# Game Physics

Dynamics of Particles

# Overview

- Physics in video games
- Dynamics of particles
  - Linear motion
  - Force and motion
  - Collision response

# Game Physics

- Physics is a broad discipline
  - Only some areas are useful for video games
  - For objects to appear believable in games
    - Need an understanding of how things behave in the real world, then try to model and simulate this
    - Physics simulations do not necessarily make the game fun
      - Inaccurate and confusing behaviour can destroy the game experience
  - Classical (Newtonian) mechanics
    - Describes how objects move and interact
    - Dynamics is the process of determining this over time
    - Calculate based on various properties
      - Force, mass, friction, drag, etc.

# Game Physics

- Physics in games
  - ➢ Dynamic simulations
    - Allow motion to be imparted to objects in a highly interactive and naturally chaotic manner
  - ➢ Effects difficult to achieve using animation clips
    - Less predictable compared to fixed animations
    - May give rise to emergent behaviours
      - E.g. rocket launcher jump trick (Team Fortress Classic)
    - However, dynamic simulations are harder to control
      - E.g. tweaking a value may indirectly affect the behaviour of other objects in the game
      - Hard to visualise the actual behaviour of changing a value
      - Need lots of testing and fine-tuning

# Game Physics

- Physics in games
  - ➢ Based on Newton's laws of motion
  - ➢ Closely integrated with a collision system
    - In a virtual game world, programmers must explicitly ensure that objects do not pass through one another
    - Determine whether any objects come into contact or are intersecting
    - Information about the contact
  - ➢ Collision detection and collision resolution are different
    - Some games may not simulate proper physics but may still have a collision detection system

# Game Physics

- Physics engines
  - ➢ Examples
    - Bullet, PhysX, Havok
  - ➢ Reusable technology
    - Avoids hard-coding physics behaviour for specific instances/games
    - Data driven
  - ➢ Real-time simulation
    - Can be rather computationally intensive
  - ➢ Instability issues
    - Numerical or due to various constraints
  - ➢ Particles, rigid-bodies, soft-bodies

# Dynamics of Particles

- Units and measures
  - ➢ Physics is based on measurements
    - Some SI (International Systems of Units) base quantity and units

| Quantity | Unit Name | Unit Symbol |
|---|---|---|
| Length | Metre | m |
| Mass | Kilogram | kg |
| Time | Second | s |

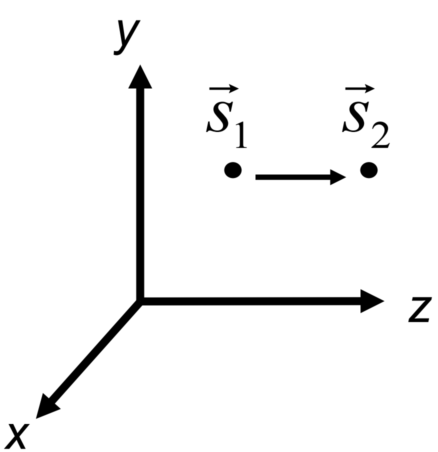
    - Combined for derived units
      - E.g., force is defined in kg.m/$s^2$ or N (Newtons)
  - ➢ Measurements are often relative in games
    - Normally still correlates to real-world measurements and units
    - Time is important in calculations, regardless of frame rate
    - Game physics engines are normally based on SI units for consistency and re-usability

# Dynamics of Particles

- Dynamics of particles (point mass)
  - From a physics simulation point of view
    - Only consider its position only
  - The object’s orientation, size, shape and structure
    - Irrelevant in the context of a particle
    - E.g., simulating a bullet’s motion
- Kinematics
  - The study of motion, not considering that which causes it
- Kinetics
  - Deals with forces and interactions that produce or affect motion

# Dynamics of Particles

- Linear motion
  - Described in terms of
    - Displacement, time, velocity, and acceleration
  - Displacement
    - Change from one position to another
    - Vector quantity
      - Has a magnitude (distance) and a direction

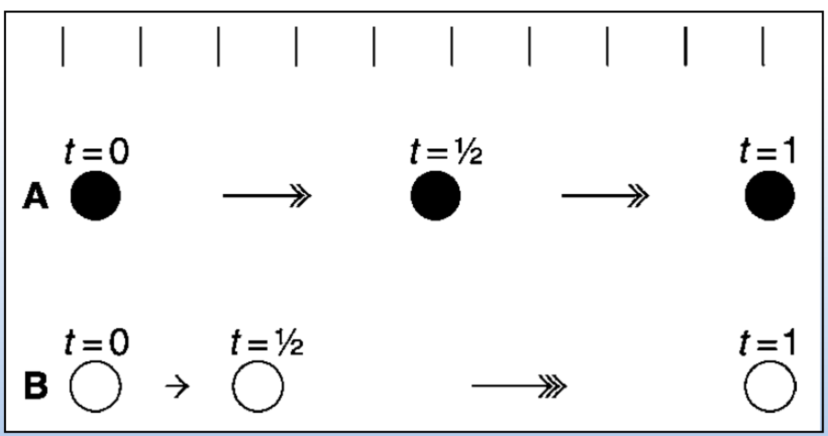


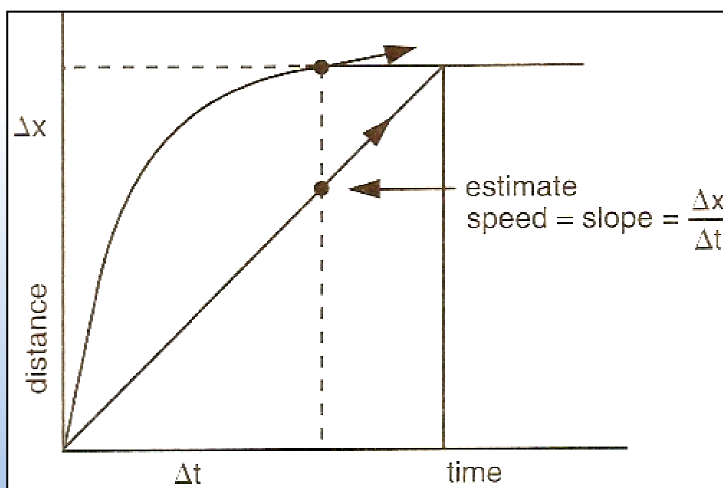

$$
\begin{aligned}
\Delta\vec{s} &= \vec{s}_2 - \vec{s}_1 \\
&= (\Delta s_x, \Delta s_y, \Delta s_z) \\
&= (s_{2x} - s_{1x}, s_{2y} - s_{1y}, s_{2z} - s_{1z})
\end{aligned}
$$

# Dynamics of Particles

- Linear motion
  - ➢ Interested in changes over time
    - A moving object changes position
  - ➢ Average velocity (m/s)
    - Change in position over the time period taken for that change
    - An object can change speed over the time period ($\Delta t$)
    - Doesn't represent exact velocity of an object at any point in time

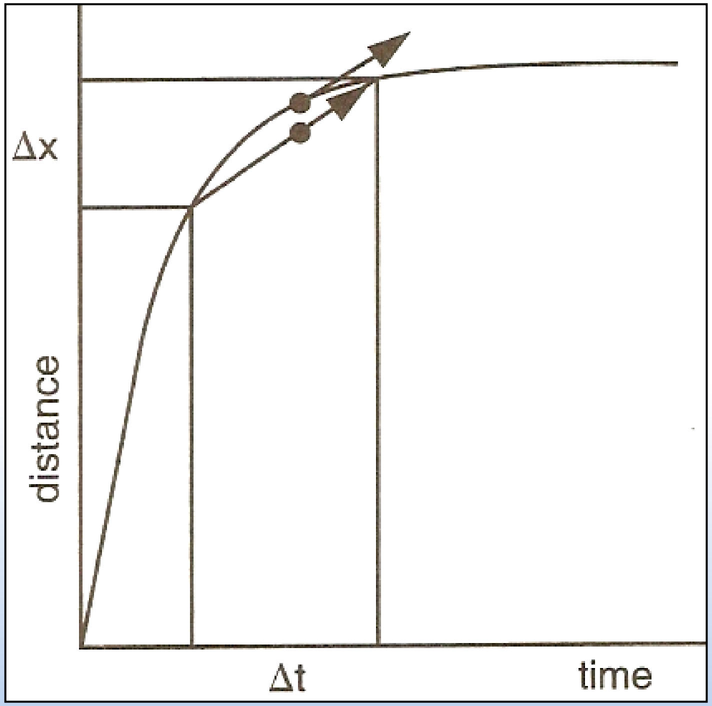
$$\vec{v}_{average} = \frac{\Delta \vec{s}}{\Delta t}$$



# Dynamics of Particles

- Linear motion
  - Velocity (instantaneous velocity)
    - Shrink the time interval (Δt) closer and closer to 0
    - The limit as Δt approaches 0
    - Differentiate with respect to time




$$\vec{v} = \lim_{\Delta t \to 0} \frac{\Delta \vec{s}}{\Delta t} = \frac{d\vec{s}}{dt}$$

# Dynamics of Particles

- Linear motion
  - Acceleration
    - Rate at which velocity is changing
      - An object accelerating quickly, is rapidly increasing in velocity
      - Decelerating, decreasing in velocity (i.e. slowing down)
    - A positive value represents speeding up
    - Zero acceleration means no change in velocity
    - A negative value represents slowing down
    - Differentiate velocity with respect to time

$$\vec{a} = \lim_{\Delta t \to 0} \frac{\Delta \vec{v}}{\Delta t} = \frac{d\vec{v}}{dt} = \frac{d^2 \vec{s}}{dt^2}$$

# Dynamics of Particles

- Linear motion
  - In mathematics, integration is the opposite of differentiation
    - If something is differentiated, then integrating it will return it back to its starting form
  - Integration is used to update position and velocity
    - With velocity, can work out the position at a given time
    - With acceleration, can work out the velocity at a given time
    - While integration is quite complex
      - Game physics engines perform updates in a time-stepped manner (numerical integration)

# Dynamics of Particles

- Linear motion
  - Equations for linear motion
    - Only valid for motion with constant or approximately constant acceleration

$$\vec{s} = \vec{v}_{average} t$$

$$\vec{v} = \vec{v}_{initial} + \vec{a} t$$

$$\vec{s} = \vec{v}_{initial} t + \frac{1}{2} \vec{a} t^2$$

$$\vec{v}^2 = \vec{v}_{initial}{}^2 + 2 \vec{a} \vec{s}$$

# Dynamics of Particles

- Unity
  - Kinematics can be implemented manually using a GameObject's Transform class
    - Updating velocity

```
velocity += acceleration * deltaTime;
```

    - Updating displacement

```
displacement = (velocity + 0.5 * acceleration *
                    deltaTime) * deltaTime;
```

    - Moving the particle

```
transform.Translate(displacement, Space.World);
```

# Dynamics of Particles

- Accuracy versus computational complexity
  - Updating displacement

```
displacement = (velocity + 0.5 * acceleration *
                    deltaTime) * deltaTime;
```

  - When updating every frame, `deltaTime` will be small
  - For example, if updating at 60 frames per second
    - `deltaTime` ~ 0.016667
    - `0.5 * deltaTime * deltaTime` ~ 0.000139
    - Acceleration unlikely to have much impact
  - If acceleration is not too high, ignoring acceleration results in a reasonably accurate approximation

```
displacement = velocity * deltaTime;
```

# Dynamics of Particles

- Force and motion
  - ➢ Kinetics
    - Why does a particle accelerate?
  - ➢ Force
    - An influence which tends to change the constant velocity motion of an object
  - ➢ Newton’s laws of motion
    - Three physical laws
      - – Provide relationships between forces acting on an object and the motion of the object

Use the force

# Dynamics of Particles

- Newton's first law (Law of Inertia)
  - An object will remain at rest, or continue to move at a constant velocity, unless acted upon by a net force
    - A stationary object will not move until a net force acts upon it
    - An object that is in motion will not change its velocity (will not accelerate) until a net force acts upon it
- Inertia
  - The resistance of an object to any change in its state of motion

# Dynamics of Particles

- Newton’s second law (Law of Acceleration)
  - Any change in motion involves an acceleration
    - The Newton’s first law is a special case of the second law for which the net force is zero
  - A net force acting on an object produces acceleration that is proportional to the object’s mass

$$\vec{F} = m\vec{a}$$

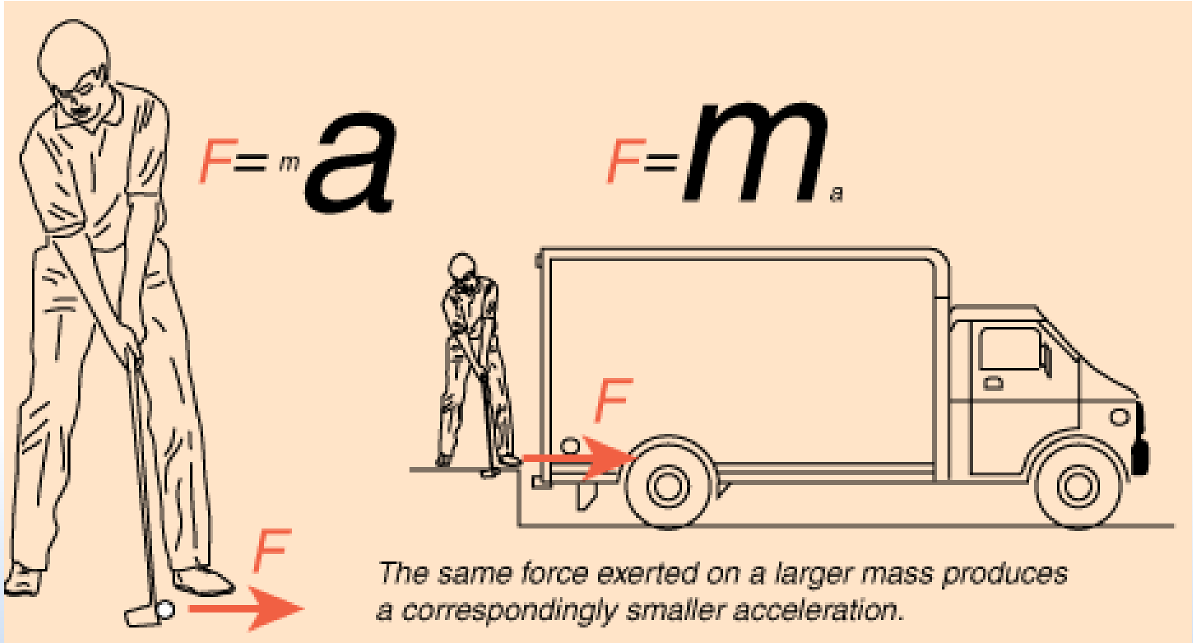
  - Mechanism by which forces alter the motion of an object
    - A force is something that changes the acceleration of an object
  - Apply a force to a particle

```
acceleration = force/mass;
```

# Dynamics of Particles

- Mass
  - Property of an object that makes it resist acceleration (inertia)
  - Measured in kg

$$\vec{F} = m\vec{a}$$




http://hyperphysics.phy-astr.gsu.edu/hbase/newt.html#fma

# Dynamics of Particles

- Newton's third law
  - If object 1 exerts a force on object 2, object 2 also exerts a force on object 1
    - The forces are equal in magnitude and opposite in direction
  - A mechanism for calculating collision responses


$$\vec{F}_1 = -\vec{F}_2$$

# Dynamics of Particles

- D'Alembert's principle
  - Multiple forces can act on an object at the same time
    - Can replace a set of forces acting on an object with a single force

$$\vec{F} = \sum_i \vec{F}_i$$

    - Simply add forces together using vector addition
    - Use a force accumulator to add each applied force
      - Final value (after accumulation) will be the resulting force applied to the object

# Dynamics of Particles

- Unity
  - Rigidbody
    - Used to control an object's movement and position through Unity's physics engine
    - In a script, the `FixedUpdate()` method is recommended as the place to apply forces and change Rigidbody settings
      - The reason being physics updates are carried out in measured time steps that don't coincide with the frame update
      - `FixedUpdate()` is called immediately before each physics update and so any changes made there will be processed directly

# Dynamics of Particles

- Rigidbody class
  - Some public variables
    - `isKinematic` – controls whether physics affects the rigidbody
    - `mass` – the rigidbody's mass
    - `useGravity` – controls whether gravity affects the rigidbody
    - `velocity` – rate of change of the rigidbody's position
  - Some public methods
    - `AddForce` – adds a force to the rigidbody
    - `AddExplosionForce` – simulates an explosion force
    - `AddForceAtPosition` – a force at a position results in a force and a torque applied to the rigidbody
    - `GetAccumulatedForce` – returns the accumulated force on the rigidbody before the simulation step

# Dynamics of Particles

- Collision response
  - Complex in the real world
    - Needs to be approximated
  - When objects collide, collision simulation must ensure
    - Objects do not interpenetrate
    - Response looks realistic
  - Involves understanding a number of concepts
    - Momentum, work, energy, etc.
  - Collision detection
    - Detects collision and computes contact information required for the collision response

# Dynamics of Particles

- Momentum (linear momentum) of a particle
  - ➢ Defined as $\vec{p} = m\vec{v}$
  - ➢ Newton's second law $\vec{F} = m\vec{a}$
    - Can also be expressed as $\vec{F} = \dfrac{d\vec{p}}{dt}$
      - – The rate of change of the momentum of a particle is proportional to the net force acting on the particle in the direction of that force
  - ➢ In a closed, isolated system

$$\frac{d\vec{p}}{dt} = 0 \qquad \frac{\Delta\vec{p}}{\Delta t} = 0$$

    - No particles enter/leave the system
    - Sum of external forces acting on the system is zero
    - No change in momentum

$$\Delta\vec{p} = \vec{p}_{final} - \vec{p}_{initial} = 0$$

# Dynamics of Particles

- Conservation of momentum
  - A tool for analysing collisions
  - Sum of momenta of all objects in the system cannot be changed by interactions within the system
    - The momentum of an isolated system is a constant
  - For a system with two particles

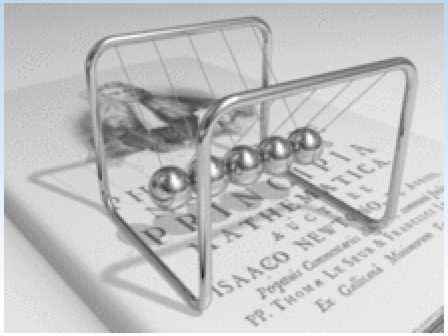
$$\boxed{\begin{aligned}&\vec{p}_{initial} = \vec{p}_{final}\\ &\vec{p}_1 + \vec{p}_2 = \vec{p}_1' + \vec{p}_2'\\ &m_1\vec{v}_1 + m_2\vec{v}_2 = m_1\vec{v}_1' + m_2\vec{v}_2'\end{aligned}}$$

# Dynamics of Particles

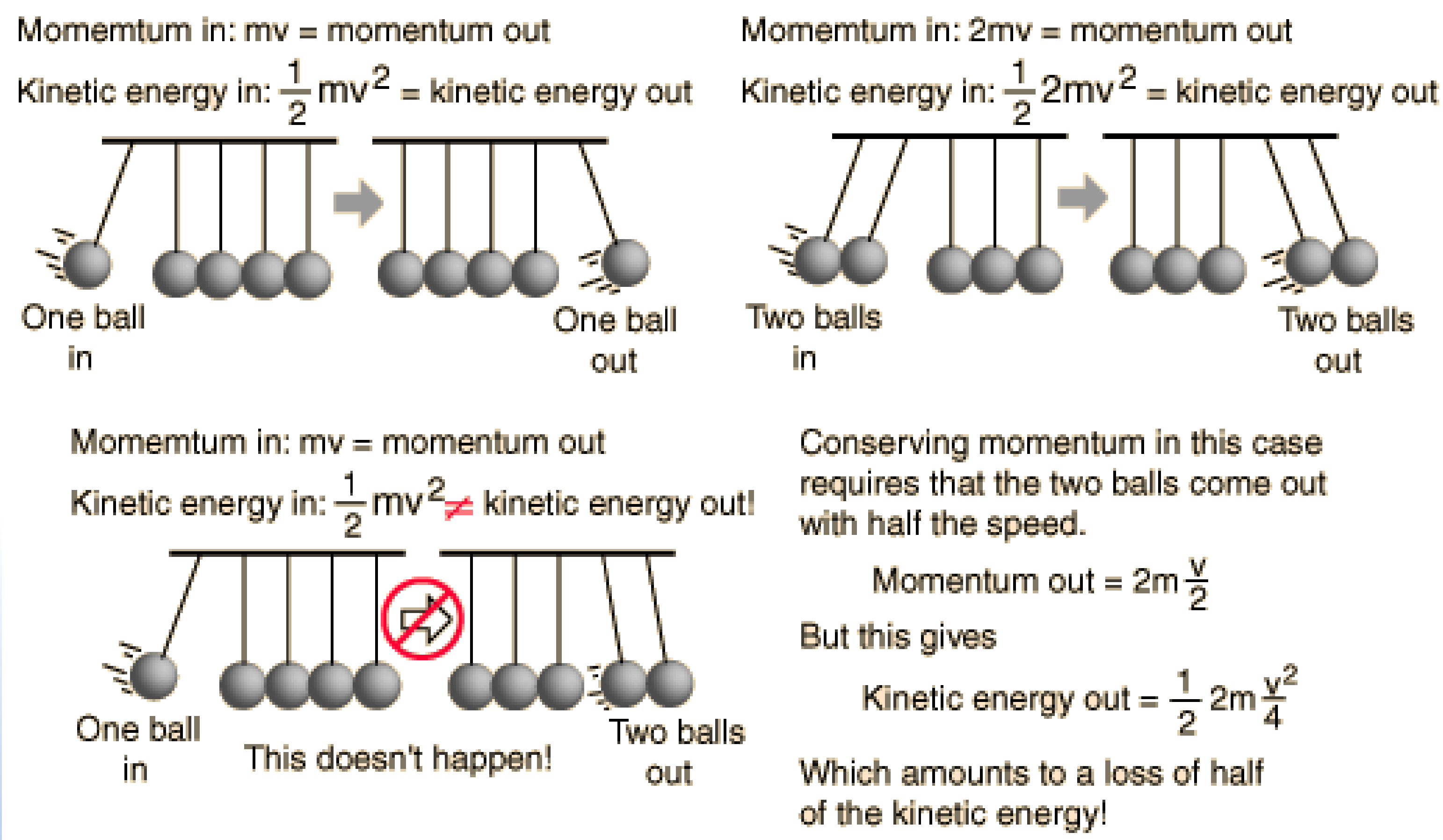
- Conservation of energy
  - Energy can neither be created nor destroyed
  - If there are no external forces on a system (isolated system)
    - Total energy of the system is constant
- Kinetic energy
  - Energy of an object in motion $K = \frac{1}{2}mv^2$
- Elastic collision
  - In perfectly elastic collisions there is no loss of kinetic energy
    - Kinetic energy is conserved

$$K_1 + K_2 = K_1' + K_2'$$

# Dynamics of Particles



- Conservation of energy




http://230nsc1.phy-astr.gsu.edu/hbasees/hframe.html

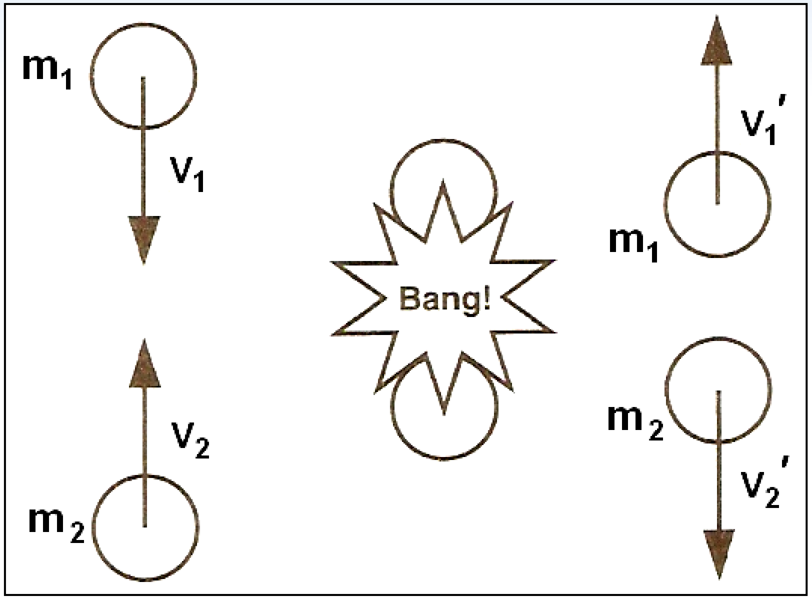
# Dynamics of Particles

- Elastic collision
  - Conservation of momentum

$$m_1\vec{v}_1 + m_2\vec{v}_2 = m_1\vec{v}_1' + m_2\vec{v}_2'$$

  - Conservation of kinetic energy

$$\frac{1}{2}m_1{v_1}^2 + \frac{1}{2}m_2{v_2}^2 = \frac{1}{2}m_1{v_1}'^2 + \frac{1}{2}m_2{v_2}'^2$$



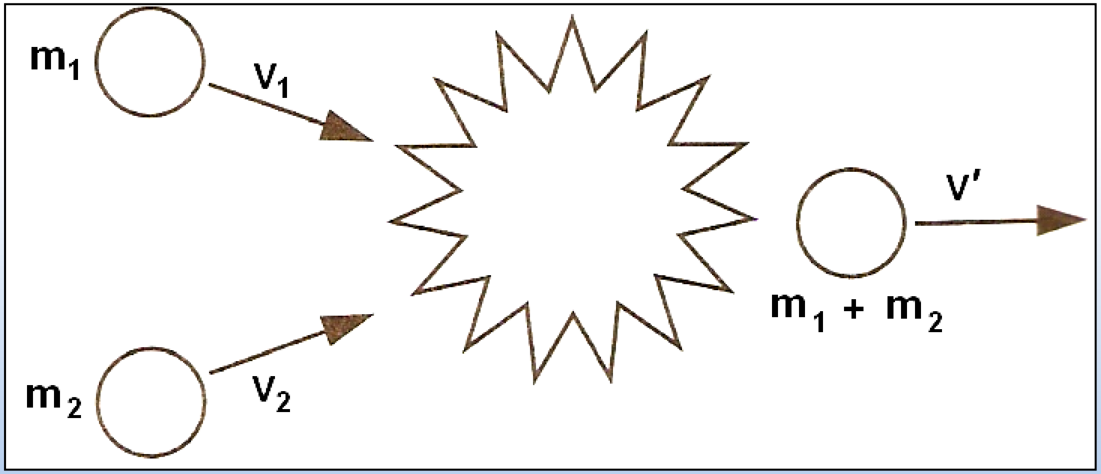

$$v_1' = \frac{(m_1 - m_2)v_1}{m_1 + m_2} + \frac{2m_2v_2}{m_1 + m_2}$$

$$v_2' = \frac{(m_2 - m_1)v_2}{m_1 + m_2} + \frac{2m_1v_1}{m_1 + m_2}$$

# Dynamics of Particles

- Inelastic collision
  - If objects stick together, collision is completely inelastic
  - Kinetic energy is changed to some other form of energy
    - Total energy conserved but *kinetic energy* **not** conserved
  - Conservation of momentum

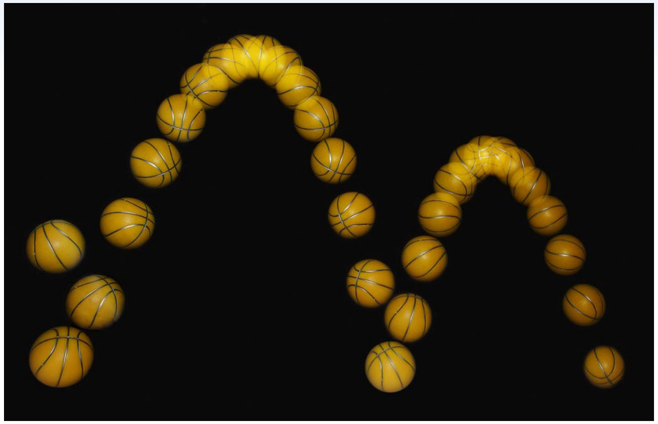
$$m_1\vec{v}_1 + m_2\vec{v}_2 = (m_1 + m_2)\vec{v}'$$




$$\vec{v}' = \frac{m_1\vec{v}_1 + m_2\vec{v}_2}{m_1 + m_2}$$

# Dynamics of Particles

- Coefficient of restitution, e
  - Represents the elasticity of a collision
    - "Bounciness"
  - Collisions in the real world are neither perfectly elastic or inelastic
    - Perfectly inelastic collision, e = 0
      - Difference in final velocities is zero
    - Perfectly elastic collision, e = 1
      - Velocities before and after collision equal in magnitude opposite in direction



$$e = \frac{-(v_1' - v_2')}{v_1 + v_2}$$

$$v_1' = \frac{(m_1 - e m_2)v_1 + (1+e)m_2 v_2}{m_1 + m_2}$$

$$v_2' = \frac{(m_2 - e m_1)v_2 + (1+e)m_1 v_1}{m_1 + m_2}$$



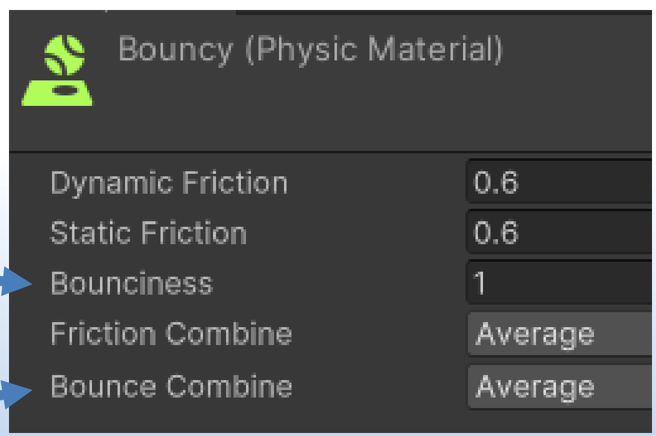
http://en.wikipedia.org/wiki/Coefficient_of_restitution

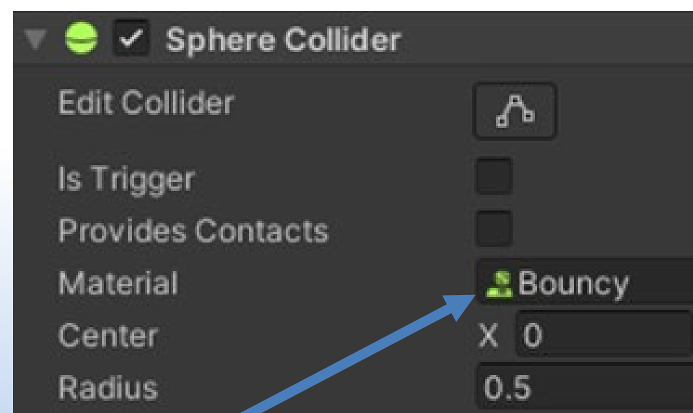
# Dynamics of Particles

- Unity
  - ➢ Physic material
    - Adjusts friction and bouncing effects of colliding GameObjects
    - Applied to a GameObject's Collider
    - Two colliding objects may have different coefficient of restitutions
      - – Can set a method for combining them

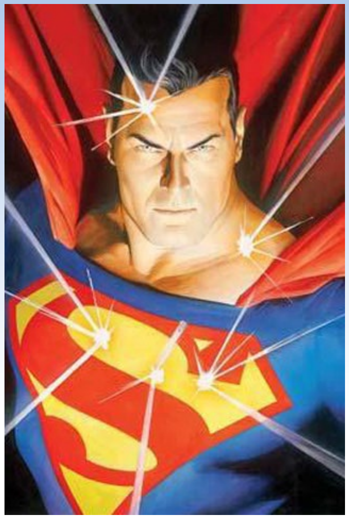
Coefficient of restitution

Combining



# Dynamics of Particles



- Impulse of force
  - When two particles collide
    - Equal but opposite forces act on the particles
    - These forces will change their linear momentum over a very short period of time
  - Impulse
    - Strength and duration of the collision force $\vec{J} = \int_{t_i}^{t_f} \vec{F}(t)dt$
  - Impulse momentum theorem
    - Change in momentum of each object in a collision is equal to the impulse that acts on that object

$$\vec{p}_f - \vec{p}_i = \Delta\vec{p} = \vec{J}$$



http://www.freeiphonewallpapers.net/wallpaper/Superman-Bullets/

# Dynamics of Particles

- Unity
  - `ForceMode`
    - To specify how to apply a force using `Rigidbody.AddForce()`
    - Ignores mass
      - `Acceleration` : adds a continuous acceleration to the rigidbody
      - `VelocityChange` : Adds an instant velocity change to the rigidbody
    - Uses mass
      - `Force` : adds a continuous force to the rigidbody
      - `Impulse` : adds an instant impulse to the rigidbody

# References

- Among others, material sourced from

  - https://unity.com/
  - https://docs.unity3d.com
  - Jason Gregory, Game Engine Architecture, A.K. Peters
  - Ian Millington, Game Physics Engine Development, Morgan Kaufmann
  - David Conger, Physics Modeling for Game Programmers, Thomson Learning
  - C.R. Nave, HyperPhysics, http://hyperphysics.phy-astr.gsu.edu/hbase/hph.html
  - David Halliday, Robert Resnick and Jearl Walker, Fundamentals of Physics, Wiley