



## Ray Casting

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## Ray Casting Algorithm

For each pixel

1. Compute ray from eye through pixel
2. For each primitive
  - Test for ray-object intersection
3. Shade pixel using nearest primitive (or set to background color)

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## Computing the Rays

Choose eye point, view direction, up direction, fields of view (x and y)

$p_t = \text{eye} + t * v$  ( $v$  typically normalized)

Compute rays to two opposite corners

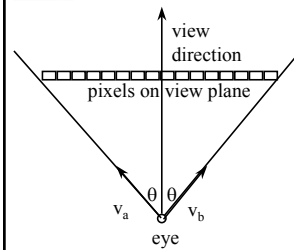
Compute step sizes,  $\Delta x$  and  $\Delta y$  to go from pixel to pixel

To compute new ray: take step, then normalize

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## 2D ray calculation



view is normalized  
view direction

$\text{right} = (\text{view}_y, -\text{view}_x)$   
 $v_a = \text{view} - \tan\theta * \text{right}$   
 $v_b = \text{view} + \tan\theta * \text{right}$   
 $\text{step} = (v_b - v_a) / \text{num\_pixels}$   
 $v_0 = v_a + \text{step} / 2$   
 $v_i = v_{i-1} + \text{step}$

Note: take equal-sized steps in viewing plane, not equal angles!

In 3D, we have an additional step size and field-of-view angle as well as an up vector.

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## Computing Intersections

Ray is in parametric form ( $t$  is parameter)

$$x = p + t * v$$

Represent primitive in implicit form:

$$f(x,y,z) = 0$$

(any  $(x,y,z)$  on surface evaluates to zero)

Substitute  $(x,y,z)$  of ray into  $f(x,y,z)$  and solve for  $t$

- degree  $n$  implicit function will be degree  $n$  in  $t$
- quadric surfaces may be solved with quadratic equation -- pick real solution closest to eye

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## Example Quadric Functions

$$\text{Sphere: } (x-a)^2 + (y-b)^2 + (z-c)^2 - r^2 = 0$$

Circular cylinder (parallel to  $z$ -axis):

$$(x-a)^2 + (y-b)^2 - r^2 = 0$$

Hyperbolic paraboloid:

$$y^2/b^2 - x^2/a^2 - z = 0$$

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## General Quadrics

General quadric has form:

$$Ax^2 + 2Bxy + 2Cxz + 2Dx + Ey^2 + 2Fyz + 2Gy + Hz^2 + 2Iz + J = 0$$

or...

$x^t Q x = 0$ , where  $x^t = [x \ y \ z \ 1]$  and

$$Q = \begin{bmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \\ D & G & I & J \end{bmatrix}$$

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## Quadric Intersections

Quadric:  $x^t Q x = 0$

Ray:  $x = p + tv$

Substituting ray for  $x$ :

$$(p + tv)^t Q (p + tv) = 0$$

$$p^t Q p + p^t Q tv + tv^t Q p + tv^t Q tv = 0$$

$$(v^t Q v)t^2 + (p^t Q v + v^t Q p)t + p^t Q p = 0$$

$$(v^t Q v)t^2 + (2v^t Q p)t + p^t Q p = 0$$

(Q is symmetric)

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## Common Ray-tracing Primitives

Sphere, ellipsoid

Cylinders

Plane, triangle

- $Ax + By + Cz + D = 0$

Torus

Bezier/Nurbs patches

- parametric, so use implicit form of ray

—intersection of two planes

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## Local Illumination Shading

Compute normal at closest intersection

- $\nabla f = (\partial_x, \partial_y, \partial_z)$  is normal vector field for implicit function,  $f$

For each light

- Use position and normal to compute light contribution
- Accumulate light contributions

Color pixel

- Clamp to avoid overflow

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## Shadows

Only add contribution from a light if it is visible from the point (and vice versa)

- test for intersections along ray in L direction
- accumulate contribution if no occlusion

(illumination is no longer totally local)

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## Truncating Primitives


Use another implicit function

- Test which side of the implicit function the intersection is on
- Keep intersection only if it is on the correct side

For example, truncate a cylinder using two plane equations (or perhaps a sphere)

- then cap using the two planes truncated by the cylinder

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen

 **Constructive Solid Geometry**

---

**Perform hierarchical set operations on primitives**


**Union:**  $\cup$

**Intersection:**  $\cap$

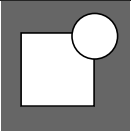
**Difference:**  $-$


---

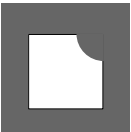
Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen

 **CSG Operators**

---


$\text{Square} \cup \text{Circle} =$  

$\text{Square} \cap \text{Circle} =$  

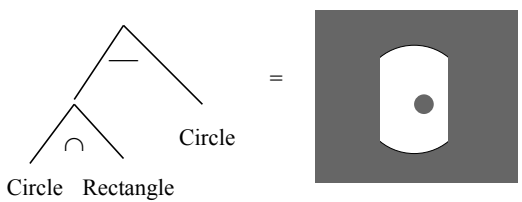
$\text{Square} - \text{Circle} =$  

---

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen


 **CSG Hierarchy**

---



---

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen

 **Ray Tracing CSG**

---

**Each "object" may be a primitive or a CSG hierarchy**


**Find all ray-primitive intersections for hierarchy**

**Use CSG operators to determine which intervals are solid or vacant**

**Use start of nearest solid interval as ray-object intersection**

---

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen

 **CSG Tracing Algorithm**

---

**Start at root of CSG Hierarchy**


**Trace ray through left child - result is ordered list of intersections, forming solid and vacant intervals**

**Trace ray through right child**

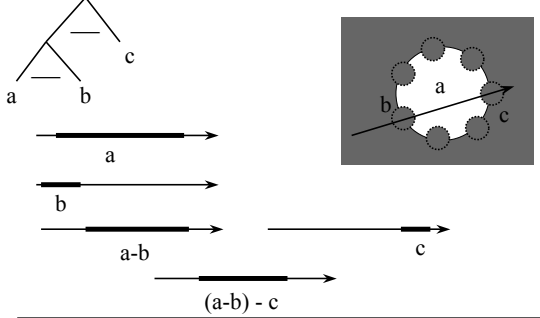
**Merge lists of intersections/intervals by applying CSG operator of current node**

---

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen

 **CSG Example - golf ball**

---



---

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



## **Some CSG Details**

---

**Each interval endpoint associated with  
intersection of ray with some surface**

**Normal computed from surface of  
intersection**

**Material parameters may come from either  
primitive**

---

Johns Hopkins Department of Computer Science  
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen