A9: Particle Systems

一、概述

**1.目标：**

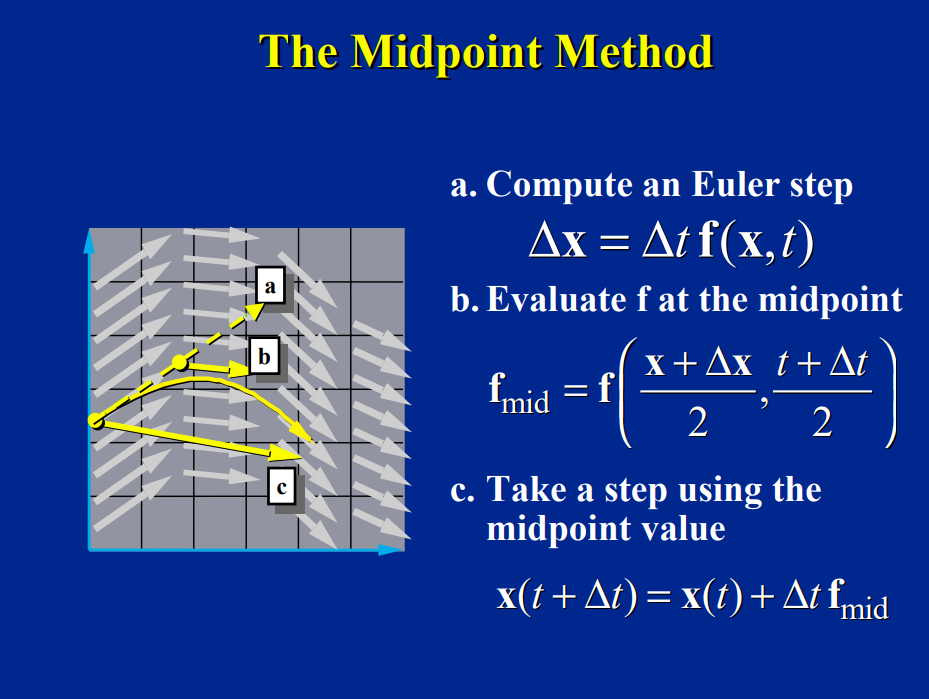
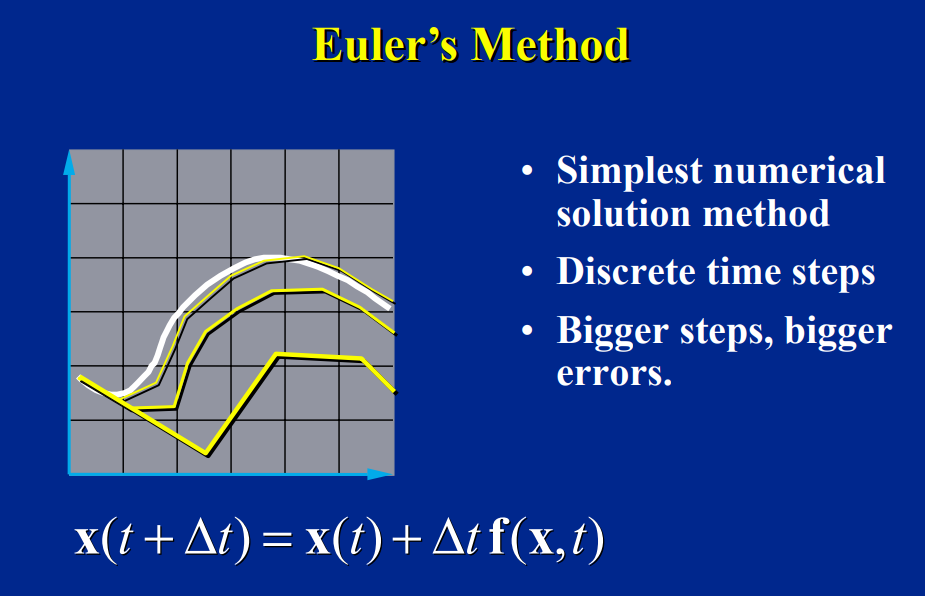
（1）实现多种力场对粒子的影响：

Gravity Forcefield，Radial ForceField, Constant ForceField, Vertical ForceField。

（2）实现两种粒子生成器的绘制：hose generator和ring generator。

（3）实现两种积分器：Euler integrator和Midpoint integrator。

**2.知识点介绍：**

1. 粒子系统动画的实现方法：
2. 使用OpenGL的定时器，每隔一定时间间隔就调用该粒子系统的integrator中的Update函数。
3. 在Update函数中根据dt和力场进行计算，更新已有粒子的位置和速度。同时调用generator的函数生成新粒子，调用particlesSet的函数去掉用完生命周期的粒子。
4. 每个粒子的位置在随着时间而变化，也就形成了动画。
5. 两种integrator：计算x(t+dt)和v(t+dt)的方式不同。
6. 不同力场下加速度的计算方法：

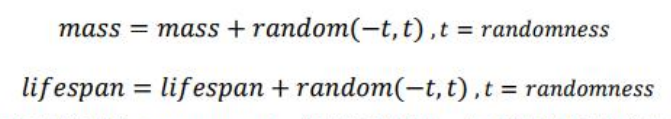
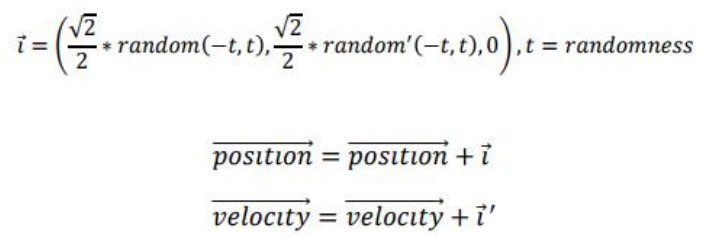
* Gravity Forcefield： a = gravity / mass;
* Radial ForceField： a=position\*magnitude/ mass\*（-1）

径向力场, 将粒子拉向原点，力的大小和到原点的距离成正比.

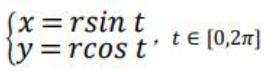
* Constant ForceField：a=force/ mass;
* Vertical ForceField：a= force(0,position.y()\*magnitude,0) / mass \*（-1）

垂直力场，将粒子拉向y轴负方向

1. 通过generator确定产生新粒子的个数、位置、速度、生命周期：

* 为了在整个仿真过程中，都保持粗略上一致的粒子数 (desired\_num\_particles），每步应产生（dt \* desired\_num\_particles / lifespan) 个粒子。
* 产生一个新粒子需要随机产生如下属性：position，velocity，mass，lifespan。每个属性都有其对应的随机参数。查找资料后，hose generator有一种产生随机粒子的公式如下：

作业中提供的random->next ()函数可以产生（0,1）之间的一个随机浮点数。

* 对于环形粒子生成器，可以考虑借助圆的参数方程，通过随机改变圆的半径和下一个点旋转的角度产生随机位置。为了保证每一圈比较均匀，还需要随着时间增加产生的粒子数，可以给原来得到新粒子数的公式上乘一个系数。

二、实现细节

**1.** **积分器：**

* Euler integrator

|  |
| --- |
| void EulerIntegrator::Update (Particle \*p, ForceField \*forcefield, float current\_time, float dt){  color = Vec3f (1, 0, 0);  Vec3f v = p->getVelocity ();  Vec3f position = p->getPosition () + v \* dt;//x(t+dt)=x(t)+v\*dt  p->setPosition (position);  float mass = p->getMass ();  Vec3f a = forcefield->getAcceleration (p->getPosition (), mass, current\_time);//a=F/m  Vec3f velocity = v + a \* dt;//v(t+dt)=v(t)+a\*dt  p->setVelocity (velocity);  p->increaseAge (dt);  } |

* Midpoint integrator

|  |
| --- |
| void MidpointIntegrator::Update (Particle \*p, ForceField \*forcefield, float current\_time, float dt){  color = Vec3f (0, 1, 0);  float mass = p->getMass ();  Vec3f a = forcefield->getAcceleration (p->getPosition (), mass, current\_time);  Vec3f v = p->getVelocity ();  Vec3f pm = p->getPosition () + v \* (dt / 2);//xm=x(t)+v\*dt/2  Vec3f vm = v + a \* (dt / 2);//vm=v(t)+a\*dt/2  Vec3f position = p->getPosition () + vm \* dt;//x(t+dt)=x(t)+vm\*dt  p->setPosition (position);  Vec3f velocity = v + forcefield->getAcceleration (pm, mass, current\_time + dt / 2)\*dt;// v(t+dt) = v(t)+ a(xm, t+dt/2) \* dt  p->setVelocity (velocity);  p->increaseAge (dt);  } |

**2.** **粒子生成器：**

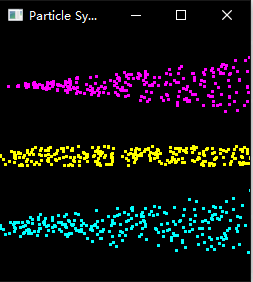
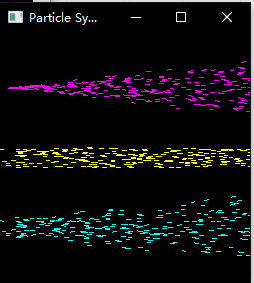
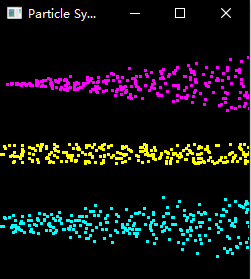
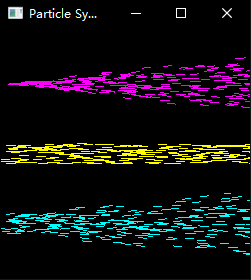
* HoseGenerator

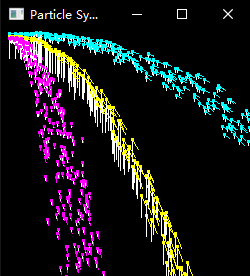
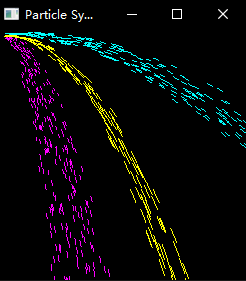
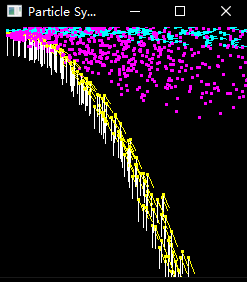
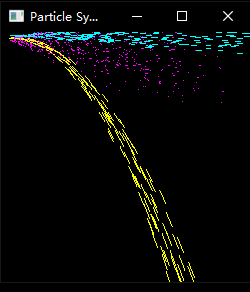
|  |
| --- |
| Particle\* HoseGenerator::Generate (float current\_time, int i){  //软管生成器：  //随机位置  Vec3f position\_random = position + Vec3f (MID\_SQRT2 \* (2 \* random->next () - 1.0f) \* position\_randomness, MID\_SQRT2 \* (2 \* random->next () - 1.0f) \* position\_randomness, 0);  //随机速度  Vec3f velocity\_random = velocity + Vec3f (MID\_SQRT2 \* (2 \* random->next () - 1.0f) \* velocity\_randomness, MID\_SQRT2 \* (2 \* random->next () - 1.0f) \* velocity\_randomness, 0);  //随机质量  float mass\_random = mass + (2 \* random->next () - 1.0f) \* mass\_randomness;  //随机生命值  float lifespan\_random = lifespan + (2 \* random->next () - 1.0f) \* lifespan\_randomness;  Particle\* particles = new Particle (position\_random, velocity\_random, color, dead\_color, mass\_random, lifespan\_random);  return particles;  } |

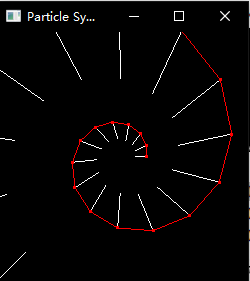
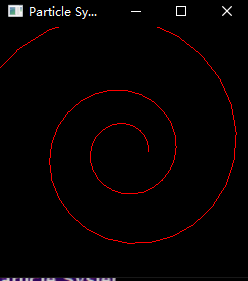
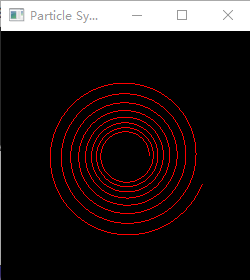
* RingGenerator

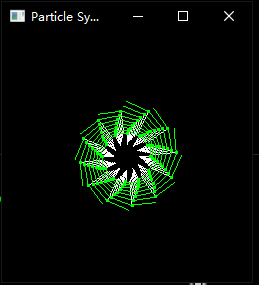
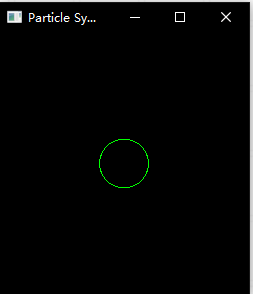
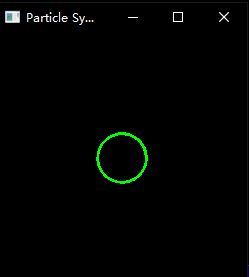
|  |
| --- |
| Particle\* RingGenerator::Generate (float current\_time, int i){  //环形生成器 :  float radius = 2.5f + (1.0f \* random->next () - 0.5f);  float theta = random->next () \* 2.0f \* M\_PI;  Vec3f position\_random=Vec3f(radius\*sinf(theta),0.0f,radius\*cosf(theta));//位置  Vec3f velocity\_random = velocity + Vec3f (0.707f \* (2 \* random->next () - 1.0f) \* velocity\_randomness, 0.707f \* (2 \* random->next () - 1.0f) \* velocity\_randomness, 0.0f);//随机速度  float mass\_random=mass+(2 \* random->next () - 1.0f) \* mass\_randomness;//随机质量  float lifespan\_random=lifespan+(2\*random->next()-1.0f)\*lifespan\_randomness; Particle\* particles = new Particle (position\_random, velocity\_random, color, dead\_color, mass\_random, lifespan\_random);  return particles;  } |

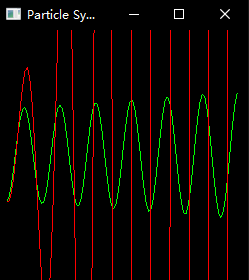
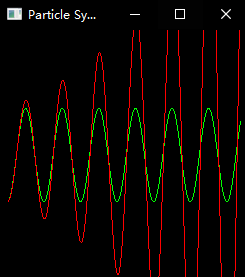
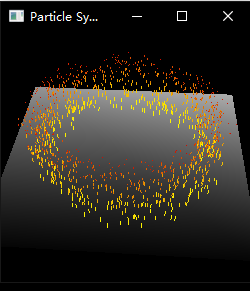
三、结果展示

1.Hose generator和motion blur测试：

2. Hose generator和力场测试：

3.Ring generator 和Euler integrator方式测试：

4. Ring generator 和Midpoint integrator方式测试：

5.以上两种integrator的其他测试：

四、心得体会

当根据时间来改变很多个质点的位置和速度之后，就可以形成粒子动画。而加上各种力场、各个属性的随机值，可以产生千变万化的效果！