Ray Intersections

CS 319
Advanced Topics in Computer Graphics
John C. Hart
What about the normal?

- Let \( \mathbf{n} = [a \ b \ c \ d] \) be a tangent plane
- Let \( \mathbf{x} = [x \ y \ z \ 1]^T \) be a point
- Plane-point duality
  - Planes are row vectors
  - Points are column vectors
- Point \( \mathbf{x} \) in plane \( \mathbf{n} \) \( \iff \mathbf{n} \mathbf{x} = 0 \)
- Need to find \( \mathbf{n}' \) such that \( \mathbf{n}'^T \mathbf{x} = 0 \)
- Notice \( \mathbf{n} \mathbf{T}^{-1} \mathbf{T} \mathbf{x} = 0 \)
- New normal \( \mathbf{n}' = \mathbf{n} \mathbf{T}^{-1} = (\mathbf{T}^{-1})^T \mathbf{n}^T \)
- Could also use the adjoint \( \mathbf{n}' = \mathbf{n} \mathbf{T}^* \)
  - \( \mathbf{n}' \) not necessarily unit length even if \( \mathbf{n} \) is
  - But we’ll need the inverse anyway
Normals and implicit surfaces

- Affine coordinates
- Homogenous coordinates
Matrix Inverse

\[ A^{-1} = \begin{bmatrix} a & b & c & x \\ d & e & f & y \\ g & h & i & z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = (ST)^{-1} = \begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{T} \]

\[ \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \end{bmatrix} \begin{bmatrix} ei - hf & ch - bi & bf - ce \\ fg - di & ai - cg & cd - af \\ dh - eg & bg - ah & ae - bd \end{bmatrix} \]

Don’t need 1/|S| if just need direction of transformed normal. Will have to renormalize anyway if S not special unitary.

\[ S^{-1} = [\text{minors of } S]^T \]
Scene Graph

- Hierarchical representation of all objects in scene
- Transformation nodes
  - Intersect kids by $T^{-1} \mathbf{r}$
  - Returned normal $(T^{-1})^T \mathbf{n}$
  - Maintain $T^{-1}$ (not $T$)
Instancing

- Scene graph is a hierarchy
- Not necessarily a tree
- Directed acyclic graph (DAG)
- Nodes may have multiple parents
- *Instance*: Appearance of each node’s geometry in scene
Fun with Instancing

Image courtesy John Amanatides
Torus

• Product of two implicit circles
  \[(x - R)^2 + z^2 - r^2 = 0\]
  \[(x + R)^2 + z^2 - r^2 = 0\]
  \[
  ((x - R)^2 + z^2 - r^2)((x + R)^2 + z^2 - r^2)
  = (x^2 - 2Rx + R^2 + z^2 - r^2) (x^2 + 2Rx + R^2 + z^2 - r^2)
  = x^4 + 2x^2z^2 + z^4 - 2x^2r^2 - 2z^2r^2 + r^4 - 2x^2R^2 +
    2z^2R^2 - 2r^2R^2 + R^4
  = (x^2 + z^2 - r^2 - R^2)^2 + 4z^2R^2 - 4r^2R^2
  \]

• Surface of rotation: replace \(x^2\) with \(x^2 + y^2\)
  \[f(x,y,z) = (x^2 + y^2 + z^2 - r^2 - R^2)^2 + 4R^2(z^2 - r^2)\]

• Quartic!!!
• Up to four ray torus intersections
Ray-Object Intersection

- Returns intersection in a hit record
- “Next” field enables hit record to hold a list of intersections
- List only non-negative intersection parameters
- Ray always originates outside
  - If first $t = 0$ then ray originated inside
- Parity classifies ray segments
  - Odd segments “in”
  - Even segments “out”
Constructive Solid Geometry

- Construct shapes from primitives using boolean set operations
- Union: $A \cup B$, $A + B$, $A$ or $B$
- Intersection: $A \cap B$, $A*B$, $A$ and $B$
- Difference: $A\setminus B$, $A–B$, $A$ and not $B$
CSG Intersections

- List of $t$-values for A, B w/in-out classification
  - A.t_list = \{0.9, 3.1\} = \{0.9_{\text{in}}, 3.1_{\text{out}}\}
  - B.t_list = \{2.5, 4.5\} = \{2.5_{\text{in}}, 4.5_{\text{out}}\}
    - Use dot(r,d,n) to determine in,out
- Merge both lists into a single $t$-ordered list
  - \{ 0.9_{\text{in}}, 2.5_{\text{out}}, 3.1_{\text{in}}, 4.5_{\text{out}} \}
    - Keep track of A and B in/out classification
- Use Roth table to classify $t$-values
  - A+B = \{0.9_{\text{in}}, 2.5_{\text{in}}, 3.1_{\text{in}}, 4.5_{\text{out}}\} = \{0.9, 4.5\}
  - A*B = \{0.9_{\text{out}}, 2.5_{\text{in}}, 3.1_{\text{out}}, 4.5_{\text{out}}\} = \{2.5, 3.1\}
  - A-B = \{0.9_{\text{in}}, 2.5_{\text{out}}, 3.1_{\text{out}}, 4.5_{\text{out}}\} = \{0.9, 2.5\}
Accelerating Ray Intersections

• Q: Why is basic ray tracing so slow?
• A: It intersects every ray with every primitive in every object
• Q: How can we make ray tracing faster?
• A: Coherence

*Image coherence* – neighboring pixel probably display same object

*Spatial coherence* – neighboring points probably exhibit same appearance

*Temporal coherence* – Pixels in neighboring frames probably display same object

Stanford Bunny
~70K triangles

Do we need 70K ray-triangle intersections for each ray?
Shadow Caching

• Any interloper between surface point \( x \) and the light source \( s \) will cast a shadow
  – Doesn’t matter how many
  – Doesn’t matter which is closest
  – Stop ray intersections once \( any \) intersection found

• Neighboring shadowed surface points \( x \) and \( x' \) probably shadowed by the same object
  – Start shadow ray intersection search with object intersected in last shadow search
Bounding Volume

- Ray-bunny intersection takes 70K ray-triangle intersections even if ray misses the bunny
- Place a sphere around bunny
  - Ray $A$ misses sphere so ray $A$ misses bunny without checking 70K ray-triangle intersections
  - Ray $B$ intersects sphere but still misses bunny after checking 70K intersections
  - Ray $C$ intersects sphere and intersects bunny
- Can also use axis-aligned bounding box
  - Easier to create for triangle mesh
Bounding Volume Hierarchy

- Associate bounding volume with each node of scene graph
- If ray misses a node’s bounding volume, then no need to check any node beneath it
- If ray hits a node’s BV, then replace it with its children’s BV’s (or geometry)
- Breadth first search of tree
  - Maintain heap ordered by ray-BV intersection $t$-values
  - Explore children of node w/least pos. ray-BV $t$-value
Grids

- Encase object in a 3-D array of cubic cells
- Each cell contains list of all triangles it contains or intersects
- Rasterize ray to find which cells it intersects
  - 3D Bresenham algorithm
  - All cells that contain any part of ray
- Working from first ray-cell to last…
  - Find least positive intersect of ray with triangles in cell’s list
  - If no intersection, move on to next cell
Tagging

- Ray-object intersection test valid for ray with entire object
  - not just portion of object inside current cell
- Need only intersect object once for each ray
- In cell A – list = {#1}
  - Intersect r with #1? Yes
    - Miss [X] Tag #1 with no-intersection
- In cell B – list = {#2}
  - Intersect r with #2? Yes
    - ray r hits object #2 but later in cell C
    - Tag object #2 with intersection-at-C
- In cell C – list = {#1,#2}
  - Intersect r with #1? No (no-intersection)
  - Intersect r with #2? No (intersection-at-D)
- In cell D – list = {#2}
  - Intersect r with #2? No (intersection-at-D)
Other Partitioning Structures

- Octree
  - Ray can parse through large empty areas
  - Requires less space than grid
  - Subdivision takes time
- Binary Space Partition (BSP) Tree
  - Planes can divide models nearly in half
  - Trees better balanced, shallower
  - Added ray-plane intersections