Voxels & Collisions

Today

- Implicit Surfaces
- Voxels
- Collisions
- Implicit Integration

Implicit Surfaces

- For a sphere:
  \[ H(x,y,z) = x^2 + y^2 + z^2 - r^2 \]
  - If \( H(x,y,z) = 0 \), on surface
  - If \( H(x,y,z) > 0 \), outside surface
  - If \( H(x,y,z) < 0 \), inside surface

Level Sets

- Efficient method for computing signed distance field

Marching Cubes

- Polygonization: extract triangle mesh from signed distance field
“Marching Tetrahedra”

Jules Bloomenthal
“An implicit surface polygonizer”
Graphics Gems IV

“When the Blobs Go
Marching Two by Two”,
Jeff Lander, Gamasutra

Similarly, we can create volumetric models:

“Marching Tetrahedra”

“Interval volume tetrahedrization”
Visualization ’97
Nielsen & Sung

“Questions?”

Today
- Implicit Surfaces
- Voxels
- Collisions
- Implicit Integration

Collisions
- Detection
- Response
- Overshooting problem
  (when we enter the solid)

Detecting Collisions
- Easy with implicit equations of surfaces
  \( H(x,y,z) = 0 \) at surface
- \( H(x,y,z) < 0 \) inside surface
- So just compute \( H \) and you know that you’re inside if it’s negative
- More complex with other surface definitions
Collision Response
- tangential velocity $v_t$ unchanged
- normal velocity $v_n$ reflects:
  \[ v = v_t + v_n \]
  \[ v_\perp = v_t - \varepsilon v_n \]
- coefficient of restitution (1 for elastic, 0 for plastic)
- change of velocity $= -(1+\varepsilon)v$
- change of momentum Impulse $= -m(1+\varepsilon)v$

Collisions - Overshoooting
- Usually, we detect collision when it’s too late: we’re already inside
- Solutions: back up
  - Compute intersection point
  - Compute response there
  - Advance for remaining fractional time step
- Other solution:
  Quick and dirty fixup
  - Just project back to object closest point

Collision Detection for Solids
- How to detect collision between 2 polyhedra?
- Need an inside/outside test
- Test if a vertex is inside the other polyhedron
- But treat also edge-edge intersection

Cost of Detection?
- Test each edge with each face?
  - $O(N^2)$
- How would you detect collision between two bunnies?
  - $O(N^2)$ is too expensive!
  - Use spatial hierarchy

Conservative Bounding Region
- First check for an intersection with a conservative bounding region
- Early reject

Conservative Bounding Regions
- tight $\rightarrow$ avoid false positives
- fast to intersect
**Overlap test**

- Overlap between two axis-aligned boxes?
  - Check if the intervals along the 3 dimensions overlap
- Overlap test between two spheres?
  - $D(\text{center}_1, \text{center}_2) < r_1 + r_2$

**General Collision Detection**

- Put a hierarchy around your objects
- Use the fast overlap test recursively
- Handle exact case at the leaves (when necessary)
- More difficult for self-collision (e.g. cloth)
  - Because there is more overlap

**Reduced Deformation**

- Collisions are expensive
- Deformation is expensive
- This is a lot of geometry!
- Simplify the simulation model

**Cloth Collision**

- A cloth has many points of contact
- Stays in contact
- Requires
  - Efficient collision detection
  - Efficient numerical treatment (stability)

**Cloth Collision Challenges**

- Detecting and resolving prior unavoidable intersections

**Questions?**
Today

- Implicit Surfaces
- Voxels
- Collisions
- Implicit Integration

Euler Integration

- Timestep $h$, move in the direction of $f(X, t)$

Stiffness & Stability

- With an explicit/forward integration scheme:
  $$y_{k+1} = y_k + h \cdot g(y_k)$$
  we must use a very small timestep to simulate stable, stiff cloth.

- Alternatively we can use an implicit/backwards scheme:
  $$y_{k+1} = y_k + h \cdot g(y_{k+1})$$
  $$y_k = y_{k+1} - h \cdot g(y_{k+1})$$
  Solving one step is much more expensive (Newton’s Method, Conjugate Gradients, …) but overall faster than the thousands of explicit timesteps required for very stiff springs.

Cloth

- Dynamic motion driven by animation

Questions?