浙江科技学院学报,第 18 卷第 2 期,2006 年 6 月
Journal of Zhejiang University of Science and Technology
Vol. 18 No. 2, June 2006

基于 DirectX9 技术的三维撞球游戏开发

潘钦员,周群一
(浙江科技学院 信息与电子工程学院,杭州 310023)

摘　要: DirectX 是微软公司一组专门用于开发游戏等高性能多媒体软件的底层应用程序接口。基于 DirectX 9.0 技术实现了一个三维撞球游戏,其中使用 DirectX Graphics 实现了三维游戏场景的绘制,使用 DirectInput 实现了游戏键盘的输入控制,使用 DirectSound 产生游戏的各种音效。游戏中设计的碰撞检测算法简单、实用。游戏具有一定的娱乐性和真实性。

关键词: DirectX;碰撞检测;透视投影;游戏
中图分类号: TP311;TS952.83　　**文献标识码**: A　　**文章编号**: 1671-8798(2006)02-0107-04

A 3D Bump-Ball Game Development Based on DirectX9

PAN Qin-yuan, ZHOU Qun-yi
(School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China)

Abstract: Aiming at high-performance multimedia software, such as game, DirectX is a collection of low level application programming interfaces developed by Microsoft company. A three dimensions bump-ball game was developed using DirectX 9.0 techniques. Three dimensions scene was rendered using DirectX Graphics, keyboard was controlled by using DirectInput, and sound effect in game was implemented by using DirectSound. Collision detection algorithm in game is simple and practical. In conclusion, this game has certain entertainment and reality.

Key words: DirectX; collision detection; perspective projection; game

DirectX 是微软公司一组专门用于开发游戏和其他高性能多媒体软件的底层应用程序接口,它提供标准接口来与图形卡和声卡、输入设备等进行交互。如果没有这组标准 API,则需要为图形卡和声卡的每个组合和每种类型的键盘、鼠标和游戏杆编写不同的代码。DirectX 从具体的硬件中抽象出来,并且将一组通用指令转换成硬件的具体命令,同时为游戏和其他多媒体软件程序提供更高性能。目前 DirectX 技术广泛应用于三维图形开发[1]、计算机仿真[2,3]、虚拟现实[4]、嵌入式系统[5]等领域,在三维游戏开发中也有很大的市场前景。

1　DirectX9 技术简介

当前最新的 DirectX 版本为 DirectX9.0。Di-

收稿日期: 2006-04-03

作者简介: 潘钦员(1982—　),男,浙江苍南人,浙江科技学院计算机科学与技术专业 02 级本科生;周群一(1976—　),男,安徽合肥人,讲师,博士,主要从事模式识别、数字媒体技术和嵌入式系统研究。

rectX9.0包括8个主要组件。其中,DirectPlay提供多人网络游戏的功能和方便快捷的网络数据交互;DirectMusic则为音乐音轨、MIDI或者其他由DirectMusic Producer创作的非音乐音轨提供一套完整的解决方案;DirectShow可以对媒体数据流进行高质量的采集与回放;DirectSetup能够实现DirectX组件的自动安装;DirectX Media Objects提供数据流对象的读写支持,包括视频和音频解码器及其效果。DirectSound用于播放和捕获音频波形的高性能音频应用软件的开发;DirectInput支持各种输入设备如键盘、鼠标、操作杆等,并且完全支持力反馈技术。DirectX Graphics组合了过去DirectX版本中的DirectDraw和Direct3D两个组件,使其成为一个适用于所有图形程序的独立的应用程序接口,并简化了图形编程任务。

2 游戏功能与模块设计

笔者基于DirectX9.0的DirectX Graphics、DirectSound和DirectInput技术,开发了一个三维撞球游戏。玩家通过键盘控制一块母板的左右移动,反弹一个小球在地面以上空间中不断移动。当小球碰撞到固定砖块时,砖块消失并获得一定游戏奖励分。如果小球落地,则游戏失败;如果所有固定砖块都被碰撞销毁,则游戏过关。主要的游戏对象及游戏规则如下:

移动小球　碰到场景的边界、移动母板或固定砖块后会反弹;碰到场景的下边界则生命值失去一次。

母板　在同一水平高度上左右移动,用来反弹移动小球。

固定砖块　不会移动,在小球碰撞一次后消失,同时游戏分数增加。

场景边界　上、左、右三个边界,可以反弹移动小球。

文字信息　包括游戏已运行时间,游戏一共绘制的帧数,每秒绘制帧数,以及剩余的生命值和已获得的游戏分数。

基于此,本文采用面向对象语言C++,主要类设计如图1所示。

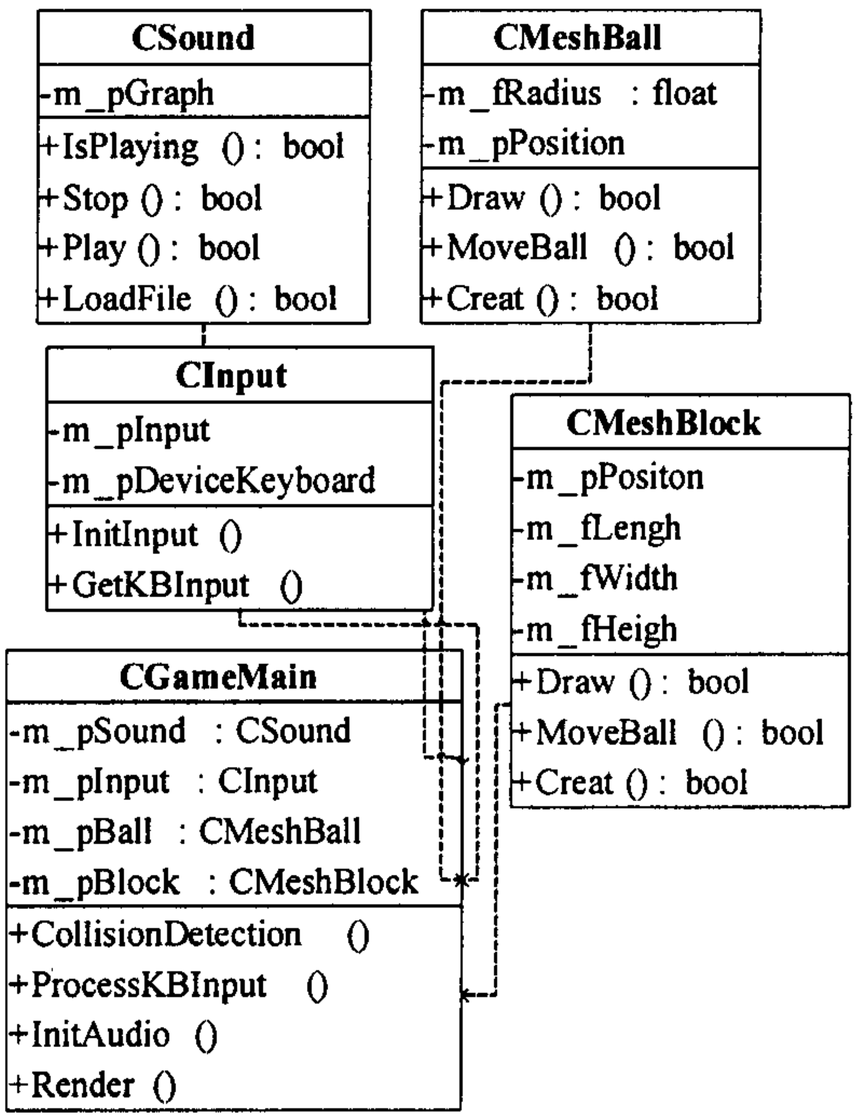

图1　游戏主要类设计图

CGameMain:此类为游戏的入口类,并控制其他类行为的调用,同时实现了碰撞检测和场景的三维绘制等。

CMeshBall:此类用于封装移动小球的属性和行为,包括小球的位置、移动方向和速度等。

CMeshBlock:此类用于封装游戏中的母板与固定砖块的属性和行为。

CSound:此类为游戏的声音控制类,使用DirectSound技术实现,用来处理各种游戏状态的声音播放,如游戏的背景音乐,碰撞时的音效,支持WAV和MP3等声音格式。

CInput:此类为游戏的输入设备类,通过它获取用户的键盘输入,使用DirectInput技术实现,控制母板的左右移动和视坐标位置等。

3 软件实现

笔者使用的DirectX版本为DirectX 9.0c 2005Apr版,使用的集成开发环境为Visual Studio.Net 2003。整个开发解决的关键技术如下。

3.1 游戏绘制

通常,电影通过24帧/s的速度连续播放图像,相邻图像之间差别不大,由于人眼视觉残留的生理特性,看到的画面感觉就是连续的动画。这种方法与DirectX3D的基本绘制原理类似。DirectX3D的绘制步骤如下:

首先,下一帧将要显示的场景将被绘制到一个不可见的内存区域,该内存区域称为“后缓冲区”。绘制完成后,快速地将该“后缓冲区”翻动到用于显示的“前缓冲区”,并不断重复这个过程,这样,便达到视觉上的连续效果。代码框架如下:

```
//开始绘制后缓冲区
m_pD3DDevice->BeginScene();
//此处进行碰撞检测和绘制后缓冲区(代码略)
//结束绘制后缓冲区
m_pD3DDevice->EndScene();
//将后缓冲区翻转到前缓冲区
m_pD3DDevice->Present();
```

3.2　透视投影

透视投影类似于人对客观世界的观察方式，它的特点是距离视点近的物体比较大，而距离远的物体相对比较小，这种投影方式的视景体可以被认为是一个棱台。它广泛应用于三维地形模拟、飞行穿越仿真等模拟人眼视觉效果的研究领域。笔者的游戏中也采用了透视投影变换，增强了游戏的三维显示效果。

透视投影的基本原理是：在游戏世界中存在一个视点，就如同人眼一样。所有物体都相对于这个视点显示，而视点具有一个发散角度，且在距离视点固定的位置存在一个投影面即视平面，如图 2 所示。其中，Z_{prp} 为视点，P 为观察点，P' 为 P 点在视平面上的投影点。

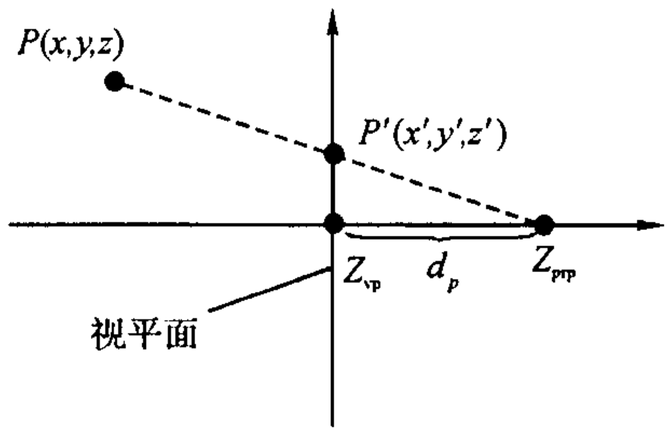

图 2　透视投影

很容易得到 P' 点的参数方程：

$$\begin{cases} x' = x - x \cdot u \\ y' = y - y \cdot u \\ z' = z - (z - z_{prp}) \cdot u \\ u = (z_{vp} - z)/(z_{prp} - z) \end{cases}$$

即：

$$\begin{cases} x' = x \cdot (z_{prp} - z_{vp})/(z_{prp} - z) \\ \quad = x \cdot d_p/(z_{prp} - z) \\ y' = y \cdot (z_{prp} - z_{vp})/(z_{prp} - z) \\ \quad = y \cdot d_p/(z_{prp} - z) \end{cases}$$

这样实现了物体从世界坐标系到视坐标系的坐标变换。透视投影变换在 DirectX 中的基本实现如下：

```
//设置视矩阵
D3DXMatrixLookAtLH ( &matrixaView, &viewEyePt,
&viewLookatPt,&viewUpVector);
//进行视变换
m _ pd3dDevice-> SetTransform ( D3DTS _ VIEW,
&matrixaView);
//设置透视投影矩阵
D3DXMatrixPerspectiveFovLH ( &matrixaProjection,
D3DX_PI/4,1.0f,1.0f,100.0f);
//进行透视投影变换
m _ pd3dDevice-> SetTransform ( D3DTS _ PROJEC-
TION,&matrixaProjection);
```

3.3　碰撞检测

在游戏中，只有移动小球会与其他物体发生碰撞。球的碰撞有三种：球与母板之间的碰撞、球与固定砖块之间的碰撞、球与场景边界的碰撞，但这三种碰撞的检测原理是一样的。

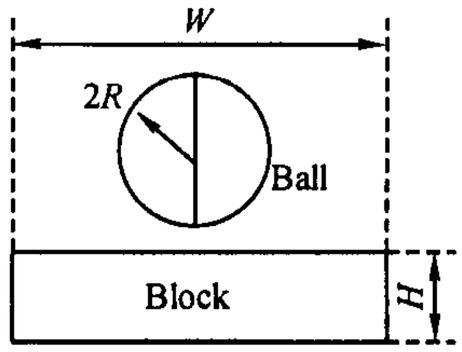

图 3　碰撞检测

图 3 中，R 为 Ball(移动小球)的半径，H 和 W 分别代表 Block(待检测碰撞物体)的高度和宽度。设移动小球球心与待检测碰撞物体重心之间的距离为 D(t)，则移动小球与待检测碰撞物体发生碰撞，必须同时满足以下两个条件：

(1) D(t). Height$\leqslant R+H/2$

(2) (Ball. $X \geqslant$ Block. $X-W/2$) AND (Ball. $X \leqslant$ Block. $X+W/2$)

显然，这样的处理存在误差，因为在相邻两帧的时间内，小球与物体可能已经发生了碰撞。因此，为了减少碰撞检测误差，本文对碰撞进行预测，每次的绘制都是预先检测过的。主要思路如下：

```
//判断游戏是否结束
while(! m_IsGameEnd){
//游戏运行后的第一次绘制时需要特殊处理
if(m_IsFirstFrame){
    //初始移动不需要进行碰撞检测
    m_pMeshBall->FirstMove();
    //绘制"后缓冲区"
    Render();
    //预移动，即计算出下一次移动
    //球将到达的位置，但不立即绘制
```

```
        m_pMeshBall->NextMove();
        //预检测是否将发生碰撞
        //并保存是否碰撞的标识
        CollisionDetection();
    }else{
        //如果碰撞标识表明已发生碰撞,
        //则改变被撞物体和小球的状态,
        //如物体是否显示、小球新的速度
        //和方向等
        if(m_bCollided)ChangeState();
        //无论是否碰撞,都要进行绘制
        Render();
        m_pMeshBall->NextMove();
        CollisionDetection();
    }
}
```

此外,降低碰撞检测的次数可以节省每一帧内的处理时间。对于那些不可能发生碰撞的帧,不必每次都进行复杂的碰撞检测。显然,当小球处在某固定砖块外围,并且远离该固定砖块的方向移动时,是不会有机会与该固定砖块发生碰撞,因此不必进行碰撞检测。主要思路如下:

```
//将本帧小球与砖块的距离与前一帧的进行比较
//当小球在 x 轴或者 y 轴方向上远离某个砖块时
if((distanceX> distancePreX)||(distanceY> distancePreY))
{
    //当小球位于固定砖块的包围盒以外时
    if(((ballPosX> blockRightX)||(ballPosX<blockLeftX))&&((ballPosY> blockTopY)||(ballPosY<blockBottomY)))
        //标志为不需要进行碰撞检测
        {bCollided=false;}
        //否则标志为需要进行碰撞检测
    else{bCollided=true;}
}
else{bCollided=true;}
```

利用 bCollided 标志,直接可以判断该固定砖块是否需要进行碰撞检测,从而节省了时间。

4 讨 论

笔者基于微软公司的 DirectX9.0 系列技术实现了一个三维撞球游戏。该游戏已经在安装了 DirectX9.0 运行时环境的 Windows2000/XP 系统下成功应用。

该游戏具有一定的娱乐性,较好地实现了游戏中常见的碰撞检测问题,同时,可以从不同视角观察游戏三维场景,并伴有游戏音效处理,使游戏本身更加真实。

文中提出的碰撞检测算法在本游戏实际应用中正确且实用,但仍存在一些需改进之处:

(1) 可以将整个未被碰撞的固定砖块区域,包括游戏边界,作为一个大的不规则物体。使用包围盒技术,对小球和该不规则物体整体上进行碰撞检测,显然,这将极大程度地节省碰撞检测的次数。

(2) 目前,碰撞检测还是基于二维平面上的处理。如果考虑 Z 轴方向上的物体运动变化,碰撞检测算法还需要做更多的改进。

参考文献:

[1] 茅忠明,王行骏,陈玮. 基于 DirectX 软件包进行三维图形的开发应用[J]. 上海理工大学学报,2002,24(1):48-52.

[2] 倪世宏,张瑞峰,史忠科,等. 基于 DirectX 技术的飞机飞行过程再现[J]. 计算机工程,2004,30(24):131-133.

[3] 李俊涛,李学仁,李永宾. 基于 DirectX 的虚拟仪表技术在飞行仿真中的应用[J]. 空军工程大学学报:自然科学版,2004,5(6):1-3.

[4] 岳璐璐,岳毅. 虚拟工程实时漫游系统技术研究[J]. 计算机与现代化,2004(11):15-18.

[5] 陆云峰,李强,母其勇. 基于 WindowsCE.NET 的嵌入式 PC 视频监控系统[J]. 计算机工程,2004,30(21):79-182.