Principles of Computer Game Design and Implementation

Lecture 18
We already learned

• Collision detection
• Collision response
Outline for today

• Physics engines
• The usage of physics engine in jMonkey
Classes of Physics Engines

• High-precision physics engines:
  – usually used by scientists and computer animated movies.
  – more processing power to calculate very precise physics

• Real-time physics engines
  – used in video games and other forms of interactive computing
  – use simplified calculations and decreased accuracy to compute in time for the game to respond at an appropriate rate for gameplay.
Real-Time Game Physics

• We had a look at just some aspects of the use of physics in computer games
  – Particle motion
    • Newtonian physics
  – Simple collision
    • Ball-Plain
    • Ball-Ball

• Rigid-body physics, soft-body physics, fluid mechanics, etc
Physics Engine

• A prebuild solution
  – Typically provides above mentioned functions
  – Supports collision detection
    • Part of physics?
    • Part of graphics?
    • All-in-one solutions exist
      – jME v3.0
Physics Engine vs Home Tools

• Advantages of game engines
  – Complete solution from day 1
  – Proven, robust code base (in theory)
  – Lower costs

• Advantages of home-grown solutions
  – Choose only the features you need
  – Opportunity for more game-specific optimizations
  – Greater opportunity to innovate
Hardware support

• Hardware acceleration for physics processing is now usually provided by graphics processing units that support more general computation, a concept known as General Purpose processing on Graphics Processing Unit.

• AMD and NVIDIA provide support for rigid body dynamics computations on their latest graphics cards.

• Migrating data into graphical form and then using the GPU to scan and analyze it can create a large speedup.
Some Physics Engines

This is an incomplete list of physics engines available on the market.

• Open Source
  – Bullet
    • jBullet – a Java port
  – Box2D
  – Newton Game Dynamics
  – Open Dynamics Engine (ODE)
Commercial Projects

- Havoc
- PhysX
- Euphoria
- ...

10
Bullet

• Features: Multiplatform support, various shapes for collision detection, rigid and soft-body dynamics, discrete and continuous collision detection, constraints and motors, plugins.
• improved support for robotics, reinforcement learning learning and VR.
• Showcase:
  – Movies:
    • How to train your dragon, Megamind, Shrek, Sherlock Holmes, Bolt …
  – Games:
    • Toy story 3
    • HotWeels: Battle Force 5
    • …
Newton Dynamics

• Scene management, collision detection, dynamic behaviour

• Showcase: A number of games including *Penumbra, Mount&Blade*
ODE

- Features:
  - Rigid body dynamics
  - Collision detection engine

- Showcase:
  - Call of Juarez
  - World of Goo
  - ...

Havoc

• Features: rigid body dynamics, collision detection

• Lots (over 150) of games, including
  – Halo (2, 3, Wars, Reach)
  – Bioshock (1, 2)
  – Fable (2, 3)
  – Battlefield: Bad Company (1, 2)
  – …
PhysX

• Fully-fledged physics engine with hardware acceleration

• Showcase:
  – Metro 2033
  – Mafia II
  – ...

15
Euphoria

• Features: on the fly 3D character animation.
• Showcase:
  – GTA IV
  – Red Dead Redemption
  – Star Wars: The Force Unleashed
jBullet

• A re-implementation of the Bullet physics engine
  – `most of Bullet 2.72 base features`
  – Bullet is now at version 2.83
• jMonkeyEngine integration
• jMonkey also supports native Bullet
  – Functionality purposely limited to that of jBullet
jME3AppState

- jME is natively a multithreaded application
- A separate control loop associated with anAppState
- Physics engine is one such control loop
public void simpleInitApp() {
    bulletAppState = new BulletAppState();
    stateManager.attach(bulletAppState);
}
Rigid Body Mechanics

```java
Sphere s = new Sphere(60, 60, 1.5f);
Geometry ball = new Geometry("Sphere", s);
boss.setMaterial(mat);
boss.move(15, 30, 0);

RigidBodyControl myControl = new RigidBodyControl(1f);
boss.addControl(myControl);

bulletAppState.getPhysicsSpace().add(myControl);

rootNode.attachChild(boss);
```
Tuning the Behaviour

RigidBodyControl myControl = new RigidBodyControl(1f);
ball.addControl(myControl);

myControl.setRestitution(.8f);
myControl.setFriction(2);
myControl.setDamping(0, 0.1f);

Make it bounce
Linear
Roll
Demo
Global Properties

bulletAppState =
    new BulletAppState();
stateManager.attach(bulletAppState);
bulletAppState.getPhysicsSpace().
    setGravity(Vector3f.ZERO);
Demo
Static, Dynamic and Kinematic Objects

• An object of zero mass is static

```java
RigidBodyControl scenePhy = 
    new RigidBodyControl(0f);
```

• Dynamic and Kinematic objects have non-zero mass

```java
RigidBodyControl paddleControl = 
    new RigidBodyControl(100f);
paddleControl.setKinematic(true);
```
Controlling Static or Kinematic Entities

- setLocalTranslation
- setLocalRotation
- move
- rotate
- ...

The difference:
Kinematic objects update their physical state as they move
## Controlling a Dynamic Entity (1)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setAngularVelocity(v)</code></td>
<td>This sets the current rotational speed of the object. The x, y, and z components of the vector are the speed of rotation around the respective axis. (Rotation)</td>
</tr>
<tr>
<td><code>setLinearVelocity(v)</code></td>
<td>This sets the current linear speed of this object. (Translation)</td>
</tr>
<tr>
<td><code>applyCentralForce(v)</code></td>
<td>This pushes an object over time with an additional moment ( v ), expressed as <code>Vector3f</code>, applied to the center. (Translation)</td>
</tr>
<tr>
<td><code>applyForce(v,p)</code></td>
<td>This pushes an object over time with additional force ( v ), applied to a non-central point ( p ). (Translation)</td>
</tr>
<tr>
<td><code>applyTorque(v)</code></td>
<td>This twists an object over time additionally around its axes. The x, y, and z components of the <code>Vector3f</code> ( v ) specify the torque around the respective axis. (Rotation)</td>
</tr>
</tbody>
</table>
Controlling a Dynamic Entity (2)

applyTorqueImpulse(v)  
This applies an instantaneous torque \( v \) to the object. The x, y, and z components of the \( \text{Vector3f} \) \( v \) specify the torque around the respective axis. (Rotation)

applyImpulse(v,p)  
This applies an instantaneous impulse \( v \), expressed as \( \text{Vector3f} \), to the object at a point \( p \) relative to the object. (Translation)

clearForces()  
This cancels all forces and stops all current motion.
Physics-Based Collision Detection in jME3

```java
public class Example06 extends SimpleApplication implements PhysicsCollisionListener {

...

}

public void collision(PhysicsCollisionEvent event) {

...

}
```
public void collision(PhysicsCollisionEvent event) {
    if ((event.getNodeA().getName().equals("Sphere") &&
        event.getNodeB().getName().equals("paddle")) ||
        (event.getNodeB().getName().equals("Sphere") &&
        event.getNodeA().getName().equals("paddle"))) {
        Material mat = new Material(assetManager,
            "Common/MatDefs/Light/Lighting.j3md");
        mat.setBoolean("UseMaterialColors", true);
        mat.setColor("Ambient",
            ColorRGBA.randomColor());
        paddle.setMaterial(mat);
    }
}
Demo
Many Other Features

• E.g Hinges

```java
HingeJoint joint =
    new HingeJoint(
        hC, // A
        bC, // B
        // pivot point local to A
        new Vector3f(0f, 0f, 0f),
        // pivot point local to B
        new Vector3f(0f, 10f, 0f),
        new Vector3f.UNIT_Z,  // DoF Axis of A (Z axis)
        Vector3f.UNIT_Z); // DoF Axis of B (Z axis)

bulletAppState.getPhysicsSpace().add(joint);
```
Game Physics

• Getting a physics engine (and even free of charge) for your project is not a big deal
  – Integrating a physics engine into your system is a different matter
Repeated:
Physics Engine vs Home Tools

• Advantages of game engines
  – Complete solution from day 1
  – Proven, robust code base (in theory)
  – Lower costs

• Advantages of home-grown solutions
  – Choose only the features you need
  – Opportunity for more game-specific optimizations
  – Greater opportunity to innovate