

Managed DirectX 三维建模技术中的类实现

邢静宜¹, 王鹏飞¹, 谭小峰¹, 顾瑞春²

(1. 内蒙古科技大学 机械工程学院, 内蒙古 包头 014010;

2. 内蒙古科技大学 信息工程学院, 内蒙古 包头 014010)

摘 要: 三维建模技术是进行三维仿真的基础, 使用高级语言和图形接口进行三维建模的开发是根本的解决方法。针对目前 .NET 技术的快速发展与应用, 采用 .NET 语言和 Managed DirectX 技术进行开发, 对基于 Managed DirectX 的三维建模技术中的三维模型类进行了研究及实现, 并在 .NET 环境下开发了三维模型基类库, 其中使用了参数化建模方法。最终实现了基本的三维建模类, 在此基础上可进一步开发出通用的三维建模器。Managed DirectX 作为 DirectX 中重要的成员, 为基于 .NET 框架开发三维图形应用程序提供了极大的方便。

关键词: Managed DirectX; .NET; 三维建模; 类实现

中图分类号: TP311

文献标识码: A

文章编号: 1673-629X(2007)04-0208-03

Implementation of Class in 3D Modeling Technology Based on Managed DirectX

XING Jing-yi¹, WANG Peng-fei¹, TAN Xiao-feng¹, GU Rui-chun²

(1. Department of Mechanical Engineering, Inner Mongolia University of Science and Technology, Baotou 014010, China;

2. Department of Information Engineering, Inner Mongolia University of Science and Technology, Baotou 014010, China)

Abstract: 3D modeling is the base of 3D simulation. The solution is development using advanced language and graphical interface. Aiming at the extend of .NET application, the 3D modeling class of 3D modeling technology is researched. The technology is based on Managed DirectX with the RAD ability of .NET framework. Class's implementation of 3D modeling is completed. The base class library of 3D modeling is built in .NET framework. Through the course of the technology development researched Managed DirectX and interrelated arithmetic of 3D graphics, 3D modeling system could be developed based on the technology. As the important member, Managed DirectX provides a great convenience for the development of 3D graphical application based on .NET framework.

Key words: managed directX; .NET; 3D modeling; class Implementation

0 引言

DirectX 软件开发工具包 (SDK) 提供了一套优秀的应用程序编程接口 (APIs), 是一款强大的 3D、网络、音频专业 SDK。可以提供开发高质量、实时的应用程序所需要的各种资源。它可以让以 Windows 为作业平台的多媒体应用程序获得更高的执行效率, 还可以加强 3D 图形成像和丰富的声音效果^[1]。

Managed DirectX (以下简称为 MDX) 为 DirectX 提供了受控接口。它具有所有托管代码的新特性, 可以使得开发人员利用 .NET 下的各种特性、C# 的简洁和强大, 以及 Visual Studio .NET 的 RAD 和 UI 方面的支

持, 来快速开发丰富的高质量图形应用程序^[2]。

在机械工程学的三维图形仿真研究中, 三维建模技术是关键共性技术之一。文中将三维图形复杂的框架和技术难点隐藏起来, 使用 .NET 技术开发出一套交互式的三维模型基类库, 由此可以进一步构建三维建模系统。这只需要对基类库进行必要的扩展, 并进行部分封装与调用。

1 Managed DirectX

由于 MDX 新近发布, 相关资料欠缺, 基于此技术的开发与应用也较少。以往进行 DirectX 开发, 都需要使用 C++, 因为 DirectX 本身及其 SDK 都是基于 C++ 和 COM+ 的。由于在 MDX 中去除了 COM 组件层, 使用 MDX 可以改进应用程序的执行性能, 并减少代码量和提高系统的开发效率。由于继承了功能强大而易

收稿日期: 2006-07-14

基金项目: 国家科技攻关计划资助项目 (2001BA201A46)

作者简介: 邢静宜 (1979-), 女, 内蒙古包头人, 助教, 研究方向为机电一体化、CAD/CAM 技术。

用的.NET 构架公共类,使界面更加直观,同时把开发者从麻烦的内存管理任务中解放出来^[3]。

1.1 Device 类

Device 类是 Managed DirectX 里所有绘图操作所必需的。场景里所有其他图形对象都依赖于 Device。在 Managed DirectX3D 里,应用程序可以控制任意多个 Device。Device 共有三个构造函数,其中一个如下:

```
public Device(int adapter, DeviceType deviceType,
Control renderWindow, CreateFlags behaviorFlags, Pre-
sentParameters[] presentationParameters);
```

参数 adapter 表示系统将要使用的物理图形卡,参数 DeviceType 表示要创建的 device 类型,renderWin-
dow 表示把设备绑定到的窗口,参数 CreateFlags 用来
描述设备创建之后的行为,最后一个参数表示设备把
数据呈现到显示器的方式。

1.2 变换矩阵

变换是把几何体位置从一个坐标系转到另一个坐
标系的系列矩阵。用于 Device 的三个主要变换是
world, view 以及 projection 变换,但也有一些其他变
换。比如用来控制 texture stages 的变换,它依赖于一个 255 的世界矩阵。

1.3 Mesh 类

Mesh 类是 MDX 里可以封装并且加载顶点和索
引数据的对象,可以用来储存任何类型的图形数据,但
主要用来封装复杂的模型,其命名空间见表 1。Mesh
类有用来提高渲染物体性能的方法,所有的 Mesh 对
象都包含了一个顶点缓冲和一个索引缓存。

表 1 Managed DirectX 命名空间

命名空间	说明
Microsoft. DirectX	父命名空间,包含所有公共代码
Microsoft. DirectX. Direct3D	Direct3D 图形 API 以及 D3DX 辅 助库
Microsoft. DirectX. DirectDraw	DirectDraw 图形 API
Microsoft. DirectX. DirectPlay	DirectPlay 网络连接 API
Microsoft. DirectX. DirectSound	DirectSound 音频 API
Microsoft. DirectX. DirectInput	DirectInput 用户输入 API
Microsoft. DirectX. AudioVideo- Playback	简单音频和视频回放 API
Microsoft. DirectX. Diagnostics	简单诊断 API
Microsoft. DirectX. Security	DirectX 代码访问安全的底层结构

2 三维建模基类库的实现

2.1 类库结构设计

2.1.1 扩展类库

在 Direct SDK 类库的基础上进行基础绘制、控制
显示等基本类实现,其底层调用为 DirectX 类,是进行

高级扩展应用类库建立的基础,包括图元基类。

2.1.2 单元类库

在系统扩展类库的基础上进行系统单元类库的建
立,此类属于应用层的功能扩展。在此类中可以进行
系统基本单元的绘制,包括图元类、辅助类、控制类,各
类间关系如图 1 所示。

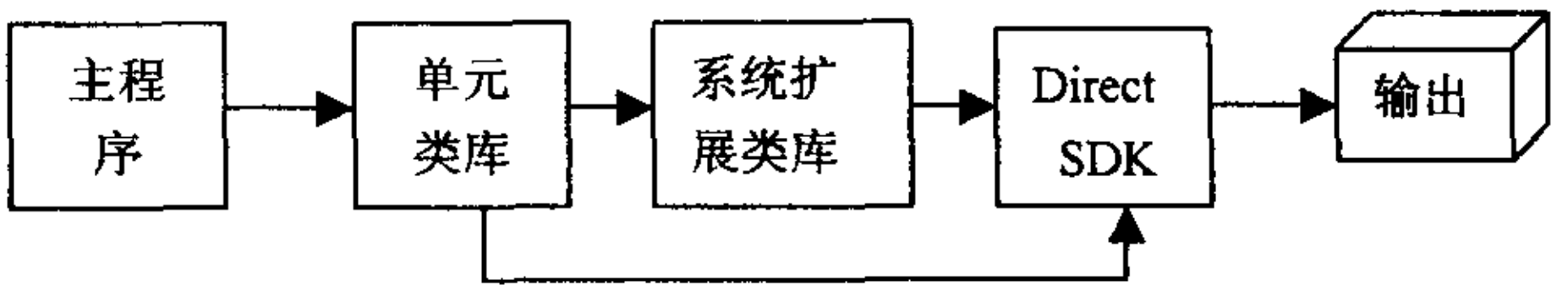


图 1 各类间的调用关系

2.2 基类库内容

基类库包括进行三维建模所需要的基本类,主要
包括图元类、单元类、辅助类和控制类四个部分。具体
如下:

- (1)图元类对 Mesh 类进行了重新封装,提供基本
图形元素绘制,包括点、线、面、体。其中基本体元素包
括长方体、圆柱体、球体、圆环体、圆锥体。
- (2)单元类是在图元类基础上实现各种单元绘制
类,包括单元链表类(UnitLink)。单元链表类是单元
类的链表数据组织类。
- (3)辅助类包括材质类和灯光类。
- (4)控制类分为对象控制和视图控制两种,操作主
要有平移、缩放、旋转。其中,对象控制类的具体定义
见表 2。

表 2 对象控制类的定义

名称(内容)	定义	描述
属性:		
positionX	Float	位置处的 X 坐标(可以是偏移量)
positionY	Float	位置处的 Y 坐标(可以是偏移量)
positionZ	Float	位置处的 Z 坐标(可以是偏移量)
positionVector	Vector3	向量(3D)
angle	Float	转角(弧度)
object	Unit	控制对象(单元类)
OriginWorld	World	原世界变换阵
方法:		
Translation		平移
Scaling		比例缩放
Rotate		旋转(有多个方法)

2.3 程序与代码实现

2.3.1 图元类与函数

除基本体元素外,也包括常用体元素。例如定义
管状体如下^[4]:

```
//绘制管状体
public Mesh Rings(
float r, //内径
float R, //外径
float length, //长度
int slices //分割份数
```



```

    )
}
Worlducs = Matrix.Translation(m-positionx, m-positiony, m-positionz); //用户坐标变换
Ring ring = new Ring(); //创建 Ring 对象
OutMesh = ring.DrawRing(m_device, r, R, length, slices); //绘制对象
OutMesh = Transform(OutMesh, Worlducs); //进行真实世界的坐标转换
return OutMesh; //返回所生成的 Mesh 对象 Rings
}

```

函数定义了对各种体元素进行基本操作的方法。例如对两对象进行布尔和操作:

```
public static Mesh Add(Mesh mesh1, Mesh mesh2)
```

对体元素进行真实的世界坐标变换的方法:

```
public Mesh Transform(Mesh oldmesh, Matrix ucs)
```

2.3.2 单元类及其链表类

单元类是绘制各种体单元类的基类,可以派生出各个体单元类。

```

public class Unit
{
    public Device m_device; //绘图参数:显卡
    //参数变量
    public int m_unitid; //单元编号
    public int m_sectionid; //截面类型编号
    public int m_materialid; //材质编号
    public Unit m_next; //链表中下一个单元
    public Unit m_previous; //链表中上一个单元
    .....
    public virtual void Show() { ..... }
}

```

单元链表类是将单元类实例进行数据组织的类。作用是将已产生的实例用数据链表的结构组织数据,并进行显示处理^[5]。

```

public class UnitLink
{
    private Unit first; //链表中第一个单元
    public bool IsEmpty(); //判断链表是否为空
    public void Add(Unit Item); //在当前链表中新增一个单元实例
    public Unit Remove(int UnitID); //从当前链表中删除一个单元实例
    public Unit Find(int UnitID); //从当前链表中查询一个单元实例
    public void Draw(); //对链表中实例进行绘制
}

```

2.3.3 辅助类与控制类

辅助类中的材质类继承自 Material 类,并可自由扩展。灯光类采用合用 DX 中的灯光类产生灯光基

类,并增加组合环境光类的实现。控制类中的视图控制类描述如下:

```

public CtrlClass() //视图控制类
{
    //私有变量\类属性
    private Device m_device; //device 设置定义
    private Matrix m_viewMatrix; //观察矩阵
    private Vector3 m_oldEye; //原观察点
    private Vector3 m_oldLookat; //原视点
    private Matrix transformMatrix; //观察变换阵
    .....
    //类方法
    private void getTransformMatrix(float X, float Y, float Z, float Yaw, float Pitch, float Roll) //获取变换阵
    public void Work(float X, float Y, float Z, float Yaw, float Pitch, float Roll, float WinScale) //对视图进行变换
    private void SetLights() //光线设置
    public void ScreenToVector(float screenPointX, float screenPointY, float width, float height, float R) //屏幕二维点转换到三维坐标系中
    .....
}

```

3 结束语

文中实现的类库是基于 .NET 与 Managed DirectX 技术开发的交互式三维建模类库,是进一步进行三维建模平台系统开发的基础。在类库的实现中涉及了较多的计算机图形学技术及微分几何学理论。本类库在更加方便的用户控制和实体的运算部分还有不足之处,有待进一步完善。

Managed DirectX 作为 DirectX 中重要的成员,为基于 .NET 框架开发三维图形应用程序提供了极大的方便^[6],由于降低了开发难度,所以给快速开发相应的图形应用系统带来了机遇。

参考文献:

- [1] Microsoft. .NET Framework[EB/OL]. 2005. www.microsoft.com/china/msdn/.
- [2] Microsoft. Directx 9.0 SDK Document[EB/OL]. 2005. www.microsoft.com/downloads/.
- [3] Miller T. Managed DirectX 9 Kick Start: Graphics and Game Programming[M]. Indiana: Sams Publishing, 2003.
- [4] Telles M. C# 技术内幕[M]. 北京:中国水利水电出版社, 2002.
- [5] 甄 镭. .NET 与设计模式[M]. 北京:电子工业出版社, 2005.
- [6] 华才健, 蔡 勇, 申玉斌. 基于 DirectX 的交互式三维图形开发工具[J]. 电脑开发与应用, 2004, 17(12): 32-33.