：TP317.4 **文献标识码**：A **文章编号**：1000-7024 (2008) 21-5580-03

Research and implementation of binding of physics engine and graphic rendering engine

OUYANG Hui-qin, CHEN Fu-min
(Computing Center, Tongji University, Shanghai 200092, China)

Abstract: In recent years, physics engine plays a more and more important role in areas of computer simulation, computer games and so on. With the amalgamation of professional graphic rendering engine and physics engine, the more natural effect of the virtual reality scene is displaied. The basic contents and design principles of object-oriented graphics rendering engine and Newton game dynamics is researched, and how to bind these two kinds' engines is analyzed. And the class library's structure and implementing method of the binding system is given. Finally, there are two instances about the virtual scene based on the binding platform, which display physics attributes of virtual objects.

Key words: physics engine; OGRE; Newton game dynamics; binding; collision detection

0 引 言

计算机诞生的最初目的之一，就是计算火炮的弹道，这实际上就是最初的物理引擎演示。物理特性的计算也广泛应用于从计算机科学设计到仿真模拟的各个方面。在越来越盛行的计算机游戏领域，物理引擎的作用也愈加显得重要。除了拥有好的渲染画面，动听的音乐，引人入胜的剧情外，严谨完善的物理引擎也是构成一款出色游戏的必要组成部分。物理引擎使用对象物理属性(动量、扭矩或者弹性等)来模拟实体行为，这不仅可以得到更加真实的结果，对于开发人员来说也比编写行为脚本要更加容易掌握。

OGRE(object-oriented graphics rendering engine)作为一个开源的面向对象的图形渲染引擎，拥有着强大的功能，诸如虚拟场景的管理与渲染，光照和材质处理，角色动画，粒子系统与自然模拟等特效的实现等。

但是，当虚拟现实场景需要实现比较复杂的物体碰撞、滚动、滑动或者弹跳的时候，OGRE这个专注于图像处理的图形引擎必须与专业的物理引擎API相绑定，才能实现实体物理特性的展示。

1 Newton物理引擎

Newton物理引擎是一个具有良好跨平台性和精准碰撞检测的物理环境实时仿真引擎。它的API提供场景管理、碰撞检测和动态行为管理等功能，并且短小、快速、稳定，能够很容易的嵌入到其它应用程序中。

1.1 基本原理

物理引擎的工作流程就是对渲染场景的每一帧进行物理模拟，再对具有物理特性的模拟实体进行状态与受力分析。然后，进行碰撞检测，找出运动实体间相互的约束信息，并通过牛顿力学原理计算出每个实体新的位移和速度，从而更新实体的方位，得到新的虚拟场景[1]。基本处理步骤如下：

(1)物理模拟。以牛顿力学定理为基础，根据实体对象的物理属性，计算各种力对实体的作用，从而确定其运动状态。

(2)计算各个实体对象所受的合力(Force)和合力矩(Torque)。合力采用向量叠加法计算，合力矩可用平行轴定理来计算。

(3)进行碰撞检测。根据实体的运动状态检测它们之间的交互情况。一般的碰撞检测分为两个检测阶段：粗略检测阶段主要是利用物体的包围盒(AABB，OBB，Sphere)，配合一种空

收稿日期：2007-11-15 **E-mail**：feiqi515@163.com
作者简介：欧阳慧琴 (1984－)，女，江西永新人，硕士研究生，研究方向为图形图像处理、虚拟现实；陈福民 (1945－)，男，上海人，教授，博士生导师，研究方向为图形图像处理、虚拟现实。

间划分法(BSP, OBBTree)来进行粗略的层次型碰撞检测。目的是快速过滤掉没有发生碰撞的物体。如果物体的包围体发生碰撞，那么就进入到精细检测阶段进行更精细的判断。精细检测一般是基于三角形的碰撞检测，目的是找到确切的碰撞点坐标和碰撞法线。

(4)如果有碰撞发生，响应碰撞并进行碰撞处理。一般用冲量定理来计算刚体碰撞后的速度。

另外，物理引擎不知道也不关心它进行模拟的对象是如何显示的。它根据对象的物理(而非图形)描述对这些对象的运动和交互过程进行模拟，该描述信息有可以用于生成“追踪”模拟的显示。对象物理特性的图形显示则交给图形渲染引擎来处理。

1.2 基本实现框架

Newton物理引擎根据以上的基本原理，嵌入底层图形接口OpenGL，并提供了大量的功能特性接口函数。它的具体实现主要分为3步[2]：①引入Newton引擎库文件，将所嵌入的图形接口OpenGL初始化。②创建物理场景。初始化物理引擎中的主要构成要素，如世界、实体、关节等。初始化流程如图1所示。③执行图形渲染引擎主循环，即按帧绘制场景。它的实现过程具体可描述为：设置场景更新时间步，在时间步内进行受力分析和物理模拟，进行碰撞检测并响应碰撞，最后更新实体状态，重新渲染场景。

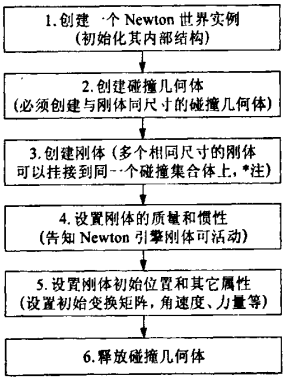

图1 Newton物理引擎初始化步骤

注：当要创建多个刚体时，Newton物理引擎引入了回调机制(Callback Functions)来管理刚体的创建、设置和销毁工作。刚体信息发生变化时，引擎通过这些回调函数与应用程序通信，减轻了应用程序的负担。

1.3 主要构成元素

• 世界(World)：最先创建，用以存放其它所有的对象。

• 实体(Body)：物理引擎中可相互交互的基本对象。包括刚体、非刚体和一些特殊形态的物体等。

• 碰撞几何体(Collision Primitives)：刚体等实体在创建前需要定义一个与其大小相同的碰撞几何体来约束它的形状。多个尺寸相同的实体可以同时挂接到一个碰撞几何体上。

Newton物理引擎提供几种方式来描述碰撞几何体：基本形状(Primitive Shapes)，如盒状体、椭圆体、圆筒等；凸包(Convex Hulls)，凸包用空间的一系列点来创建一个最小的凸型；碰撞树(Tree Collisions)，从碰撞树创建的刚体都被自动赋予无限大的质量，所以它被用来创建那些不需要运动的背景模型。

• 关节(Joint)：两个刚体的连接部位，它可以影响和约束不同刚体之间的行为和运动方式。

根据关节自由度的不同，Newton物理引擎提供了4种基本的关节，它们是：杵臼关节(ball and socket)，连接刚体能在一定角度范围内向各方向运动；铰链(hinge)，刚体可围绕一个轴转动；滑动关节(slider)，刚体可在一个轴上随意滑动；螺旋状关节(corkscrew)，兼具铰链和滑动关节的功能。

• 材质(Material)：这里的材质不同于物体本身所具有的材质属性(即对光是何种反射方式)。Newton引擎中的材质表示当两个不同材质的物体相互碰撞时所表现出来的一种碰撞特性。它定义了两个部分，MaterialID和MaterialPair。MaterialID代表系统中用到的材质(图形材质)，MaterialPair则描述了两种材质的物体发生碰撞所产生的行为。

• 常用附加功能：光线跟踪(Ray Casting)，用它可以实现虚拟场景中物体的拾放、目标的射击及其它人工智能算法。迭代器功能(Iterators functions)，用在场景管理器需要对场景中每一个物体提取相关信息的情况下。另外还有活动物体实时跟踪回馈机制等。

2 OGRE与Newton物理引擎的绑定

2.1 OGRE图形引擎的扩展性

OGRE是用C++开发的面向对象且使用灵活的图形渲染引擎。它的目的是让开发者能更方便和直接地开发基于3D硬件设备的应用程序或游戏。引擎中的类库对更底层的系统库(如：Direct3D和OpenGL)的全部使用细节进行了抽象，并提供了基于现实世界对象的接口和其它类。

作为一个功能强大的图形引擎，OGRE提供了与其它程序相融合的接口，可以很容易的与其它库文件绑定，从而实现物理、音效、网络连接等更加强大的功能。OGRE源代码中提供了一个ReferenceAppLayer框架类，用来与其它插件库文件集成[3]。要插入物理系统，具体是在ReferenceAppLayer框架类中引入物理引擎库文件，再进行一些数据结构方面的转换，比如将Newton物理引擎的4×4阶矩阵换作OGRE中的四元数和向量。之后的工作就是在OGRE图形引擎应用框架中使用物理引擎提供的各种组成元素的接口函数。

2.2 OGRE与Newton物理引擎的绑定

2.2.1 OgreNewt类库

OGRE本身提供的ReferenceAppLayer框架类可以用来与物理引擎集成，但为了体现OGRE面向对象的接口特性，简化用户应用程序的操作，Walaber编辑了一个基于OGRE和Newton引擎的OOP类库——OgreNewt[4]。OgreNewt类库将Newton SDK中的所有物理特性接口函数嵌入到一组基于OGRE平台的面向对象的类中，使得Newton物理引擎中的碰撞几何体绑定到OGRE实体网格上，从而实现了物体的物理特性与渲染特性的结合，为实现OGRE和Newton引擎的相互绑定提供了一个很好的层次类结构和实现平台。

具体来说，OgreNewt类库是将设置的一组融入Newton物理引擎特性接口的类全部放入一个名为OgreNewt的名字

空间，其中具体的类结构关系如图2所示。从图中可以清楚的看出各种特性类的层次关系和包含结构。为了方便应用程序的调用，这组特性类的头文件被包含进一个名为OgreNewt.h的头文件。而每一个特性头文件都包含进Newton物理引擎头文件 Newton.h 和主面向应用的 OGRE 头文件 Ogre.h。本文就是在参考 OgreNewt 类库的基础上，对其内部的类结构进行了一定的修改，调整了一些特性类的相关函数，使其与 Newton 物理引擎的融合更加简单明了，调用的API函数功能更加内聚，并实现了不同特性类之间的松耦合。同时，将 OGRE 引擎的相关功能类的实例作为绑定系统中各特性类的数据成员，在进行场景创建和图形渲染时，只需调用这些 OGRE 类对象的相关实现函数，就能够将虚拟物体的物理特性与场景的渲染特性相结合，从而实现OGRE和Newton引擎的相互绑定。

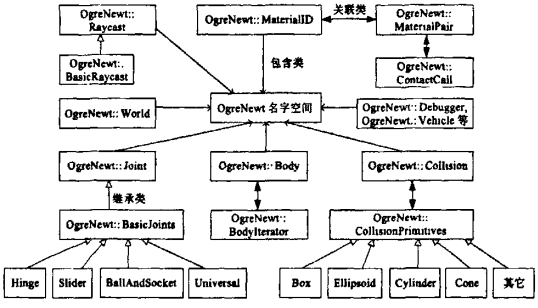

图 2　OgreNewt 名字空间类结构关系

2.2.2　绑定平台上应用程序的具体实现

通过对原有 OgreNewt 类库的局部调整与修改，并在 IDE 中设置相应的库文件和包含目录，就能够在实现绑定后的引擎平台上创建体现渲染场景中实体物体特性的应用程序。

(1)在应用程序头文件中包含进 OgreNewt.h 头文件(仍引用原 OgreNewt 类库头文件名)。

(2) 新建应用程序类。它继承自 OGRE 中的示范程序类(class ExampleApplication)。应用程序类中包括建立场景 createScene()和创建帧监听器 createFrameListener()两个重要的成员函数。createScene()函数负责 OGRE 渲染场景的建立和物理引擎中相关实体的设置，然后再把物理实体与 OGRE 场景中的 Node 相挂接。Newton 引擎中支持的物理特性和基本功能都可以通过在createScene()函数中添加相应的函数接口来实现。createFrameListener() 函数负责在嵌入物理引擎的渲染场景中监听外部输入设备的操作，从而实现场景的实时交互功能。具体可见图3主程序伪代码。另外，还必须创建类数据成员。基本的数据成员包括 Newton 的 World、OGRE 的场景节点 SceneNode 和 FrameListener 指针数据成员等。

(3)建立主函数。创建类实例对象，实现嵌入物理引擎机制的图形引擎平台上的场景渲染。

图4的代码文件展示了一个简单应用程序的头文件。以上3步基本上建立起绑定平台上应用程序的框架。利用这个框架并在类中加以具体要实现的实体属性及其功能，就可以轻松实现展现实体物理特性的逼真性虚拟场景的实时渲染。

```
//主程序伪代码
createScene()
{
    创建 Ogre 天空盒 SkyBox;
    创建地形实体和其它场景实体;
    物理引擎碰撞几何体的创建;
    设置刚体;
    创建关节、材质、光线跟踪等物理特性;
    ...
    设置照相机和灯光位置;
}
createFrameListener()
{
    调用Ogrenewt帧监听器创建类，创建Ogre帧监控器;
}
```

图 3　OgreNewt 应用程序的主程序伪代码

```
//包含 ogre 应用程序头文件
#include<ExampleApplication.h>
//包含 OgreNewt 头文件
#include<OgreNewt.h>
class OgreNewtonApplication:
public ExampleApplication
{
public:
    OgreNewtonApplication(void);
    ~OgreNewtonApplication(void);
protected:
    void createFrameListener();
    voed createScene();
private:
    OgreNewt::World* m_World;
    Ogre::SceneNode* msnCam;
    Ogre::FrameListener* mNewtonListener;
    ...
};
```

图 4　一个简单的应用程序头文件

3　实例实现

通过上文的分析，在Newton物理引擎和OGRE图形渲染引擎相互绑定的平台下，实现了一个模拟传送带的OGRE渲染场景，其中包括传送带的滚动、场景材质的设置，固定大小球形刚体的生成，刚体之间以及刚体与传送带之间的碰撞反应等。另外，还实现了一个简易小车的运动场景。小车可以朝前后运动，并可朝左右拐弯。根据其与地面的摩擦力与实时施加力的大小来确定其实时运动速度和方位。该场景中还加入了一个利用 Joint 关节连接的轮胎盒体链，以及可以向渲染场景中投掷圆柱形刚体的物理效果。通过键盘方向键的控制可以变换场景的观测位置，鼠标键则可以改变摄像机的位置。具体实现效果如图5、图6所示。

4　结束语

本文对国外专业的图形渲染引擎 OGRE 和 Newton 物理引擎的基本内容和设计原理进行了分析和研究，并利用OGRE的扩展性和Newton引擎API接口的可移植性，结合OgreNewt类库实现了图形引擎与物理引擎的绑定，并在绑定系统平台

(下转第 5620 页)

这个效果不能通过初始 Catmull-Clark 算法获得。选择λ使得$1-\lambda$是一很小的正数，极限曲面仍然接近于初始网但是却消除了尖锐的边和顶点。实验结果如图 4 所示。

(a) λ=0.1 (b) λ=0.3 (c) λ=0.6 (d) λ=0.9

图 4 对应于不同的λ值二次细分算法产生的曲面

4 结束语

该算法不仅简单、直观，而且对曲面的调整范围大。在分段连接处不是严格要求连续的情况下直接采用该细分算法，可以方便地调节曲面的位置和形状，使曲面的生成更加灵活；该算法可以在很大程度上减小调整的工作量，具有较高的实用价值。

参考文献:

[1] Ma W.Subdivision surfaces for CAD-an over view [J].Computer Aided Design,2005,37(7):693-709.

[2] Catmull, Clark. Recursively generated B-spline surfaces on arbitrary topological meshes[J].Computer Aided Design,1978,10 (6):350-355.

[3] Doo D,Sabin M.Behaviour of recursive division surfaces near extraordinary points [J]. Computer Aided Design, 1978,10 (6): 356-360.

[4] Dyn N,Levin D,Gregory J A.A butterfly subdivision scheme for surface interpolation with tension control[J].ACM Transactions on Graphics, 1990,9(2):160-169.

[5] Loop C. Smooth subdivision surfaces based on triangle[D].Dept Math, Univ of Utah,1987.

[6] Kobbeltl.$\sqrt{3}$-Subdivision [C]. New York, NY, USA: ACM SIGGRAPH,2000:103-112.

[7] Levin A.Interpolating nets of curves by smooth subdivision surfaces[C].New York.NY,USA:ACM SIGGRAPH,1999:57-64.

[8] Sederberg T. Non-uniform recursive subdivision surfaces [C]. Orando,Fl: Computer Graphics Proceedings of Annual Conference Series, ACM SIGGPAPH,1998:387-394.

[9] 陈为.加权 Catmull-Clark 曲面[J].计算机辅助设计与图形学报, 2000,12(4):277-280.

[10] 高山,卢汉清,周万宁.基于细节的自适应网格简化[J].计算机辅助设计与图形学学报,2003,15(9):1122-1127.

[11] Ciarene U,Griebel M,Rumpfm M.Feature sensitive multiscale editing on surfaces [J]. The Visual Computer, 2004,20 (5): 329-343.

(上接第 5582 页)

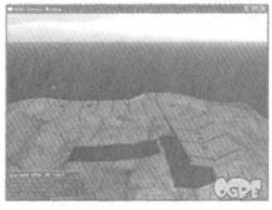

图 5 传送带效果

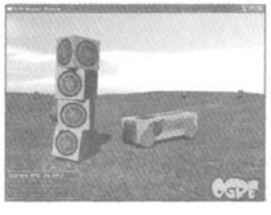

图 6 简易小车效果

上实现了两个体现物体物理特性的虚拟场景实例。但是，绑定系统的实现是建立在现有的物理引擎的基础上，调用的是它提供的一系列应用程序接口。因此，必须了解现有物理引擎接口函数的相关功能，还要考虑到它与具体图形引擎如何融合与交互。不仅繁杂难懂，使用起来也有一定的约束性。

研究现有物理引擎的基本内容和架构模式，构建一个独立的物理引擎系统并提供一层简单易用的 API 接口供上层应用程序调用，从而应用到一个统一的虚拟仿真平台上，这是笔者今后的研究方向和重点。

参考文献:

[1] David M Bourg.游戏开发物理学(physics for game developers) [M].O"Reilly Taiwan 公司,译.北京:电子工业出版社,2004.

[2] Walaber. Unofficial Newton game dynamics WIKI [EB/OL]. http://walaber.com/newton_wiki/index.php? title=Main_Page.

[3] OGRE Main Page.OGRE Tutorials[EB/OL].http://www. ogre3d. org/wiki/index.php/Ogre_Tutorials.

[4] Walaber.OgreNewt Tutorials[EB/OL].http://walaber.com/.

[5] Nourian Saeid, Shen Xiaojun, Georganas Nicolas D. XPHEVE: An extensible physics engine for virtual environments [C]. Canada: Electrical and Computer Engineering,2006:1546-1549.

[6] Chris Hecker. Physics in computer games[J].Communications of the ACM,2000,43(7):35-39.

[7] Thomas Y Yeh,Petros Faloutsos,Glenn Reinman. Enabling real-time physics simulation in future interactive entertainment[C]. USA:Association for Computing Machinery Inc,2006.

[8] Tomas Akenine-Möller,Eric Haine.实时计算机图形学[M].普建涛,译.2 版.北京:北京大学出版社,2004.