CS425: Game Programming 1
Lecture 8: Rigid-Body System
9/25/2013, 10/7/2013

Lecturer: Jyh-Ming Lien

1 Rigid-Body System

1.1 Elements

State The state $Y(t)$ of a rigid-body system at time t usually comprises of the following elements:

$$Y(t) = \begin{bmatrix} X(t) \\ R(t) \\ T(t) \\ F(t) \end{bmatrix}$$

$X(t)$ is the positions of the center of mass (COMs) of the rigid objects, $R(t)$ is the orientations, $T(t)$ and $F(t)$ are torques and forces, respectively.

Examples Here is an example:

ODE solver Assume that an ODE solver is provided, this can be Euler's method, midpoint method, or RK4. The ODE solver should have the following form ODE(Y[] , int len, double h, DYDTFUN dydt)

Function DYDTFUN the derivative function is defined as:

typedef void (* DYDTFUN)(double t, double Y[], double dY[])

Rigid vs. Particle The main differences between a particle system and a rigid-body system are:

- more states to track due to rotation
- more complex $dydt$ due to rotation

2 Rotation

Euler Angles Denoted as (r_x, r_y, r_z) where r_x, r_y, r_z are angles specifying rotations around x, y and z axes, respectively. However, the order that you apply rotation matters: ex. xyz or zyx

Rotation Matrix Denoted as

$$R(t) = \begin{bmatrix} r_{xx} & r_{yx} & r_{zx} \\ r_{xy} & r_{yy} & r_{zy} \\ r_{xz} & r_{yz} & r_{zz} \end{bmatrix}$$

Each volume of $R(t)$ specifies the orientation of x, y, and z axes. To show this, we can multiply each axis with $R(t)$:

$$R(t)\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix}, R(t)\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix}, R(t)\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{bmatrix}$$

For a matrix to be a rotation matrix the following requirement must be met.

- Each column must be orthogonal to each other
- Each column is a unit vector

To maintain these properties, $R(t)$ must be corrected after $R(t)$ is derived by ODE.

Quaternions A quaternion has four values: (s, v_s), where s is scalar and v_s is a 3d vector. Using a quaternion $q = (s, v_s)$ to represent orientation, q has the following properties:

- If q specifies a rotate θ radian about a unit vector u then

$$q = (\cos(\frac{\theta}{2}), \sin(\frac{\theta}{2})u)$$

- The inverse of q is q^{-1} which is simply $-q = (s, -v)$.
- the multiplication of two quaternions q_1, q_2 is:

$$q_1 \times q_2 = (s_1 s_2 - v_1 v_2, s_1 v_2 + s_2 v_1 + v_1 \times v_2)$$

- rotate a point p by q is written as

$$q \times p \times q^{-1}$$

.

- rotate p by q_1 and then by q_2 is written as

$$(q_2 q_1) p (q_2 q_1)^{-1}$$

- quaternions are better than matrix because
  - q can be normalized easily (this is the most important property for rigid body simulation)
  - q can be interpolated by treating q as a point on a unit sphere. More specifically, the arc connecting two such points on sphere is:

$$slerp(q_1, q_2, t) = \frac{\sin\left((1-t)\theta\right) q_1 + \sin(t\theta) q_2}{\sin(\theta)}$$

  - q can be converted to rotation matrix easily

3 Rigid body status

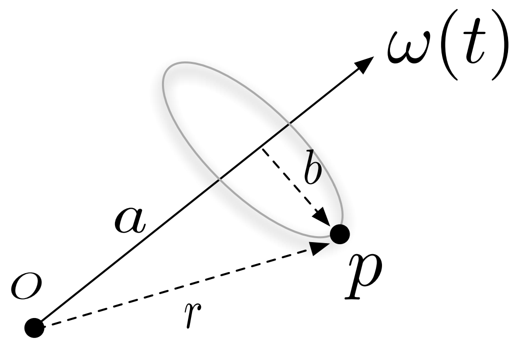
Linear velocity $V(t) = \frac{\mathrm{d}}{\mathrm{d}t}X(t)$ is the linear velocity of the center of mass.

Angular velocity $\omega(t)$ is the direction that the body is spinning about. $||\omega(t)||$ is how fast the body rotates about $\omega(t)$. Remember that $\omega(t)$ is a 3D vector.

3.1 Relationship between $\omega(t)$ and $R(t)$

Imagine that the rigid body is made of a collection of points. Let p be one of them and let O be the center of mass of this rigid body.

$\omega(t)$ and $R(t)$ Let $r = p - O$, the velocity of p at time t caused by rotation $\omega(t)$ is:



Decomposition Break r into $r = a + b$ where a is parallel to $\omega(t)$ and b is perpendicular to $\omega(t)$.

Relationship The vector a has no effect on p's velocity. The velocity must be perpendicular to b and $\omega(t)$ (i.e. tangent to the circle in the figure above). Therefore, the velocity of p is defined as

x, y and z Axes Given a rotation matrix $R(t)$, recall that each column of $R(T)$ represents the direction (orientation) of x, y and z axes. Therefore, the velocity of x axis due to $\omega(t)$ is similarly we can formulate

the same for y and z axes. Finally, we can say that

3.2 Combine $V(t)$ and $\omega(r)$

At time t, the location of the rigid body is $x(t)$ and linear velocity is $v(t)$, and the location of a point on the body is $p(t)$.

The derivative of $p(t)$ The velocity of $p(t)$ is

And, we know that the position is $p(t) = R(t)p(t = 0) + x(t)$. Finally, we can rewrite $\dot{p}(t)$ as:

4 Force and Torque

Force Similar to particle systems, there will force applied to the rigid bodies (gravities, wind, collision, ..)

Apply force Forces are applied to single points on the rigid body. Let us denote $F_p(t)$ as the force applied to point p at time t.

Torque Force $F_p(t)$ produces torque:

Note that $\tau_p(t)$ is perpendicular to $F_p(t)$ and to $(p(t)-x(t))$, therefore, it $F_p(t)$ is parallel to $(p(t)-x(t))$, $\tau_p(t) = 0$.

Total force and torque Total force in the system on the body B is

5 Linear Momentum and Angular Momentum

Instead of storing velocities, a typical rigid body system usually stores momentum due to the fact that angular momentum will stay constant of the torque is zero (this is not true for angular velocity).

Linear momentum $P_p = m_p v$ describes the resistance to the change of linear motion of a particle p. Let us see how this is formulated for the rigid body.

Derivative of Linear momentum We know that $\dot{P} = F(t)$ (newton's 2nd law), so to compute $P(t)$ we integrate $F(t)$ using ODE solver.

Angular momentum Angular momentum $L(t)$ is a measure of resistance to the change of rotation. The idea of $L(t)$ is very similar to linear momentum. More specifically,

where $I(t)$ is a 3 by 3 matrix called **inertia**. Inertia is similar to mass but different in the way that $I(t)$ is the *distribution* of mass. More on this later.

Derivative of Angular momentum $\dot{L}(t) = \tau(t)$, so to compute $L(t)$ we integrate $\tau(t)$ using ODE solver.

6 inertia Tensor

The inertia $I(t)$ is a 3 by 3 matrix describing the *distribution* of mass in the body B. More specifically,

$$I(t) = \sum_{p \in B} \begin{bmatrix} m_p(r_y^2 + r_z^2) & -m_p r_x r_y & -m_p r_x r_z \\ -m_p r_y r_x & m_p(r_x^2 + r_z^2) & -m_p r_y r_z \\ -m_p r_z r_x & -m_p r_z r_y & m_p(r_x^2 + r_y^2) \end{bmatrix},$$

where $r = p(t) - x(t)$ and m_p is the mass of p.

Bad news $I(t)$ changes when B rotates. It is really not a good idea to recompute $I(t)$ after every rotation. The time complexity of recomputing $I(t)$ is linear to the number of points.

What can we do? We can factor $I(t)$ so only the part related to rotation is recomputed.

$$\begin{aligned} I(t) &= \textstyle\sum_{p \in B} \begin{bmatrix} m_p(r_y^2 + r_z^2) & -m_p r_x r_y & -m_p r_x r_z \\ -m_p r_y r_x & m_p(r_x^2 + r_z^2) & -m_p r_y r_z \\ -m_p r_z r_x & -m_p r_z r_y & m_p(r_x^2 + r_y^2) \end{bmatrix} \\ &= \end{aligned}$$

Note that $\mathbb{I}$ is the identity matrix. Now let us rewrite the last sentence.

$$
\begin{aligned}
I(t) &= \textstyle\sum_{p\in B} m_p \left(r^T r \mathbb{I} - r r^T\right) \\
&=
\end{aligned}
$$

Use I_{body} The matrix I_{body} is invariant to rotation so we can compute I_{body} before simulation started and use it over and over. Recall that angular momentum $L(t) = I(t)\omega(t)$. To compute $\omega(t)$ from torque $\tau(t)$ we need $\omega(t) = I^{-1}(t)L(t)$.

$$
I^{-1}(t) \quad =
$$

Final remark The state of the rigid body system is again:

What is next? We will talk about the implementation of these on Wed.