Principles of Computer Game Design and Implementation

Lecture 17
We already learned

• Collision response
  – Newtonian mechanics
    • An application of Newtonian dynamics in targeting
  – Collision recipe
    • Ball-plain bouncing problem
Demo
Outline for today

• Collision recipe
  – Ball-ball collision problem

• Other physics simulation
  – rigid-body physics, soft-body physics, fluid mechanics, etc.

• A few examples for assignment 1
Ball-Ball Collision Recipe

• First, consider 1D case
  – No roll
  – No friction
  – No energy loss

• Then 3D

Elastic collision
1D Ball-Ball Collision Laws

- **Impulse conservation**

  \[
  m_1 V_1 + m_2 V_2 = m_1 V'_1 + m_2 V'_2
  \]

- **Energy conservation**

  \[
  \frac{m_1 V_1^2}{2} + \frac{m_2 V_2^2}{2} = \frac{m_1 V'_1^2}{2} + \frac{m_2 V'_2^2}{2}
  \]
1D Ball-Ball Collision: Different Masses

- Can be solved

\[ V_1' = \frac{V_1(m_1 - m_2) + 2m_2V_2}{m_1 + m_2} \]

\[ V_2' = \frac{V_2(m_2 - m_1) + 2m_1V_1}{m_1 + m_2} \]
1D Ball-Ball Collision: Same Mass

• If the balls have same mass (e.g. billiard balls)

\[ V'_1 = V_2 \]
\[ V'_2 = V_1 \]

Examples:

- \( V_1 = 10\text{mph}, \ V_2 = 0 \)
- \( V_1 = 10\text{mph}, \ V_2 = -10\text{mph} \)
- \( V_1 = 10\text{mph}, \ V_2 = 3\text{mph} \)
- \( V'_1 = 0, \ V'_2 = 10\text{mph} \)
- \( V'_1 = -10\text{mph}, \ V'_2 = 10\text{mph} \)
- \( V'_1 = 3\text{mph}, \ V'_2 = 10\text{mph} \)

Negative speed means that the ball moves from right to left.
Ball-Ball Inter Penetration

- $V_1 = 10\text{mph}$, $V_2 = -10\text{mph}$  \hspace{1cm} $V'_1 = -10\text{mph}$, $V'_2 = 10\text{mph}$
- $V_1 = -10\text{mph}$, $V_2 = 10\text{mph}$  \hspace{1cm} $V'_1 = 10\text{mph}$, $V'_2 = -10\text{mph}$
- $V_1 = 10\text{mph}$, $V_2 = -10\text{mph}$  \hspace{1cm} $V'_1 = -10\text{mph}$, $V'_2 = 10\text{mph}$
- $V_1 = -10\text{mph}$, $V_2 = 10\text{mph}$  \hspace{1cm} $V'_1 = 10\text{mph}$, $V'_2 = -10\text{mph}$

Move nowhere!
Ball-Ball Collision: Better Solution

• If \((V_1 - V_2 > 0)\)

\[ V_1' = V_2 \quad V_2' = V_1 \]

• Else no change in velocities

\[ V_1 = 10\text{mph}, \; V_2 = -10\text{mph} \quad V_1' = -10\text{mph}, \; V_2' = 10\text{mph} \]

\[ V_1 = -10\text{mph}, \; V_2 = 10\text{mph} \quad V_1' = 10\text{mph}, \; V_2' = -10\text{mph} \]

\[ V_1 = 10\text{mph}, \; V_2 = -10\text{mph} \quad V_1' = -10\text{mph}, \; V_2' = 10\text{mph} \]

\[ V_1 = -10\text{mph}, \; V_2 = 10\text{mph} \quad V_1' = 10\text{mph}, \; V_2' = -10\text{mph} \]
3D Ball-Ball Collision (Same Mass)

- Collision does not change the parallel component of velocity

\[
N = \frac{1}{\|P_2 - P_1\|}(P_2 - P_1)
\]

\[
V_{1N} = (N \cdot V_1)N \quad V_{2N} = (N \cdot V_2)N
\]

\[
V_{1||} = V_1 - V_{1N} \quad V_{2||} = V_1 - V_{2N}
\]
Recall: Projection

- So,

\[ \text{proj}_v \mathbf{w} = \frac{\mathbf{w} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}} \mathbf{v} \]

- If \( \mathbf{v} \) is already normalized (often the case), then becomes

\[ \text{proj}_u \mathbf{w} = (\mathbf{w} \cdot \mathbf{u}) \mathbf{u} \]
3D Ball-Ball Collision (Same Mass)

- Collision does not change the parallel component of velocity

\[
N = \frac{1}{\|P_2 - P_1\|} (P_2 - P_1)
\]

\[
V_{1N} = (N \cdot V_1) N \quad \quad V_{2N} = (N \cdot V_2) N
\]

\[
V_{1\parallel} = V_1 - V_{1N} \quad \quad V_{2\parallel} = V_1 - V_{2N}
\]

\[
V'_{1N} = (N \cdot V_2) N \quad \quad V'_{2N} = (N \cdot V_1) N
\]

\[
V'_{2} = V'_{1N} + V_{1\parallel} \quad \quad V'_{2} = V'_{2N} + V_{2\parallel}
\]
Same Mass Ball-Ball Collision jME code

...  
if(...) {
    Vector3f n = ball2.getLocalTranslation().subtract(ball1.getLocalTranslation()).normalize();
    float proj1V = velocity1.dot(n);
    float proj2V = velocity2.dot(n);
    Vector3f tan1 = velocity1.subtract(n.mult(proj1V));
    Vector3f tan2 = velocity2.subtract(n.mult(proj2V));
    if(proj1V - proj2V > 0) {
        velocity1 = tan1.add(n.mult(proj2V));
        velocity2 = tan2.add(n.mult(proj1V));
    }
}
...

Penetration Handling
Recall: Main Loop

Naïve approach:

```c
for(i=0;i<num_obj-1;i++)
    for(j=i+1;j<num_obj;j++)
        if(collide(i,j)){
            react;
        }
```

- **Issues:**
  - How
  - Can be **very** slow
Simple Newtonian Mechanics

• Accurate physical modelling can be quite complicated

• We considered simplest possible behaviours
  – Particle motion
  – Ball-plain and ball-ball collision
    • No friction, no properties of materials
Other Example: Box-Box collision

Boxes can interact in a number of ways

- Point–face contact
- Point–edge contact
- Edge–edge contact
- Edge–face contact
- Point–point contact

Hard to achieve a realistic behaviour without considering rotation, deformation, friction
Other Physical Simulations

• Rigid body (no deformation) physics
  – Rotation, friction, multiple collisions
  – Joints and links
    • Ragdoll physics
More Physics

• Soft body physics (shapes can change)
  – Cloth, ropes, hair

• Fluid dynamics
Putting It All Together

• Combine all aspects of a physical model

- Force/Torque generators
- Rigid-body update (Integrator)
- Contact generator (possibly with plug-in code)
- Create contacts

- Apply forces and torques
- Write integrated point and velocity
- Write postcollision position and velocity
- Contact resolution


• Use hardware acceleration
Decoupling Physics and Graphics

• What if we need physics simulation for something not shown?
• E.g. reconsider the targeting problem

Drag acts on the projectile
What Can We Do

• Euler steps give us the updated entity position based on the interaction with other entities and forces

• Analytical solution can be difficult to obtain
  – Quadratic drag?
  – Wind?
  – Rocket-propelled grenade?
Interactive Approach

• Compute the initial velocity as if there is no drag, wind, thrust,... (or simply pick a value)

• While not hit sufficiently close, repeat
  – Use Euler steps to see where it gets
  – If overshot, reduce speed
  – If undershot, increase speed

Fun to watch, but does it solve our problem?