# Simulation

Dynamical Simulation

Physics Engines

JBullet

Physics in jMonkeyEngine

# Dynamical Simulation

- The simulation of systems of objects that are free to move according to Newton's laws of dynamics
- Particle
- Rigid body

## Particle

- Has a mass
  - Measure of body's resistance to motion or a change in its motion
- Has position, velocity and acceleration
- No dimension
- Linear motion

## Rigid Body

- Solid body of finite size
- Deformation is neglected
- Position, velocity and acceleration is handled same way as for particles
  - At the body's center of mass
- Center of mass
  - The point in a body around which the mass of the body is evenly distributed

## Rigid Body

- Has an orientation
- Has angular velocity and acceleration
- Torque causes rotational acceleration
- Moment of inertia
  - A measure of an object’s resistance to change in its rotation rate
  - Think of it as rotational mass

## Procedure

- Calculate body's mass properties
- Gather all forces and torques (moments) acting on body
- Solve the equations of motion for linear and angular acceleration
- Integrate with respect to time to find linear and angular velocity
- Integrate with respect to time to find linear and angular displacement

## Procedure Example

- Integrate using Euler’s method
- Calculate acceleration given forces and mass
  $F = ma$
  $a = F/m$
- Calculate new velocity
  $v_2 = v_1 + at$
- Calculate new position
  $s_2 = s_1 + vt$
- Can use duration of the last frame as time

# Collision Detection

- Collision detection determine whether two or more objects have collided
- May need to calculate
  – Time of Impact
  – Closest Points
  – Penetration Depth
- Mesh vs mesh not easy to implement robustly
- Moving collision objects are often represented by convex shapes instead of triangle meshes
  -box, sphere, capsule, cylinder, cone, convex hull
- Improves performance and quality

# Collision Detection

- Detect collisions after they happen when objects are intersecting
  - The collision response has to “fix” the intersecting objects
- Find when objects collide and stops right before they intersect
  - Is more difficult to determine when objects collide than if they collide

## Collision Response

- Collision response determines the motion of the objects after they have collided
- Penalty methods
  - Inserts temporary springs between objects at point of contact
- Analytical methods uses classical (Newtonian) impact principles

## Collision Response

- An elastic collision loses no kinetic energy
  - Billiard balls
- In an inelastic collision the kinetic energy is used to deform the object or is converted to other type of energy
  - Velocity is reduced in the direction perpendicular to the surface
- Friction
  - Changes the linear velocity in the tangential direction
  - Produce torque that changes the angular velocity

# Physics Engine

- Performs the simulation for you
- Has it's own data structure that is separate from the rendering
- Have to set up the objects and add it to the engine
  - Collision shape
  - Start position
  - Mass properties
- Apply forces to the objects when simulation is running
- Changing the position or velocity directly may break the simulation
  - Becomes unstable
  - Blows up
- Every frame the object transformations are copied from the physics engine to the scene graph

## Physics Engine

- Constraints (joints) may be added
- Connects two bodies by constraining translation or rotation in some way
- A hinge can be used to connect a door to a frame
- Point to point constraint can be used to connect the bones in a human body
  - Ragdoll
- Constraints may have limits
  - Define how much the door can be opened
- Motors used to apply forces to constraints

## Closed Source Physics Engines

- NVIDEA PhysX
  - Binary is free to use
  - Supported by GeForce GPUs
  - Used in lots of commercial games
- Havok Physics
  - Another popular physics engine used in lots of commercial games
  - Binary is free for non commercial use

## Open Source Physics Engine

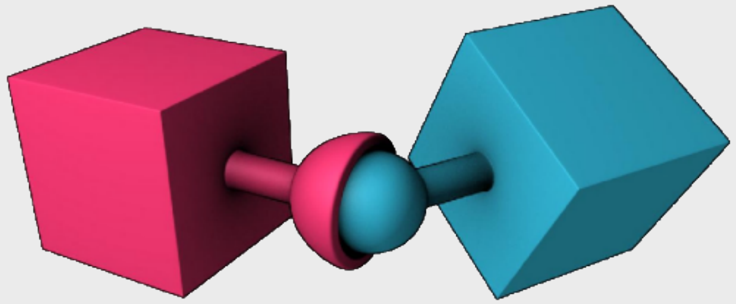
- Open Dynamics Engine (ODE)
  - BSD license
- Bullet
  - zlib license
  - Used by Blender 3D and commercial games
- JBullet
  - A partial Java port of Bullet

## JBullet

- Supported shapes
  - static plane
  - Box
  - Sphere
  - Capsule
  - Cylinder
  - Cone
  - convex hull
  - compound shape
  - static and moving triangle mesh
  - uniform scaling shape

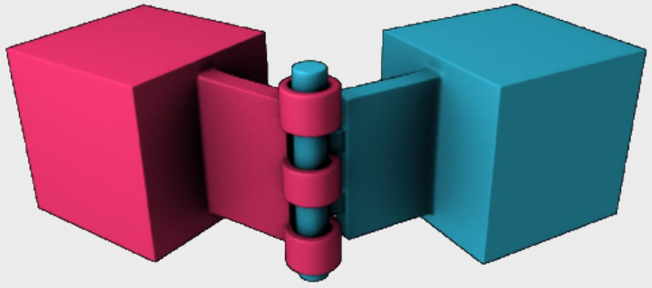
# JBullet

- Supported joints
  - Point to Point
    - Ball socket join
    - Limits translation so local pivots match in world space
  - Cone Twist
    - Point to Point with cone and twist axis limits
  - Generic 6-DOF
    - Can configure linear and angular motion axis
    - Each axis can be locked, free or unlimited
  - Raycast Vehicle
    - Entire vehicle is a single rigidbody

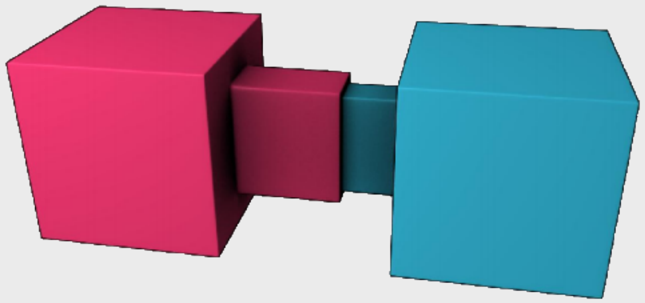


## JBullet

- Supported joints
  - Hinge
    - Can only rotate around one axis



  - Slider
    - Rotate around one axis and translate along this axis



# JBullet

- DynamicsWorld
  - Holds all the objects
  - Performs the simulation
    - You must call stepSimulation every frame
  - Constructed from
    - BroadphaseInterface
    - Dispatcher (narrow phase)
    - ConstraintSolver
    - CollisionConfiguration

# JBullet

- BroadphaseInterface
  - Finds overlapping pairs of AABB
  - AxisSweep3 is a sweep and prune algorithm
  - SimpleBroadphase is brute force O(n^2)
- Dispatcher
  - calculates exact collision given a list of possible colliding pairs
- ConstraintSolver
  - Solves contacts and joints

## JBullet

- MotionState
  - Way to get world transform of simulated objects
  - Simulation happens at center of mass
  - Have to adjust model if it doesn’t match
  - DefaultMotionState
    - Common implementation that supports center of mass offset

## JBullet

- General Tips
  - Keep size of moving collision objects between 10 cm and 5 meters
  - Avoid large mass ratios

# Physics in jMonkeyEngine 3

- Has built-in support for jBullet
- First you need a BulletAppState

```
public void simpleInitApp() {
        bulletAppState = new BulletAppState();
        stateManager.attach(bulletAppState);
}
```

- It synchronizes the scenegraph with the physics engine
- Use it to gain access to PhysicsSpace
  - Available after BulletAppState is added to StateManager
- PhysicsSpace contains all the physics object

## Physics in jMonkeyEngine 3

- For each physical spatial
  - Create a CollisionShape
  - Create the PhysicsControl from the CollisionShape and a mass value
  - Add the PhysicsControl to its Spatial
  - Add the PhysicsControl to the PhysicsSpace
  - Attach the Spatial to the rootNode (as usual)

# Physics in jMonkeyEngine 3

- CollisionShape is the (simplified) shape used by physics engine
- CollisionShapeFactory has methods to create shape from jME subgraph
  - Creates a CompoundCollisionShape with a child for each geometry in the subgraph
  - createBoxShape(Spatial) uses BoxCollisionShape
  - createMeshShape(Spatial) uses MeshCollisionShape
  - createDynamicsMeshShape(Spatial) uses HullCollisionShape

## Physics in jMonkeyEngine 3

- PhysicsControl connects a Spatial to a physics object
  - RigidBodyControl
    - Dynamic, kinematic and static objects
    - Set mass to 0 to make it static
  - GhostControl
    - Collision and intersection detection
  - CharacterControl
  - VehicleControl and PhysicsVehicleWheel
  - RagDollControll

# Physics in jMonkeyEngine 3

- Examples
  - Box falls and bounces on a static ground box
  - Ground box is replaced by model of a control room
  - Adds a sphere that can be controlled by the keyboard
  - Detect collisions between the box and sphere
  - Move the sphere with the mouse
  - Connect sphere and box using Point2PointJoint
  - Connect sphere and box using HingeJoint
  - Move a character around the room
  - Compare shapes created by CollisionShapeFactory