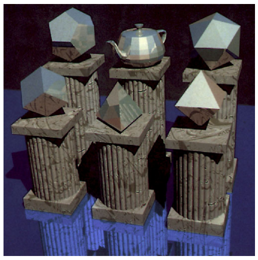
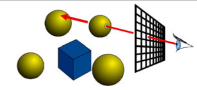
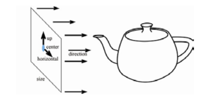
Ray Casting II

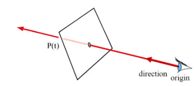


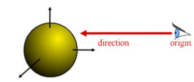
Last Time?

- Ray Casting / Tracing
- Orthographic Camera
- Ray Representation
  - P(t) = origin + t * direction
- Ray-Sphere Intersection
- Ray-Plane Intersection
- Implicit vs. Explicit Representations







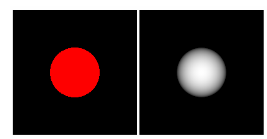


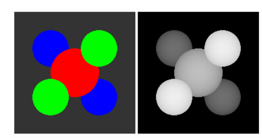
Explicit vs. Implicit?

- Explicit
  - Parametric
  - Generates points
  - Hard to verify that a point is on the object
- Implicit
  - Solution of an equation
  - Does not generate points
  - Verifies that a point is on the object

Assignment 1: Ray Casting

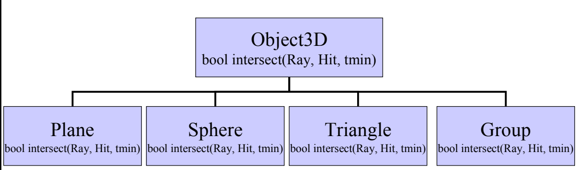
- Write a basic ray caster
  - Orthographic camera
  - Sphere Intersection
  - Main loop rendering
  - 2 Display modes: color and distance
- We provide:
  - Ray (origin, direction)
  - Hit (t, Material)
  - Scene Parsing





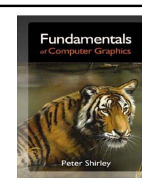
Object-Oriented Design

- We want to be able to add primitives easily
  - Inheritance and virtual methods
- Even the scene is derived from Object3D!



Graphics Textbooks

- Recommended for 6.837:
  Peter Shirley
  Fundamentals of Computer Graphics
  AK Peters



- Ray Tracing

Linear Algebra Review Session

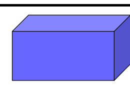
- Monday Sept. 20 (this Monday!)
- Room 2-105 (we hope)
- 7:30 – 9 PM

Questions?

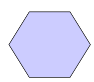
Image by Henrik Wann Jensen

Overview of Today

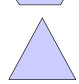
- Ray-Box Intersection



- Ray-Polygon Intersection



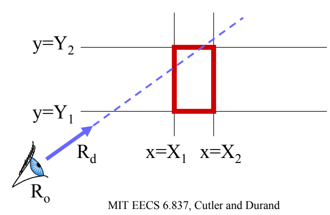
- Ray-Triangle Intersection



- Ray-Bunny Intersection
  & extra topics…

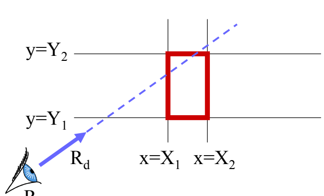
Ray-Box Intersection

- Axis-aligned
- Box: $(X_1, Y_1, Z_1) \rightarrow (X_2, Y_2, Z_2)$
- Ray: $P(t) = R_o + tR_d$



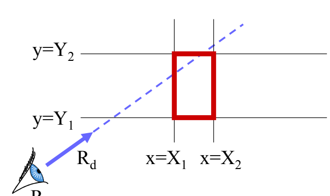
Naïve Ray-Box Intersection

- 6 plane equations: compute all intersections
- Return closest intersection inside the box
  - Verify intersections are on the correct side of each plane: $Ax+By+Cz+D < 0$



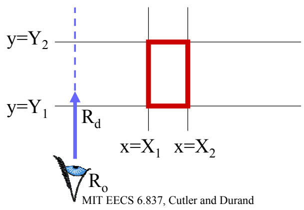
Reducing Total Computation

- Pairs of planes have the same normal
- Normals have only one non-0 component
- Do computations one dimension at a time



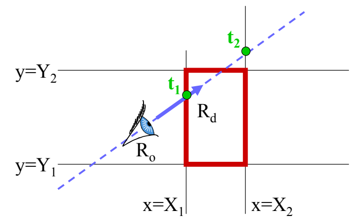
Test if Parallel

- If $R_{dx} = 0$ (ray is parallel) AND $R_{ox} < X_1$ or $R_{ox} > X_2$ → **no intersection**



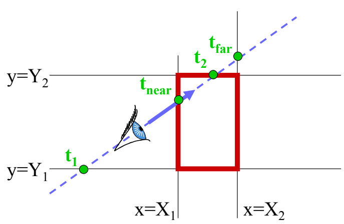
Find Intersections Per Dimension

- Calculate intersection distance t_1 and t_2
  - $t_1 = (X_1 - R_{ox}) / R_{dx}$
  - $t_2 = (X_2 - R_{ox}) / R_{dx}$



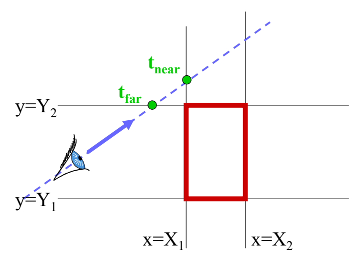
Maintain t_{near} & t_{far}

- Closest & farthest intersections *on the object*
  - If $t_1 > t_{near}$, $t_{near} = t_1$
  - If $t_2 < t_{far}$, $t_{far} = t_2$



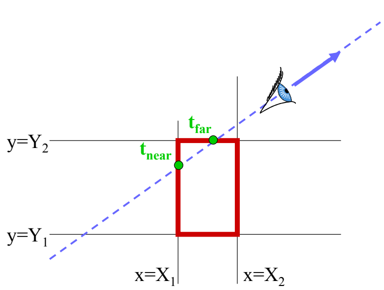
Is there an Intersection?

- If $t_{near} > t_{far}$ → **box is missed**



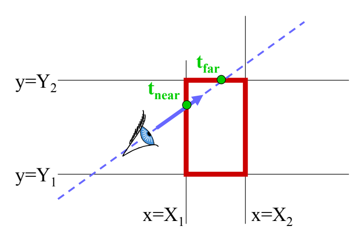
Is the Box Behind the Eyepoint?

- If $t_{far} < t_{min}$ → **box is behind**



Return the Correct Intersection

- If $t_{near} > t_{min}$ → **closest intersection at t_{near}**
- Else → **closest intersection at t_{far}**



Ray-Box Intersection Summary

- For each dimension,
  - If $R_{dx} = 0$ (ray is parallel) AND $R_{ox} < X_1$ or $R_{ox} > X_2$ → **no intersection**
- For each dimension, calculate intersection distances t_1 and t_2
  - $t_1 = (X_1 - R_{ox}) / R_{dx}$ $t_2 = (X_2 - R_{ox}) / R_{dx}$
  - If $t_1 > t_2$, swap
  - Maintain t_{near} and t_{far} (closest & farthest intersections so far)
  - If $t_1 > t_{near}$, $t_{near} = t_1$ If $t_2 < t_{far}$, $t_{far} = t_2$
- If $t_{near} > t_{far}$ → **box is missed**
- If $t_{far} < t_{min}$ → **box is behind**
- If $t_{near} > t_{min}$ → **closest intersection at t_{near}**
- Else → **closest intersection at t_{far}**

Efficiency Issues

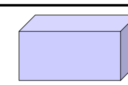
- $1/R_{dx}$, $1/R_{dy}$ and $1/R_{dz}$ can be pre-computed and shared for many boxes
- Unroll the loop
  - Loops are costly (because of termination if)
  - Avoid the t_{near} & t_{far} comparison for first dimension

Questions?

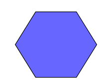
Image by Henrik Wann Jensen

Overview of Today

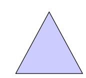
- Ray-Box Intersection



- Ray-Polygon Intersection



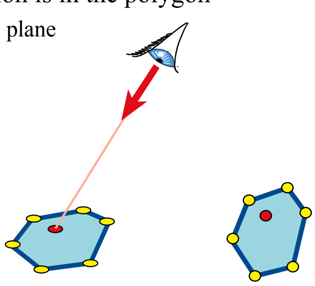
- Ray-Triangle Intersection



- Ray-Bunny Intersection & extra topics…

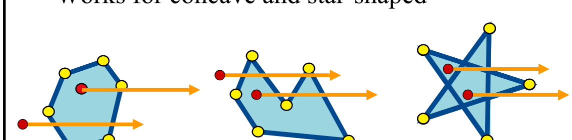
Ray-Polygon Intersection

- Ray-plane intersection
- Test if intersection is in the polygon
  - Solve in the 2D plane



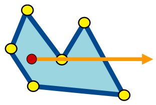
Point Inside/Outside Polygon

- Ray intersection definition:
  - Cast a ray in any direction
    - (axis-aligned is smarter)
  - Count intersections
  - If odd number, point is inside
- Works for concave and star-shaped



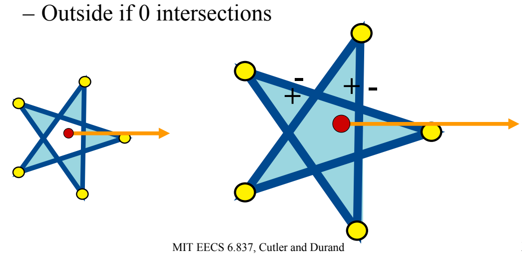
Precision Issue

- What if we intersect a vertex?
  - We might wrongly count an intersection for exactly one adjacent edge
- Decide that the vertex is always above the ray



Winding Number

- To solve problem with star pentagon:
  - Oriented edges
  - *Signed count* of intersections
  - Outside if 0 intersections

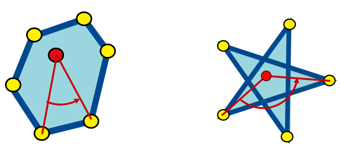


Alternative Definition

- Sum of the signed angles from point to edges

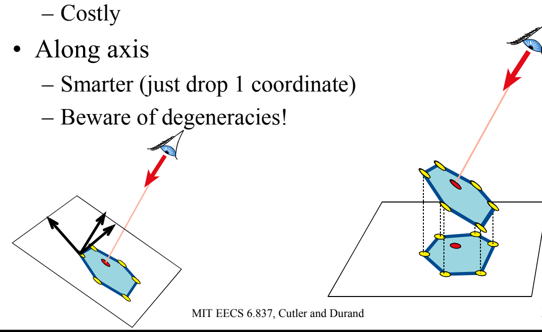
  ±360°, ± 720°, … **→ point is inside**

  0° **→ point is outside**



How Do We Project into 2D?

- Along normal
  - Costly
- Along axis
  - Smarter (just drop 1 coordinate)
  - Beware of degeneracies!



Questions?

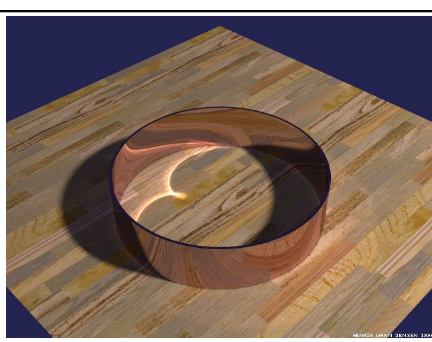
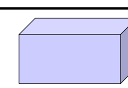


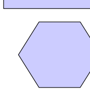
Image by Henrik Wann Jensen

Overview of Today

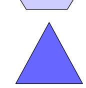
- Ray-Box Intersection



- Ray-Polygon Intersection



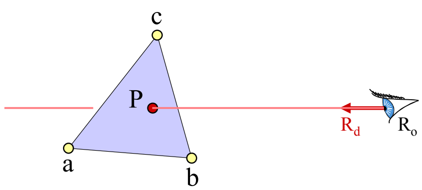
- Ray-Triangle Intersection



- Ray-Bunny Intersection
  & extra topics…

Ray-Triangle Intersection

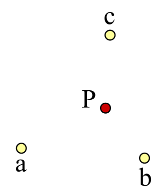
- Use ray-polygon
- Or try to be smarter
  - Use barycentric coordinates



Barycentric Definition of a Plane

[Möbius, 1827]

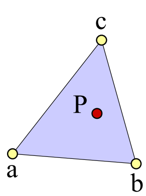
- $P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$
  with $\alpha + \beta + \gamma = 1$
- Is it explicit or implicit?



P is the *barycenter*: the single point upon which the plane would balance if weights of size α, β, & γ are placed on points a, b, & c.

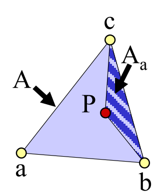
Barycentric Definition of a Triangle

- $P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$
  with $\alpha + \beta + \gamma = 1$
- AND $0 < \alpha < 1$ & $0 < \beta < 1$ & $0 < \gamma < 1$



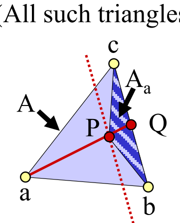
How Do We Compute α, β, γ ?

- Ratio of opposite sub-triangle area to total area
  - $\alpha = A_a/A$ $\beta = A_b/A$ $\gamma = A_c/A$
- Use signed areas for points outside the triangle



Intuition Behind Area Formula

- P is barycenter of a and Q
- A_a is the interpolation coefficient on $\overline{aQ}$
- All points on lines parallel to $\overline{bc}$ have the same α (All such triangles have same height/area)



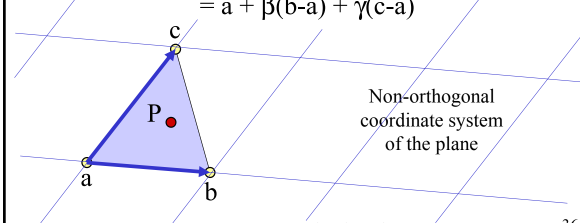
Simplify

- Since $\alpha + \beta + \gamma = 1$, we can write $\alpha = 1 - \beta - \gamma$

$P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$ rewrite

$P(\beta, \gamma) = (1-\beta-\gamma)a + \beta b + \gamma c$

$= a + \beta(b-a) + \gamma(c-a)$



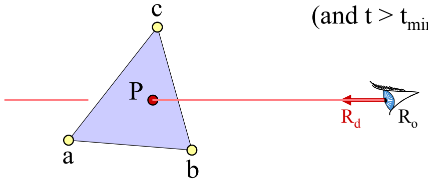
Intersection with Barycentric Triangle

- Set ray equation equal to barycentric equation

$$P(t) = P(\beta, \gamma)$$

$$R_o + t * R_d = a + \beta(b-a) + \gamma(c-a)$$

- Intersection if $\beta + \gamma < 1$ & $\beta > 0$ & $\gamma > 0$ (and $t > t_{min}$ …)



Intersection with Barycentric Triangle

- $R_o + t * R_d = a + \beta(b-a) + \gamma(c-a)$

$$R_{ox} + tR_{dx} = a_x + \beta(b_x - a_x) + \gamma(c_x - a_x)$$
$$R_{oy} + tR_{dy} = a_y + \beta(b_y - a_y) + \gamma(c_y - a_y)$$
$$R_{oz} + tR_{dz} = a_z + \beta(b_z - a_z) + \gamma(c_z - a_z)$$

3 equations, 3 unknowns

- Regroup & write in matrix form:

$$\begin{bmatrix} a_x - b_x & a_x - c_x & R_{dx} \\ a_y - b_y & a_y - c_y & R_{dy} \\ a_z - b_z & a_z - c_z & R_{dz} \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} a_x - R_{ox} \\ a_y - R_{oy} \\ a_z - R_{oz} \end{bmatrix}$$

Cramer's Rule

- Used to solve for one variable at a time in system of equations

$$\beta = \frac{\begin{vmatrix} a_x - R_{ox} & a_x - c_x & R_{dx} \\ a_y - R_{oy} & a_y - c_y & R_{dy} \\ a_z - R_{oz} & a_z - c_z & R_{dz} \end{vmatrix}}{|A|} \qquad \gamma = \frac{\begin{vmatrix} a_x - b_x & a_x - R_{ox} & R_{dx} \\ a_y - b_y & a_y - R_{oy} & R_{dy} \\ a_z - b_z & a_z - R_{oz} & R_{dz} \end{vmatrix}}{|A|}$$

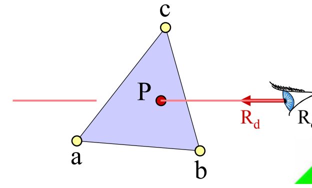
$$t = \frac{\begin{vmatrix} a_x - b_x & a_x - c_x & a_x - R_{ox} \\ a_y - b_y & a_y - c_y & a_y - R_{oy} \\ a_z - b_z & a_z - c_z & a_z - R_{oz} \end{vmatrix}}{|A|}$$

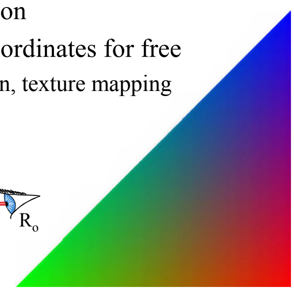
| | denotes the determinant

Can be copied mechanically into code

Advantages of Barycentric Intersection

- Efficient
- Stores no plane equation
- Get the barycentric coordinates for free
  - Useful for interpolation, texture mapping



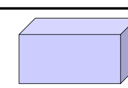


Questions?

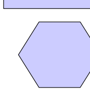
- Image computed using the RADIANCE system by Greg Ward

Overview of Today

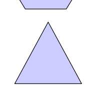
- Ray-Box Intersection



- Ray-Polygon Intersection



- Ray-Triangle Intersection



- Ray-Bunny Intersection & extra topics…

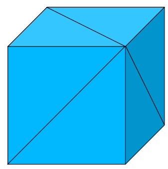
Triangle Meshes (.obj)

vertices

```
v -1 -1 -1
v  1 -1 -1
v -1  1 -1
v  1  1 -1
v -1 -1  1
v  1 -1  1
v -1  1  1
v  1  1  1
```

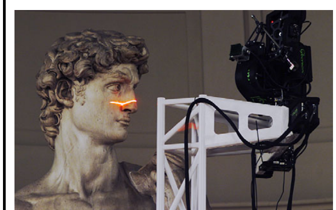
triangles

```
f 1 3 4
f 1 4 2
f 5 6 8
f 5 8 7
f 1 2 6
f 1 6 5
f 3 7 8
f 3 8 4
f 1 5 7
f 1 7 3
f 2 4 8
f 2 8 6
```



Acquiring Geometry

- 3D Scanning

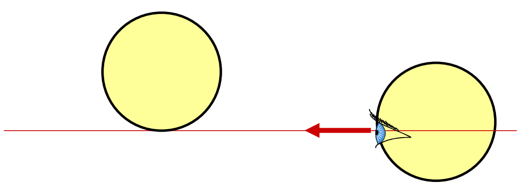


Digital Michealangelo Project (Stanford)

Cyberware

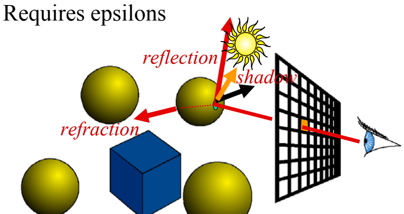
Precision

- What happens when
  - Origin is on an object?
  - Grazing rays?
- Problem with floating-point approximation



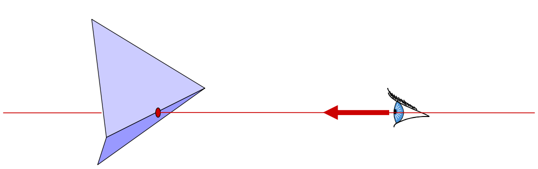
The evil ε

- In ray tracing, do NOT report intersection for rays starting at the surface (no false positive)
  - Because secondary rays
  - Requires epsilons



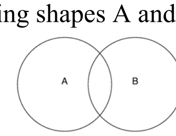
The evil ε: a hint of nightmare

- Edges in triangle meshes
  - Must report intersection (otherwise not watertight)
  - No false negative

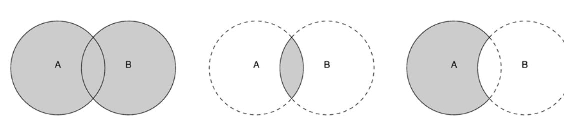


Constructive Solid Geometry (CSG)

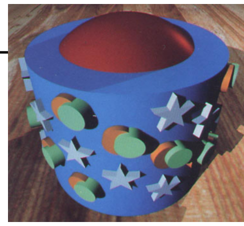
Given overlapping shapes A and B:

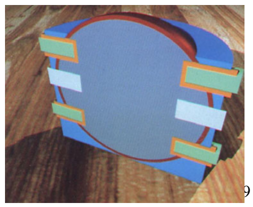


Union Intersection Subtraction

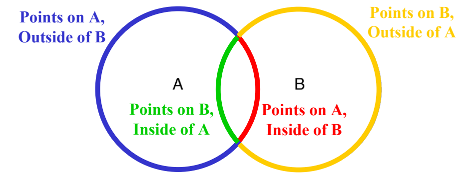


For example:

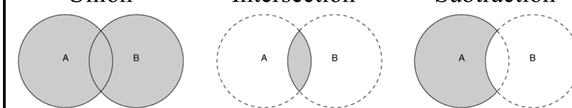




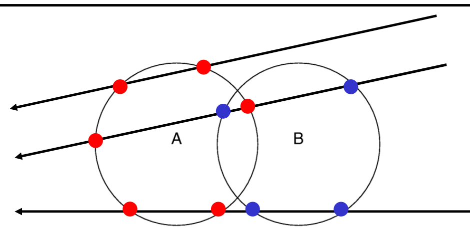
How can we implement CSG?



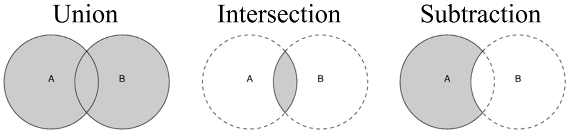
Union | Intersection | Subtraction



Collect all the intersections

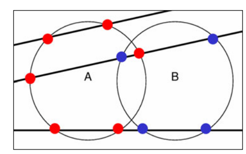


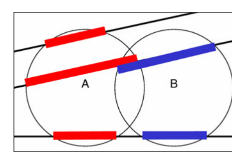
Union | Intersection | Subtraction



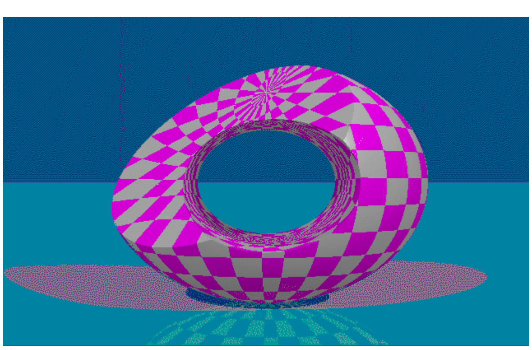
Implementing CSG

1. Test "inside" intersections:
   - Find intersections with A, test if they are inside/outside B
   - Find intersections with B, test if they are inside/outside A
2. Overlapping intervals:
   - Find the intervals of "inside" along the ray for A and B
   - Compute union/intersection/subtraction of the intervals





Early CSG Raytraced Image



Questions?

Next week: Transformations