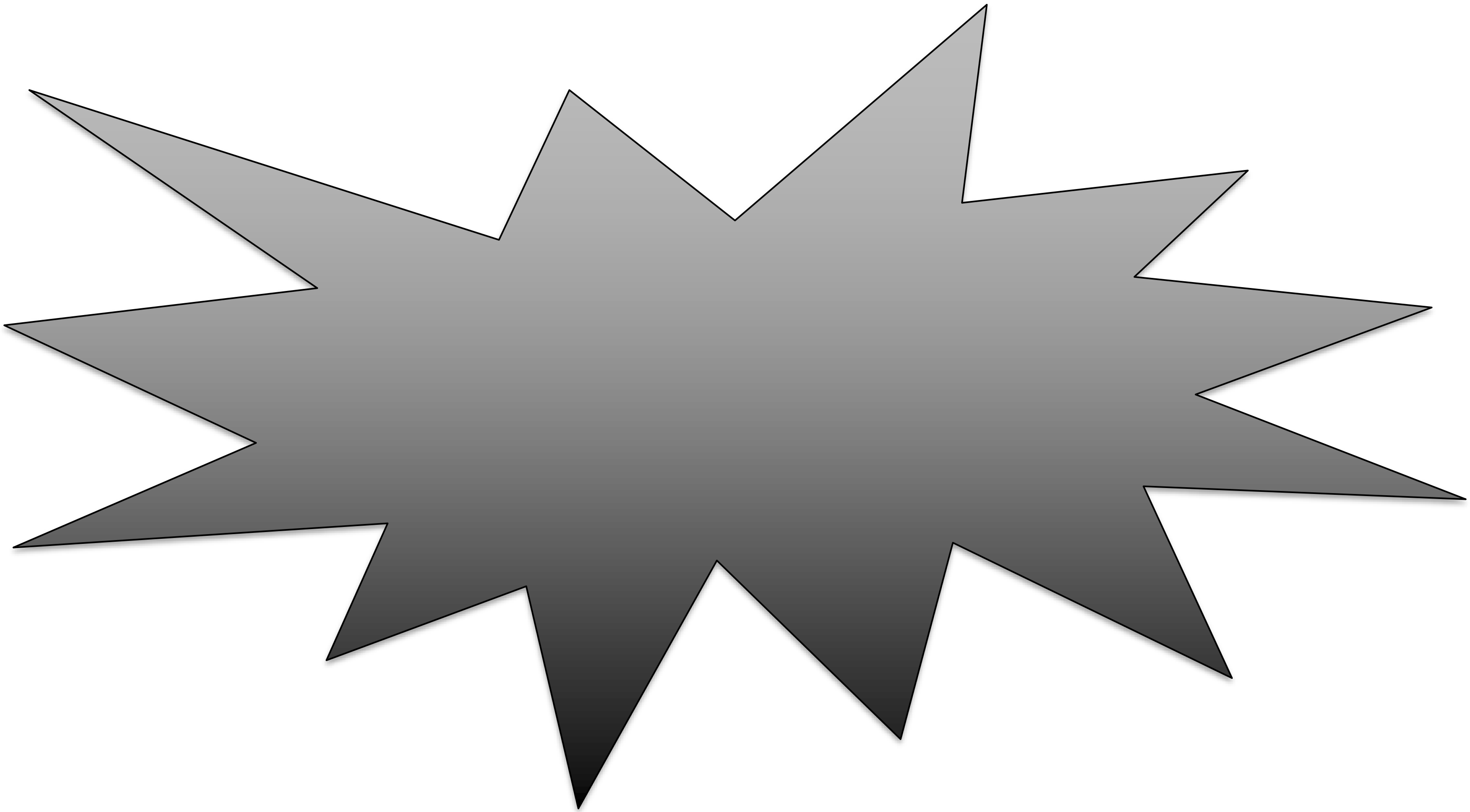
Principles of Computer Game Design and Implementation

Lecture 17

We already learned

- Collision response
  - Newtonian mechanics
    - An application of Newtonian dynamics in targeting
  - Collision recipe
    - Ball-plain bouncing problem

Demo

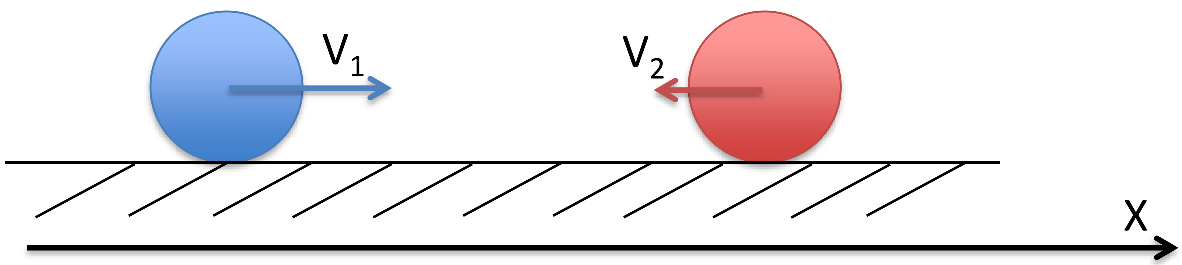


Outline for today

- 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

Elastic collision

- Then 3D

1D Ball-Ball Collision Laws

- Impulse conservation

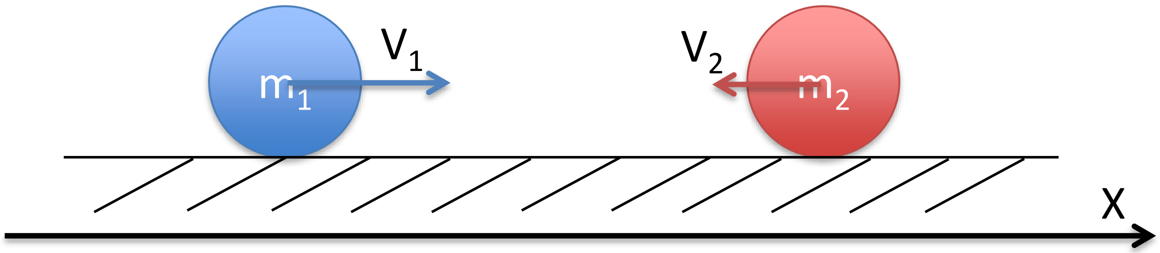
Before collision

After collision

$$m_1 V_1 + m_2 V_2 = m_1 V_1' + m_2 V_2'$$

- Energy conservation

$$\frac{m_1 V_1^2}{2} + \frac{m_2 V_2^2}{2} = \frac{m_1 {V_1'}^2}{2} + \frac{m_2 {V_2'}^2}{2}$$

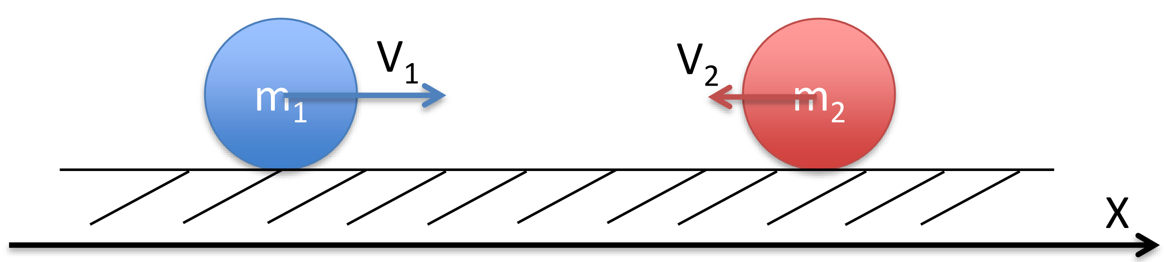


1D Ball-Ball Collision: Different Masses

- 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

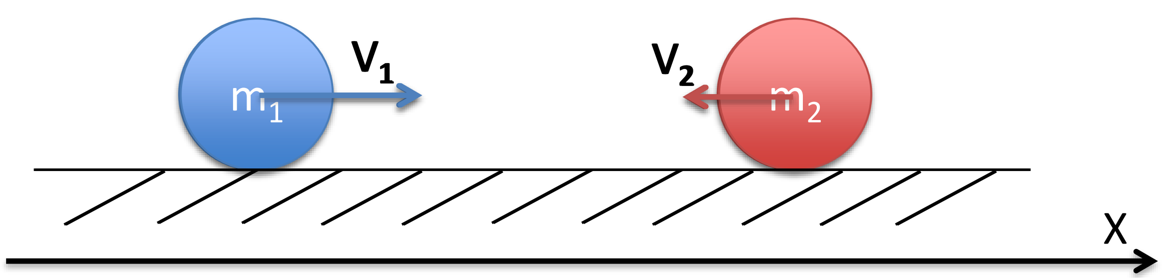
- If the balls have same mass (e.g. billiard balls)

$$V_1' = V_2 \qquad V_2' = V_1$$

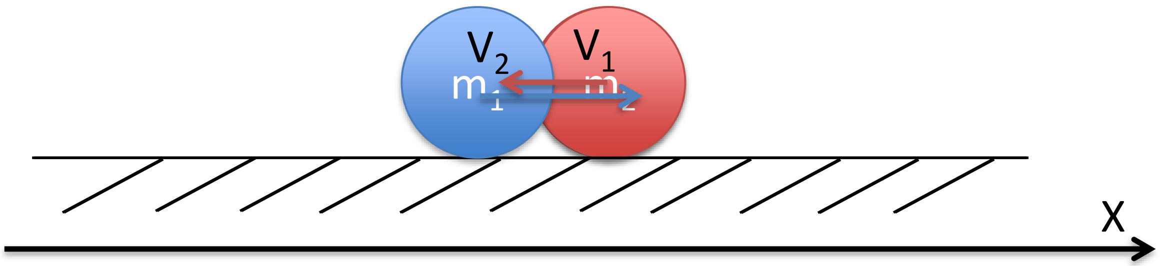
Examples:

V_1 = 10mph, V_2 = 0	V'_1 = 0, V'_2 = 10mph
V_1 = 10mph, V_2 = -10mph	V'_1 = -10mph, V'_2 =10mph
V_1 = 10mph, V_2 = 3mph	V'_1 = 3mph, V'_2 =10mph

Negative speed means that the ball moves from right to left



Ball-Ball Inter Penetration



- V_1 = 10mph, V_2 = -10mph $\qquad$ V'_1 = -10mph, V'_2 =10mph
- V_1 = -10mph, V_2 = 10mph $\qquad$ V'_1 = 10mph, V'_2 =-10mph
- V_1 = 10mph, V_2 = -10mph $\qquad$ V'_1 = -10mph, V'_2 =10mph
- V_1 = -10mph, V_2 = 10mph $\qquad$ V'_1 = 10mph, V'_2 =-10mph

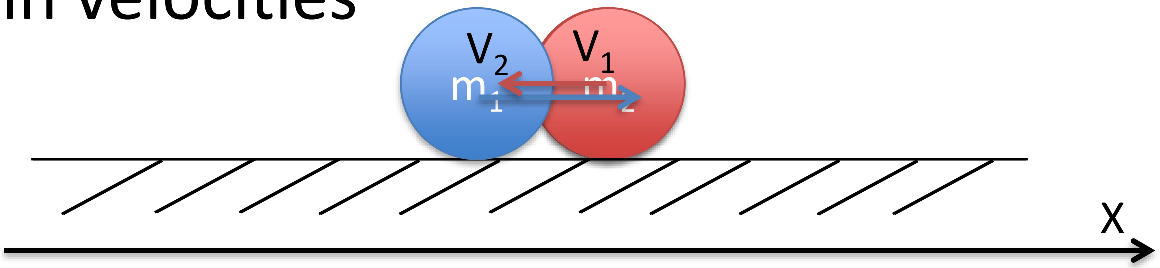
Move nowhere!

Ball-Ball Collision: Better Solution

- If $(V_1 - V_2 > 0)$

$$V_1' = V_2 \qquad V_2' = V_1$$

- Else no change in velocities



- V_1 = 10mph, V_2 = -10mph V'_1 = -10mph, V'_2 =10mph
- V_1 = -10mph, V_2 = 10mph V'_1 = 10mph, V'_2 =-10mph
- V_1 = 10mph, V_2 = -10mph V'_1 = -10mph, V'_2 =10mph
- V_1 = -10mph, V_2 = 10mph V'_1 = 10mph, V'_2 =-10mph

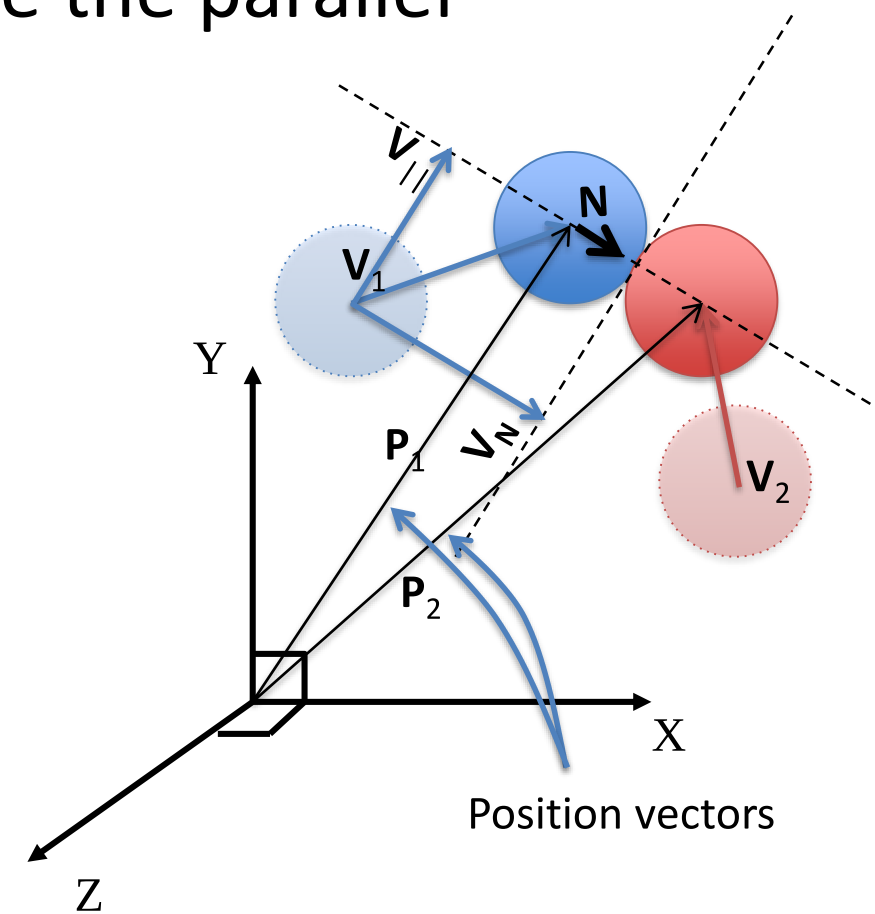
3D Ball-Ball Collision (Same Mass)

- Collision does not change the parallel component of velocity

$$\mathbf{N} = \frac{1}{\|\mathbf{P}_2 - \mathbf{P}_1\|}(\mathbf{P}_2 - \mathbf{P}_1)$$

$\mathbf{V}_{1N} = (\mathbf{N} \bullet \mathbf{V}_1)\mathbf{N}$ $\qquad$ $\mathbf{V}_{2N} = (\mathbf{N} \bullet \mathbf{V}_2)\mathbf{N}$

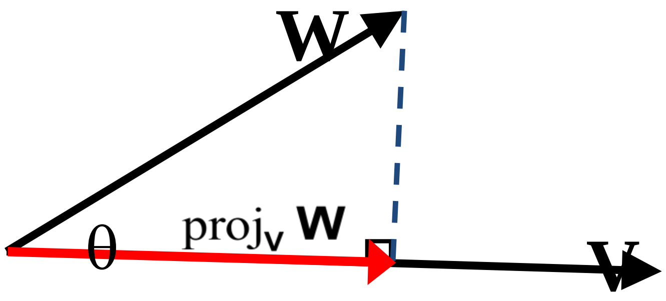
$\mathbf{V}_{1||} = \mathbf{V}_1 - \mathbf{V}_{1N}$ $\qquad$ $\mathbf{V}_{2||} = \mathbf{V}_1 - \mathbf{V}_{2N}$



Recall: Projection

- So,

$$\mathrm{proj}_{\mathbf{V}}\,\mathbf{W} = \frac{\mathbf{W} \bullet \mathbf{V}}{\mathbf{V} \bullet \mathbf{V}}\mathbf{V}$$

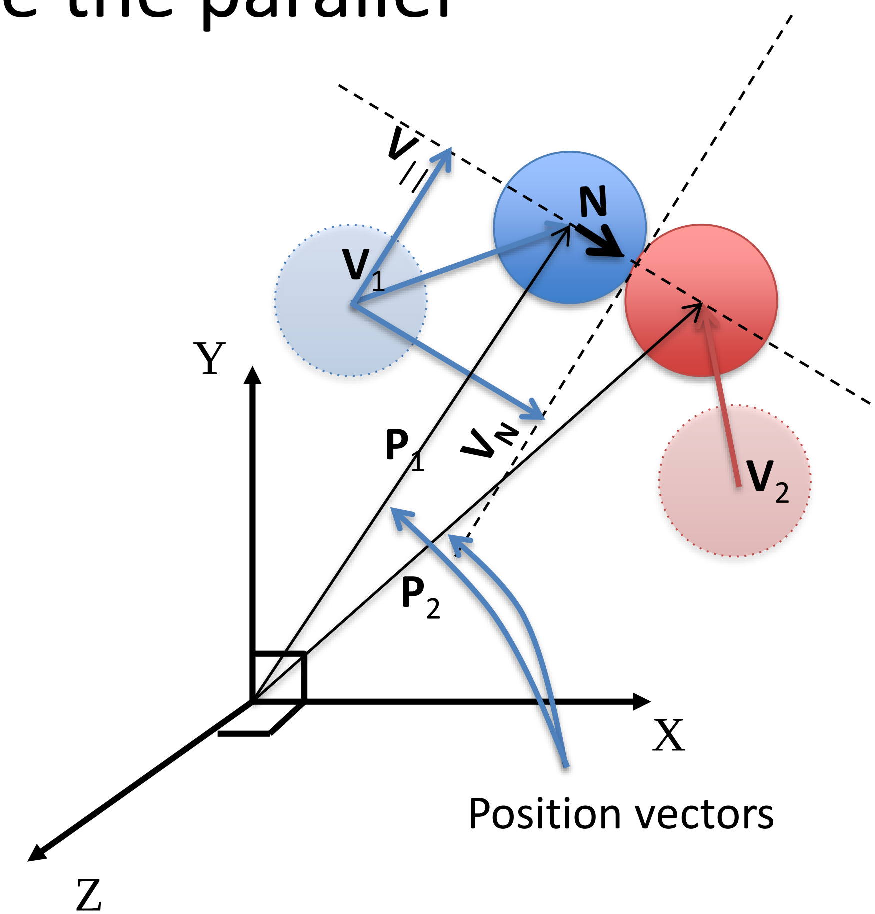


- If **V** is already normalized (often the case), then becomes

$$\mathrm{proj}_{\mathbf{U}}\,\mathbf{W} = (\mathbf{W} \bullet \mathbf{U})\mathbf{U}$$

3D Ball-Ball Collision (Same Mass)

- Collision does not change the parallel component of velocity



$$\mathrm{N} = \frac{1}{\|\mathbf{P}_2 - \mathbf{P}_1\|}(\mathbf{P}_2 - \mathbf{P}_1)$$

$\mathbf{V_{1N} = (N \bullet V_1)N}$

$\mathbf{V_{1||} = V_1 - V_{1N}}$

$\mathbf{V_{2N} = (N \bullet V_2)N}$

$\mathbf{V_{2||} = V_1 - V_{2N}}$

$\mathbf{V'_{1N} = (N \bullet V_2)N}$

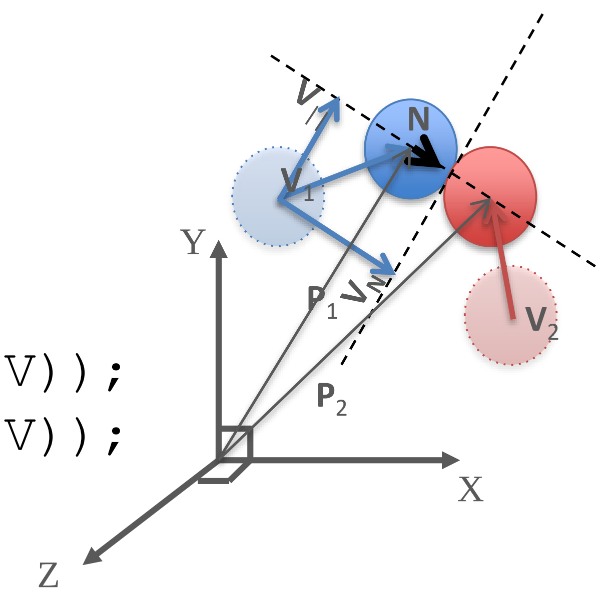
$\mathbf{V'_2 = V'_{1N} + V_{1||}}$

$\mathbf{V'_{2N} = (N \bullet V_1)N}$

$\mathbf{V'_2 = V'_{2N} + V_{2||}}$

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));
  }
}
…
```



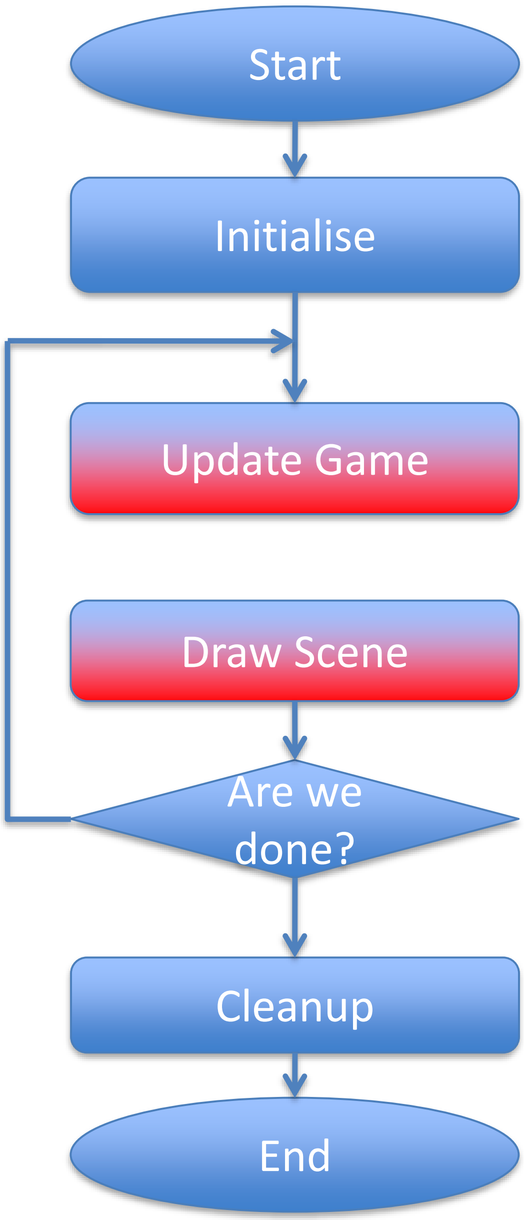
Penetration Handling

Recall: Main Loop

Naïve approach:

```
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

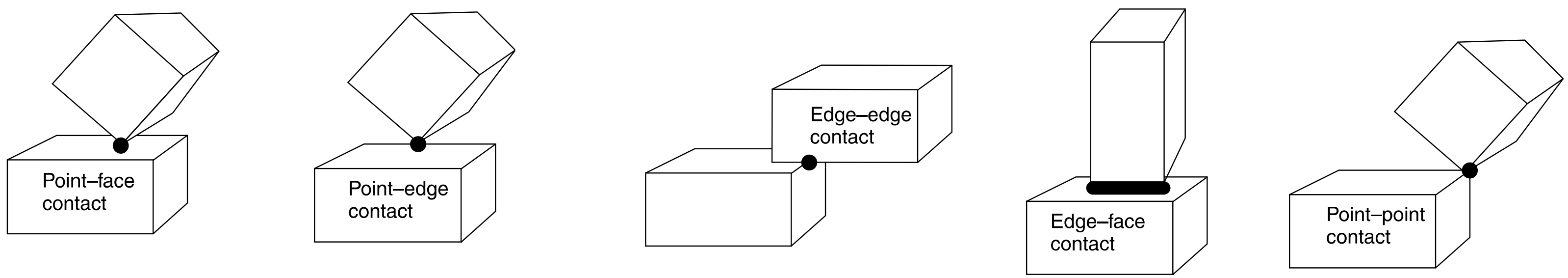


Simple Newtonian Mechanics

- 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

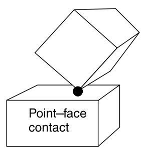
Other Example: Box-Box collision

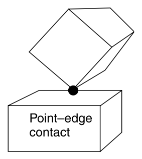
Boxes can interact in a number of ways

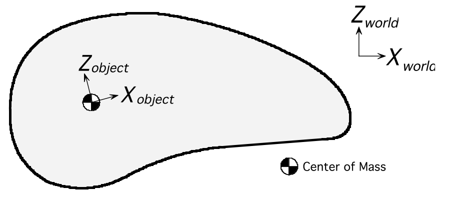


Hard to achieve a realistic behaviour without considering rotation, deformation, friction

Other Physical Simulations



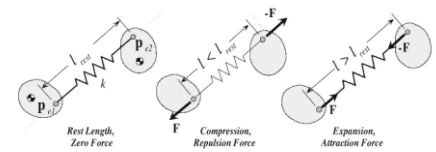




- Rigid body (no deformation) physics
  - Rotation, friction, multiple collisions
  - Joints and links
    - Ragdoll physics

More Physics

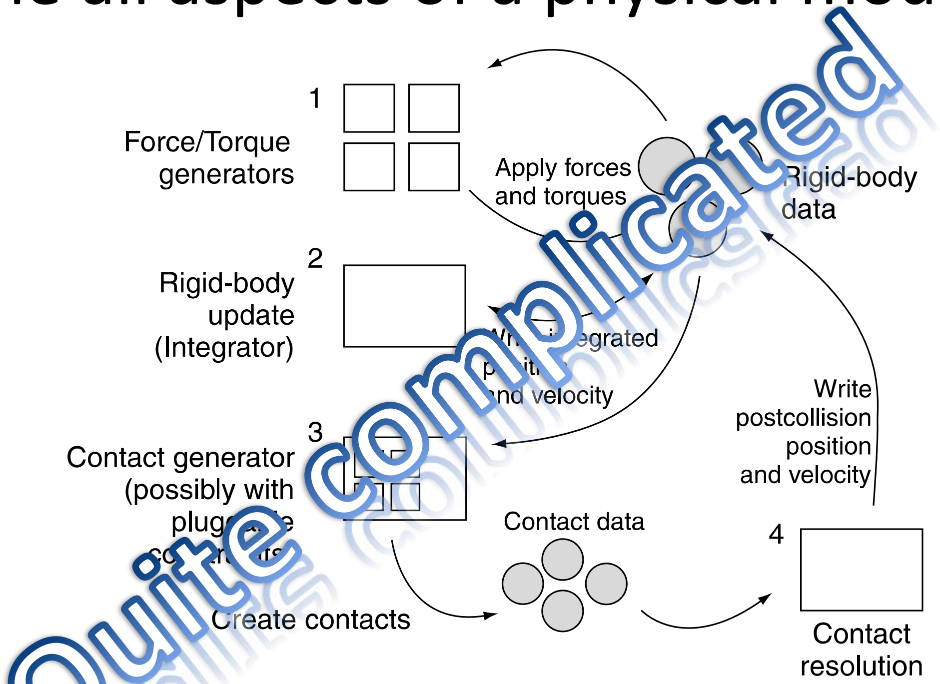
- Soft body physics (shapes can change)
  - Cloth, ropes, hair



- Fluid dynamics

Putting It All Together

- Combine all aspects of a physical model



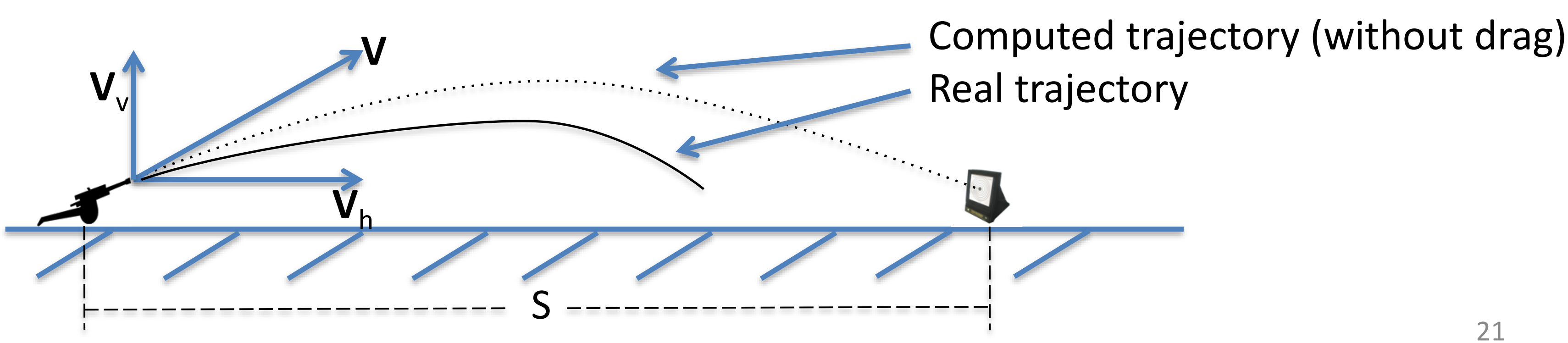
I. Millington. *Game Physics Engine Development.*

- 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?