粒子系统

郑华

2016 年 7 月 3 日

1 概念

1.1 产生

如何逼真地模拟自然景物一直是图形学中的一个热门研究课题和难点问题。火焰、云烟、滴、雪花等动态自然景物的模拟，在航空航天、影视广告、虚拟场景中有着广泛的应用。然而多数景物的外形是随机变化的，很难用常规的建模方法及模拟技术来描述。因此自然景物的模拟一直以来都是虚拟现实领域研究的热点和重点。随着近年来研究的不断深入，各种自然景物模拟算法不断涌现，模拟结果也越来越具有真实感。其中，粒子系统方法是迄今为止被认为模拟不规则模糊自然景物最为成功的一种生成算法。

随着应用的不断扩展, 在虚拟现实系统的设计与实现中, 有一些景观很难用简单的几何图元来表示, 这类景观主要是一些离散的或者动态的自然景观和人文景观, 例如烟、星星、喷泉和烟花等等

1985 年由 W. T. Reeves 等首次系统地提出了粒子系统方法。此方法被认为是迄今为止模拟不规则模糊物体，最为成功的一种图形生成算法。在计算机虚拟仿真领域，应用粒子系统模拟不规则模糊物体的方法，即一种模拟不规则自然景物生成和动画系统，已经得到了广泛应用。

1.2 什么是粒子系统

粒子系统是一种典型的物理建模系统，它是用简单的体素完成复杂运动的建模。

粒子系统由大量称为粒子的简单体素构成，每个粒子具有位置、速度、颜色和生命期等属性，这些属性可根据动力学计算和随机过程得到。而一个粒子需要被赋予哪些属性，主要取决于被模拟对象。

粒子系统是最实用的过程动画技术之一。所谓过程动画是指物体的运动或变形可由一个过程来描述。最简单的过程动画是用一个数学模型去控制物体的几何形状和运动，如旗帜, 水波随风的运动。较复杂的过程动画则是包括物体的变形、弹性理论、动力学、碰撞检测在内的物体的复杂运动。

1.3 粒子系统的发展

为了实现对这类不规则物体的真实感显示，国内外学者一直在努力探索, 先后提出了多种方法。例如，L 系统、分形结构图等。各种方法相比而言，粒子系统产生的图像质量高，运算代价不高，且适合于中低档次硬件平台，所以，已被大量运用在大量的 3D 软件中。

1.4 粒子系统研究方向

随机粒子系统 主要通过可控制的随机过程，控制粒子属性的变化。1983 年 Reeves 首次系统地提出了应用粒子系统，模拟虚拟场景中不规则物体的方法，模拟出烟花绽放的过程，并在电影 StarTrek 1 中绘制了星系爆炸的场面。从那之后，人们对粒子系统使用范围进行了进一步的拓展，使得粒子系统能够拟火焰、烟花、烟雾、飞机飞行的特效、喷泉，甚至是水中航行的船只的航行轨迹。

流体粒子系统 粒子的运动轨迹受流体力学的影响。比如，模拟地下煤矿矿井内气流，可以描述矿井发生火灾时火焰的扩散情况，类似的还有陶瓷辊道窑内进行陶瓷烧制时火焰轨迹的建模。粒子系统也能够对液体进行建模，Liu Xue-mei 用粒子系统模拟外科手术出血情况。粒子系统甚至能够在高密度、高粘度、高压力、高温度等极端的条件下，模拟出了岩浆流动的情况。

方向粒子系统 考虑粒子间的相互影响，粒子除了具有速度和位置等动态属性外，还必须有方向属性。这样的粒子系统，主要模拟织物、可变形物体和刚体等。1987 年，Reynolds 把粒子系统看成是一组相互影响的粒子组成的，每一个粒子的具体位置，受与其相关粒子的影响。在这个理论基础上 OlafEtzmuss 通过对连续介质的小改动，得到了耦合粒子系统，这个粒子系统中的粒子是相互关联的，并用以此来模拟变形的物体。Bernhard Eberhardt，基于耦合粒子系统系统理论，对纺织物进行了建模。同时，Szeliski 和 Richard 提出基于粒子系统的模拟弹性表面的方法，也可以对纺织物进行建模。

结构化粒子系统 主要用来模拟具有一定结构的物体或现象。1985 年，Reeves 发展了粒子系统，他们用“Volume filling”基本单元，去生成随时间改变形状但又基本保持不变的实体，如随风飘动的花草树叶。袁琪在 2007 年利用粒子系统虚拟作物器官。Rhys Goldstein 在生物学医药领域，用粒子系统来模拟神经元末梢，并且取得了很好的效果。法国的 Eric Galin 用粒子系统模拟出吊灯、雕塑品等具有隐性曲面造型的物体。

2 粒子系统的建模与仿真

粒子系统是一个动态的模型，粒子在系统中要经过“产生”、“运动”和“消亡”这 3 个阶段。随着时间的推移，系统中旧的粒子不断消失，新的粒子不断加入。系统中“存活”的粒子，其位置及生命值亦随时间变化而变化，其正常运行的关键，是确定粒子的初始属性、粒子的变化规律和绘制等因素

在粒子生命期的每一刻，都要完成以下 5 步工作

1. 粒子源产生新粒子，并赋予粒子属性后加入系统中
2. 根据粒子的动态属性，对粒子进行移动和交换，同时更新粒子属性
3. 判断粒子的生命值
4. 删除那些已经超过其生命周期的粒子
5. 绘制并显示由有生命的粒子组成的图形

2.1 粒子的生成

粒子系统的生成步骤一般分为下面几步：

1. 根据产生效果的物理特性，建立数学模型，即构建粒子的运动方程;
2. 确定粒子的个体属性
3. 产生粒子系统
4. 粒子活动
5. 粒子的死亡

1. 构建粒子系统的运动方程 总结并研究粒子可能的运动形式，研究其运动方程的求解方法，在不影响整体效果前提下尽量使得算法简化，计算量减少以满足实时性的要求。

在整个数学模型的构建过程中，为了使得产生的图像更具有真实感，应该考虑风力、重力等因素。

2. 确定粒子的个体属性及产生 为表达粒子系统的随机性，Reeves 采用了一些简化的随机过程来控制粒子在系统中的形状、特征及运动。

对每一粒子参数均确定其变化范围，然后在该范围内随机地确定它的值，而其变化范围则由给定的平均期望值和最大方差来确定，粒子系统中的粒子的产生，采用随机过程函数来控制。每一帧产生的粒子数目，直接影响到画面的效果，常使用下面两种方法来进行定义

第 f_n 帧产生新粒子数目 NParts(f_n) 定义为 ：

$$NParts_f = MeanParts_f + Rand() \times VarParts_f \tag{1}$$

- $Rand$ 是在区间为 [-1.0，1.0] 上均匀分布的随机函数
- $MeanParts_f$ 是新产生粒子的平均值
- $VarParts_f$ 是新产生粒子的方差

控制粒子的层次细节和绘制效率 为了有效控制粒子的层次细节和绘制效率，还可以根据单位屏幕面积所具有的平均粒子数和方差，来确定进入粒子系统的粒子数，则式 1可以修改为

$$NParts_f = (MeanParts_f + Rand() \times VarParts_f) * ScreenArea \tag{2}$$

- $Rand$ 是在区间为 [-1.0，1.0] 上均匀分布的随机函数
- $MeanParts_f$ 是新产生粒子的平均值
- $VarParts_f$ 是新产生粒子的方差
- $ScreenArea$ 是粒子系统的屏幕面积

使粒子系统在强度上增加或者减少 在式 2中，能够有效地避免用大量粒子来模拟在屏幕上投影面积很小的景物，因而大大提高了算法的绘制效率。为了能够使粒子系统在强度上增加或者减少，设计者一般采用线性函数公式，使得每一时刻每帧中平均粒子的数量不同

$$MeanParts_f = InitialMeanParts_{sa} + DeltaMeanParts_{sa} * \ f - f_0 \tag{3}$$

- f 是现在的帧数
- f_0 是粒子系统中最初的帧数
- $InitialMeanParts$ 是粒子系统最初帧中的平均粒子数
- $DeltaMeanParts_{sa}$ 是粒子数的变化率

对于设计者 因而,为了控制粒子系统中粒子的产生,设计者只要指 f_0 或者 $InitialMeanParts$,$DeltaMeanParts$，$Varparts$ 或者 $InitialMeanParts_{sa}$，$DeltaMeanParts_{sa}$，$Varparts_{sa}$ 的参数即可

粒子的运动 粒子一旦产生，并具有一定的初始属性之后，它们便开始运动。粒子需要在初始运动属性的基础上，推导出其他时刻的运动属性。

粒子产生后，就要根据所确定的运动模型进行运动。一帧接一帧，直至死亡。这种运动有自己的规律，也就是说所有粒子的属性有共同的地方，这是产生整个现象的基础，但每个粒子又有自己的不同，这种不同就要有随机函数来体现.

粒子的运动轨迹为 ：

- 位置：$P(f_i) = P(f_{i-1}) + V(f_{i-1}) * (f_i - f_{i-1})$
- 速度：$V(f_i) = MV + RAND() * VV + A * (f_i - f_{i-1})$
- 颜色：$C(f_i) = MV + RAND() * VC + \Delta C * (f_i - f_{i-1})$
- 透明度：$T(f_i) = MT + RAND() * VT + \Delta T * (f_i - f_{i-1})$
- 生存期：$L(f_i) = L(f_{i-1}) - \Delta L$

f_i 是粒子系统中的帧号 (i=0,1,2,3...),i = 0 时为初始帧，A、ΔC ΔT ΔL 分别是粒子的加速度，颜色变化率，透明度变化率，生命递减值，并且都可以定义为常数。

2.2 粒子的属性

任何新产生的粒子，都必须赋予它们一定的初始属性。粒子的初始属性，可以进行如下描述:

1. 初始位置
2. 初始速度（速率和方向）
3. 初始大小
4. 初始颜色
5. 初始透明度
6. 形状
7. 生命期

不同的特效对粒子的个体属性要求的侧重点不同。例如在模拟星光灿烂的宇宙时，为了产生星光闪烁的效果，就要考虑粒子的位置，大小，透明度和颜色属性，而对加速度，速度不作为考虑重点。

2.3 粒子的绘制和渲染

粒子绘制的技术主要有

1. 点粒子的控制
2. 面粒子的控制
3. 线性粒子的控制
4. 随机形状粒子的控制

实现粒子渲染的手段，主要采用光照、阴影、浓淡以及消隐处理技术。有些情况下，粒子被认为是光源，此时可以忽略消隐，简化浓淡处理，只以粒子的灰度或者颜色，来加亮相应的象素。

2.4 粒子的死亡

当粒子一旦产生之后，就被赋予了生存期，一般都是用帧来计算粒子的生存期，其随着粒子的运动而递减。当递减到零时，粒子“死亡”。此时，当从系统中将该“死亡”粒子删除

在实际的应用中，也可以采用其他的方式来度量粒子的死亡。例如, 当粒子的颜色和透明度低于系统设定值，或者粒子的运动超出了规定的区域，这些情形都可以根据实际的要求，认为粒子已经死亡

2.5 粒子系统的程序设计

粒子系统的程序设计一般分为 4 步，如下：

- 定义一个粒子结构
- 粒子初始化
- 计算出粒子的运动轨迹
- 模糊效果生成

1. 定义一个粒子结构 ：

```
typedef struct Particle
{
   D3DVECTOR   _postion;  //当前坐标 三维的x,y,z 二维的x,y
   float       _angle;  //方向
   float       _speed;  //速度
   float       _angleAdjust;  //角加速度
   float       _life;      //生命值
   float       _deltaLife;  //生命衰减变化值
}Particle;
```

2. 粒子的初始化 当一个粒子诞生时，我们需要对它进行一些必要的初始化，对于不同的效果，初始化的内容也是不一样的，下面给出一种最简单的运动粒子的初始化：

```
void InitParticle(Particle& p, D3DVECTOR postion)
{
  p._position = position;
  p._speed    = 1+(Rand()*3);
  p._angle    = Rand()*2*pi;
  p._angleAdjuset = -1/40.0f+Rand()/20;
  p._life = 1.0f;
  p._deltaLife    = 0.01f+Rand()/20;
}
```

3. 计算粒子运动轨迹

```
void  MoveParticle(Particle& p)
{
  if(p._life <= 0) return;

  p._position += cos(p._angle)*p._speed;
  p._angel  += p._angleAdjust;
  p._life -= p._deltaLife;
}
```

4. 产生模糊效果 这个根据具体情况具体示例，一般将临近像素进行取平均值操作。

3 粒子系统的优化技术

粒子系统涉及的计算量相当庞大和复杂，所以粒子系统优化研究也越来越受重视，因为这样可以节省计算资源，提高计算效率。一般来说有以下几种优化的技术

3.1 绘制效率优化

粒子系统运行的最终结果，是在屏幕上绘制每个粒子形态这个过程，需要一定的时间，影响了系统的实时性。绘制效率优化，就是采用特定方法，减少粒子绘制时间，进而提高速度。优化包括系统级和代码级两种：系统级主要采用高速缓存保存编译过的绘图代码，使用时不需再次编译；代码级主要基于人眼分辨率限制，少绘制或不绘制人眼不敏感或看不到的区域。绘制效率优绘制效率的优化包括为:

1. 显示列表
2. 公告板技术
3. 几何形体优化
4. 多级粒子系统

3.2 计算复杂性优化

粒子系统运行过程中需要大量计算，如碰撞检测、速度调整、内存分配、算术运算、循环判断等。这些计算消耗系统资源，间接表示时间复杂度大小，对系统实时性影响比较大，处理不好将严重影响系统速度。在计算复杂度的优化方面主要有:

1. 碰撞检测与处理优化
2. 存储方式优化
3. 运动过程优化
4. 线性表
5. 局部力场

3.3 粒子数量的优化

粒子数量优化，就是在不影响系统真实感的前提下，尽可能地减少粒子数量，减少空间复杂度。粒子数量的优化我们主要采用的技术包括

1. 结构化粒子
2. LOD 技术
3. 伪粒子系统