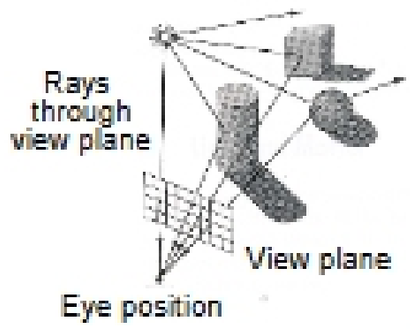
Ray Casting

Tom Funkhouser
Princeton University
COS 426, Spring 2006

3D Rendering

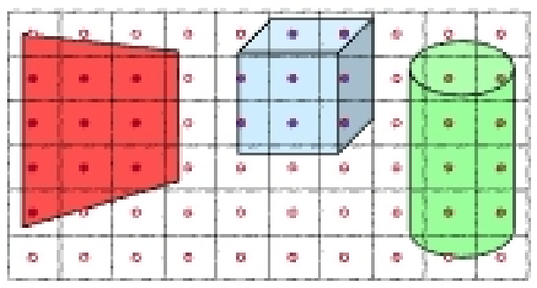
- The color of each pixel on the view plane depends on the radiance emanating from visible surfaces

Simplest method is ray casting



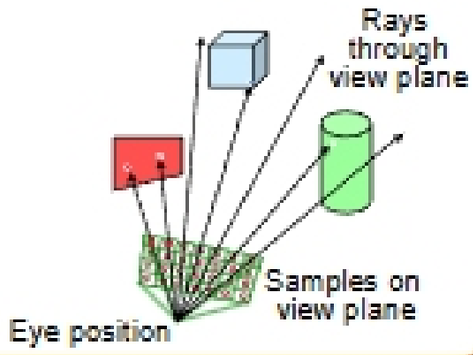
Ray Casting

- For each sample …
  - Construct ray from eye position through view plane
  - Find first surface intersected by ray through pixel
  - Compute color sample based on surface radiance



Ray Casting

- For each sample …
  - Construct ray from eye position through view plane
  - Find first surface intersected by ray through pixel
  - Compute color sample based on surface radiance



Ray Casting

- Simple implementation:

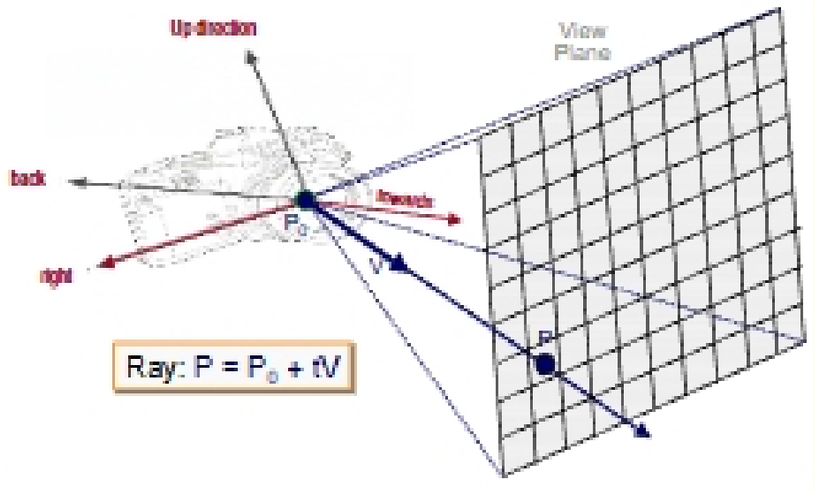
```
Image RayCast(Camera camera, Scene scene, int width, int height)
{
    Image image = new Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Intersection hit = FindIntersection(ray, scene);
            image[i][j] = GetColor(hit);
        }
    }
    return image;
}
```

Ray Casting

- Simple implementation:

```
Image RayCast(Camera camera, Scene scene, int width, int height)
{
    Image image = new Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Intersection hit = FindIntersection(ray, scene);
            image[i][j] = GetColor(hit);
        }
    }
    return image;
}
```

Constructing Ray Through a Pixel



Constructing Ray Through a Pixel

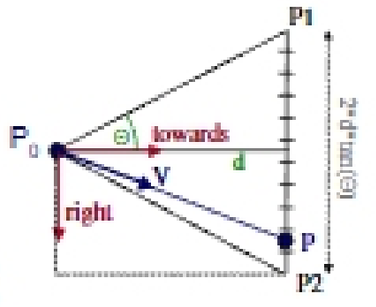
- 2D Example

Θ = frustum half-angle
d = distance to view plane

right = towards x up

$P1 = P_0 + d*towards - d*tan(\Theta)*right$
$P2 = P_0 + d*towards + d*tan(\Theta)*right$

$P = P1 + ((i + 0.5) / width) * (P2 - P1)$
$V = (P - P_0) / \|P - P_0\|$



Ray: $P = P_0 + tV$

Ray Casting

- Simple implementation:

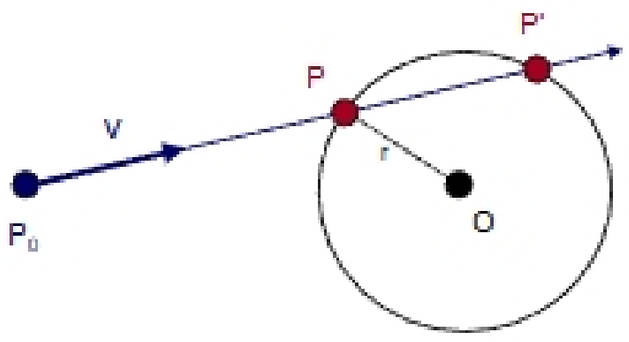
```
Image RayCast(Camera camera, Scene scene, int width, int height)
{
    Image image = new Image(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            Ray ray = ConstructRayThroughPixel(camera, i, j);
            Intersection hit = FindIntersection(ray, scene);
            image[i][j] = GetColor(hit);
        }
    }
    return image;
}
```

Ray-Scene Intersection

- Intersections with geometric primitives
  - Sphere
  - Triangle
  - Groups of primitives (scene)
- Acceleration techniques
  - Bounding volume hierarchies
  - Spatial partitions
    - » Uniform grids
    - » Octrees
    - » BSP trees

Ray-Sphere Intersection

Ray: $P = P_0 + tV$
Sphere: $|P - O|^2 - r^2 = 0$



Ray-Sphere Intersection I

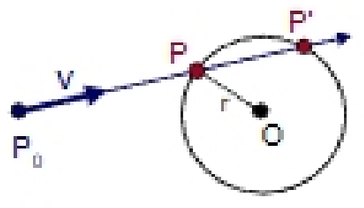
Ray: $P = P_0 + tV$
Sphere: $|P - O|^2 - r^2 = 0$

Algebraic Method

Substituting for P, we get:
$|P_0 + tV - O|^2 - r^2 = 0$

Solve quadratic equation:
$at^2 + bt + c = 0$
where:
$a = 1$
$b = 2V \cdot (P_0 - O)$
$c = |P_0 - C|^2 - r^2 = 0$

$P = P_0 + tV$



Ray-Sphere Intersection II

Ray: $P = P_0 + tV$
Sphere: $|P - O|^2 - r^2 = 0$

Geometric Method

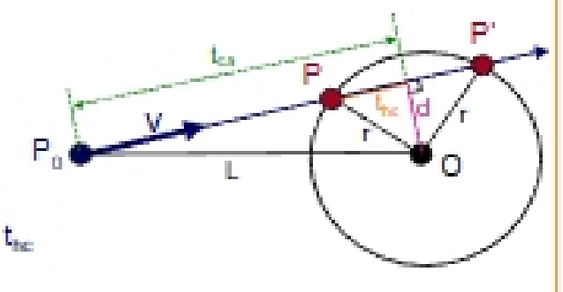
$L = O - P_0$

$t_{ca} = L \cdot V$
if ($t_{ca} < 0$) return 0

$d^2 = L \cdot L - t_{ca}^2$
if ($d^2 > r^2$) return 0

$t_{hc} = \sqrt{r^2 - d^2}$
$t = t_{ca} - t_{hc}$ and $t_{ca} + t_{hc}$

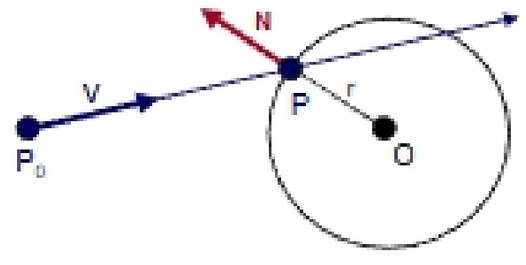
$P = P_0 + tV$



Ray-Sphere Intersection

- Need normal vector at intersection for lighting calculations

$N = (P - O) / \|P - O\|$

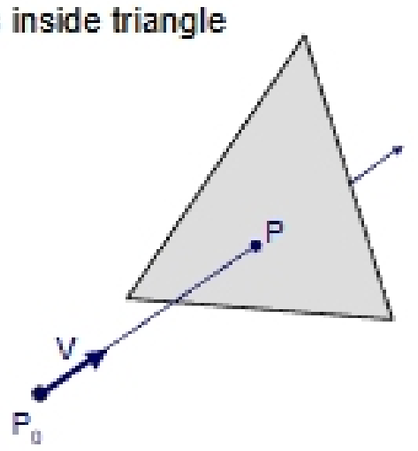


Ray-Scene Intersection

- Intersections with geometric primitives
  - Sphere
  - » Triangle
  - Groups of primitives (scene)
- Acceleration techniques
  - Bounding volume hierarchies
  - Spatial partitions
    - » Uniform grids
    - » Octrees
    - » BSP trees

Ray-Triangle Intersection

- First, intersect ray with plane
- Then, check if point is inside triangle



Ray-Plane Intersection

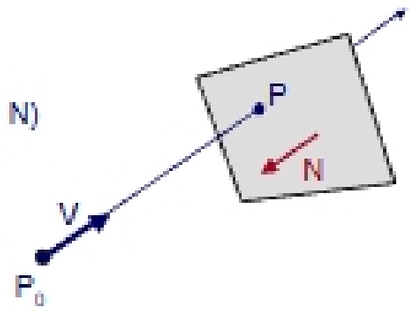
Ray: $P = P_0 + tV$
Plane: $P \cdot N + d = 0$

Algebraic Method

Substituting for P, we get:
$(P_0 + tV) \cdot N + d = 0$

Solution:
$t = -(P_0 \cdot N + d) / (V \cdot N)$

$P = P_0 + tV$



Ray-Triangle Intersection I

- Check if point is inside triangle algebraically

For each side of triangle
  $V_1 = T_1 - P$
  $V_2 = T_2 - P$
  $N_1 = V_2 \times V_1$
  Normalize N_1
  if $((P - P_0) \cdot N_1 < 0)$
    return FALSE;
end