第七章 过程动画

过程动画

❖用数学公式或者算法驱动物体运动或者随时间作某些改变的动画技术（Maya和Softimage中称为表达式动画）

- 过程纹理造型
  - ✓不规则的自然景物
  - ✓动态变化的纹理细节
- 过程纹理动画
  - ✓参数随时间的变化
  - ✓艺术和物理表现形式的完美结合

过程动画技术

- 3D纹理映射与过程纹理
- Fourier合成技术
- 基于语法的造型和动画(L系统)
- 粒子系统
  - 模拟不规则模糊物体最为成功的一种图形生成算法

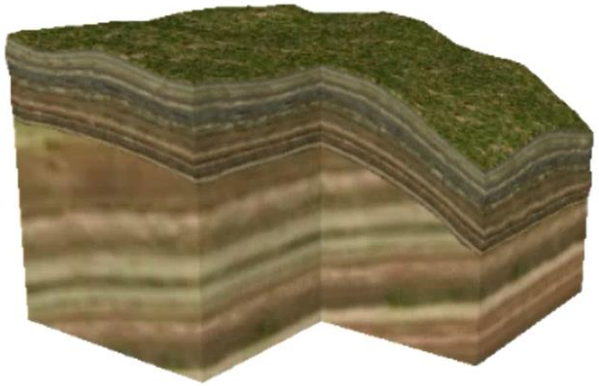
第七章 过程动画

三维纹理映射技术

二维纹理映射

- 由纹理表面至景物表面的映射是一种非线性映射，在曲面上曲率变化比较大的区域可能发生纹理的非均匀变形，导致不真实的视觉效果
- 对由多个曲面拼接而成的景物表面进行二维纹理映射时，很难保证相邻曲面片之间纹理的连续性

三维纹理映射技术



- 三维纹理映射也称为体纹理映射
- 将三维纹理函数直接定义在三维纹理空间中

$$T=T(x, y, z)$$

- 纹理空间同景物空间在数学上是等同的，可通过空间嵌入实现纹理映射
  - 将景物变换到三维纹理空间
  - 雕刻木头时出现的木质纹理

三维纹理的构造

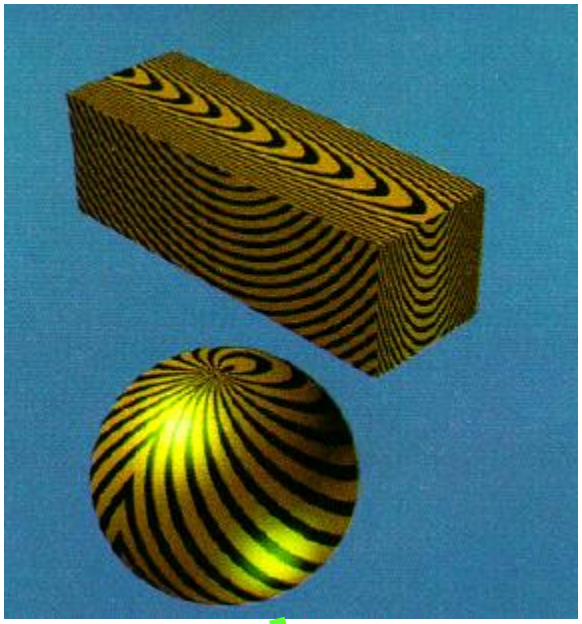
- 基于离散采样的数字化纹理
  - 二维纹理映射中广泛采用的方式
  - 对于高分辨率的三维纹理，存储量巨大
- 利用数学模型计算生成的纹理
  - 利用简单的过程迭代函数生成复杂的纹理，称为过程纹理
  - 木材、大理石、云彩、火焰、石块…

过程纹理造型技术

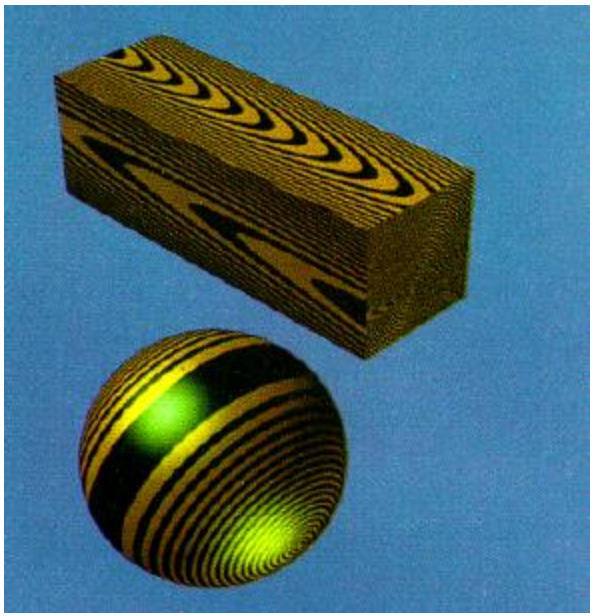
- 过程纹理是解析表达的数学模型，能够用一些简单的参数来逼真地描述复杂的自然纹理细节。它们一般为经验模型。

  - 木纹函数
  - 分形随机函数
  - 三维噪声函数
  - 湍流(turbulence)函数
  - 气态过程纹理

木纹函数—木质品的纹理效果



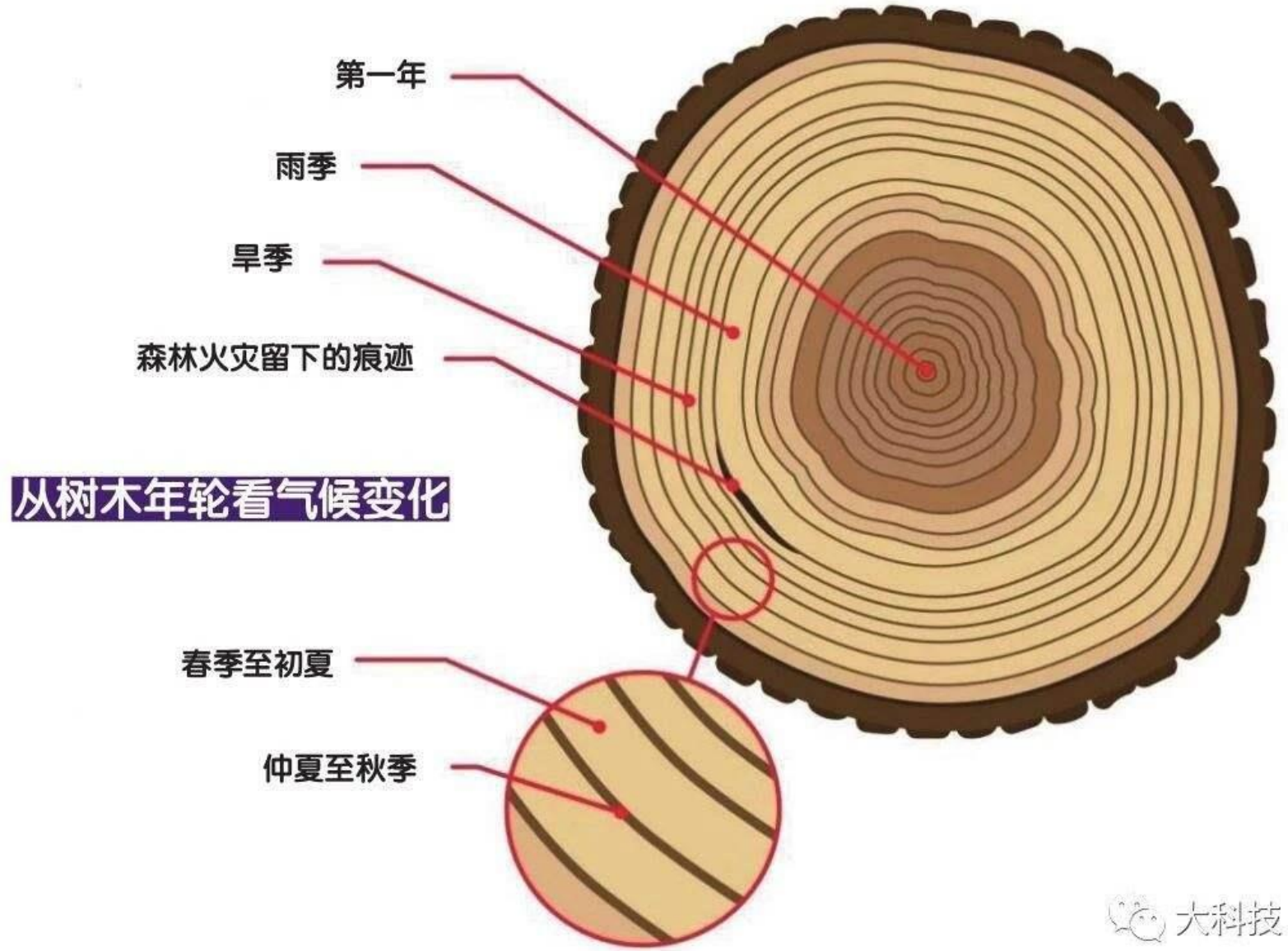
2D纹理映射法



木纹函数法

木纹函数—木质品的纹理效果

木纹函数—木质品的纹理效果



木纹函数—木质品的纹理效果

木纹函数—木质品的纹理效果

- 用一组共轴圆柱面定义体纹理函数，将位于相邻圆柱面之间点的纹理函数值交替地取为“明”和“暗”。
- 增加非规则性
  - 扰动(perturbing)：用正弦函数扰动半径
  - 扭曲（twisting）:圆柱轴方向增加小扭曲量
  - 倾斜(tilting):将圆柱的轴沿木块的长轴方向倾斜

木纹函数—木质品的纹理效果

$$r_1 = \sqrt{u^2 + v^2}$$

$$r_2 = r_1 + 2\sin(a\theta + v/b) \quad \theta = \arctan(u/w)$$

$$(x, y, z) = tilt(u, v, w)$$

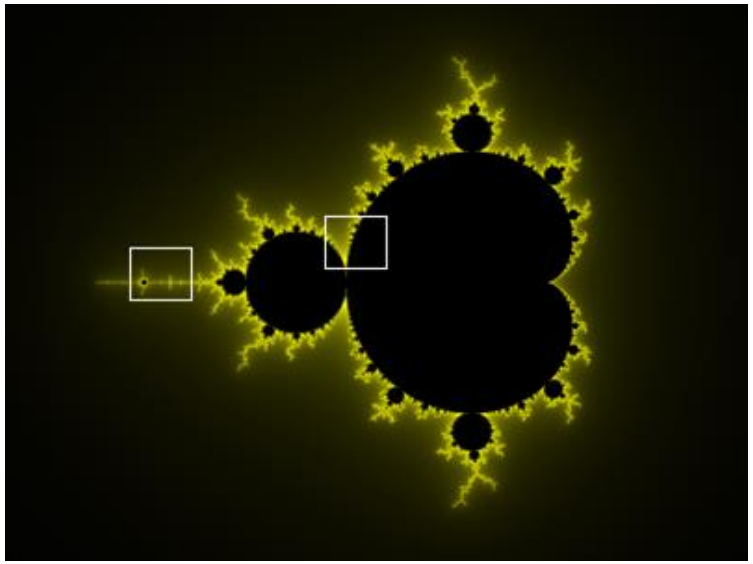
倾斜函数：纹理空间到景物空间的映射

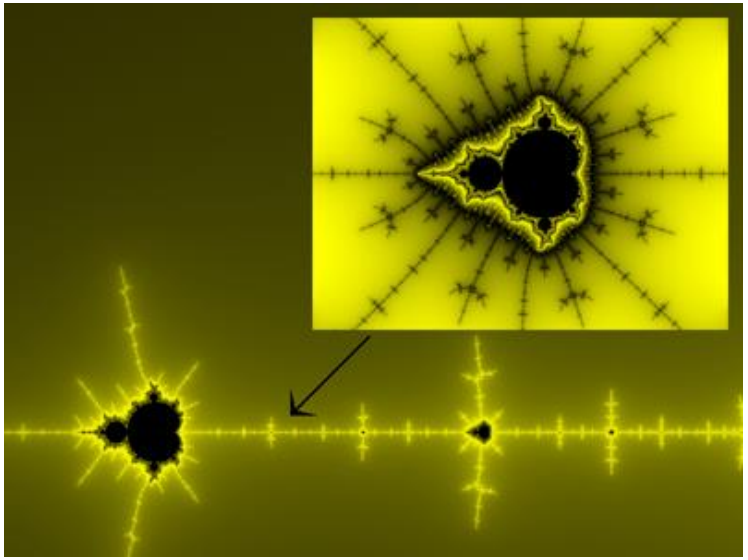
分形 —— http://www.fractal.net.cn

“分形”一词译于英文Fractal，系分形理论的创始人曼德尔布罗特（B.B.Mandelbrot）于1975年由拉丁语Frangere，一词创造而成，词本身具有“破碎”和“不规则”的含义。

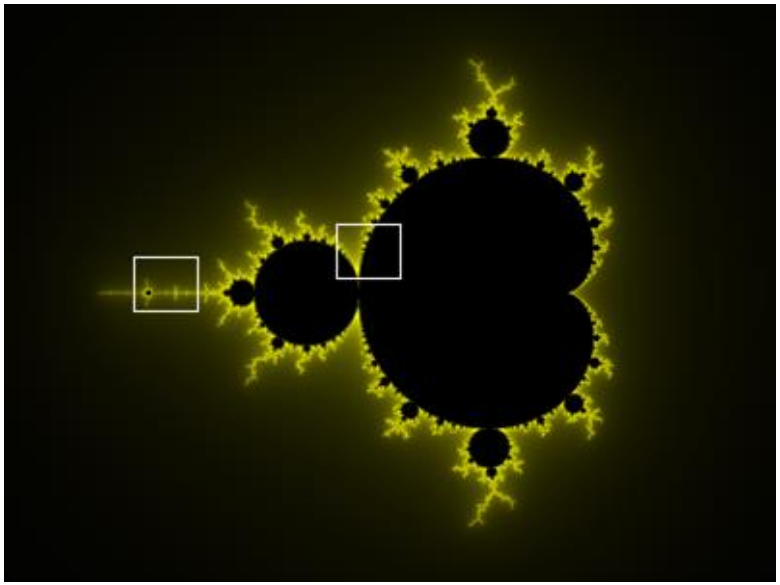
Mandelbrot研究中最精彩的部分是1980年他发现的并以他的名字命名的集合，他发现整个宇宙以一种出人意料的方式构成自相似的结构。

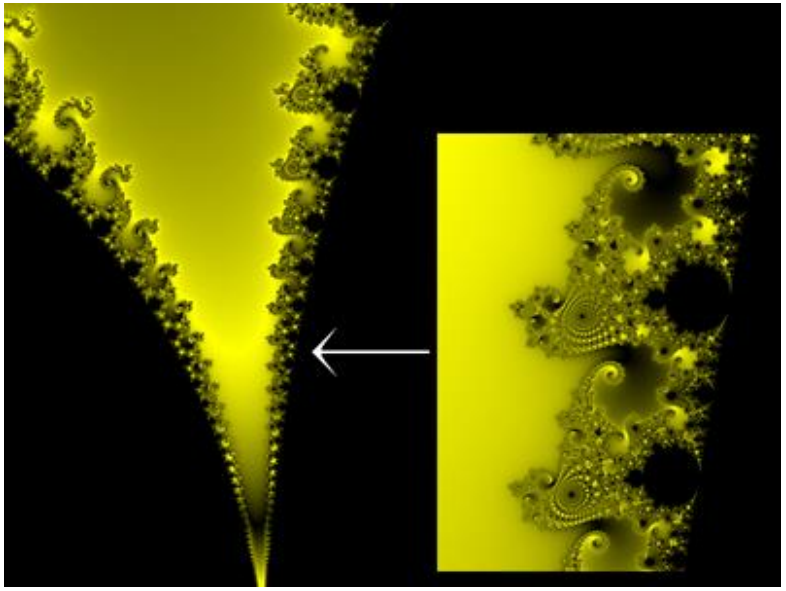
Mandelbrot集合





Mandelbrot集合





分形的研究对象

- 自然界和非线性系统中出现的不光滑和不规则的几何形体
  - 起伏蜿蜒的山脉、坑坑洼洼的地面、曲曲折折的海岸线、层层分叉的树枝、支流纵横的水系、翻腾变幻的浮云、地壳上的褶皱、密布人体周身的血管、满天闪烁的繁星、撕裂夜空的闪电、魔鬼般跳跃的火焰、船尾湍急的涡流、拍岸的惊涛与浪花、金属和非金属材料的断面、生物的大分子结构、分子光谱分布以及电磁波噪声分布等等。。。

分形的一些应用

由于分形能够用递推函数加以描述（斐波那契序列就是一个递推的例子，它的每个项都等于前两项的和），所以用计算机生成分形是理想的。像电影《星际旅行II：可汗的愤怒》中新行星的诞生以及《吉地的返回》中行星在空间飘浮等壮观的场面，就是由彼克沙公司在一台计算机上完成的（1986年）。分形还能用于描述和预示不同生态系统的演化（如乔治亚洲奥克芬诺沼泽地和生态变化）

分形艺术---http://www.fractal.net.cn

- 分形艺术作品的特点
  - 图案古怪、神奇、无穷无尽
  - 结果的不可预测性

牛顿分形---http://www.fractal.net.cn

牛顿分形图形中的颜色显示每个答案的种类及性质，即迭代到目的地花费的时间

分形艺术---http://www.fractal.net.cn

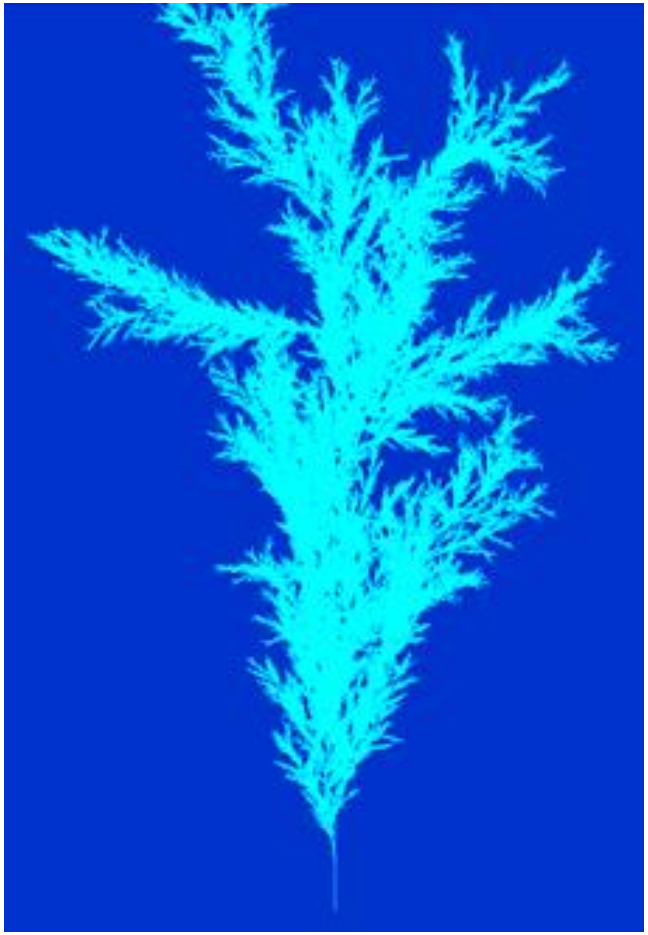
分形艺术---http://www.fractal.net.cn

分形艺术---http://www.fractal.net.cn

分形艺术---http://www.fractal.net.cn

分形艺术---http://www.fractal.net.cn

分形艺术---自然界



分维随机函数

分维布朗运动(fbm:fractional Brownian motion)是由mandelbrot和Van Ness 于1968年提出的一类一维高斯随机过程，被广泛地应用于自然界中的许多时间序列模型，如地貌表面的纹理。

利用fbm模型可以模拟具有不同自相似性的不规则自然纹理，特别是对地域和各种星球表面纹理的模拟比较成功。

分维布朗运动(fbm)

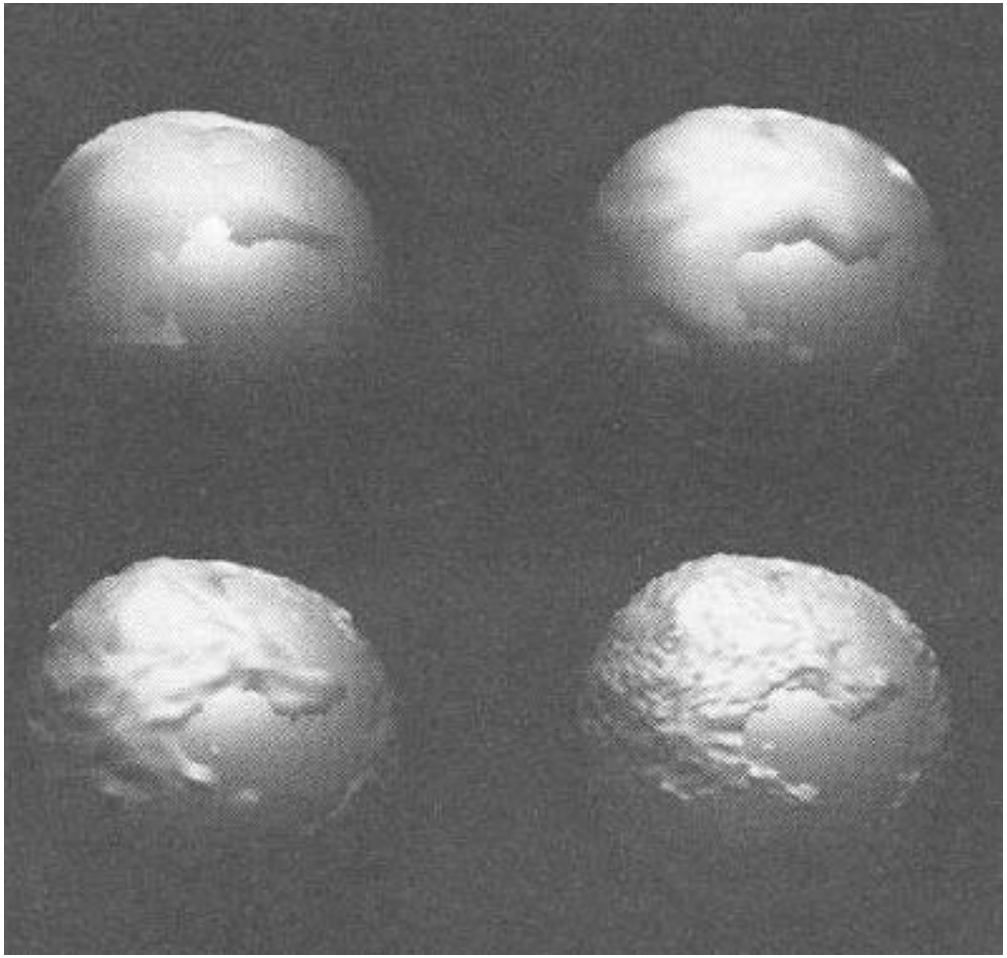
$$B_H(0, w) = b_0$$

$$B_H(u, w) - B_H(0, w) = [1/\Gamma(H + 0.5)]$$

$$\left\{ \int_{-\infty}^{u} [(u - s)^{H-0.5} - (-s)^{H-0.5}]\, dB(s, w) \right.$$

$$\left. + \int_{0}^{u} (u - s)^{H-0.5}\, dB(s, w) \right\}$$

Planets by fbm



三维噪声函数

- 三维噪声函数是一个被广泛应用的过程纹理函数

$$f=noise(x,y,z)$$

- 良好的噪声函数应满足的条件

  旋转统计不变性

  平移统计不变性

  频率带宽很窄

移动和旋转噪声函数后，总体特征保持不变

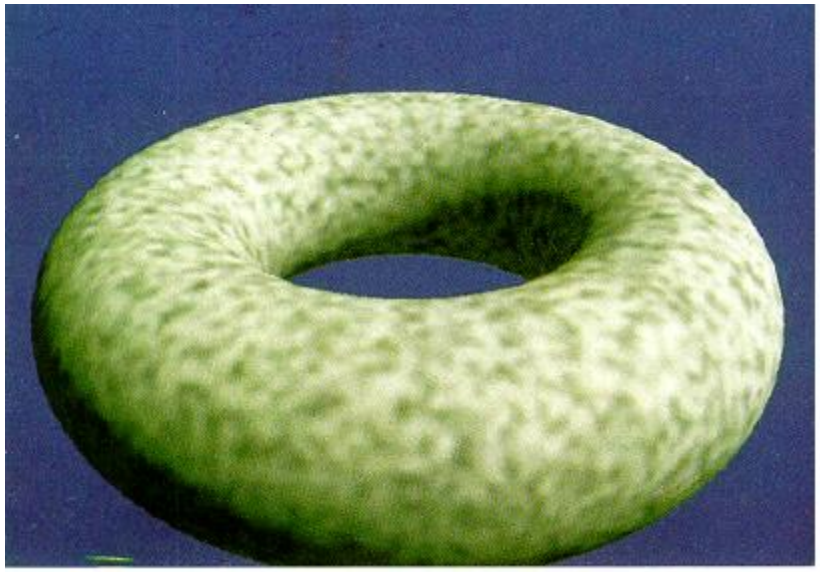
构造三维噪声函数的一种快速方法

采样三维整数网格定义噪声函数。在每一个整数网格点(i, j, k)上定义一个随机灰度值(0, 255). 非整数网格点处的函数值由其周围的8个整数网格点处的函数值插值得到。

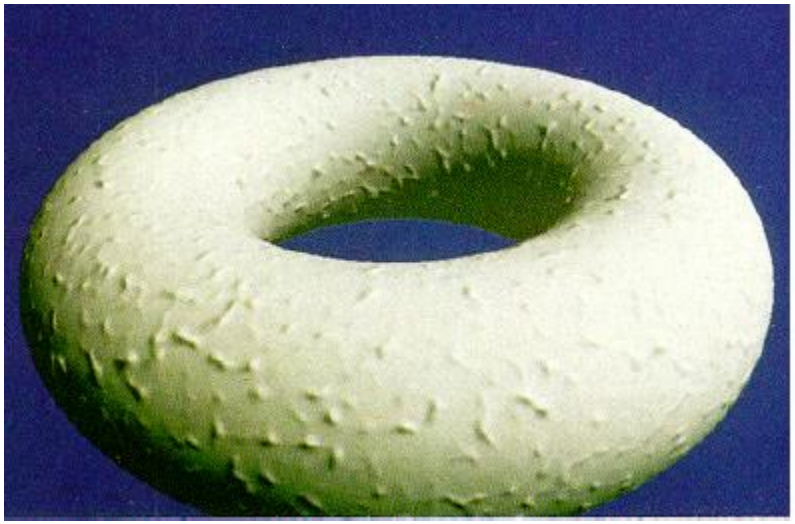
该方法得到的噪声纹理是连续函数，但在相邻网格界面上的梯度不连续，因此在某些方向上存在人工痕迹，原因在于该噪声纹理不满足旋转统计不变性。

实际中使用噪声函数生成其他自然纹理，而不是直接绘制噪声纹理。

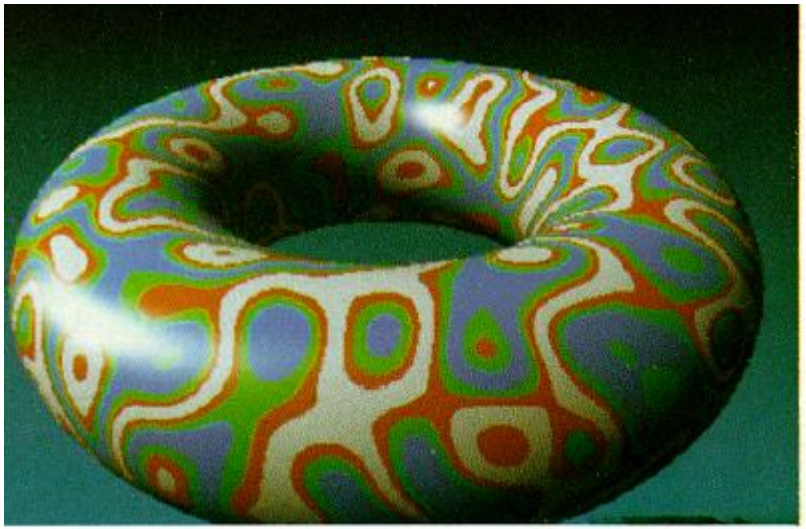
三维噪声函数



三维噪声函数



三维噪声函数



湍流函数

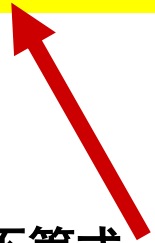
- Perlin于1985年提出的一种关于湍流函数的经验模型，可成功地用于模拟大理石、火焰、云彩等自然纹理
- Perlin的湍流函数由一系列三维噪声函数叠加而成



湍流函数

$$turbulence(P) = \sum_{i=0}^{k} | Noise(2^i P) / 2^i |$$

$$1/2^{k+1} < \text{象素长度}$$



K为满足不等式 的最小整数

P表示纹理空间中的一点(x, y, z)

湍流函数

湍流函数构造的特点：

任取和式的相邻两项，后一项噪声函数的变化频率总是前一项变化频率的两倍，其变化幅度为前一项的一半，因此其对湍流函数的贡献缩小了一半。

用湍流函数模拟自然景物纹理

1. 选取简单的函数描述自然纹理的基本结构特征。一般说来，这些函数是连续的，且具有变化较大的一阶导数
2. 用湍流函数扰动基本函数中的某些参数，从而产生复杂的纹理细节

用湍流函数模拟自然景物纹理

1. 描述大理石纹理不同材质的颜色以及大理石纹理的周期形态

Marble(P)=marble-color(sin(x))

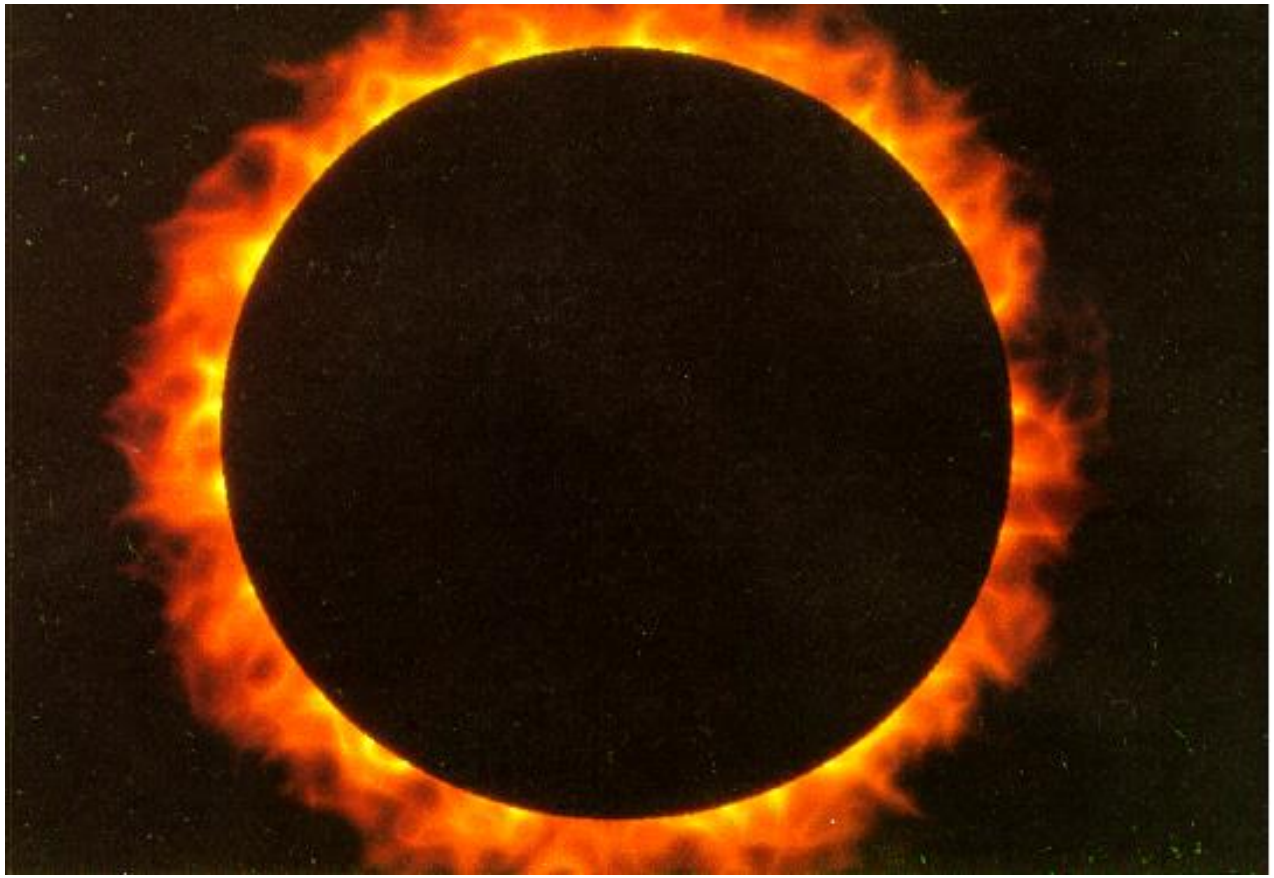
2. 将基本参数经过湍流函数扰动

Marble(P)=marble-color(sin(x+turbulence(P)))

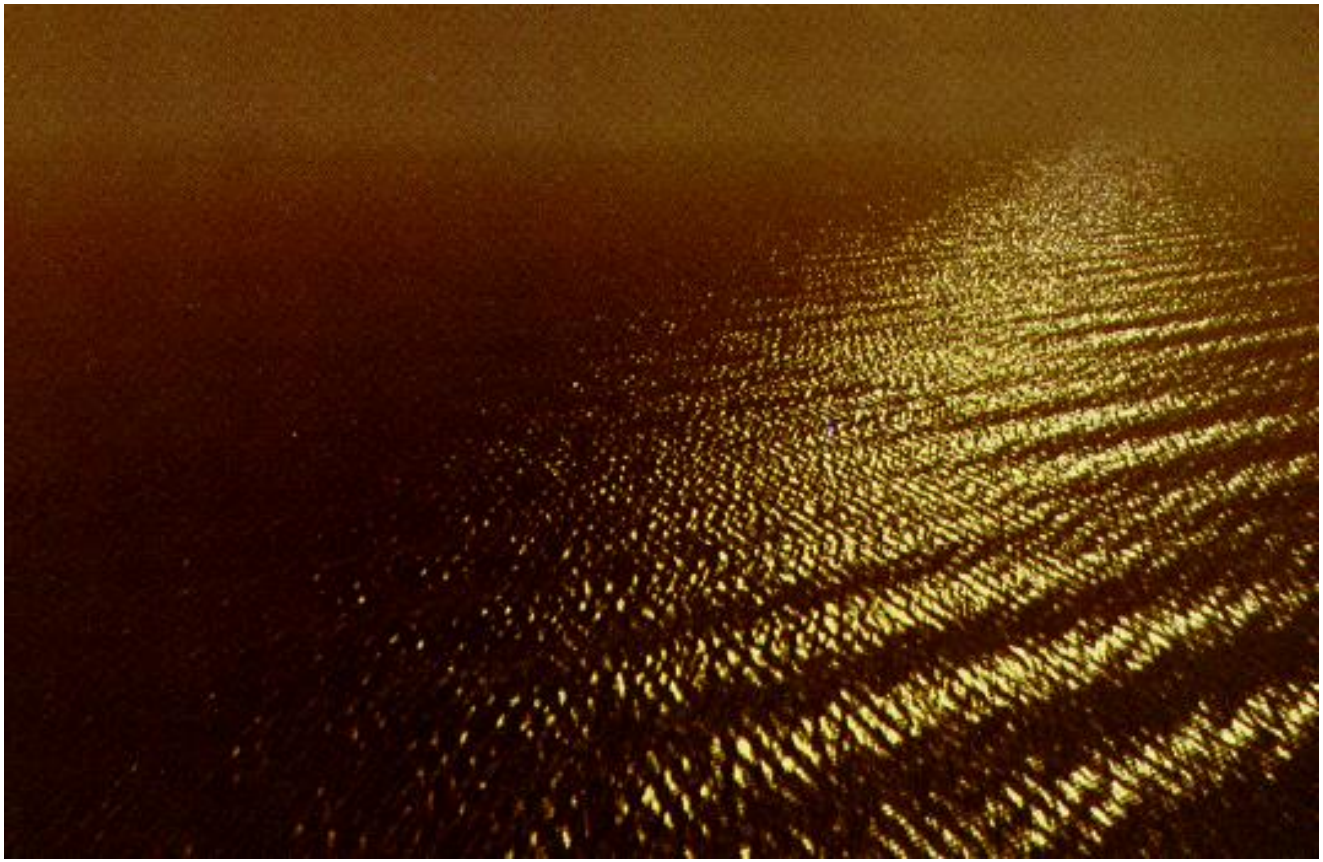
marble-color()是一个彩色滤波函数，它将[-1,+1]区间中的任何一实数值映射为RGB空间的一个颜色。一般可将其表示为三条独立的样条曲线

用湍流函数模拟自然景物纹理

用湍流函数模拟自然景物纹理



用湍流函数模拟自然景物纹理



三维噪声函数&湍流函数

参考资料： perlin.pdf

Perlin K. An image synthesizer. Computer Graphics,1985,19(3):287~296

气态过程纹理

- 与其他自然景物不同，很难用几何造型的方法描述气体的基本几何形状
- 气态过程纹理

  建立气体的基本体密度函数

  利用湍流函数扰动纹理空间中点的位置

  计算扰动后的点的密度

基本气体外形的描述

利用幂函数描述气体的几何形状

```
Basic-gas(pnt,density,parms)
Xyz-td pnt; float *density,*parms;
{
  float turb;  int i;
 float pow-table[POW-TABLE-SIZE];
 for (I= POW-TABLE-SIZE-1;I>=0;I--)
 Pow-table[I]=pow(((double(i)/(POW-TABLE-    SIZE-
1)*parms[1]*2.0,(double)parms[2]);
```

基本气体外形的描述

```
turb=turulence(pnt);
*density=pow-table[(int)(turb*(0.5*(POW-TABLE-SIZE-1)))];
}
```

Parms[]是一个参数数组。Parms[1]是气体密度的上限值，界于0和1之间

Parms[2]是幂函数的指数

Parms[1]固定时，parms[2]越大，气体越接近羽毛状

茶杯的蒸汽上升效果的模拟

- 使用basic-gas定义的函数。由于蒸汽的密度不大，可以取parms[1]=0.57,parms[2]=6.0
- 效果不逼真
  - ❖ 蒸汽没有完全限制于茶杯的上方
    - ➢ 以杯中水面为中心对密度分布作**球状**衰减
  - ❖ 蒸汽的密度没有随着高度的增加而下降
    - ➢ 对空间点的高度，采用**指数衰减**函数

烟柱的模拟

- **烟柱的特征**

  上升时逐渐扩散，并慢慢地由规则形状变得紊乱

- **引入湍流到圆柱的中心，使烟柱看上去逼真**

  为模拟湍流和空气气流的效果，湍流量将作为烟柱高度的函数增加，其密度作为高度的函数降低，从而确保扩散效果的真实性

- **烟雾上升时，其形状应该是弯曲和盘绕的**

  在计算点到圆柱中心的距离之前，使用一个垂直的螺旋线移动该点可以得到满意的效果

过程纹理的动画技术

1. 随时间变化改变纹理空间

- 将时间作为参数，改变纹理空间中纹理的定义

2. 在纹理空间中移动绘制点

- 不改变纹理空间

湍流运动的模拟

随时间变化改变纹理空间（第一种方法）

由于湍流函数由噪声函数所定义，需要定义四维噪声函数。在四维空间中构造一个整数网格结构，则四维噪声函数noise(P,t)在其每个整数网格点(i,j,k,l)的取值为一随机灰度，非网格点处的值由四线性插值函数计算得到。

以四维噪声函数取代三维噪声函数便可以得到四维湍流函数turbulence(P,t)。

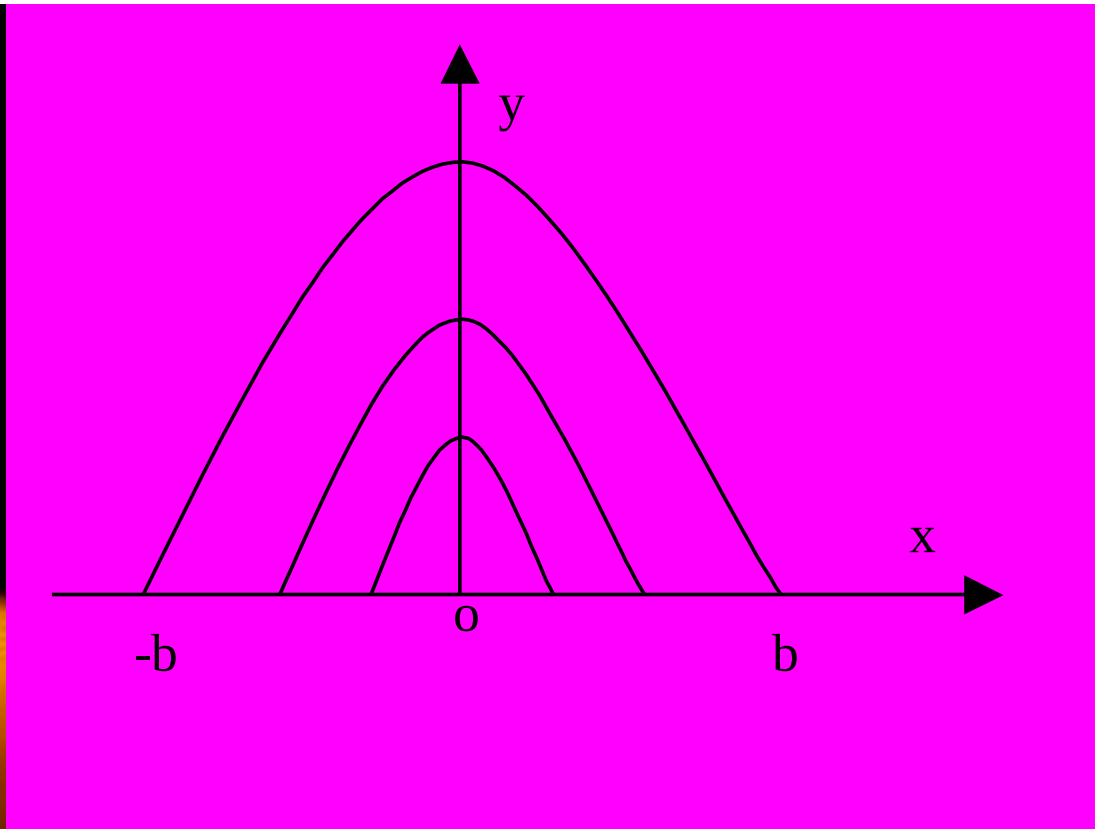
湍流运动的模拟

为增加真实性，可以将第二种纹理动画技术引入到四维湍流函数中，此时，空间点P亦是时间的函数，因此四维湍流函数可以表达为turbluence(P(t),t)

以二维火焰动画效果的模拟为例

二维火焰动画效果的模拟

- 二维火焰的几何形状用抛物线形区域表示



二维火焰动画效果的模拟

二维火焰的颜色分布

Flame(P)=(1-y/h)flame-color(|x/b|)

flame-color()函数是一条三维颜色空间中的参数曲线，分别表示R,G,B三条分量曲线。这三条分量曲线在x=0处达到最大，在x=1处取值为0。

当x从0变到1时，绿色和蓝色分量较红色分量曲线衰减的快

二维火焰动画效果的模拟

- 引入湍流函数后，火焰的动态色彩分布函数

Flame(P,t)=(1-y/h)flame-color(|x/b|+turbulence(P,t))

- 引进第二种动画技术，改变火焰的位置

Flame(P,t)=(1-y/h)flame-color(|x/b|+turbulence(P＋△P(t),t))

△P(t)=(0, t △y,0)

△y为火焰所在的长方形沿y方向关于纹理空间的相对运动速度

第七章 过程动画

Fourier合成技术

- **Fourier合成技术是一种有效的过程动画技术**
  - ❖ **水波、云彩、山脉、森林…**
- **基本思想**
  - ❖ **将一系列不同频率和相位的正弦（余弦）波叠加起来产生所需要的纹理模式**

云彩纹理

- 云彩的形状
  - 薄层状------云层平面
  - 厚团状－－－系列的椭球面
- 云彩的光亮度和透明度
  - 一系列三角函数构成的三维纹理函数

三维纹理函数

$$T(x,y,z)=\sum_{i=1}^{n}C_i[Sin(FX_i*x+PX_i)+T0]*$$
$$\sum_{i=1}^{n}C_i[Sin(FY_i*y+PY_i)+T0]*$$
$$\sum_{i=1}^{n}C_i[Sin(FZ_i*z+PZ_i)+T0]$$

N=6 时 可生成丰富的纹理细节

三维纹理函数

$$FX_{i+1} = 2FX_i$$

$$PX_i = \frac{\pi}{2}\sin(FZ_{i-1}z)$$

$$FY_{i+1} = 2FY_i$$

$$PY_i = \frac{\pi}{2}\sin(FX_{i-1}x)$$

$$FZ_{i+1} = 2FZ_i$$

$$PZ_i = \frac{\pi}{2}\sin(FY_{i-1}y)$$

$$C_{i+1} = 0.707C_i$$

$$T_0 = 1$$

光照模型

$$I_1 = (1-s)*I_d + s*I_s$$

$$I_2 = (1-t)*I_1 + t*T(x,y,z)$$

$$I = (1-a)*I_2 + a*I_a$$

Id和Is是云层平面或者椭球球面上可见点处的漫反射和镜面反射分量 （Phong 模型…）

S, t和a分别是云层的镜面，纹理和泛光系数

透明度函数

- 为了获得真实感的云彩效果，云彩的边缘必须趋于透明，因此需要设计合适的透明度函数

- 平面云层的每一可见点处定义透明度函数V:

$$V=\begin{cases}0 & T(x,y,z)\geq V_1+D\\ 1-\dfrac{T(x,y,z)-V_1}{D} & V_1\leq T(x,y,z)<V_1+D\\ 1 & others\end{cases}$$

V1 和D决定了透明度从0变化到1的区域

透明度函数

- 团状云彩的透明度函数

$$V = 1 - \frac{T(x,y,z) - V_1 - (V_2 - V_1)(1 - g(x,y,z))}{D}$$

其中，g(x, y, z)是一个标量函数，它于椭球在图像平面的投影中心处取值为1，在其投影边界处取值为0；

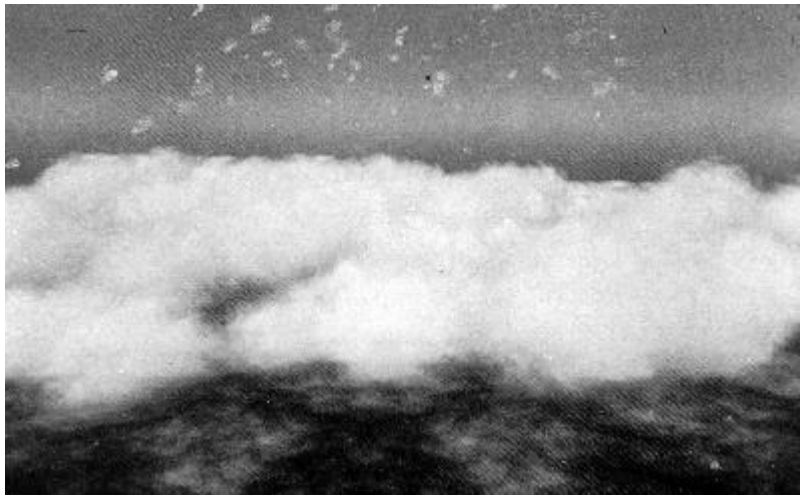
V_1和V_2分别为投影中心和边界处的透明度

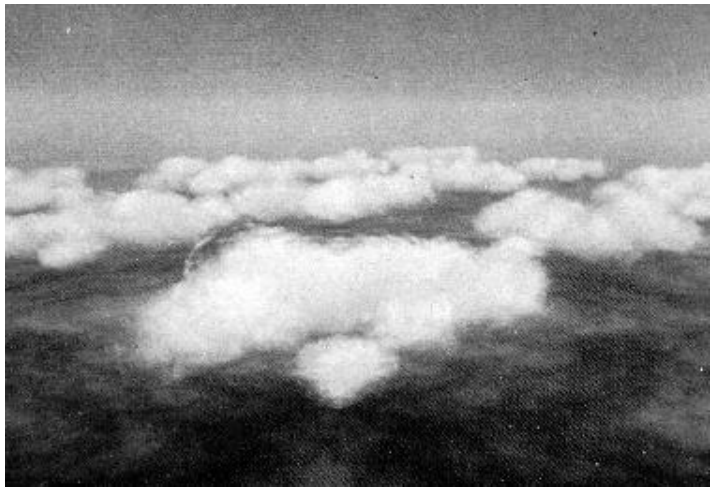
云彩纹理

光照模型和透明度函数结合，可以计算云彩表面每一可见点处的色彩。
Gardner用这种方法生成了非常逼真的三维云彩，地貌和森林景色，但该方法的计算量比较大，因为三维纹理函数T(x,y,z)的计算比较耗时

云层

云团





参考文献

Gardner G. VISUAL SIMULATION OF CLOUDS. Computer Graphics,1985,19(3):297~303

Gardner.pdf

水波纹理

- 正弦波叠加技术生成水波纹理是Fourier合成纹理的一个实例
- 优点
  - 非常容易实现水的波动
- 缺点
  - 只能模拟小振幅的水波。水波函数的振幅逐渐变大时，纹理的真实感会逐渐变差

水波纹理

- 在该景物表面产生水波效果的两种方法
  1. 将波纹函数生成的纹理作为凹凸纹理映射到景物表面，该方法简单高效
  2. 将波纹函数作为位移函数扰动景物表面的采样点
     - 需预先剖分景物表面
     - 剖分生成的采样点分布稀疏时，会出现严重的波纹走样现象

平行水波的波纹函数

$$z(x,y)=\sum_{i=1}^{n} Ai\cos(u_i(x-C_{i1}t)+v_i(y-C_{i2}t))$$

Ai是水波振幅，t为时间，(u_i,v_i)是第I个正弦波的前进方向；(C_{i1},C_{i2})为第I个正弦波的速度

$(C_{i1},C_{i2})=C_i(u_i,v_i)$, Ci是一个常数

平行水波的波纹函数

为避免水波纹理呈现周期形态，函数中的参数必须适当选取。逐一确定每一个余弦波的参数很麻烦，一个可行的办法是为每一个参数设置一上、下界，然后用随机数生成器生成每一正弦波的各个参数。实践证明，这种方法可以生成较为逼真的水波纹理效果

平行水波的运动

上面的方程实际上定义了t时刻水波的一个高度场。当t变量连续变化时，便得到了水波的运动，将它们作用于景物的表面，便可以产生所需要的水波纹理的动态效果。

同心水波函数

$$Z(x,y)=A\cos(k|r|-Ct)$$

$$r=(x,y)-(c_1,c_2)=(x-c_1,y-c_2)$$

（C1,C2）为同心水波的中心

C为扩散速度

http://graphics.stanford.edu/courses/cs348b-competition/cs348b-01/ocean_scenes/

http://graphics.stanford.edu/courses/cs348b-competition/cs348b-01/ocean_scenes/

第七章 过程动画

基于语法的造型技术

字符串表示绘图规则

F：表示在当前的位置画一条长为k的直线段。k是由用户事先任意设定的数值，表示基本线段的长度。

+：表示逆时针旋转一个角θ，θ的数值由用户事先确定；

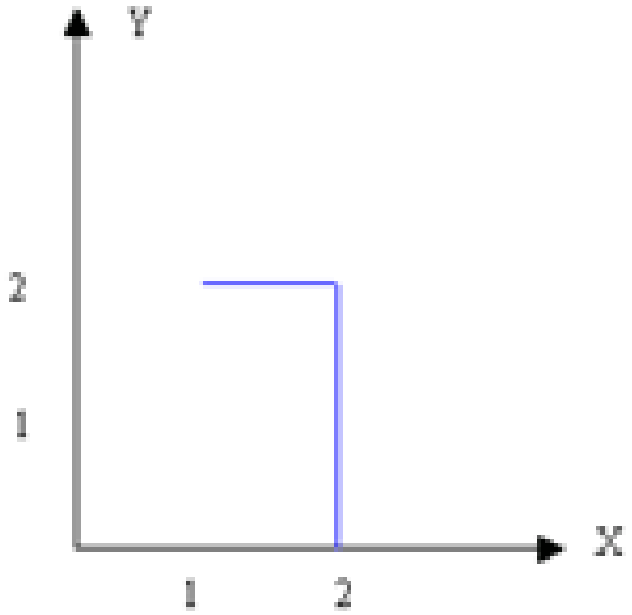
−：表示顺时针旋转一个角θ；

[：表示暂时保存当前的画图状态

]：表示提取出保存的画图状态。“[”和“]”要成对的出现。

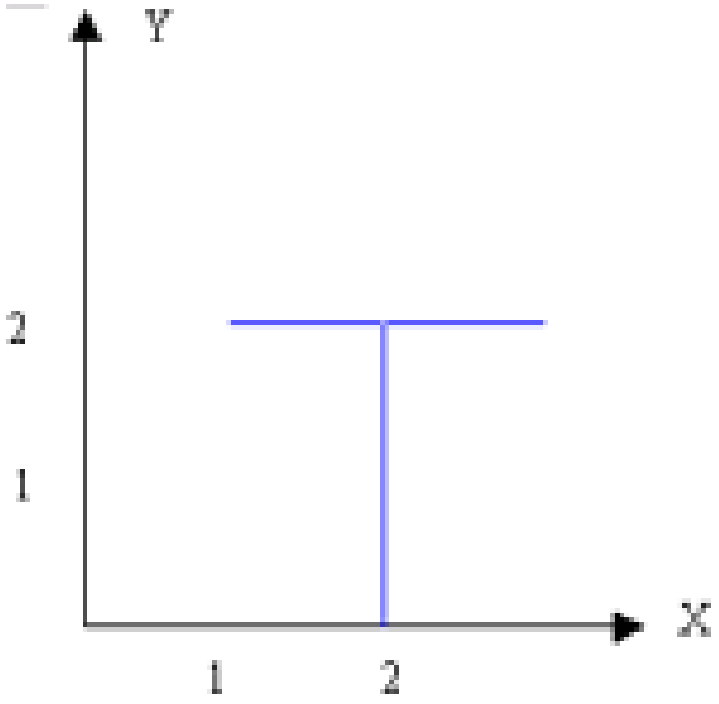
字符串表示的绘图规则

确定了开始的坐标和方向，由上面符号组成的任意的一系列指令就能指导画图了。比如：长度k=1，θ=90度角，开始坐标是（2，0），开始方向角是90度， FF+F字符串画出来的图为：



字符串表示的绘图规则

FF[+F][-F]



给定一个符号串就能对应的画出这个符号串代表的图形

基于语法的造型技术—L系统

- L-系统指美国生物学家Lindenmayer A. 提出的研究生物形态与生长的描述方法。后来Smith等人将L系统引进图形学，形成一类自然景物模拟的有效方法。它可以描述树种、植物及植物叶片等自然景观及生长过程。
- 一棵树的结构可以表示为由一些语法规则定义的语言中的一个语句，语法规则由一个可图示的产生式集确定，最后基于这些产生式规则将语句转化为图像

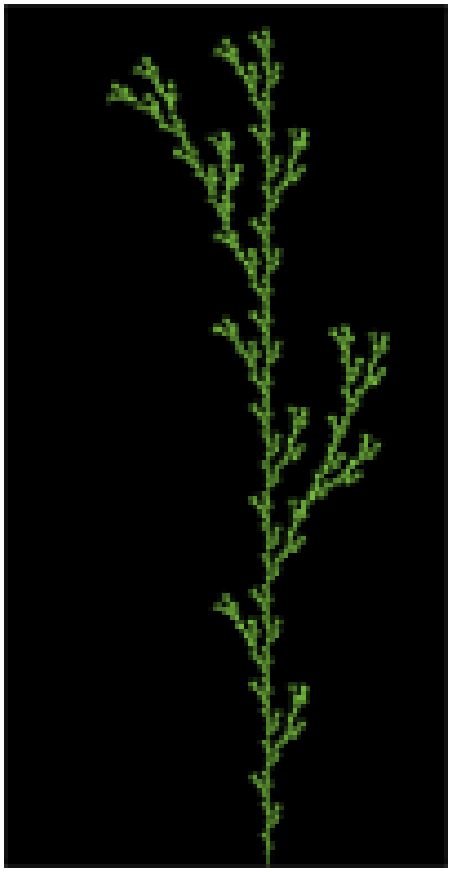
产生式系统

产生式系统就是由若干个产生式构成的一组语句，各个产生式之间可以相互替换字符串

产生式（也叫做规则）是形如X->Y的式子，其中X叫做左件，Y叫做右件。X->Y表示X能够推导出Y。如果X是一个字符串，Y也是字符串，那么X->Y表示能够用Y来替换X。例如如果给定一个初始时刻的字符串ABXXTT，那么运用规则X->Y就能把这个字符串变成ABYYTT。如果产生式的右件多于一个字符，那么就能推导出比原来字符串更长的字符串来。例如如果X->YYY，那么ABXXTT会被替换成ABYYYTT，显然后来的字符串比原来的长。

产生式系统

产生式系统：X->YF，Y->+FX，开始时刻的字符串是X，用这两个规则跌代1次就能得到字符串：+FXF，迭代2次就是+F+FXFF，3次是+F+F+FXFFF。。。



F->F[+F]F[-F]F，开始的时候字符串是F，转角为-25.7341度。迭代6次就可以得到最终的画图指令字符串，再根据这个指令字符串画出图形，就能得到左边的图

L系统生成的2D树

L系统生成的2D树

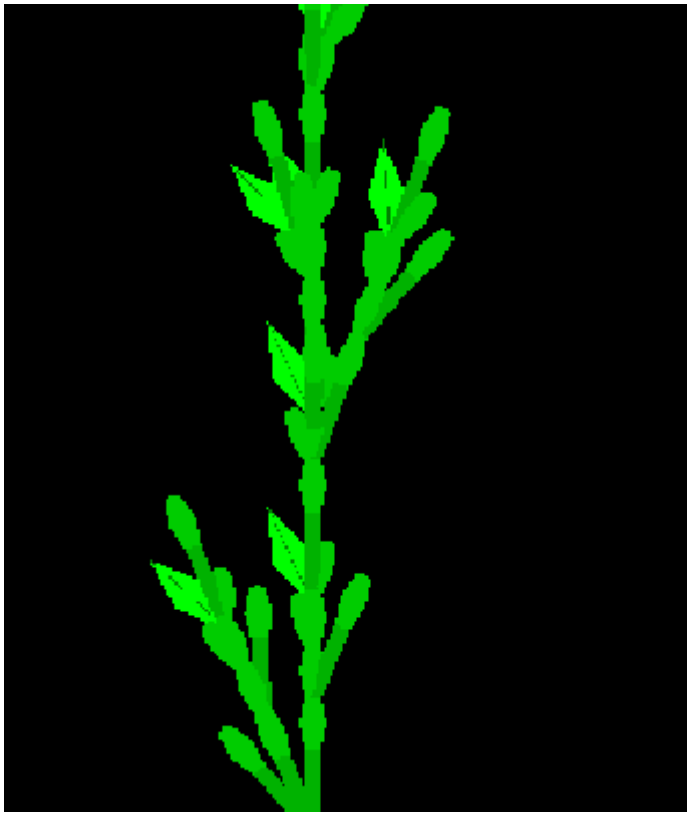
L系统生成的2D树

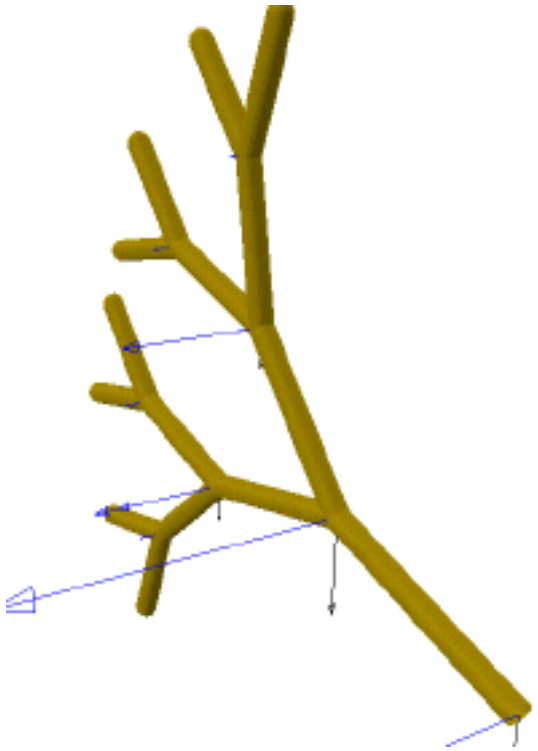
http://www.swarmagents.com/complex/bottomup/fractal.htm

L系统生成3D树

二维植物结构说明了如何采用语法规则来控制隐含的几何结构。该方法可以方便地推广到三维植物的造型，此时，需要确定三维空间的产生式规则以及描述体素的复杂几何和渲染属性。例如，可以用具有适当弯曲度的圆柱体描述树枝，将圆柱体截面半径的减少定义为树枝长度的函数，并引入纹理图像或者纹理函数来描述树皮的纹路。

L系统生成的3D树





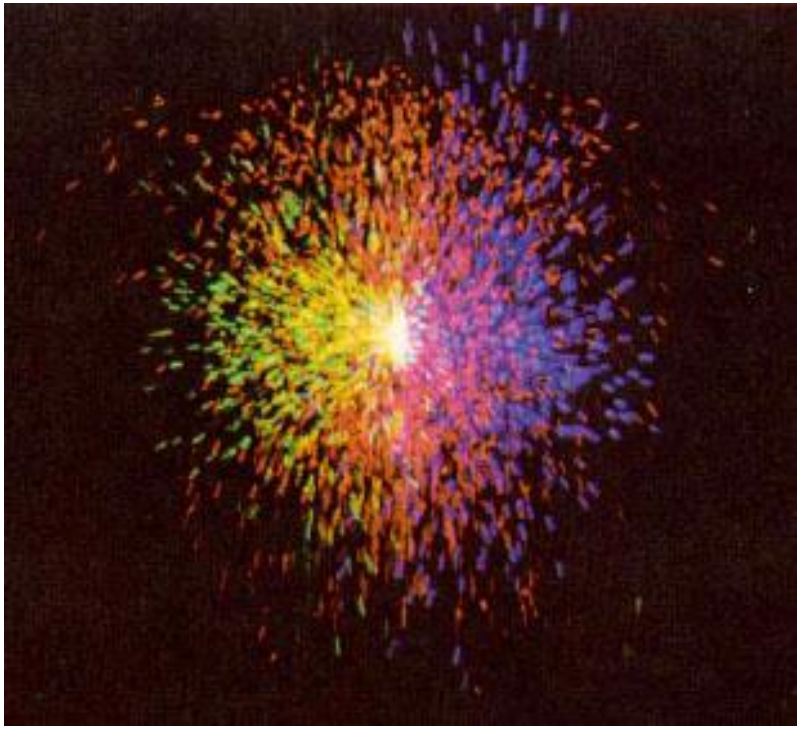
第七章 过程动画

粒子系统

- 粒子系统是模拟不规则模糊物体的模糊性和动态性的最成功的一种图形生成方法
  - 景物被定义为由成千上万个不规则的、随机分布的粒子所组成。每个粒子都具有一定的生命周期，不断地改变形状，不断地运动
  - 动画效果体现在景物的整体形态和特征的动态变化，而不是各个粒子本身
  - 火、云、水、森林。。。

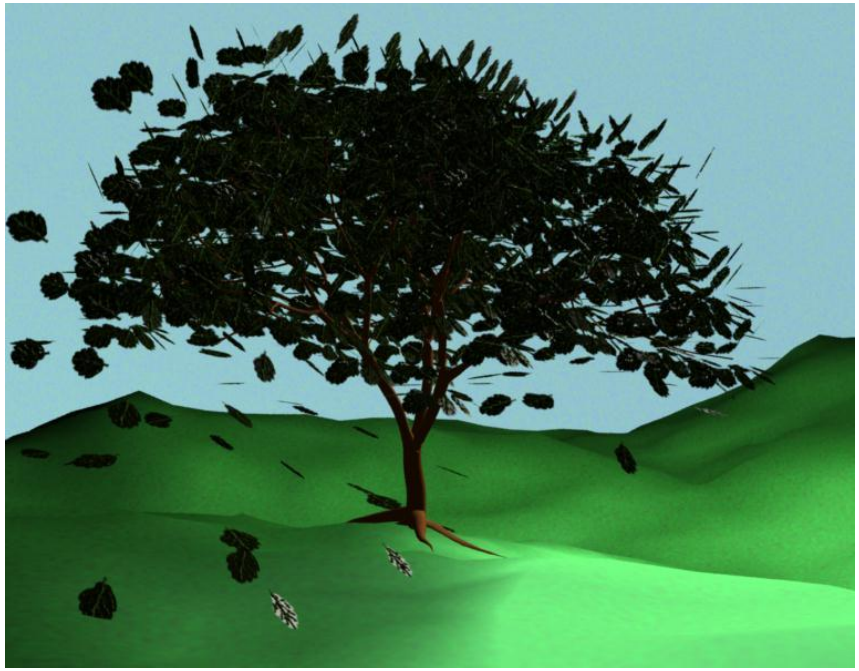
粒子系统

- A collection of many minute particles that together represent a fuzzy object
- Use points to define shapes rather than polygons

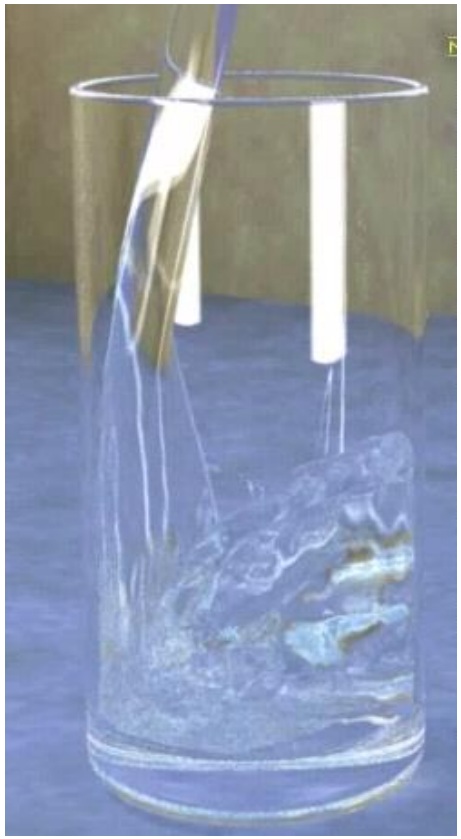


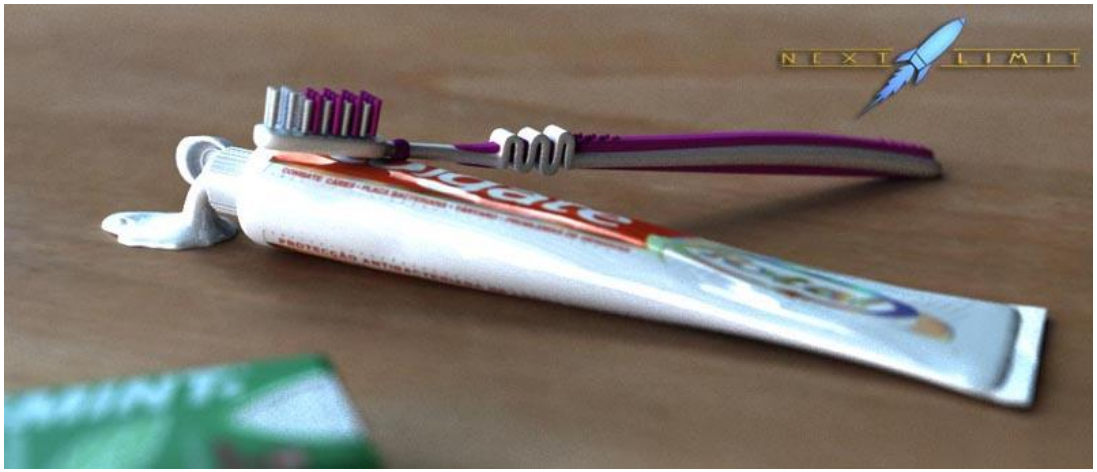
优点

- Simple – points rather than polys
- Procedural
- Random
- Models that are “alive”



Particle Systems and Fuzzy Shapes

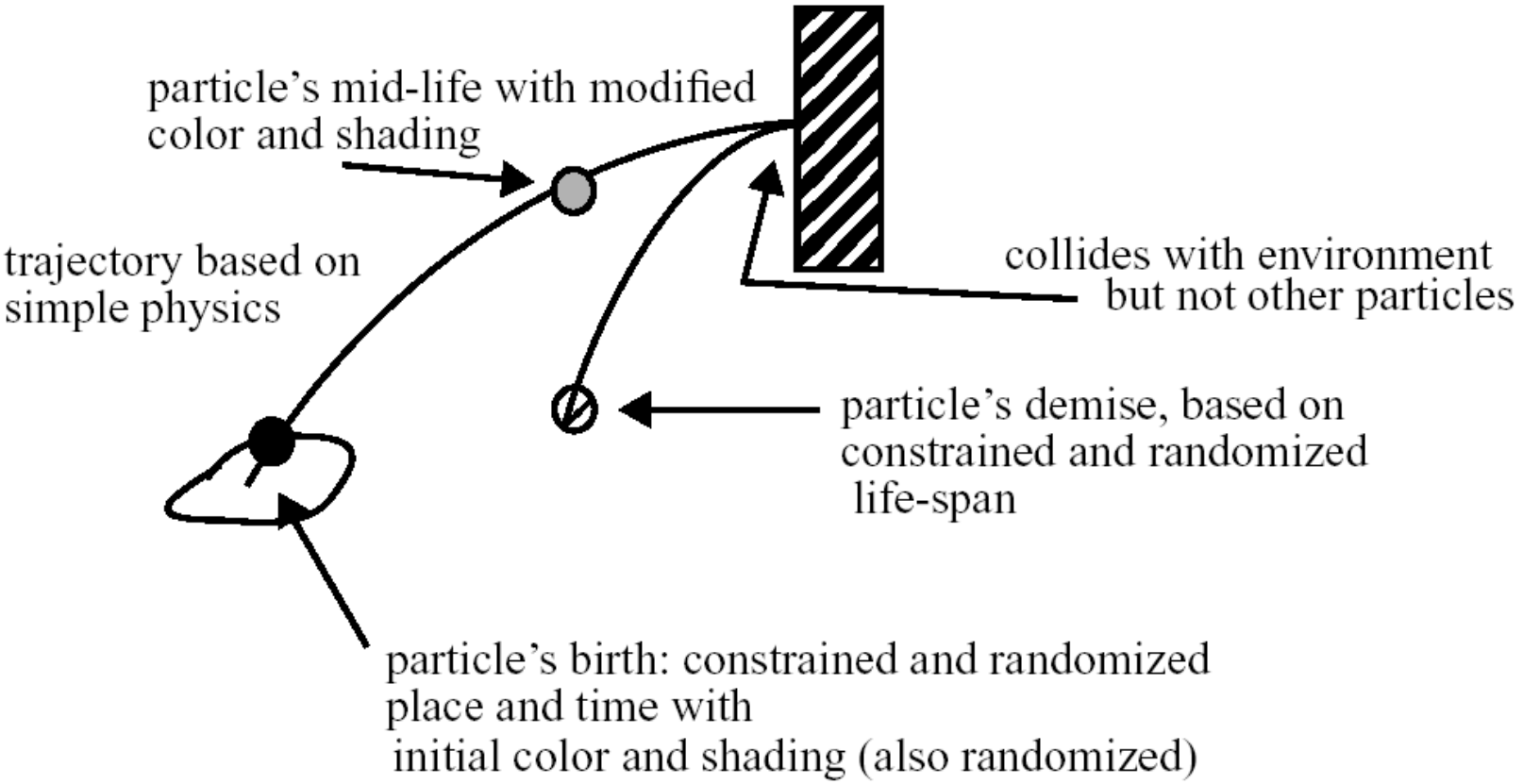




Fuzzy objects

- Do not have smooth, well-defined, and shiny surfaces
- Irregular, complex, and ill-defined
- Soft, deformable objects

Life of a Particle



粒子的生命期

- Lifetime of a particle defined at birth to be a certain number of frames
- Or, kill particles
  - That are not visible
  - When they are a certain distance from the origin
  - Below a threshold intensity

粒子的特点

- 每个粒子均被赋予一定的生命周期
  - 出生—>成长—>衰老—>死亡
- 与粒子有关的每一个参数将受到一个随机过程的控制
  - 粒子不断改变其形状，不断运动
  - 粒子系统表示的景物具有良好的随机性和动态性

粒子系统画面生成的基本步骤

1. 生成新的粒子并将其加入到系统中
2. 赋予每一新粒子以一定的属性
3. 删除那些已经超过其生命周期的粒子
4. 根据粒子的动态属性对粒子进行移动和变换
5. 渲染并显示由有生命的粒子组成的图形

粒子的运动与随机性

生成粒子系统画面时, 可以与任何与描述物体运动和特征的模型结合起来, 如粒子的**运动**, **变换**可以用一个**偏微分方程**描述.

对于粒子系统的**随机性**, 可以采用一些简化的**随机过程**来控制粒子在其所在系统中的**形状**, **特征**和**运动**.

粒子的运动与随机性

Parameter=MeanParameter+Rand()*VarParameter

Parameter:粒子系统中任一需要随机确定的参数

Rand(): [-1,1]中的随机数函数

MeanParameter: 该参数的均值

VarParameter: 该参数的方差

粒子的数目的控制

方法1：确定每一时刻进入系统的粒子数

Nparticles=MeanParticles+Rand()*VarParticles

方法2：单位屏幕面积上具有的平均粒子数和方差

Nparticles=(MeanParticles+Rand()*VarParticles)*ScreenArea

避免用大量粒子模拟在屏幕上投影面积很小的景物

!!! 为了使粒子系统在光强上有变化,可以将进入系统的平均粒子数看作是时间的函数

MeanParticles=InitialMeanParticles+△MeanParticles*(t-t0)

粒子的基本属性

位置和大小

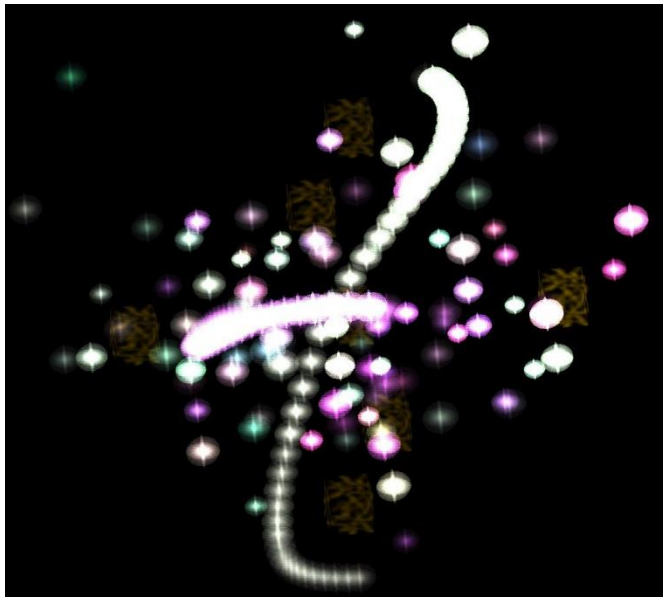
运动速度和方向

颜色

透明度

形状

生命周期

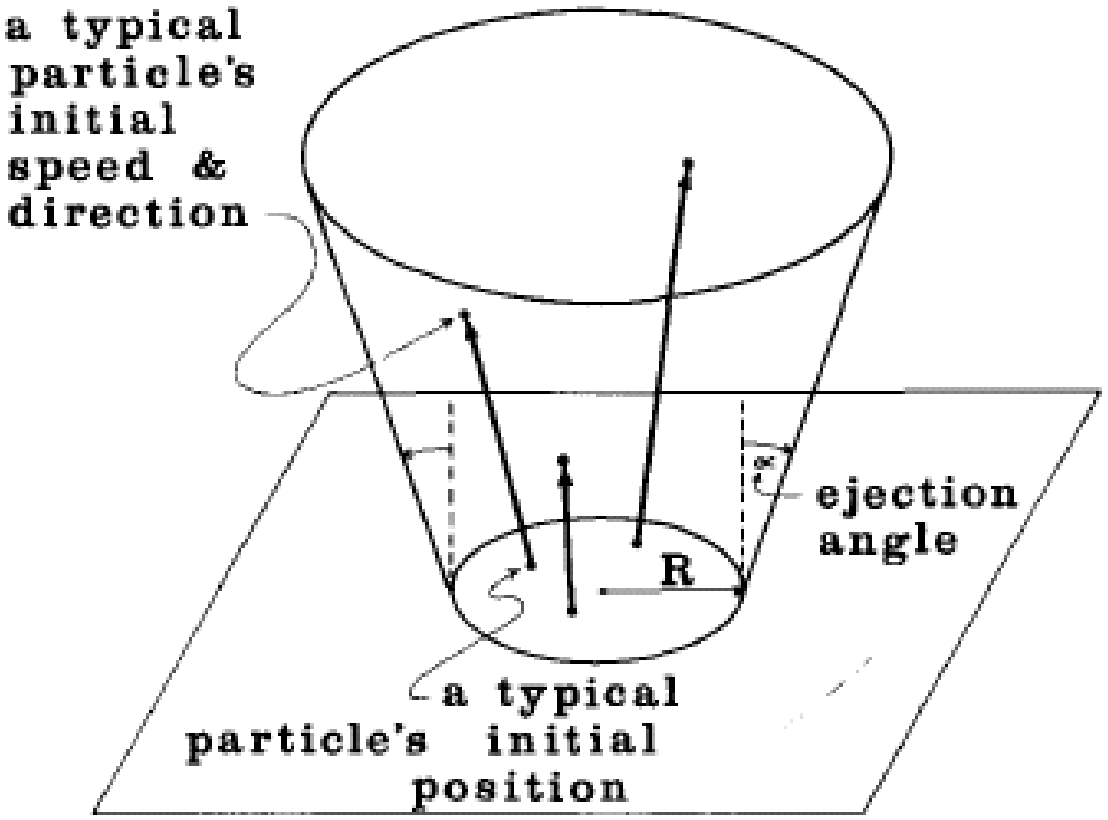


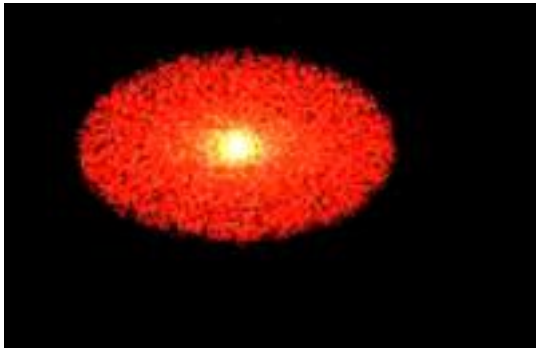
粒子的基本属性

为定义粒子的初始位置和运动方向, 可以采用一个规则物体来描述粒子系统的基本生成形状, 定义关于其局部坐标系原点的一个区域， 新生成的粒子就随机地放置在该区域内. 可以用粒子发射时的一些角度度量定义新粒子的初始运动方向.

常用的基本形状:

球面， x-y平面上的圆, 长方体, 分形中的吸引子...


a typical
particle's
initial
speed &
direction
ejection
angle
R
a typical
particle's initial
position



Particles are laid out in a circular disc structure and given an initial upwards velocity. When the particle's lifetime is expired it is removed. If the particle collides with the ground it remains stationary.

Similar to above example , but particles are initially laid out in a square structure

Similar to above example , but particles are initially laid out in a ring structure.

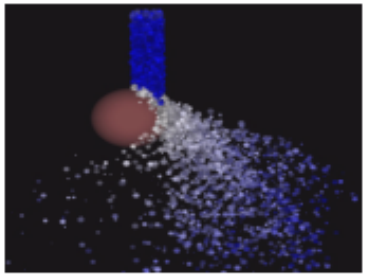
In this example the particles are initially clustered in a point, but particles are constantly being regenerated at the start point. The effect looks very much like a fire or torch burning.

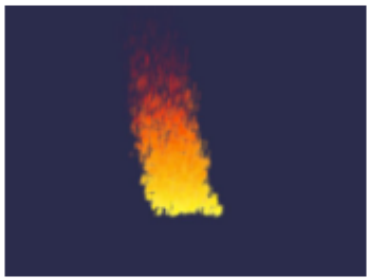
粒子系统的渲染

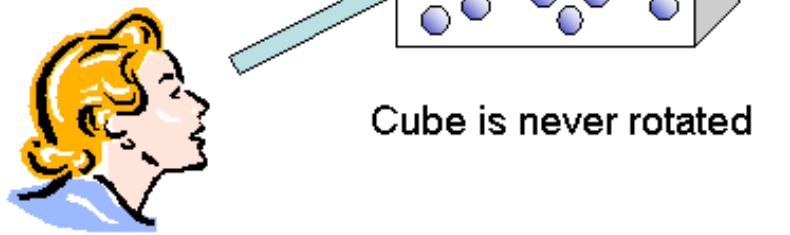
- 传统渲染方法
  - 粒子数目很大,导致复杂的消隐计算
- 简化的渲染方法[Reeves]
  - 用粒子系统表示的自然景物不与任何几何模型相交, 因而粒子系统的渲染与其他景物的渲染可分别进行.
  - 每一粒子都被假设为一个点光源, 它们都对所覆盖的象素有光亮度贡献, 其大小决定于粒子的颜色和透明度

Rendering a Particle

Rendering a Particle





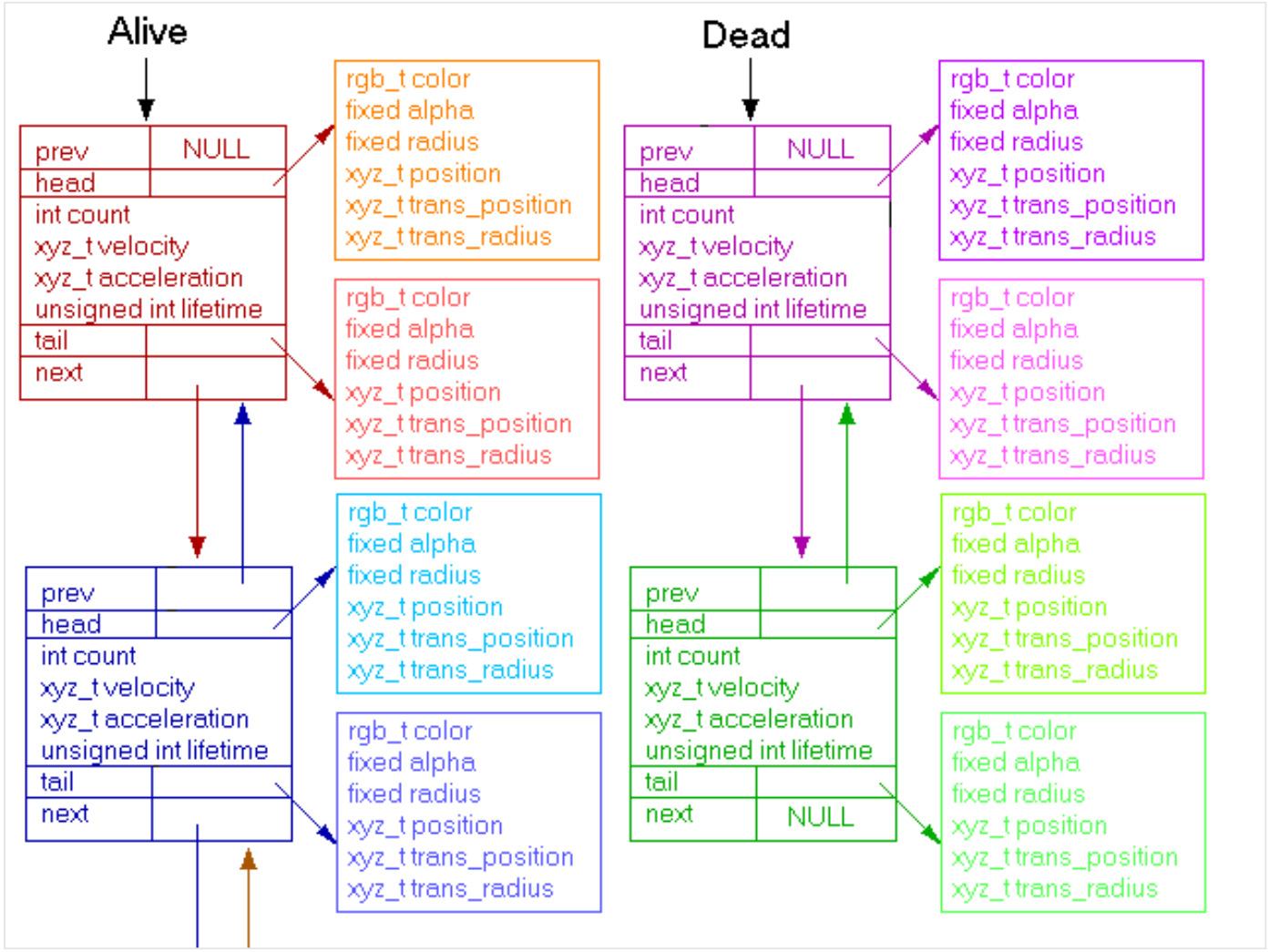


Cube is never rotated

- Each particle → a single point, line, sphere, or complex polygonal object.
- The advantage of **spheres** is that no matter how you rotate them, if there is no texturing on them, they will look the same.

- Doubly linked list → particles can be easily moved from one pool to another.
- Alive pool → contains particles that should be rendered to the screen because they are currently active
- Dead pool → reserve of allocated particle space that may be needed later or have exceeded their lifetime.

Rendering a Particle



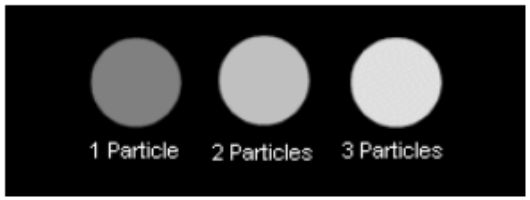
Rendering a Particle

Rendering a Particle → Alpha Blending

Without alpha-blending : particles are simply textured rectangles.

Alpha-blending is enabled

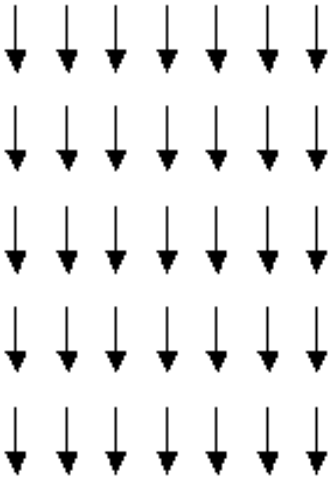
$$FinalColor = DestColor \times DestBlendFactor + SrcColor \times SrcBlendFactor$$



The more particles are on top of each other, the brighter the color in that area should be

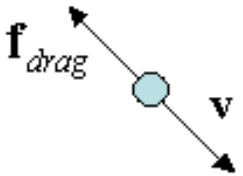
粒子所受的力

Unary forces -
forces that only depend on 1 particle



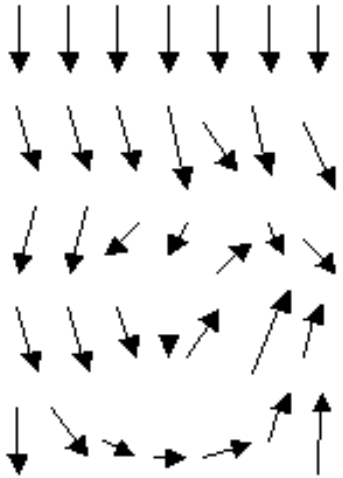
Gravity

$$\mathbf{f} = m\,\mathbf{g}$$



Dampening

$$\mathbf{f}_{drag} = -k_d\,\mathbf{v}$$

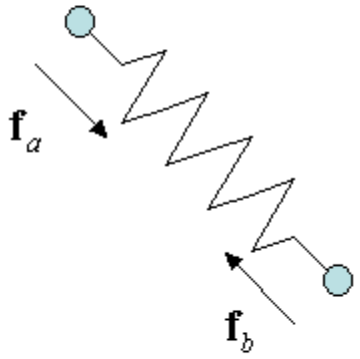


Wind Fields

$$\mathbf{f} = k\,\mathbf{v}_{wind}$$

粒子所受的力

Binary forces -
forces that only depend on 2 particles



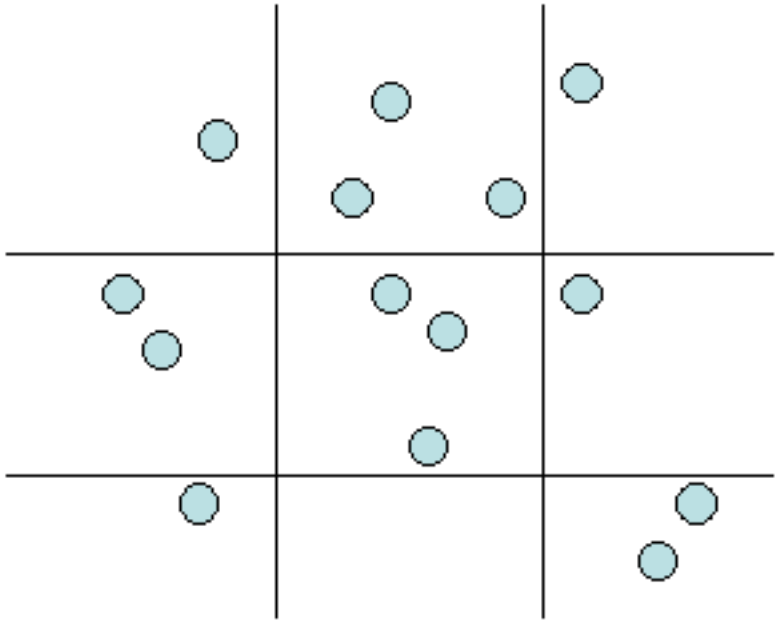
Springs

$$\mathbf{f}_a = -k_s(|\mathbf{x}_a - \mathbf{x}_b| - l_0)\frac{\mathbf{x}_a - \mathbf{x}_b}{|\mathbf{x}_a - \mathbf{x}_b|}$$

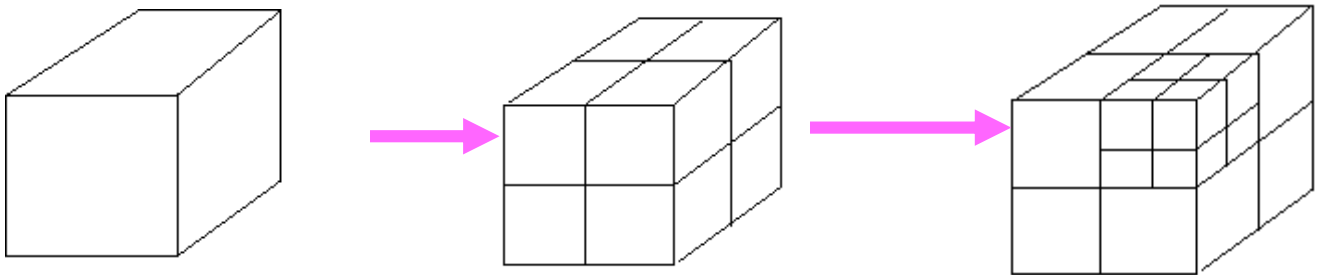
$$-k_d\left(\frac{(\mathbf{v}_a - \mathbf{v}_b)\cdot(\mathbf{x}_a - \mathbf{x}_b)}{|\mathbf{x}_a - \mathbf{x}_b|}\right)\frac{\mathbf{x}_a - \mathbf{x}_b}{|\mathbf{x}_a - \mathbf{x}_b|}$$

粒子所受的力

Spatial forces -
forces that depend on local particles



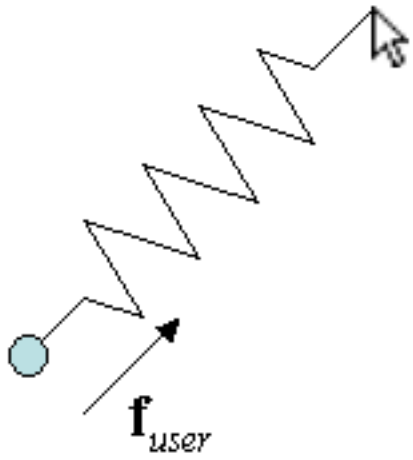
A better method → *octree space subdivision*



–The essence of it is that you clump particles together as one virtual mass when computing the effect of that clump of particles on a particle or clump of particles far away.

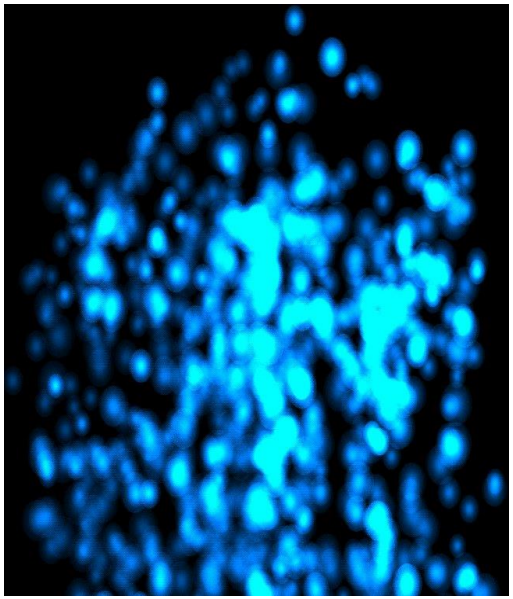
粒子所受的力

User Interaction forces -
forces applied to particles by the user



Dumb Particles

- Particles that do not interact with each other
- i.e. vortices(漩涡), smoke, rain, fire



Smart Particles

- Particles interact with each other
- Useful for simulating

  -Fluids

  - Mass,Density, pressure, viscosity per particle

例子

Spray.dsw: 一个简单的粒子系统的源代码 (Particle.cpp)

ParticleIllusion: 一个功能比较丰富的2D粒子系统效果的软件

http://www.wondertouch.com/

思考题：

1. 什么是过程动画？过程动画技术有哪些？
2. 常用的三维过程纹理造型技术有哪些？过程纹理动画技术有哪些？
3. 什么是Fourier合成技术？
4. 请用L系统生成一棵二维树木模型（实验4，选做）。
5. 粒子系统动画的过程是怎样的？

结束