Simulation of the unstable rotation of a cuboid

Books in space

Ivar Postma Eamon Nerbonne

Introduction to Computational Science
School for Computing and Cognition
University of Groningen

December 1, 2011

Experiment in space

- http://www.youtube.com/watch?v=GgVpOorcKqc

Stable and unstable rotation

- The book can stably rotate around two of its axis
- Rotation around the third axis is not stable

Model

State

A point is in a certain state

$x(t)$	position
$R(t)$	orientation
$P(t)$	total linear momentum
$L(t)$	total angular momentum

Model

Updating the state

$$
\begin{aligned}
\dot{x}(t) &= v(t) && \text{velocity} \\
\dot{R}(t) &= \omega^*(t)R(t) && \text{angular velocity} \\
\dot{P}(t) &= F(t) && \text{force} \\
\dot{L}(t) &= \tau(t) && \text{torque (moment of force)}
\end{aligned}
$$

- $\omega^*(t)$ is the tensor notation of $\omega(t)$
- $\dot{\alpha}$ is the notation for the time derivative of α

Modelling the behaviour

Simplifications

We make a few simplifications

- The book is a cuboid
- The density is uniform

Modelling the behaviour

More simplifications

When a book is rotating in space, the model is simple because

- there is no force, $F(t) = 0$
- there is no torque, $\tau(t) = 0$
- $x(t)$ is chosen to be the centre of mass, which is stationary

Thus, we only need to calculate the angular velocity

Model

Updating the state

$$
\begin{aligned}
x &= \text{constant} \\
R(t+\epsilon) &= R(t) + \epsilon\ \omega^*(t) R(t) \\
P &= 0 \\
L &= \text{constant}
\end{aligned}
$$

Angular speed is given by

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

$I(t)$ depends on $R(t)$ and the moment of inertia of the body I_{body}

$$I(t) = R(t) I_{body} R(t)^T$$

Moment of inertia

- The moment of inertia of an object is a measure of the resistance of an object to changes in the rotation
- For a cuboid the moment of inertia is given by

$$I_{body} = \frac{1}{12}\begin{pmatrix} M(b^2+c^2) & 0 & 0 \\ 0 & M(a^2+c^2) & 0 \\ 0 & 0 & M(a^2+b^2) \end{pmatrix}$$

where a, b and c are the dimensions of the cuboid

Methods to solve the calculations

- Euler intergration
- Midpoint method
- 4th order Runga Kutta

Implementation

We intend to implement this in C++ using OpenGL for the visualisation

Questions?