Haptic Systems
530-655

Mohsen Mahvash
Lecture 6
17/1/06
Haptic rendering of contact

- The god-object method
- A simple collision detection method
- Sliding contact
- Spherical proxy
- Improving collision detection algorithms
- References
Haptic simulation of contact with a virtual wall

\[ \vec{F} = 0 \]

\[ t = 0\, ms \]

\[ \vec{F} = 10N \]

\[ t = 1\, ms \]

\[ \vec{F} = -k(\vec{x} - \vec{c}) \]

\[ t = 2\, ms \]

god object
proxy
contact point
A physics-based method

The tip of a virtual tool comes into a contact with a stiff deformable body at point $c$ and then the contact point slides over the surface of the virtual object.
The god-object method

The god-object is moved to a point that locally minimizes the distance between the god-object and the tip of the haptic device, subject to a surface constraint.

\[
\text{min} \ Q = \frac{1}{2} | \vec{c} - \vec{x} |^2
\]

\[
\vec{n}_1 . \vec{c} = \vec{d}_1
\]

(equation of a plane)
Surface representation of 3D objects

Many complex objects can be represented by triangles.
A simple collision detection method

Points $\vec{x}_0$ and $\vec{x}_1$ define a line segment.

Determining the triangles which collide with the line segment.

For a triangle that collides:

$$(\vec{x}_0 - \vec{v}_1).\vec{n} > 0$$
$$(\vec{x}_1 - \vec{v}_1).\vec{n} < 0$$
$$(\vec{c} - \vec{v}_i).\vec{n}_i < 0 \; \forall i = 1 : 3$$
Sliding contact (convex objects)

\[
\begin{align*}
\vec{n}_1 \cdot \vec{c} &= d_1 \\
\vec{n}_1 \cdot \vec{c} &= d_1 \\
\vec{n}_2 \cdot \vec{c} &= d_2
\end{align*}
\]

\[
\min Q = \frac{1}{2} |\vec{c} - \vec{x}|^2
\]
Sliding contact (concave objects)

Several surface constraints are active when the god object crosses an edge of a concave surface.

\[ \vec{n}_1 \cdot \vec{c} = d_1 \]
\[ \vec{n}_2 \cdot \vec{c} = d_2 \]

\[ \min Q = \frac{1}{2} | \vec{c} - \vec{x} |^2 \]
An optimization problem

\[
\min Q = \frac{1}{2} |\vec{c} - \bar{x}|^2 \quad \text{when} \quad \vec{n}_1 \cdot \vec{c} = d_1 \quad \vec{n}_2 \cdot \vec{c} = d_2 \quad \vec{n}_3 \cdot \vec{c} = d_3
\]

\[
\min L = \frac{1}{2} |\vec{c} - \bar{x}|^2 + \lambda_1 (\vec{n}_1 \cdot \vec{c} - d_1) + \lambda_2 (\vec{n}_2 \cdot \vec{c} - d_2) + \lambda_3 (\vec{n}_3 \cdot \vec{c} - d_3)
\]

\[
\frac{\partial L}{\partial c_i} = 0 \quad \frac{\partial L}{\partial \lambda_i} = 0
\]

\[
A \begin{bmatrix} \bar{c} \\ \bar{\lambda} \end{bmatrix} = \begin{bmatrix} \bar{x} \\ \bar{d} \end{bmatrix}
\]
Spherical proxy

Falling through holes of a virtual object can be avoided by using a spherical proxy.
Improving collision detection algorithms

OBBTrees Method: an oriented bounding box surrounds all objects under a subtree.

If the line segment does not collide with the box it will not collide with the objects inside the box.
References:


Tomorrow

Force shading and rendering of friction