Particle Systems

Computer Animation and Visualisation
Lecture 8
Taku Komura
Overview

• Particle System
  – Modelling fuzzy objects (fire, smoke)
  – Modelling liquid
  – Modelling cloth
  – Integration:
    – implicit integration,
    – Verlet integration
  – Rigid body dynamics by particle systems
Particle Systems

• Modelling objects by a number of small particles

• We can model fuzzy objects such as
  – fire,
  – fireworks
  – clouds,
  – smoke,
  – water, etc.

• But we can also model
  – cloth
  – hair
  – rigid objects
Particle System

• Initially developed by Reeves to create scenes for Star Trek II.

http://www.youtube.com/watch?v=13b7TSiidaM&feature=channel_page
Reeves ’83,’85
Particle Dynamics

- A particle's position in each succeeding frame can be computed by its velocity.
- This can be modified by an acceleration force for more complex movements, e.g., gravity simulation.

\[ x_{\text{new}} = x_{\text{old}} + v_{\text{old}} \Delta t \]
\[ v_{\text{new}} = v_{\text{old}} + a \Delta t \]
\[ ma = F \]
Modelling fuzzy objects with particle system

For each frame of an animation sequence the following steps are performed:

1. New particles are generated
2. Each new particle is assigned its own set of attributes
3. Any particles that have existed for a predetermined time are destroyed
4. The remaining particles are transformed and moved according to their dynamic attributes
5. An image of the remaining particles is rendered
For Modelling Fire and Smoke

- Particles are generated using stochastic methods.
- The designer controls the mean number of particles generated per frame and the variance.
- So the number of particles generated at frame $F$ is:

$$N_{partsF} = N_{mean} + \text{rand()} \cdot \text{Variance}$$

$$-1.0 \leq \text{rand()} \leq 1.0$$  a uniformly distributed random number
Particle Attributes

Each new particle has the following attributes:

– initial position
– initial velocity (speed and direction)
– initial size
– initial colour
– initial transparency
– shape
– lifetime
Modelling Water

• To model liquid, viscosity and friction has to be taken into account.

• The external forces can be divided into components:
  – adhesion forces: attract particles to each other and to other objects,
  – viscosity forces: dampen movement of particles in contact with objects in the environment
  – friction forces: limit shear movement of particles in the liquid.
Adhesion Force

Attract particles to each other and to other objects

The adhesion force between (left) honey-honey, (middle) honey-ceramic and (right) non-mixing liquid

Figure 2: Examples of adhesion functions used in our system. The function domains are specified in particle radius units so that a simulation can be run with larger or smaller numbers of particles by just changing the standard particle radius. Generally, we define adhesion functions with a support of 4 particle radii.
Viscosity Force

- Limit shear movement of particles in the liquid
- The momentum between the neighbour particles are exchanged

\[ P_{\text{exch}} = \frac{v \cdot k(d_{pn})(P_p - P_{pn})\Delta t}{2 \sum_{n=1}^{n} k(d_i)} \]

\( v \): viscosity coefficient
\( k() \): A weighting function (Gaussian)
\( d_i \): distance from the processed particle
\( P_p, P_{pn} \): momentum of \( p \) and \( pn \)
\( P_{\text{exch}} \): momentum exchanged

\[ P_p = P_p + P_{\text{exch}} \]
\[ P_{pn} = P_{pn} - P_{\text{exch}} \]
Friction Force

– Dampen movement of particles in contact with objects in the environment.
– Simply scale down the velocity by a constant value

\[ \hat{v}_t = f v_t \]

– Or we can compute the penetration depth, and adjust the force according to the depth  (explained later)
Rendering liquid

- Once the particle positions have been determined, next the scene must be rendered.
- There are two categories of methods to do this:
  - rendering individual particles, or
  - rendering the surface of the object the particles
- Rendering individual particles works well for modelling waterfalls or spray
- A surface rendering will give a more accurate description but is generally much slower

http://www.youtube.com/watch?v=n5lOjME8B6M
http://www.youtube.com/watch?v=isXNkTiiAYQ
Rendering Liquid by Blobs (metaballs)

- Create an implicit function similar to the electron density maps

\[ D(x, y, z) = \sum b_i \exp(-a_i r_i) \]

\[ r_i = \sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2} \]

\((x_i, y_i, z_i)\): the center of the particle i

\(F(x, y, z) = D(x, y, z) - T\)

- We can define the surface where

\(F(x, y, z) = 0\)

Blinn ‘92
Advantages of simulating fuzzy objects by particle systems

• Complex systems can be created with little human effort.
• The level of detail can be easily adjusted.
  – If a particle system object is in the distance, then it can be modelled to low detail (few particles)
  – If it is close to the camera, then it can be modelled in high detail (many particles)
Cloth Simulation

- Need to generate animations of cloth of the characters appearing in the games
- Can use particles for this purpose
Cloth

- We model the cloth by mass-spring models (particles connected with springs)
  - Mass pulled by the interconnecting springs
Hair

- Linear set of particles
- Length structural force
- Deformation forces proportional to the angle between segments

http://www.youtube.com/watch?v=gj2UdZ-G0vg&feature=related
Integration for Simulation

Euler Method

- $h$: time step
- $M$: mass matrix,
- $f_o$: external force
- $\Delta x$: pos, vel of particles

\[
\begin{pmatrix}
\Delta x \\
\Delta v
\end{pmatrix}
= \Delta t
\begin{pmatrix}
v_0 \\
M^{-1}f_0
\end{pmatrix}
\]
Problem with Euler Method

- When the time step is too large, the system can blow off (diverge) very quickly
Implicit Integration (Baraff ’98)

- Very stable
- Assuming the force in the next time step also results in the velocity increment at this time step
- Need to calculate $\Delta x \Delta v$ that satisfies this equation

$$\begin{pmatrix} \Delta x \\ \Delta v \end{pmatrix} = \Delta t \begin{pmatrix} v_0 + \Delta v \\ M^{-1} f (x_0 + \Delta x, v_0 + \Delta v) \end{pmatrix}$$
Taylor series expansion of the force

\[ f(x_0 + \Delta x_0, v_0 + \Delta v) = f_0 + \frac{\partial f}{\partial x} \Delta x + \frac{\partial f}{\partial v} \Delta v \]

- Substituting this into the last equation, we get

\[
\begin{pmatrix}
\Delta x \\
\Delta v
\end{pmatrix} = \Delta t \begin{pmatrix}
v_0 + \Delta v \\
M^{-1} (f_0 + \frac{\partial f}{\partial x} \Delta x + \frac{\partial f}{\partial v} \Delta v)
\end{pmatrix}
\]
Finally, computing the velocity update

- Substituting $\Delta x = \Delta t (v_0 + \Delta v)$

We can finally compute $\Delta v$ by

$$
\Delta v = \frac{\Delta t M^{-1} \left( f_0 + \Delta t \frac{\partial f}{\partial x} v_0 \right)}{\left( I - \Delta t M^{-1} \frac{\partial f}{\partial v} - \Delta t^2 M^{-1} \frac{\partial f}{\partial x} \right)}
$$
Advantages

• We can update $\Delta v$ by using large time steps ($h$)
• Do not have to worry about blowing off

Drawbacks

• The motions can be smoothened too much
Verlet integration (Jakobsen ’02)

\[
x' = 2x - x^* + a \Delta t^2
\]

\[
x^* = x
\]

Directly computing the next position from the acceleration

- \(x\) : current position
- \(x'\) : updated position
- \(x^*\) : The position in the previous step
- \(\Delta t\) : time step,
- \(a\) : acceleration
- Used intensely when simulating molecular dynamics. It is quite stable since the velocity is implicitly given
- Directly computing the next location from the current acceleration
Advanced Character Physics (ACP): Multibody dynamics by particle system
Jakobsen, GDC `02

- Simulating articulated objects such as human bodies, robotic arms, cars by particle systems
- Easy to implement
  - But slow when the particle number is very large
ACP: How does it work?

- Model rigid bodies by a collection of particles
- Assume the distance between the particles will remain the same if they belong to the rigid body
- Each particle will move based on the free particle dynamics
- The 3D position will be modified according to the distance constraints
ACP: Keeping the distance between the particles the same

- First integrate the position of the particles using the velocity and acceleration information
- To keep constant lengths of the links between points, the two particles are simply translated along the line connecting them offline

\[
\begin{align*}
\Delta x_1 &= \frac{m_1^{inv}}{m_1^{inv} + m_2^{inv}} \Delta d \\
\Delta x_2 &= -\frac{m_2^{inv}}{m_1^{inv} + m_2^{inv}} \Delta d
\end{align*}
\]
ACP: Modeling joints

It is possible to connect multiple rigid bodies by hinge joints or pin joints. Simply let two rigid bodies share a particle, and they will be connected by a pin joint. Share two particles, and they are connected by a hinge.

A 3DOF pin joint (left) and 1DOF hinge joint (right)
ACP: Setting joint limits

• A method for restraining angles is to satisfy a dot product constraint:

\[(x_2 - x_0) \cdot (x_1 - x_0) < \alpha.\]

• If the constraint is about to be violated, another rod is inserted between \(x_1\) and \(x_2\)
ACP: Modeling Characters

- We can model human characters by this system
- Used for ragdoll physics in Hitman

Figure 9. The particle/stick configuration used in Hitman for representing the human anatomy
ACP: Collision detection

• The collision detection will be done based on the points and the surfaces

• The colliding particle will be pushed out from the penetrating area

• The velocity of the colliding particle will be reduced to zero
ACP: Friction Force

- According to the Coulomb friction model, friction force depends on the size of the normal force between the objects in contact.
- The penetration depth $dp$ can be first measured when a penetration has occurred (before projecting the penetration point out of the obstacle),
- and then, the tangential velocity $v_t$ is reduced by an amount proportional to $dp$ (the proportion factor being the friction constant).
Readings

- Baraff and Witkin, “Large Steps in Cloth Simulation”, SIGGRAPH 98
- Stam, “Stable Fluids”, SIGGRAPH 99