

Homework 03

Particle Transport

David C. Banks

Electrical Engineering and Computer Science
University of Tennessee

2009

Textbook

Chapter 14 Sampling
Chapter 20 Light

Baseball

A pitcher throws a baseball toward home plate. The batter swings and misses. The pitcher throws again. The batter hits the ball toward left field.

The pitcher is the “source” and the plate is the “target.” The pitching mound is a 2D disk. The ball is released at different points and with different velocities from within the disk. The ball travels toward area surrounding the plate, then exits the batter’s box with a particular velocity. If the ball is hit, it may be fouled behind the catcher (forward-scattered) or put into play (back-scattered). If the batter is struck, the ball is absorbed.

Gravity

A cloud of interplanetary debris ejects one comet at a time toward a star cluster. Sometimes a comet passes through the cluster. Sometimes it enters into orbit.

In the 2D version, the comet cloud is a disk. The comets are particles with mass shot from random locations in the cloud, in randomized directions toward the target. The target is a set of stationary particles in another disk. The comet may forward-scatter through the cluster, back-scatter from the cluster, or be absorbed by the cluster.

Golf

A golfer putts a ball toward the cup. The ball may miss the cup (no scattering) or may lip out, continuing past the cup (forward scattering) or around toward the golfer (back scattering). If the ball goes in the cup, it is absorbed.

Tennis

Player A serves the tennis ball to player B. Player B may return the ball to player A (back scattering), may fail to reach the ball (no scattering), or may mis-hit with the ball glancing off the edge of the racket and continuing past player B (forward scattering). If the ball goes into the net, it is absorbed.

Light

A photon is emitted from a luminaire. It travels in some direction until it reaches a target of atoms. The photon may pass through the atoms (no scattering) or may be absorbed. The particles may re-radiate a photon roughly in the direction it was traveling (forward scattering) or toward the luminaire (back scattering).

Traffic

A car travels toward an intersection. It may pass through (no scattering), turn left or right or make a u-turn (scattering). It may be stopped by a collision or by a red light (absorption).

Third person shooter

A blaster shoots at an alien. The shot misses (no scattering) or is deflected by the alien's shield (back scattering) or is absorbed by the alien.

Simple game engine

Shootem: a 3rd person remote-controlled shooter game

Write a game *shootem* that emits particles from a source at a target and measures the outcome.

Particle transport

Shoot particles at a target and collect the results

- Probe particles come from a disk

- 1 particle at a time

Target particles lie in a disk

- Target particles don't move

Detector surrounding target prints velocity in, out

- New probe particle released after old particle exits detector

- Particle is "absorbed" if it takes too long to emerge

File format

Example: experiment1.prt

Particle2D

```
pType { id Probe area 10 gray 230 mass 1.0 update yes }
```

```
pType { id Target area 20 gray 130 mass 1.0 update no }
```

```
particle { id Probe samples 100 x 100 y 100 positionRadius  
10 vx 0 vy 100 velocityRadius 10 }
```

```
particle { id Target samples 10 x 100 y 200 positionRadius  
10 }
```

```
sensor { detect Probe x 100 y 200 radius 50 absorbTime 2.0 }
```

Point in a square

Random sample of position in unit interval

```
x = drand48();
```

Point in a square

Random sample of position in unit-radius interval

```
x = drand48();
```

```
x = 2.0*x;
```

```
x = x - 1.0;
```

Point in a square

Random sample of position in square

```
x = drand48();  
y = drand48();
```

Point in a square

Random sample of position in unit-radius square

```
x = drand48();  
y = drand48();  
x = 2.0*x - 1.0;  
y = 2.0*y - 1.0;
```

Point in a disk

Non-uniform random sample of position in disk

```
x = drand48()*2.0-1.0;
y = drand48()*2.0-1.0;
if ( fabs(x) > fabs(y) )
{
    max = fabs(x);
    min = fabs(y);
}
else
{
    max = fabs(y);
    min = fabs(x);
}
length =  $\sqrt{x^2 + y^2}$ ;
x *= max/length;
y *= max/length;
```

Point in a disk

Non-uniform random sample of position in disk

```
r = drand48();  
 $\theta = \text{drand48()} * 2.0 * \pi;$   
x = r*cos( $\theta$ );  
y = r*sin( $\theta$ );
```

Point in a disk

Uniform random sample of position in disk

```
int pointInDisk = 0;
while ( !pointInDisk ) /* rejection test */
{
  x = drand48()*2.0-1.0;
  y = drand48()*2.0-1.0;
  pointInDisk = (  $x^2 + y^2 \leq 1.0$  );
}
x = x*radius + offsetX;
y = y*radius + offsetY;
```

Point in a disk

Difference dp from point $p=(x,y)$ to disk center $c=(cx,cy)$

Compare length $|dp|$ to radius of disk

```
dp = p-c;  
/* dp.x = p.x - c.x; */  
/* dp.y = p.y - c.y; */  
if ( length(dp) <= radius ) pointInDisk = 1;  
else pointInDisk = 0;
```

Point in a disk

Difference dp from point $p=(x,y)$ to disk center $c=(cx,cy)$

```
dp = p-c;  
pointInDisk = ( length(dp) <= radius );
```

Sensor

Detecting a particle

Does a particle enter the sensor disk?

Does a particle exit the sensor disk?

```
pointIsIn = pointInDisk ( particle.position, disk );  
if ( !pointWasIn && pointIsIn ) printVelocity(particle);  
if ( pointWasIn && !pointIsIn ) printVelocity(particle);  
pointWasIn = pointIsIn;
```

Sensor

Detecting a particle

Particle's position, velocity at boundary

```
printf ( "particle_{_id_%s_x_%f_y_%f_vx_%f_vy_%f}_\n",  
        particle.id,  
        particle.position.x - sensorCenter.position.x,  
        particle.position.y - sensorCenter.position.y,  
        particle.velocity.x, particle.velocity.y);
```

Sensor

Waiting for a particle to leave sensor disk

Particle has speed $s = |\text{velocity}|$

Distance traveled $d \approx s \cdot dt$

Sensor has diameter $2 \cdot r$

```
d += s*dt;  
if ( d/(2.0*r) > absorbTime ) /* particle is absorbed */
```

Programming components

File syntax and semantics

1. **pType** update boolean

if (update==1) { update position and velocity }

2. **particle** number int

Generate number of sample points within disk

New particle is created when old one exits sensor

3. **particle** positionRadius float

4. **particle** velocityRadius float

5. **sensor** detect char[]

6. **sensor** x float y float

7. **sensor** radius float

Print position and velocity on entry, exit

Position relative to sensor's center

Webpage

Create scene files

Vary parameters

- 1, 10, 100, 1000 target particles

- 5 values of gravity G : 2 negative, 2 positive, zero

Image of scene on webpage

- Draw disk in debug mode

Record animations of probe particles scattering from the target

- Forward, back, absorb