

文章编号:1001-9081(2010)03-0589-04

## XNA 中基于素材管道的粒子系统设计与实现

罗为君<sup>1</sup>, 林亚平<sup>1,2</sup>

(1. 湖南大学 软件学院, 长沙 410082; 2. 湖南大学 计算机与通信学院, 长沙 410082)

(percent@21cn.com)

**摘要:**粒子系统是目前游戏引擎研究领域的热点之一。介绍了粒子系统的基本原理,利用面向对象技术与扩展 XNA 的素材管道,针对自定义素材文件进行处理,使粒子系统模块化,形成了一个简单易用、容易扩展的粒子系统模块,可方便地与游戏引擎特效模块整合。给出了粒子系统的管理接口定义及层次结构关系的建模实现,通过扩展素材管道处理自定义素材文件,对粒子系统进行集成与封装,最后,使用 XNA Game Studio 实现了一些具体应用。

**关键词:**素材管道;粒子系统;XNA;游戏引擎

**中图分类号:** TP391.9      **文献标志码:** A

### Design and implementation of particle system based on content pipeline in XNA

LUO Wei-jun<sup>1</sup>, LIN Ya-ping<sup>1,2</sup>

(1. Software College, Hunan University, Changsha Hunan 410082, China;

2. College of Computer and Communication, Hunan University, Changsha Hunan 410082, China)

**Abstract:** Particle system is one of the hot topics in the game engine. In this paper, the general principle of particle system was introduced. Aiming at processing custom content files in XNA easily, by making use of object-oriented technique and extending the behavior of content pipeline, a particle system module was developed. The definition of management interface and the relation of hierarchy of particle system modeling were given. And particle system was integrated and encapsulated. At last, a concrete application was given by making use of XNA Game Studio.

**Key words:** content pipeline; particle system; XNA; game engine

## 0 引言

随着近年来显卡性能和速度的不断提高,人们已经不再满足于简单的二维游戏世界,三维游戏因其火焰、瀑布、雨水、爆炸和烟雾等特效所显示出的逼真、虚幻的场景画面,已取代了二维游戏的地位,成为市场上的主流。从技术的角度讲,这主要归功于三维游戏引擎中的特效处理模块。其中,基于粒子系统<sup>[1]</sup>的特效是目前引擎研究所关注的热点。目前,已有许多利用粒子系统模拟自然现象的研究工作,能够很好地将模拟出的雨水、雪花、火焰和烟雾等三维复杂自然景物用于游戏中。

一般而言,粒子特效系统是游戏引擎的重要组成部分,而创建特定的粒子系统是一个比较复杂的过程,不仅要考虑如何让粒子更真实地反映现实,而且需要对大量的粒子属性进行管理。

目前粒子系统存在下列不足:1)代码可复用性差,只能适用于少数简单粒子系统;2)未对粒子系统进行集成与封装,管理与维护代码不便利;3)粒子运行过程中属性管理复杂;4)粒子初始属性参数调整不方便。

本文基于粒子系统的基本原理,使用面向对象的方法<sup>[2]</sup>与 XNA Game Studio 开发工具,利用 XNA 框架的素材管道(Content Pipeline)设计了一套基本的粒子系统引擎<sup>[3]</sup>,其中定义了基本的数据结构、用于高级着色语言(High Level

Shader Language, HLSL)的顶点格式等,并将其封装成一个动态链接库(Dynamic Link Library, DLL),从而达到简单易用、管理简便和快速生成粒子特效的目的。

## 1 XNA 与素材管道简介

### 1.1 XNA 简介

XNA 是微软推出的“通用软件开发平台”,目标是降低游戏开发成本、缩短开发周期。以 DirectX 为原型,微软希望把 XNA 发展为所有游戏开发平台的通用标准,如此一来将实现游戏开发工具的无缝嵌入和平滑过渡。目前 XNA 已经发展到第 3 个版本(XNA Game Studio 3.1)<sup>[4]</sup>,整合了 Windows、Xbox 与 Windows CE 等操作系统以及 Zune 播放器。

当前游戏开发主要使用 OpenGL 和 Direct3D 等图形应用接口,它们或有高速的性能,或有强大的功能,或与运行平台无关,或与硬件设备无关,但是在实现同样效果时,与 XNA 相比,它们需要数量更为庞大的代码,需要写一些底层和低级的代码来直接与显卡、声卡和输入设备打交道。与 OpenGL 和 Direct3D 相比,XNA 不仅继承了 Direct3D 在显示、声音以及系统组件等多媒体技术方面的优势,而且 XNA 在游戏开发效率、内容与代码维护管理、平台支持等方面更为突出。

### 1.2 素材管道简介

素材管道<sup>[5]</sup>及相关的应用程序编程接口(Application Programming Interface, API)是美工和程序员用来把一个数字

收稿日期:2009-09-17;修回日期:2009-11-03。

**作者简介:**罗为君(1984-),男,湖南衡南人,硕士研究生,主要研究方向:计算机网络、数字媒体;林亚平(1955-),男,湖南邵阳人,教授,博士生导师,主要研究方向:计算机网络、机器学习。

素材创建器(Digital Content Creation,DCC)工具创建的游戏素材方便地导入到XNA工程的一组高级工具。素材管道主要的目标是为了:1)使美工可以自主选择数字素材创建器工具;2)提供一种机制来解除数字素材对所使用的游戏引擎的依赖性;3)提供一种简单的、可扩展的满足美工和程序员所需的素材创建系统。

XNA Game Studio已经提供了标准素材管道导入器和处理器(importers and processors)来支持常见的资源文件格式。当前,XNA Game Studio 素材文档对象模型(Document Object Model,DOM)提供了对诸如模型、材质效果、纹理等导入器和处理器的支持<sup>[6]</sup>。从一种游戏素材创建开始(比如一辆车或者人物模型)到素材可以被实际游戏代码访问的过程可以分为四个主要的步骤,XNA中资源文件处理过程如图1所示。

XNA中允许使用相关的素材管道API来扩展XNA Framework 素材管道,从而为特定类型的游戏资源开发自定义的导入器和处理器。创建一个新的素材管道类型,仅需要实现所需的导入器和处理器即可。扩展后的资源文件处理过程如图1所示,区别仅是素材管道使用自定义的导入器CustomImporter和处理器CustomProcessor。

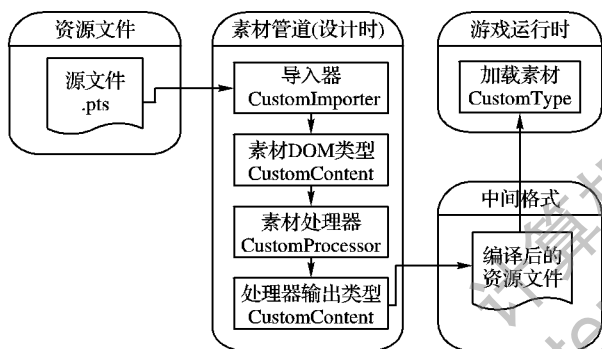


图1 XNA中资源文件处理过程

## 2 粒子系统基本原理

粒子系统是Reeves于1983年提出的,其基本思想是用许多形状简单的、具有一定生命的粒子作为基本元素聚集起来描述自然界不规则模糊景物。粒子系统中每一个粒子均具有一定的生命周期及其他属性(如大小、颜色、形状、速度和运动方向等),随着时间的推移,系统中的粒子不断地经过“产生”、“活动”和“消亡”三个阶段<sup>[7]</sup>,从而表现出景物的总体特征和形态的动态变化。

通常情况下,用粒子系统绘制一帧图像要经过下面的步骤:

- 1)生成新的粒子加入到粒子系统中,为新生粒子各属性设置初始值;
- 2)粒子根据其属性值在系统中运动;
- 3)更新粒子的属性值;
- 4)删除系统中已消亡的粒子;
- 5)绘制未消亡的粒子并显示成图像。

以上5个步骤不停地循环,有效地表示了物体的动态变化过程。

## 3 粒子系统的建模实现

### 3.1 粒子系统管理接口的定义

粒子系统是由大量单个粒子所组成的,因此,要定义一个

粒子系统,必须先定义粒子系统中粒子的属性和行为,一个典型的粒子如前所述,应该包含粒子的大小(Size)、颜色(Color)、位置(Position)、速度(Velocity)、运动方向(Direction)和生命周期(Age)等。图2中Particle为单个粒子的一个简单模型,类图给出了粒子的属性。该粒子模型尽管是最简单的,但在具体应用中可以对其扩展。

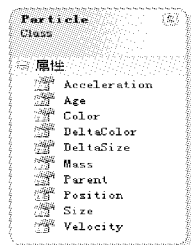


图2 单个粒子模型

在Java、C#等面向对象语言中,接口用于明确地描述系统对外提供的所有服务,能够更加清晰地把系统的实现细节与接口分离。图3为粒子系统管理接口,通过对粒子系统管理接口进行实现,生成了如图4所示的粒子系统管理类。客户程序通过这个管理类来对具体的粒子系统进行管理。

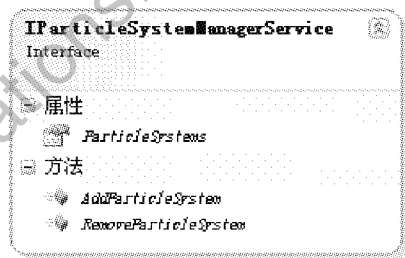


图3 粒子系统管理接口

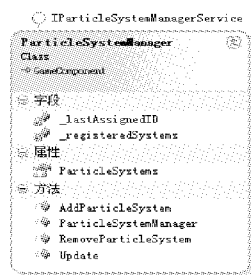


图4 粒子系统管理类

这个类给出了粒子系统生存周期中三个阶段的行为,包含了上述粒子系统全部意义上的抽象行为,是一个完备的抽象集合。Update函数用于对所有粒子系统的属性进行更新,删除已消亡的粒子系统,更新要根据粒子的活动特点和系统要表现的效果,因此其具体实现应在粒子系统类中进行;AddParticleSystem、RemoveParticleSystem函数分别用于向粒子系统管理类中添加与删除粒子系统。

### 3.2 粒子系统层次结构关系

上面定义了粒子系统管理接口,接下来定义其他类的层次结构,使粒子系统模块化。在面向对象的编程语言中,继承是其主要功能。继承是指这样一种能力:它可以使用原有类的所有功能,并在无需重新编写原有类的情况下对这些功能进行扩展。在实现具体粒子系统时,用户只需遵从继承的实现方式,使用或扩展基类的属性和方法,实现子类设定的行为。接下来主要的工作就要用子类来具体实现粒子系统。

由于粒子系统需要对多个粒子进行操作,对其产生、活动、消亡等行为进行管理,因此图5中 ParticleSystemManager 类需要与 ParticleSystem 类关联,实现创建粒子和判断粒子是否消亡的功能。而 ParticleSystem 类需要与单个粒子模型 Particle 类关联,完成更新粒子属性值的功能。

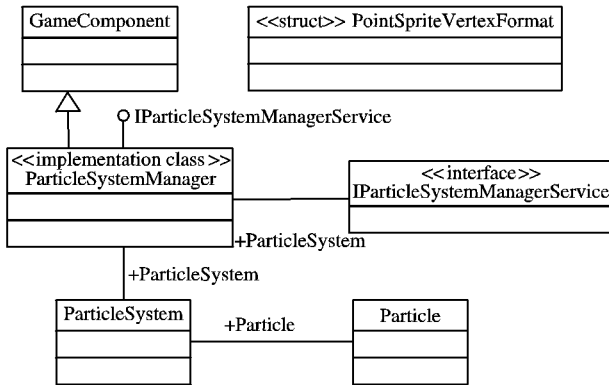


图5 粒子系统模块

粒子系统的显示大多要使用图形处理器 (Graphic Processing Unit, GPU) 的渲染功能,对于不同的粒子系统,其顶点渲染是不一样的,需要对顶点格式进行管理。在程序中,可以使用顶点结构来描述粒子的位置和颜色,并且控制每个粒子的渲染状态。PointSpriteVertexFormat 结构的具体实现形式如下:

```

/// <summary>
/// 用于绘制点精灵粒子的自定义顶点结构
/// </summary>
struct ParticleVertex
{
    public Vector3 Position;           //存储粒子的起始位置
    public Vector3 Velocity;          //存储粒子的起始速度
    public Color Random;              //随机值,用于显示粒子的颜色
    public float Time;                //粒子创建的时间
    //描述顶点结构
    public static readonly VertexElement[] VertexElements = {
        new VertexElement(0, 0, VertexElementFormat.Vector3,
            VertexElementMethod.Default, VertexElementUsage.Position, 0),
        new VertexElement(0, 12, VertexElementFormat.Vector3,
            VertexElementMethod.Default, VertexElementUsage.Normal, 0),
        new VertexElement(0, 24, VertexElementFormat.Color,
            VertexElementMethod.Default, VertexElementUsage.Color, 0),
        new VertexElement(0, 28, VertexElementFormat.Single,
            VertexElementMethod.Default,
            VertexElementUsage.TextureCoordinate, 0) };
    // 顶点结构的大小
    public const int SizeInBytes = 32;
}
  
```

PointSpriteVertexFormat 结构是用来在程序中使用点精灵 (point sprite) 绘制粒子的自定义顶点结构。用户可以根据自己的需要使用不同的顶点格式来绘制不同的粒子,从而达到多样化的目的。

粒子发射器用来控制粒子在什么位置以及如何从中心区域进行发射。新生粒子需要从发射器获得初始参数来配置自己的属性。类的具体实现形式如下:

```

/// <summary>
/// 粒子发射器
/// </summary>
public abstract class Emitter
  
```

```

{
    //粒子的位置
    public Vector3 Position { get; set; }
    //产生随机数
    private Random _random = new Random();
    public Random Random { get { return _random; } }
    //计算粒子的初始位置和方向
    public abstract void EmitPosition(out Vector3 position, out Vector3
        direction);
    //粒子发射的方向
    protected Vector3 EmitDirection( Vector3 direction, float halfAngle)
    { ... }
}
  
```

Emitter 类是虚函数,其中定义了一些基本方法供子类使用,通过这种方式,具体的粒子发射器类能够隐式地把发射器基类的所有成员当做自己的成员,从而方便扩展粒子发生器。在图6中子类 SphereEmitter、FireEmitter 和 SnowEmitter 继承了父类 Emitter,使用其中的 EmitDirection 方法,并覆写 (override) 了抽象方法 EmitPosition,从而初始化了具体粒子的初始位置并指定了运动的方向。

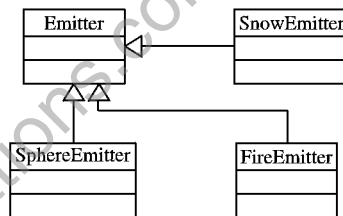


图6 粒子发射器模块

#### 4 素材管道在粒子系统中的具体应用

本文中设计的用户自定义素材管道系统组成如图7所示:此图为素材管道的代码结构与数据流向图。用户自定义素材管道使用素材管道 API 实现了导入器、素材处理器、生成资源文件、运行时的加载器等一系列功能来扩展 XNA Framework 素材管道,从而使 XNA 支持处理自定义素材文件的功能。用户自定义素材管道主要由导入器、素材处理器、资源文件生成器与资源文件加载器等模块组成。每个模块都是通过覆写素材管道父类的有关方法,来提供处理自定义素材文件的功能。最后将其编译成 dll 文件,供有关系统来调用。

这里给出在 XNA 下利用素材管道来实现多种粒子系统,自定义素材文件扩展名是 .psf,由自定义素材管道来处理,用来实现多种粒子系统;粒子系统由 ParticleSystemManager 类进行管理;粒子运动方式分别由继承 Emitter 类的子类: SphereEmitter、FireEmitter 与 SnowEmitter 类来具体实现。最后生成的几种粒子系统效果如图8~10所示。

其中,火焰效果由火焰与烟雾两种粒子系统实现,它们统一由 FireEmitter 发射器负责运动的实现,而它们的区别仅仅在于自定义素材文件的不同,即在程序加载时指定粒子系统类 fireParticle 与 smokeParticle 所使用的自定义素材文件。

以下采用素材文件方式与代码方式实现火焰粒子。

素材文件方式:

```

TextureName = Content/Textures/Particles/fire
MaxParticles = 2400
Duration = 2
DurationRandomness = 1
EmitterVelocitySensitivity = 1
  
```

```

MinHorizontalVelocity = 0
MaxHorizontalVelocity = 15
MinVerticalVelocity = -10
MaxVerticalVelocity = 10
Gravity = 0; 15; 0
EndVelocity = 1
MinColor = 10; 255; 255; 255
MaxColor = 40; 255; 255; 255
MinRotateSpeed = 0
MaxRotateSpeed = 0
MinStartSize = 5
MaxStartSize = 10
MinEndSize = 10
MaxEndSize = 40

```

普通代码编程方式:

```
class FireParticleSystem : ParticleSystem
```

```
{
protected override void InitializeSettings( ParticleSettings settings)
{
```

```

settings.TextureName = "fire";
settings.MaxParticles = 2400;
settings.Duration = TimeSpan.FromSeconds(2);
settings.DurationRandomness = 1;
settings.MinHorizontalVelocity = 0;
settings.MaxHorizontalVelocity = 15;
settings.MinVerticalVelocity = -10;
settings.MaxVerticalVelocity = 10;
settings.Gravity = new Vector3(0, 15, 0);
settings.MinColor = new Color(255, 255, 255, 10);
settings.MaxColor = new Color(255, 255, 255, 40);
settings.MinStartSize = 5;
settings.MaxStartSize = 10;
settings.MinEndSize = 10;
settings.MaxEndSize = 40;
settings.SourceBlend = Blend.SourceAlpha;
settings.DestinationBlend = Blend.One;
}

```

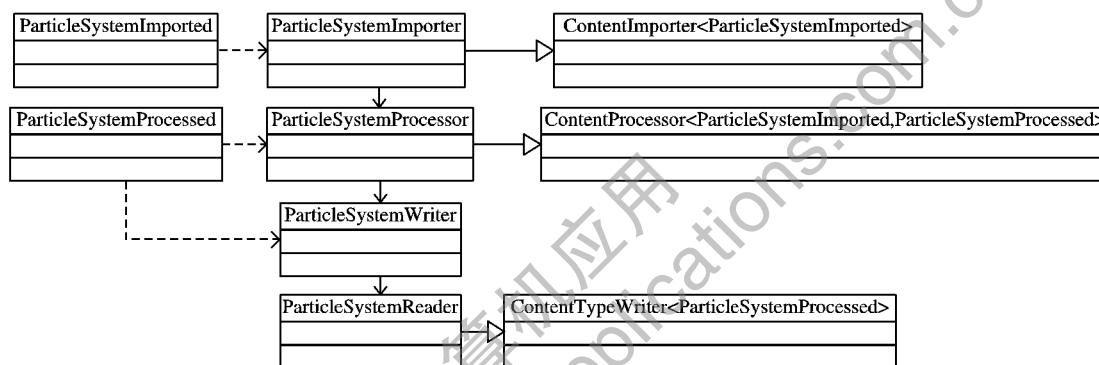


图7 素材管道的代码结构与数据流向图

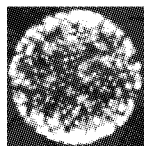


图8 粒子系统实现环境效果

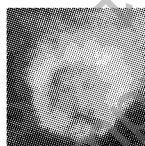


图9 粒子系统实现火焰效果



图10 粒子系统实现雪效果

从上述文件内容与代码对比可以看出,代码方式需要多次使用对象 settings 来进行参数的设置,如果要生成多个不同的粒子特效,代码重复度高,且不易更改参数;而文件方式只需要按照一定的格式进行赋值即可,不需要代码,且能方便更改参数,实现不同的粒子效果。此外,从 XNA 的工作过程来讲,自定义素材文件会被处理并编译成二进制数据文件,然后在游戏中加载,除了 XNA 引擎其他任何程序都无法读取编译过的数据文件(.xnb 文件),并且数据的加载过程也更快,因为此时所有的数据文件都已经是游戏确切所需的格式。

因此,通过素材管道这种方式能够实现快速、灵活地生成粒子特效的目的。

## 5 结语

本文根据粒子系统基本原理并通过面向对象和模块化程序设计,运用微软公司的 XNA Game Studio 以及扩展素材管

道设计出了适用于多种粒子系统的模块,为快速生成粒子特效提供了便利,并在实际使用中并取得了极好的效果。通过对 ParticleSystemManager、ParticleSystem 和 Emitter 这三者以及与素材管道的协作及实现进行具体的分析设计,达到了快速、灵活地生成粒子特效的目的。

## 参考文献:

- [1] REEVES W T. Particle system - A technique for modeling a class of fuzzy objects [J]. ACM Transactions on Graphics, 1983, 2(2): 91 - 108.
- [2] BELYAEV S Y, PLOTNIKOV M. Object-oriented high-performance particle systems [C]// Proceedings of the 6th International Workshop on Nondestructive Testing and Computer Simulations in Science and Engineering. Bellingham, WA: SPIE, 2003: 272 - 275.
- [3] REED X. Learning XNA 3.0 [M]. Sebastopol: O'Reilly, 2008: 308 - 330.
- [4] Microsoft Corporation. XNA creators club online [EB/OL]. [2009 - 06 - 11]. <http://creators.xna.com>.
- [5] Microsoft Corporation. Content pipeline [EB/OL]. [2009 - 06 - 11]. <http://msdn.microsoft.com/library/bb203887.aspx>.
- [6] Microsoft Corporation. Standard importers and processors [EB/OL]. [2009 - 06 - 11]. <http://msdn.microsoft.com/library/bb447762.aspx>.
- [7] ZHOU LI-KUN, CHEN DING-FANG. Implementation of natural scene modeling method based on physical properties and a particle system [J]. Journal of Shanghai University, 2004, 8(A02): 53 - 57.