

# 基于 DirectX 的云模拟研究

郝进亮, 陈 蕾, 娄高鸣, 侯 健

(空军航空大学 军事仿真技术研究所, 吉林 长春 130022)

**摘 要:** 虚拟场景的逼真动态显示是飞行模拟、实时动态仿真及虚拟现实等技术的基础。在飞行实时仿真系统中, 云的模拟效果对于飞行模拟和飞行训练来说是非常重要的。文中介绍了 3D 图形函数库 Direct3D, 在分析纹理映射以及粒子系统实现原理的基础上, 分别讨论了采用粒子系统建模和应用纹理动画技术来对云进行模拟, 实现了真实感较强, 渲染速度快的云层效果。文章最后对两种方法实现的云的模拟效果进行了对比, 并指出了两种方法的优缺点, 对今后的研究有一定的借鉴意义。

**关键词:** DirectX; 粒子系统; 纹理映射

**中图分类号:** TP391.9

**文献标识码:** A

**文章编号:** 1673-629X(2009)11-0195-03

## Research on Cloud Simulation Based on DirectX

HAO Jin-liang, CHEN Lei, LOU Gao-ming, HOU Jian

(Military Simulation Technology Institute, Aviation University of Air Force, Changchun 130022, China)

**Abstract:** Lifelike and dynamic displaying of virtual scene is the basis of flight simulation and real-time dynamic simulation and virtual reality. In the flight simulation system, the effects of cloud simulation are very important to flight simulation and flight training. Introduce Direct3D, and separately discuss simulation to cloud using particle system modeling and texture animation technology on the basis of analyzing the principle of texture mapping and particle system, and realize the cloud which has third dimension and fast rendering speed. In the end, contrast the effects of cloud by the two methods, point out the strongpoint and shortcoming of the two methods which has reference meaning to future study.

**Key words:** DirectX; particle system; texture mapping

## 0 引言

自然景象中十分重要的就是具有各种云彩的天空<sup>[1,2]</sup>。国内外许多学者都对云的模拟进行了很深入的研究, 其中 Blinn 提出的基于物理模型的方法以及 Kajiya 提出的基于体绘制的方法最具有代表性, 但是这些方法所实现的云模型不能兼顾系统运行的实时性和云模型的真实感, 往往都是根据需求牺牲其中的某种性能来满足另一种性能要求。

在飞行视景仿真系统中, 云的视觉效果直接影响到观察者对周围环境的感知, 文中在分析纹理映射以及粒子系统实现原理的基础上, 分别采用粒子系统建模和应用纹理动画技术, 实现了天空场景中实时生成逼真云层的效果。

## 1 Direct3D 简介

Direct3D 是 Microsoft 的 DirectX 软件开发包的组件, 是 Microsoft 自行开发的 3D 图形函数库。Direct3D 的核心结构是一个 3D 绘制引擎, 由 3 个单独的模块组成, 即变换模块、照明模块和光栅化模块, 这三个模块一起构成了 Direct3D 的渲染流水线<sup>[3]</sup>。Direct3D 提供了访问图形设备的立即模式, 该模式通过硬件抽象层 (HAL) 获得了更高的效率。HAL 还允许图形硬件在渲染、光栅化等方面保留了自己独特的性能, 从而获得更优化的显示效果<sup>[4,5]</sup>。

## 2 纹理映射及其原理

纹理映射技术是近几年来发展最快的技术之一, 广泛应用于三维真实感图形的生成与显示中<sup>[6]</sup>。一个纹理实际上就是一个位图, 纹理映射技术就是对物体表面属性进行建模, 即从二维纹理平面到三维表面的一个映射, 其关键点就是建立物体空间坐标  $(x, y, z)$  与纹理空间坐标之间的对应关系。纹理映射的本质是

收稿日期: 2009-03-09; 修回日期: 2009-06-09

作者简介: 郝进亮 (1984-) 男, 山东栖霞人, 硕士研究生, 研究方向为航空装备的仿真与运用; 陈 蕾, 硕士生导师, 副教授, 研究方向为航空装备的仿真与运用。

对三维物体进行二维参数化,即先求得三维物体表面上任一点的二维( $u, v$ )参数值,进而得到该点的纹理值,最终生成三维图形表面上的纹理图案。

在光滑曲面上添加纹理图案的核心问题是映射,因此纹理映射问题可以简化为从一个坐标系到另一个坐标系的变换<sup>[7]</sup>。其中至少涉及两个映射,一个是从纹理空间到景物空间,有时也称为曲面参数化;第二个映射是从景物空间到图像(屏幕)空间,即取景变换。通常,这两个变换被合成为一个变换。

如果纹理图案定义在纹理空间中一个正交坐标系( $u, v$ )中,曲面定义在景物空间的正交坐标系( $x, y, z$ )中,它在参数空间( $\theta, \varphi$ )的表示为 $x(\theta, \varphi), y(\theta, \varphi), z(\theta, \varphi)$ ,那么在曲面上添加纹理将涉及在两曲面之间确定或指定一个映射函数。例如,从纹理空间到参数空间的映射为:

$$\theta = f(u, v), \varphi = g(u, v)$$

从参数空间到纹理空间的逆映射为:

$$u = r(\theta, \varphi), v = s(\theta, \varphi)$$

由于 Direct3D 的纹理就是简单位图,因此任何纹理都可以被用在 Direct3D 图元上。运用纹理映射可以方便地制作真实感图形而不必花更多的时间去考虑物体的表面细节,采用纹理映射的方法可以大大简化建模的过程。

### 3 粒子系统原理及其实现

粒子系统最早是 1983 年由 William T. Reeves 提出来的,它被广泛地用来构造模拟不规则物体<sup>[8]</sup>。粒子系统的基本原理是采用大量的、具有一定属性的微小粒子图元作为基本元素,来表达不规则的模糊物体。粒子系统是一种过程模型,是利用各种计算机过程生产模型各个体素的建模技术。利用粒子系统可以将简单体素和复杂物体行为进行有机的结合,而且易于表示和实现,显示效率高。

#### 3.1 粒子的属性

粒子的运动和变化规律可以很简单也可以很复杂,这取决于所模拟的物理模型的复杂程度。对于每一个粒子,它可能具备的属性有:

**位置(坐标值)**——对于每一个粒子,最终都要把它映射到屏幕上,所以坐标成为一个粒子最为重要的属性之一;

**速度**——运动的粒子都有速度,每个粒子的速度可以各不相同。速度用来计算下一时刻粒子的位置(坐标值);

**加速度**——粒子可以做变速运动,此时加速度便会发生作用。它用来计算下一时刻粒子的速度;

**生命值**——每个粒子都有着自己的生命值,随着时间的推移,粒子的生命值不断减小,直到粒子死亡(生命值为 0)。一个生命周期结束时,另一个生命周期随即开始;

**颜色**——初始的粒子颜色是固定的,有了颜色的变化使得粒子系统更加炫目,通过颜色的变化也可以达到模拟真实变化的目的。

#### 3.2 生成过程

首先要确定云粒子的属性,然后按照粒子系统的工作原理对每个粒子进行操作,将有生命的粒子绘制在视口之内,从而实现了云模型的建立及显示。图 1 是模型中云粒子系统的工作流程图。

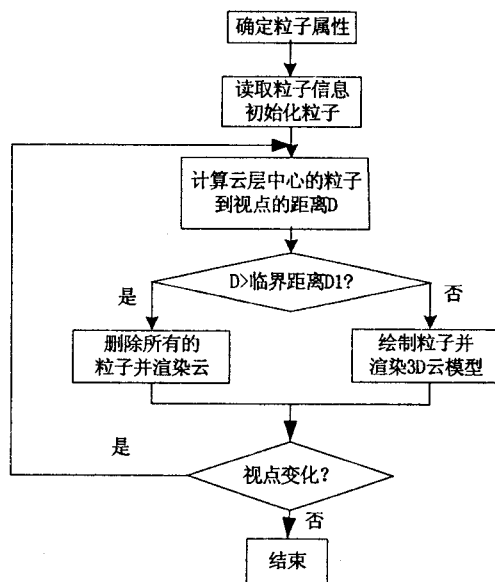


图 1 云粒子系统工作流程图

##### 3.2.1 云粒子属性

一片云里面的每个粒子都需要有大小、颜色、生命、位置等共同的属性,粒子的大小由球的半径来确定,粒子的颜色通过 RGB 颜色分量以及 Alpha 透明度来描述,通过 Alpha 值可以调整粒子的明暗程度。粒子是有生命的,该生命值通过视点距离云中心粒子的距离来描述,超出该距离,粒子死亡。粒子的位置由粒子在空间中的 XYZ 分量描述,以确定粒子在空间中的显示位置。

粒子的所有属性可以被定义为一个结构体,描述如下:

```

struct CloudParticle
{
    float Size; //大小
    Color4 Color; //颜色
    float Life; //生命
    Vector3 Position; //位置
    .....
};
  
```

### 3.2.2 粒子的初始化及属性更新

粒子的初始化通过读取模型文件里面粒子的位置、颜色、大小、生命等信息来完成。由于模拟的是静态云,所以位置和大小都不会改变,主要是对颜色值更新,颜色分为 RGB 分量和 Alpha 分量,根据 Alpha 值来计算颜色的 RGB 分量,Alpha 值根据式(1)来更新。

$$A: A_1 \times I \quad (1)$$

式中: A——粒子当前 Alpha 值;

$A_1$ ——粒子前一时刻 Alpha 值;

I——粒子的生命值。

颜色的 RGB 分量根据式(2)来更新。

$$C: 0.3 + C_1 \times P \quad (2)$$

式中: C——粒子当前的 RGB 分量;

$C_1$ ——粒子前一时刻的 RGB 分量;

P——根据光照模型得到的函数值。

### 3.2.3 粒子的死亡

根据视点到云中心粒子的距离来确定是否移除粒子,如果该距离大于预先设定的临界值,则生命变为 0,粒子从系统中删除,否则生命值为 1,绘制粒子。

### 3.3 模拟结果

该文基于粒子系统对云进行了建模,并结合光照模型对云进行渲染,最终建立了云模拟系统。通过在 PC 机上的测试,证实了该模拟系统在实时性和逼真度方面都有比较好的效果。

## 4 纹理动画原理及其实现

### 4.1 纹理动画原理

在进行动态云的模拟时,笔者是通过改变云的粒子在世界空间的位置和姿态来实现动画效果的,还有一种比较简单的方法,那就是纹理动画。纹理动画就是通过改变渲染物体时使用的纹理贴图,将多个连续的图像按照顺序以一定的时间间隔依次显示出来,这样就可以形成动画的效果。云彩的动态模拟也可以用纹理动画来实现。

### 4.2 DirectX 程序实现

用纹理动画模拟动态云的步骤及主要代码如下:

1) 定义背景天空矩形的顶点结构和顶点格式:

```
struct SKYVERTEX
{
    float x, y, z; //顶点位置
    DWORD dwColor; //顶点颜色
    float u, v; //顶点纹理坐标
};

# define D3DFVF_SKYVERTEX (D3DFVF_XYZ |
D3DFVF_DIFFUSE|D3DFVF_TEX1)
```

2) 创建背景天空矩形顶点缓冲区和云彩纹理:

```
V_RETURN(pd3dDevice->CreateVertexBuffer(4 * sizeof
(SKYVERTEX),0,D3DFVF_SKYVERTEX,D3DPOOL_MAN-
AGED,&g_pSkyVB,NULL));
SKYVERTEX vertices2[] = {.....};
```

```
.....
V_RETURN(D3DXCreateTextureFromFile(pd3dDevice, L
"ground.jpg",&g_pGroundTex));
```

```
V_RETURN(D3DXCreateTextureFromFile(pd3dDevice, L
"cloud.bmp",&g_pCloudTex));
```

3) 设置相关渲染状态:

```
pd3dDevice->SetTextureStageState(0,D3DTSS_COL-
ORAR1,D3DTA_TEXTURE);
```

```
pd3dDevice->SetTextureStageState(0,D3DTSS_COL-
ORAR2,D3DTA_DIFFUSE);
```

```
pd3dDevice->SetSamplerState(0,D3DSAMP_MINFIL-
TER,D3DTEXF_LINEAR);
```

```
pd3dDevice->SetSamplerState(0,D3DSAMP_MAGFIL-
TER,D3DTEXF_LINEAR);
```

```
pd3dDevice->SetRenderState(D3DRS_LIGHTING,
FALSE);
```

4) 移动矩阵顶点纹理坐标:

```
SKYVERTEX * pVertices;
g_pSkyVB->Lock(0,0,(void *)&pVertices,0);
for(int i=0,i<4,i++)
{pVertices[i].v-=0.2f*fElapsedTime;}
g_pSkyVB->Unlock();
```

5) 渲染背景天空矩形以及退出前释放相关资源:

```
pd3dDevice->SetTextureStageState(0,D3DTSS_COL-
OROP,D3DTOP_ADD);
```

```
pd3dDevice->SetTransform(D3DTS_WORLD,&g-
matSky);
```

```
pd3dDevice->SetTexture(0,g_pCloudTex);
```

```
pd3dDevice->SetStreamSource(0,g_pSkyVB,0,sizeof
(SKYVERTEX));
```

```
pd3dDevice->SetFVF(D3DFVF_SKYVERTEX);
```

```
pd3dDevice->DrawPrimitive(D3DPT_TRIANGLESTRIP,
0,2);
```

6) 程序退出前释放相关资源:

```
SAFE_RELEASE(g_pSkyVB);
```

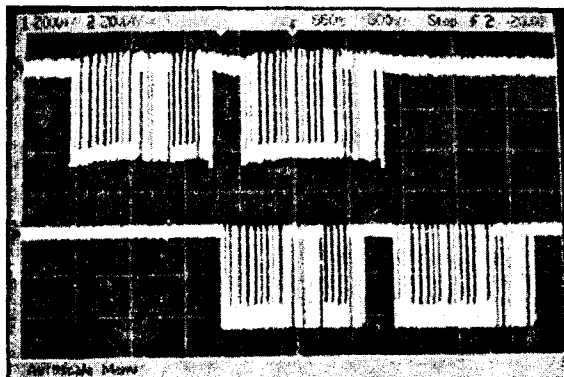
```
SAFE_RELEASE(g_pCloudTex);
```

通过纹理动画技术实现的动态云的效果见图 2。

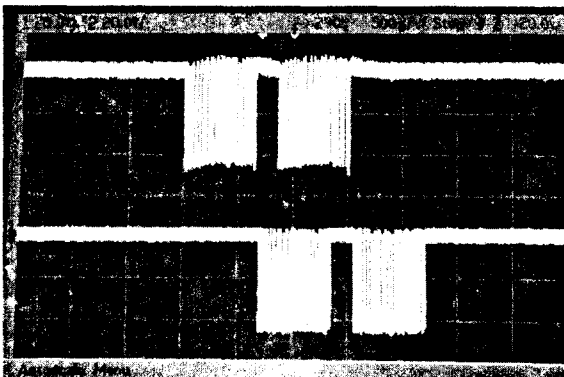
## 5 结束语

用粒子系统方法进行的云模拟虽然在逼真度与系统的实时性方面取得了一定的效果,但是生成的云还只是静态的。而根据纹理动画技术实现的云模拟虽然

(下转第 201 页)



(a) 100kbps



(b) 200kbps

图 6 相应波特率 CAN 总线中继器两个接口的波形图

#### 4 结束语

CAN 总线中继器解决了增加网络节点和延长通信距离的问题,使得 CAN 总线网络扩展更简便容易,

(上接第 197 页)

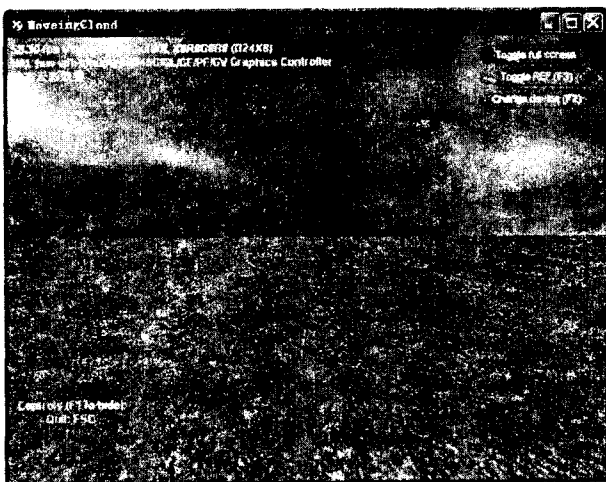


图 2 生成效果图

是动态的,但是如果纹理更新频率掌握不好,就会造成画面突变或者重复现象明显。这些问题都将在以后的工作中认真研究并加以解决。

也为两个不同波特率网络的相互通信提供了解决方案。该中继器在实际运行中可靠稳定,因为两个通道都设置了足够大的 FIFO,所以也没有出现数据溢出丢失的现象,传输延时也在可接受范围内。

表 2 在各种波特率下 CAN 中继器的时延

波特率	延时
50kbps	2600 $\mu$ s
100kbps	1320 $\mu$ s
200kbps	680 $\mu$ s
250kbps	550 $\mu$ s
500kbps	280 $\mu$ s

#### 参考文献:

- [1] 饶运涛,邹继军,郑勇芸. 现场总线 CAN 原理与应用技术[M]. 北京:北京航空航天大学出版社,2003:14-15.
- [2] 郭宽明. CAN 总线原理和应用系统设计[M]. 北京:北京航空航天大学出版社,1996:25-32.
- [3] 王黎明,夏立. CAN 现场总线系统的设计与应用[M]. 北京:电子工业出版社,2008:53-68.
- [4] 周立功. 深入浅出 ARM7——LPC213x\214x(上)[M]. 北京:北京航空航天大学出版社,2005:94-108.
- [5] PHILIPS Semiconductor. PCA82C250 CAN controller interface[S]. 2000.
- [6] 周立功. ARM 微控制器基础与实战[M]. 北京:北京航空航天大学出版社,2005:218-256.
- [7] Liu Jianxin, Guan Xuefeng, Tan Ping. The analysis and test of real-time performance for time-triggered CAN bus[J]. Proceedings of the IEEE,2008,9(3):3005-3009.
- [8] PHILIPS Semiconductor. LPC2119/2129/2194/2292/2294 USER MANUAL[S]. 2004.

#### 参考文献:

- [1] 尹勇. 自然现象的实时仿真[J]. 系统仿真学报,2002,14(9):1217-1219.
- [2] Max N. The Simulation of Natural Phenomena Panel[J]. Computer Graphics,1983,17(3):137-139.
- [3] 王德才,杨关胜,孙玉萍. 精通 DirectX 3D 图形与动画程序设计[M]. 北京:人民邮电出版社,2007.
- [4] 陈卡. Direct 3D 图形程序设计[M]. 上海:上海科学技术出版社,2003.
- [5] Sanchez J, Canton M. DirectX 3D 图形编程宝典[M]. 韩传钊译. 北京:电子工业出版社,2000.
- [6] Crow F C. Summed-Area tables for texture mapping[J]. Computer Graphics,1984,18:207-212.
- [7] 侯健,李小奇,卢永吉,等. DirectX 中纹理映射技术的实现[J]. 计算机技术与发展,2008,18(6):137-139.
- [8] 张芹,吴慧中,张健. 基于粒子系统的建模方法研究[J]. 计算机科学,2003(8):144-146.