Introduction to the Open Dynamics Engine

CSC200
Spring 2005
Marc Schraffenberger
What is ODE?

The Open Dynamics Engine (ODE) is a free, industrial quality library for simulating articulated rigid body dynamics. For example, it is good for simulating ground vehicles, legged creatures, and moving objects in VR environments. It is fast, flexible and robust, and it has built-in collision detection. ODE is being developed by Russell Smith with help from several contributors.
Rigid Bodies

• Properties which change over time
  - Position vector \((x,y,z)\) of the body's point of reference
  - Linear velocity of the point of reference, a vector \((v_x,v_y,v_z)\)
  - Orientation of a body (Rotation)
  - Angular velocity vector \((w_x,w_y,w_z)\) which describes how the orientation changes over time

• Properties which are usually constant over time
  - Mass of the body
  - Position of the center of mass with respect to the point of reference
  - Inertia matrix. Describes how the body's mass is distributed around the center of mass
Rigid Body Coordinate Frame

- Conceptually each body has an x-y-z coordinate frame embedded in it, that moves and rotates with the body.
- The origin of this coordinate frame is the body's point of reference.
- Note that the shape of a rigid body is not a dynamical property. It is only collision detection that cares about the detailed shape of the body.
Integration

The process of simulating the rigid body system through time is called integration. Each integration step advances the current time by a given step size, adjusting the state of all the rigid bodies for the new time value. There are two main issues to consider when working with any integrator

– How accurate is it? That is, how closely does the behavior of the simulated system match what would happen in real life?
– How stable is it? That is, will calculation errors ever cause completely non-physical behavior of the simulated system?
Force Accumulators

Between each integrator step the user can call functions to apply forces to the rigid body. These forces are added to "force accumulators" in the rigid body object. When the next integrator step happens, the sum of all the applied forces will be used to push the body around. The forces accumulators are set to zero after each integrator step.
**Joints and Constraints**

- In real life a joint is something like a hinge, that is used to connect two objects.

- In ODE a joint is very similar: It is a relationship that is enforced between two bodies so that they can only have certain positions and orientations relative to each other.

- Each time the integrator takes a step all the joints are allowed to apply constraint forces to the bodies they affect. These forces are calculated such that the bodies move in such a way to preserve all the joint relationships.
Joint Groups

- A joint group is a special container that holds joints in a world.
- Joints can be added to a group, and then when those joints are no longer needed the entire group of joints can be very quickly destroyed with one function call.
- However, individual joints in a group can not be destroyed before the entire group is emptied.
- This is most useful with contact joints, which are added and remove from the world in groups every time step.
Joint Error

- When a joint attaches two bodies, those bodies are required to have certain positions and orientations relative to each other.
- However, it is possible for the bodies to be in positions where the joint constraints are not met. This "joint error" can happen in two ways:
  - If the user sets the position/orientation of one body without correctly setting the position/orientation of the other body.
  - During the simulation, errors can creep in that result in the bodies drifting away from their required positions.
Error Reduction Parameter (ERP)

- There is a mechanism to reduce joint error: during each simulation step each joint applies a special force to bring its bodies back into correct alignment.
- This force is controlled by the error reduction parameter (ERP), which has a value between 0 and 1.
- The ERP specifies what proportion of the joint error will be fixed during the next simulation step.
  - If ERP=0 then no correcting force is applied and the bodies will eventually drift apart as the simulation proceeds.
  - If ERP=1 then the simulation will attempt to fix all joint error during the next time step.
  - A value of ERP=0.1 to 0.8 is recommended (0.2 is the default).
Collision Handling

- Collisions between bodies or between bodies and the static environment are handled as follows:
  - Before each simulation step, the user calls collision detection functions to determine what is touching what. These functions return a list of contact points.
  - A special contact joint is created for each contact point. The contact joint is given extra information about the contact, for example the friction present at the contact surface or how bouncy or soft it is.
  - The contact joints are put in a joint "group", which allows them to be added to and removed from the system very quickly.
  - A simulation step is taken.
  - All contact joints are removed from the system.
Typical Simulation Code

- Create a dynamics world.
- Create bodies in the dynamics world.
- Set the state (position etc) of all bodies.
- Create joints in the dynamics world.
- Attach the joints to the bodies.
- Set the parameters of all joints.
- Create a collision world and collision geometry objects, as necessary.
- Create a joint group to hold the contact joints.

(continue on next slide)
Typical Simulation Code

- Loop:
  - Apply forces to the bodies as necessary.
  - Adjust the joint parameters as necessary.
  - Call collision detection.
  - Create a contact joint for every collision point, and put it in the contact joint group.
  - Take a simulation step.
  - Remove all joints in the contact joint group.

- Destroy the dynamics and collision worlds.