Ray-Surface Intersection

CS 417 Lecture 33

Ray intersection: implicit surface

- Parametric ray

$$\mathbf{r}(t) = \mathbf{p} + t\mathbf{d}$$

- Implicit surface

$$f(\mathbf{x}) = 0$$

- Point belonging to intersection

$$\mathbf{x} = \mathbf{r}(t) \text{ and } f(\mathbf{x}) = 0$$
$$f(\mathbf{r}(t)) = 0$$
$$f(\mathbf{p} + t\mathbf{d}) = 0$$

Ray-sphere intersection: algebraic

- Begin with general form:

$$f(\mathbf{p} + t\mathbf{d}) = 0$$

- Parametric sphere:
  - assume unit sphere; other spheres in a moment…

$$\|\mathbf{x}\| = 1 \Leftrightarrow \|\mathbf{x}\|^2 = 1$$
$$f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{x} - 1 = 0$$

- Substitute:

$$(\mathbf{p} + t\mathbf{d}) \cdot (\mathbf{p} + t\mathbf{d}) - 1 = 0$$

  - this is a quadratic equation in *t*

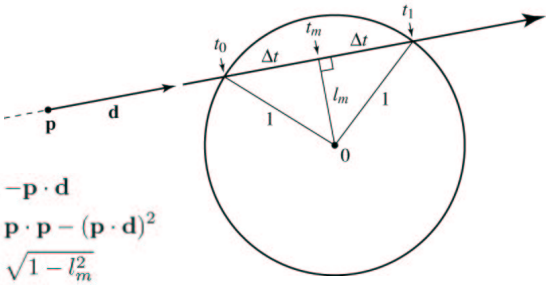
Ray-sphere intersection: algebraic

- Solution for *t* by quadratic formula:

$$t = \frac{-\mathbf{d} \cdot \mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \mathbf{p})^2 - (\mathbf{d} \cdot \mathbf{d})(\mathbf{p} \cdot \mathbf{p} - 1)}}{\mathbf{d} \cdot \mathbf{d}}$$
$$t = -\mathbf{d} \cdot \mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \mathbf{p})^2 - \mathbf{p} \cdot \mathbf{p} + 1}$$

  - simpler form holds when **d** is a unit vector
    but we won't assume this in practice (reason later)
  - I'll use the unit-vector form to make the geometric interpretation

Ray-sphere intersection: geometric



$$t_m = -\mathbf{p} \cdot \mathbf{d}$$
$$l_m^2 = \mathbf{p} \cdot \mathbf{p} - (\mathbf{p} \cdot \mathbf{d})^2$$
$$\Delta t = \sqrt{1 - l_m^2}$$
$$= \sqrt{(\mathbf{p} \cdot \mathbf{d})^2 - \mathbf{p} \cdot \mathbf{p} + 1}$$
$$t_{0,1} = t_m \pm \Delta t = -\mathbf{p} \cdot \mathbf{d} \pm \sqrt{(\mathbf{p} \cdot \mathbf{d})^2 - \mathbf{p} \cdot \mathbf{p} + 1}$$

Ray-sphere: normal and tex. coords

- Normal: gradient of *f*

$$\nabla f(\mathbf{x}) = \nabla \|\mathbf{x}\| = \mathbf{x}$$

  - this is obvious from the geometric standpoint

- Texture: use latitude-longitude parameterization
  - again for unit sphere:

$$\theta = \cos^{-1} y$$
$$\phi = \text{atan2}(-z, x)$$

  - where **x** = (*x*, *y*, *z*) is the intersection point

Ray-sphere intersection: examples

- Unit sphere, ray ((2,0,0), (-1,0,0))
- Unit sphere, ray ((2,1,0), (-1,0,0))
- Unit sphere, ray ((0,0,0), (-1,0,0))

Ray-triangle intersection

- Intersect plane first
  - implicit plane:

    $$f(\mathbf{x}) = (\mathbf{x} - \mathbf{a}) \cdot \mathbf{n}$$

  - substitute parametric ray and solve for t:

    $$(\mathbf{p} + t\mathbf{d} - \mathbf{a}) \cdot \mathbf{n} = 0$$

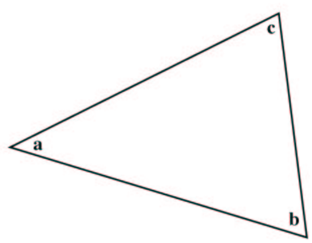
    $$t = \frac{(\mathbf{a} - \mathbf{p}) \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}$$

- Next check if point is inside triangle
  - first project both to plane
  - choose plane based on triangle normal

Ray-triangle intersection

- In plane, triangle is the intersection of 3 half spaces

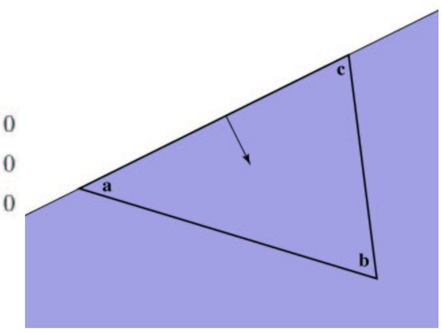
$$(\mathbf{x} - \mathbf{a}) \cdot (\mathbf{b} - \mathbf{a})^{\perp} > 0$$
$$(\mathbf{x} - \mathbf{b}) \cdot (\mathbf{c} - \mathbf{b})^{\perp} > 0$$
$$(\mathbf{x} - \mathbf{c}) \cdot (\mathbf{a} - \mathbf{c})^{\perp} > 0$$



Ray-triangle intersection

- In plane, triangle is the intersection of 3 half spaces

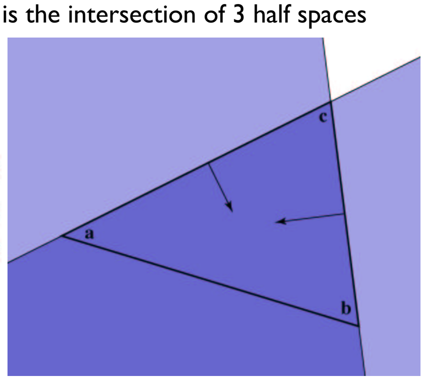
$$(\mathbf{x} - \mathbf{a}) \cdot (\mathbf{b} - \mathbf{a})^{\perp} > 0$$
$$(\mathbf{x} - \mathbf{b}) \cdot (\mathbf{c} - \mathbf{b})^{\perp} > 0$$
$$(\mathbf{x} - \mathbf{c}) \cdot (\mathbf{a} - \mathbf{c})^{\perp} > 0$$



Ray-triangle intersection

- In plane, triangle is the intersection of 3 half spaces

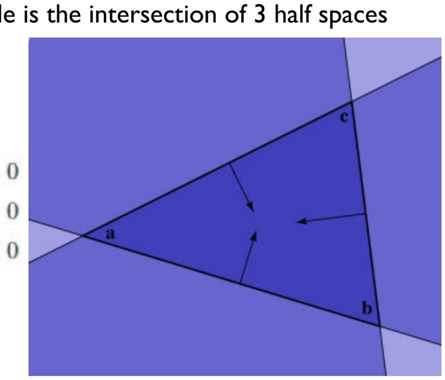
$$(\mathbf{x} - \mathbf{a}) \cdot (\mathbf{b} - \mathbf{a})^{\perp} > 0$$
$$(\mathbf{x} - \mathbf{b}) \cdot (\mathbf{c} - \mathbf{b})^{\perp} > 0$$
$$(\mathbf{x} - \mathbf{c}) \cdot (\mathbf{a} - \mathbf{c})^{\perp} > 0$$



Ray-triangle intersection

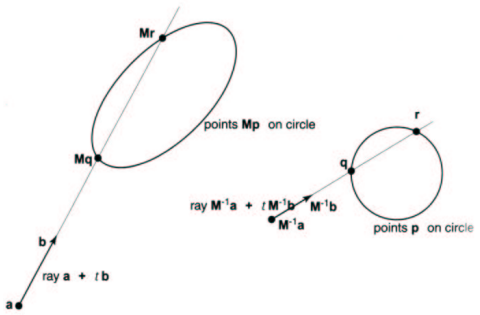
- In plane, triangle is the intersection of 3 half spaces

$$(\mathbf{x} - \mathbf{a}) \cdot (\mathbf{b} - \mathbf{a})^{\perp} > 0$$
$$(\mathbf{x} - \mathbf{b}) \cdot (\mathbf{c} - \mathbf{b})^{\perp} > 0$$
$$(\mathbf{x} - \mathbf{c}) \cdot (\mathbf{a} - \mathbf{c})^{\perp} > 0$$



Intersecting transformed primitives

- Transform ray inversely



[Shirley 2000]

Intersecting triangle meshes

- Basically, intersect each triangle
- But want to live off a compact data structure
  - this is one reason for using meshes in the first place
  - don't want to expand out to separate triangles!
  - to summarize space issue:
    - mesh: 3 ints + (vertex data) / 2 per triangle
    - separate: 3 * (vertex data) per triangle
    - position, normal, tex coords: (vertex data) = 8 floats
      - thus mesh costs 12 + 32/2 = 28 bytes/triangle
      - separate costs 32*3 = 96 bytes/triangle (3.4 x)

Intersecting triangle meshes

- Second reason for mesh: continuity
  - shared positions reduce potential for cracks
  - want shared normals for smooth shading
  - want shared texture coordinates for continuity
- This means we need to interpolate things across the triangle for every intersection calculation.
  - generally use linear interpolation
  - implement this using barycentric coordinates

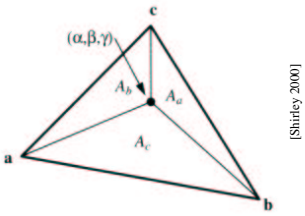
Barycentric coordinates

- A coordinate system for triangles
  - algebraic viewpoint:

$$\mathbf{p} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$
$$\alpha + \beta + \gamma = 1$$

  - geometric viewpoint (areas):
- Triangle interior test:

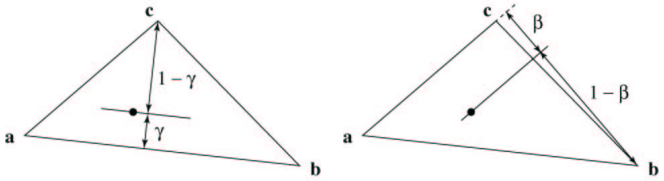
$$\alpha > 0; \quad \beta > 0; \quad \gamma > 0$$



[Shirley 2000]

Barycentric coordinates

- A coordinate system for triangles
  - geometric viewpoint: distances

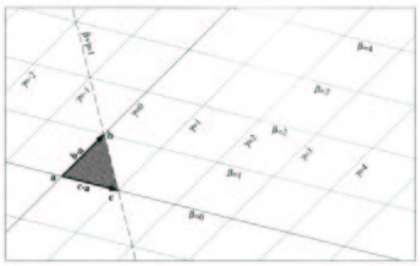


  - linear viewpoint: basis of edges

$$\alpha = 1 - \beta - \gamma$$
$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

Barycentric coordinates

- Linear viewpoint: basis for the plane



[Shirley 2000]

  - in this view, the triangle interior test is just

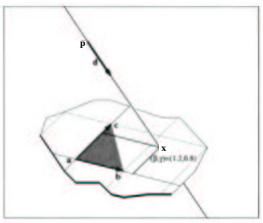
$$\beta > 0; \quad \gamma > 0; \quad \beta + \gamma < 1$$

Barycentric ray-tri. intersection

- Directly compute barycentric coords of ray-plane intersection point

$$\begin{bmatrix} | & | & | \\ (\mathbf{b}-\mathbf{a}) & (\mathbf{c}-\mathbf{a}) & -\mathbf{d} \\ | & | & | \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \mathbf{p}-\mathbf{a}$$

  - solve this 3x3 matrix system
  - result gives ray parameter and barycentric coords in one step
  - geometric interpretation



[Shirley 2000]

Interpolating across triangles

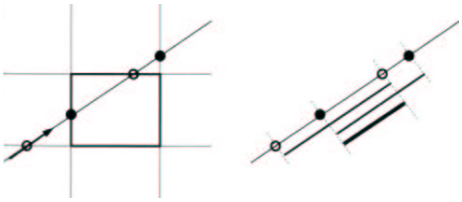
- Intersection point is written in triangle basis
  - makes inside-triangle test simple
  - also makes linear interpolation simple
- Linear interpolation
  - idea, again: values of some quantity at the three vertices define a unique affine function on the plane
  - in barycentric basis it is easy to compute this function
    - want value v_a at **a**, v_b at **b**, v_c at **c**
    - affine equation is $a + \beta b + \gamma c$
    - therefore coefficients are v_a, $v_b - v_a$, $v_c - v_a$
    - resemblance to point equation is no accident

Barycentric intersection summary

- Given: triangle (**a**, **b**, **c**) and ray **p** + *t***d**
- First form a 3x3 matrix system and solve it
  - this gives you the barycentric coords ant the ray parameter
- Next use the barycentric coords to interpolate
  - position
  - normals
  - texture coordinates
  - anything else you want
  - (all referring to the shared vertex data in the mesh)
- And return the whole lot (in an intersection record)

Intersecting other simple shapes

- Cube



[Shirley 2000]

- Cylinder
  - intersect circle in 2D, plus two end caps
- Cone
  - left as an exercise

Intersection against many shapes

- The basic idea is:

```
tMin = +infinity; hitShape = null;
for shape in shapeList {
   t = intersect(ray, shape);
   if t < tMin {
      tMin = t;
      hitShape = shape;
   }
}
```

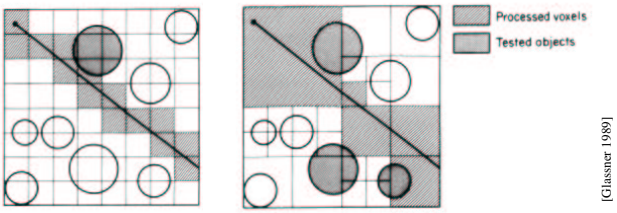
  - but this is linear in the number of shapes

Accelerating ray-scene intersection

- Intersecting a ray against all primitives spends much time intersecting against far-away primitives
- A spatial data structure helps by finding only the primitives that are near the ray
  - generally they work by partitioning 3D space into a number of regions and keeping track of which objects intersect which regions.
  - ray traversal then amounts to enumerating the regions intersected by the ray.

Acceleration by spatial subdivision

- Use a collection of axis-aligned boxes as the regions
- Variants: regular (grid) and hierarchical (kdtree/octree)



[Glassner 1989]

- Alternative: bounding volume hierarchy

More complex implicit surfaces

- E.g. defined by analytic functions or volume data
- Implicit intersection equation solved directly using root finding (secant algorithm is popular)

$$f(\mathbf{p} + t\mathbf{d}) = 0$$

Enright et al. 2002]

