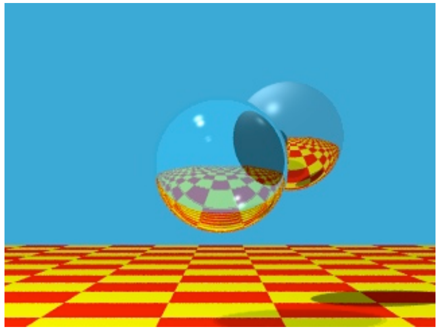
Interactive Computer Graphics

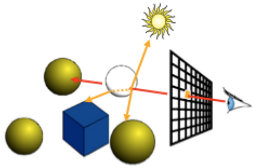
Lecture 12: Ray tracing (cont.)

Some slides adopted from
H. Pfister, Harvard

Ray tracing - Summary

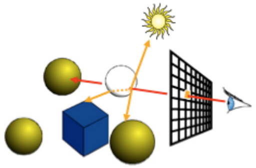


Ray tracing - Summary



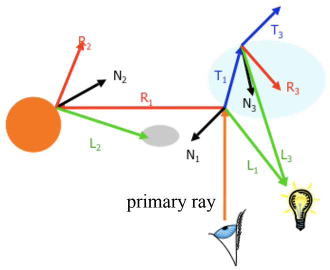
```
trace ray
      Intersect all objects
      color = ambient term
      For every light
            cast shadow ray
            col += local shading term
      If mirror
            col += k_refl * trace reflected ray
      If transparent
            col += k_trans * trace transmitted ray
```

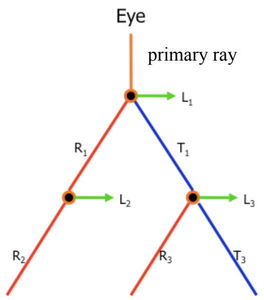
Ray tracing - Summary



```
trace ray
      Intersect all objects
      color = ambient term
      For every light
            cast shadow ray
            col += local shading term
      If mirror
            col += k_refl * trace reflected ray
      If transparent
            col += k_trans * trace transmitted ray
```

Ray tracing - Summary





R_i reflected ray
L_i shadow ray
T_i transmitted (refracted) ray

secondary rays

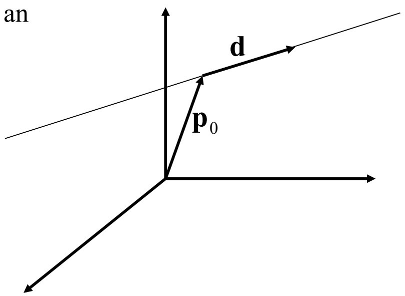
Intersection calculations

- For each ray we must calculate all possible intersections with each object inside the viewing volume
- For each ray we must find the nearest intersection point
- We can define our scene using
  - Solid models
    - sphere
    - cylinder
  - Surface models
    - plane
    - triangle
    - polygon

Rays

- Rays are parametric lines
- Rays can be defined an
  - origin $\mathbf{p_0}$
  - direction $\mathbf{d}$
- Equation of ray:

$$\mathbf{p}(\mu) = \mathbf{p}_0 + \mu\mathbf{d}$$



Ray tracing: Intersection calculations

- The coordinates of any point along each primary ray are given by:

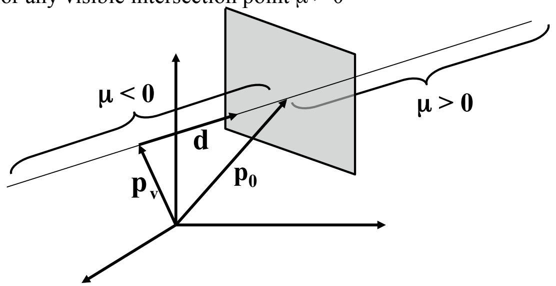
$$\mathbf{p} = \mathbf{p}_0 + \mu\mathbf{d}$$

  - $\mathbf{p}_0$ is the current pixel on the viewing plane.
  - $\mathbf{d}$ is the direction vector and can be obtained from the position of the pixel on the viewing plane $\mathbf{p}_0$ and the viewpoint $\mathbf{p}_\mathbf{v}$:

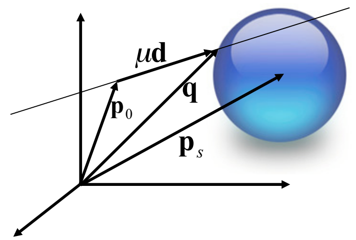
$$\mathbf{d} = \frac{\mathbf{p}_0 - \mathbf{p}_\mathbf{v}}{|\mathbf{p}_0 - \mathbf{p}_\mathbf{v}|}$$

Ray tracing: Intersection calculations

- The viewing ray can be parameterized by μ:
  - μ > 0 denotes the part of the ray behind the viewing plane
  - μ < 0 denotes the part of the ray in front of the viewing plane
  - For any visible intersection point μ > 0



Intersection calculations: Spheres



For any point on the surface of the sphere

$$|\mathbf{q} - \mathbf{p}_s|^2 - r^2 = 0$$

where r is the radius of the sphere

Intersection calculations: Spheres

- To test whether a ray intersects a surface we can substitute for **q** using the ray equation:

$$|\mathbf{p}_0 + \mu\mathbf{d} - \mathbf{p}_s|^2 - r^2 = 0$$

- Setting $\Delta\mathbf{p} = \mathbf{p}_0 - \mathbf{p}_s$ and expanding the dot product produces the following quadratic equation:

$$\mu^2 + 2\mu(\mathbf{d} \cdot \Delta\mathbf{p}) + |\Delta\mathbf{p}|^2 - r^2 = 0$$

Intersection calculations: Spheres

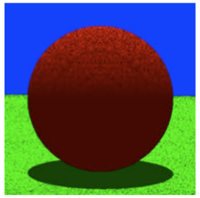
- The quadratic equation has the following solution:

$$\mu = -\mathbf{d} \cdot \Delta\mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \Delta\mathbf{p})^2 - |\Delta\mathbf{p}|^2 + r^2}$$

- Solutions:
  - if the quadratic equation has no solution, the ray does not intersect the sphere
  - if the quadratic equation has two solutions ($\mu_1 < \mu_2$):
    - μ_1 corresponds to the point at which the rays enters the sphere
    - μ_2 corresponds to the point at which the rays leaves the sphere

Precision Problems

- In ray tracing, the origin of (secondary) rays is often on the surface of objects
  - Theoretically, $\mu = 0$ for these rays
  - Practically, calculation imprecision creeps in, and the origin of the new ray is slightly beneath the surface
- Result: the surface area is shadowing itself



ε to the rescue ...

- Check if t is within some epsilon tolerance:
  - if abs(μ) < ε
    - point is on the sphere
  - else
    - point is inside/outside
  - Choose the ε tolerance empirically
- Move the intersection point by epsilon along the surface normal so it is outside of the object
- Check if point is inside/outside surface by checking the sign of the implicit (sphere etc.) equation

Problem Time

- Given:
  - the viewpoint is at $\mathbf{p}_v = (0, 0, -10)$
  - the ray passes through viewing plane at $\mathbf{p}_i = (0, 0, 0)$.
- Spheres:
  - Sphere A with center $\mathbf{p}_s = (0, 0, 8)$ and radius r = 5
  - Sphere B with center $\mathbf{p}_s = (0, 0, 9)$ and radius r = 3
  - Sphere C with center $\mathbf{p}_s = (0, -3, 8)$ and radius r = 2
- Calculate the intersections of the ray with the spheres above.

Solution

- The direction vector is $\mathbf{d} = (0, 0, 10) / 10 = (0, 0, 1)$
  - Sphere A:

    $\Delta p = (0, 0, 8)$, so $\mu = 8 \pm \text{sqrt}(64 - 64 + 25) = 8 \pm 5$

    As the result, the ray enters A sphere at (0, 0, 3) and exits the sphere at (0, 0, 13)).
  - Sphere B:

    $\Delta p = (0, 0, 9)$, so $\mu = 9 \pm \text{sqrt}(81 - 81 + 9) = 9 \pm 3$

    As the result, the ray enters B sphere at (0, 0, 6) and exits the sphere at (0, 0, 12)).
  - Sphere C has no intersections with ray.

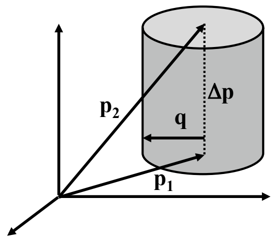
Intersection calculations: Cylinders

- A cylinder can be described by
  - a position vector $\mathbf{p_1}$ describing the first end point of the long axis of the cylinder
  - a position vector $\mathbf{p_2}$ describing the second end point of the long axis of the cylinder
  - a radius r
- The axis of the cylinder can be written as $\Delta\mathbf{p} = \mathbf{p_1} - \mathbf{p_2}$ and can be parameterized by $0 \le \alpha \le 1$

Intersection calculations: Cylinders

- To calculate the intersection of the cylinder with the ray:

$$\mathbf{p_1} + \alpha\Delta\mathbf{p} + \mathbf{q} = \mathbf{p}_0 + \mu\mathbf{d}$$



- Since $\mathbf{q} \cdot \Delta\mathbf{p} = 0$ we can write

$$\alpha(\Delta\mathbf{p} \cdot \Delta\mathbf{p}) = \mathbf{p}_0 \cdot \Delta\mathbf{p} + \mu\mathbf{d} \cdot \Delta\mathbf{p} - \mathbf{p}_1 \cdot \Delta\mathbf{p}$$

Intersection calculations: Cylinders

- Solving for α yields:

$$\alpha = \frac{\mathbf{p}_0 \cdot \Delta\mathbf{p} + \mu\mathbf{d} \cdot \Delta\mathbf{p} - \mathbf{p}_1 \cdot \Delta\mathbf{p}}{\Delta\mathbf{p} \cdot \Delta\mathbf{p}}$$

- Substituting we obtain:

$$\mathbf{q} = \mathbf{p}_0 + \mu\mathbf{d} - \mathbf{p}_1 - \left(\frac{\mathbf{p}_0 \cdot \Delta\mathbf{p} + \mu\mathbf{d} \cdot \Delta\mathbf{p} - \mathbf{p}_1 \cdot \Delta\mathbf{p}}{\Delta\mathbf{p} \cdot \Delta\mathbf{p}}\right)\Delta\mathbf{p}$$

Intersection calculations: Cylinders

- Using the fact that $\mathbf{q} \cdot \mathbf{q} = r^2$ we can use the same approach as before to the quadratic equation for μ:

$$r^2 = \left(\mathbf{p}_0 + \mu\mathbf{d} - \mathbf{p}_1 - \left(\frac{\mathbf{p}_0 \cdot \Delta\mathbf{p} + \mu\mathbf{d} \cdot \Delta\mathbf{p} - \mathbf{p}_1 \cdot \Delta\mathbf{p}}{\Delta\mathbf{p} \cdot \Delta\mathbf{p}}\right)\Delta\mathbf{p}\right)^2$$

  - If the quadratic equation has no solution:
    - ➨ no intersection
  - If the quadratic equation has two solutions:
    - ➨ intersection

Intersection calculations: Cylinders

- Assuming that μ1 ≤ μ2 we can determine two solutions:

$$\alpha_1 = \frac{\mathbf{p}_0 \cdot \Delta\mathbf{p} + \mu_1 \mathbf{d} \cdot \Delta\mathbf{p} - \mathbf{p}_1 \cdot \Delta\mathbf{p}}{\Delta\mathbf{p} \cdot \Delta\mathbf{p}}$$

$$\alpha_2 = \frac{\mathbf{p}_0 \cdot \Delta\mathbf{p} + \mu_2 \mathbf{d} \cdot \Delta\mathbf{p} - \mathbf{p}_1 \cdot \Delta\mathbf{p}}{\Delta\mathbf{p} \cdot \Delta\mathbf{p}}$$

- If the value of α1 is between 0 and 1 the intersection is on the outside surface of the cylinder
- If the value of α2 is between 0 and 1 the intersection is on the inside surface of the cylinder

Intersection calculations: Plane

- Objects are often described by geometric primitives such as
  - triangles
  - planar quads
  - planar polygons
- To test intersections of the ray with these primitives we must whether the ray will intersect the plane defined by the primitive

Intersection calculations: Plane

- The intersection of a ray with a plane is given by

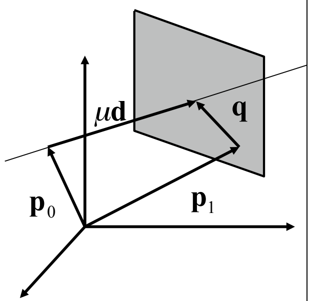
$$\mathbf{p}_1 + \mathbf{q} = \mathbf{p}_0 + \mu\mathbf{d}$$

where $\mathbf{p_1}$ is a point in the plane. Subtracting $\mathbf{p_1}$ and multiplying with the normal of the plane **n** yields:

$$\mathbf{q} \cdot \mathbf{n} = \mathbf{0} = (\mathbf{p}_0 - \mathbf{p}_1) \cdot \mathbf{n} + \mu\mathbf{d} \cdot \mathbf{n}$$

- Solving for μ yields:

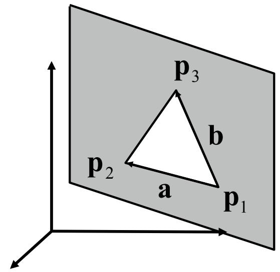
$$\mu = -\frac{(\mathbf{p}_0 - \mathbf{p}_1) \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}$$



Intersection calculations: Triangles

- To calculate intersections:
  - test whether triangle is front facing
  - test whether plane of triangle intersects ray
  - test whether intersection point is inside triangle
- If the triangle is front facing:

$$\mathbf{d} \cdot \mathbf{n} < 0$$



Intersection calculations: Triangles

- To test whether plane of triangle intersects ray
  - calculate equation of the plane using

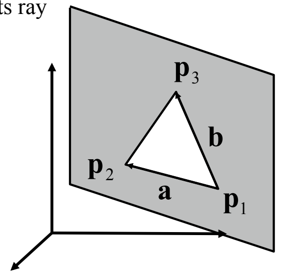
$$\mathbf{p}_2 - \mathbf{p}_1 = \mathbf{a}$$
$$\mathbf{p}_3 - \mathbf{p}_1 = \mathbf{b}$$

  - calculate intersections with plane as before

$$\mathbf{n} = \mathbf{a} \times \mathbf{b}$$

- To test whether intersection point is inside triangle:

$$\mathbf{q} = \alpha\mathbf{a} + \beta\mathbf{b}$$



Intersection calculations: Triangles

- A point is inside the triangle if

$$0 \le \alpha \le 1$$
$$0 \le \beta \le 1$$
$$\alpha + \beta \le 1$$

- Calculate α and β by taking the dot product with **a** and **b**:

$$\alpha = \frac{(\mathbf{b}\cdot\mathbf{b})(\mathbf{q}\cdot\mathbf{a}) - (\mathbf{a}\cdot\mathbf{b})(\mathbf{q}\cdot\mathbf{b})}{(\mathbf{a}\cdot\mathbf{a})(\mathbf{b}\cdot\mathbf{b}) - (\mathbf{a}\cdot\mathbf{b})^2}$$

$$\beta = \frac{\mathbf{q}\cdot\mathbf{b} - \alpha(\mathbf{a}\cdot\mathbf{b})}{\mathbf{b}\cdot\mathbf{b}}$$

Ray tracing: Pros and cons

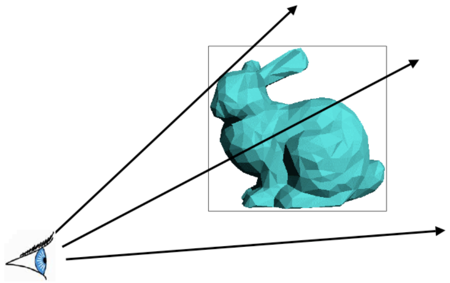
- Pros:
  - Easy to implement
  - Extends well to global illumination
    - shadows
    - reflections / refractions
    - multiple light bounces
    - atmospheric effects
- Cons:
  - Speed! (seconds per frame, not frames per second)

Speedup Techniques

- Why is ray tracing slow? How to improve?
  - Too many objects, too many rays
  - Reduce ray-object intersection tests
  - Many techniques!

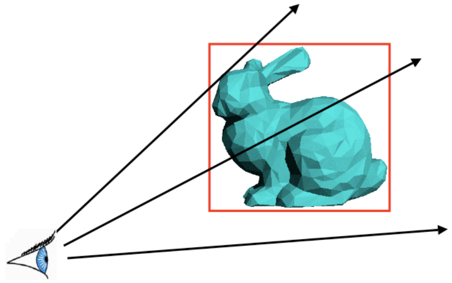
Acceleration of Ray Casting

- Goal: Reduce the number of ray/primitive intersections



Conservative Bounding Region

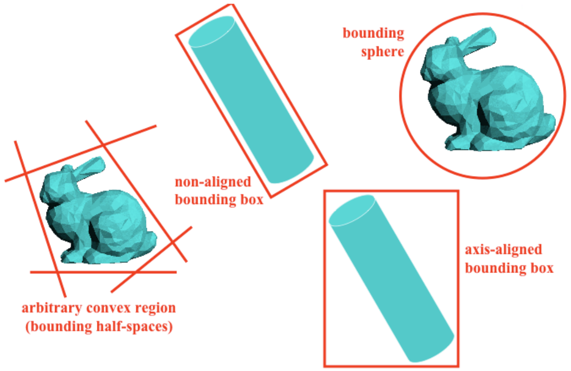
- First check for an intersection with a conservative bounding region
- Early reject



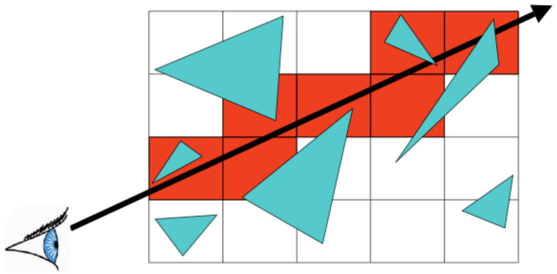
Bounding Regions

- What makes a good bounding region?

Conservative Bounding Regions

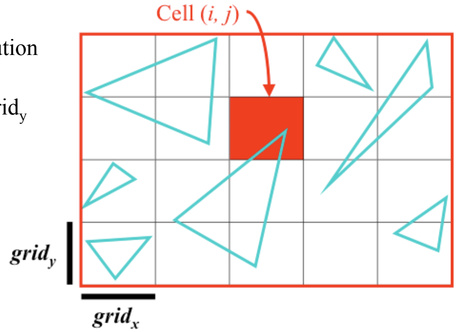


Regular Grid



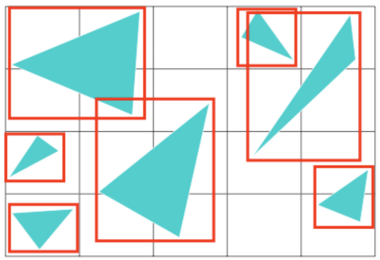
Create Grid

- Find bounding box of scene
- Choose grid resolution (n_x, n_y, n_z)
- $grid_x$ need not = $grid_y$



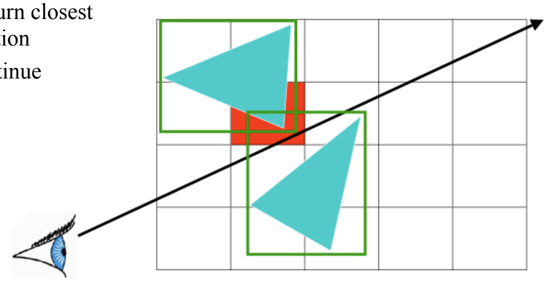
Insert Primitives into Grid

- Primitives that overlap multiple cells?
- Insert into multiple cells (use pointers)



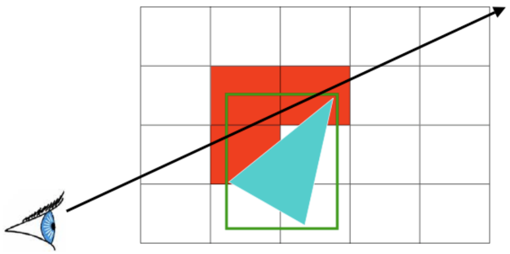
For Each Cell Along a Ray

- Does the cell contain an intersection?
- Yes: return closest intersection
- No: continue



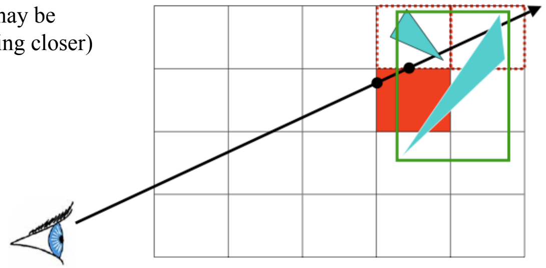
Preventing Repeated Computation

- Perform the computation once, "mark" the object
- Don't re-intersect marked objects



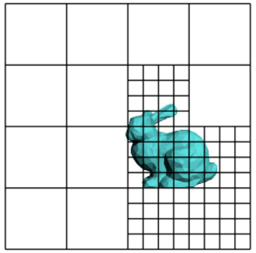
Don't Return Distant Intersections

- If intersection t is not within the cell range, continue (there may be something closer)

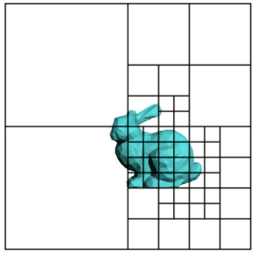


Adaptive Grids

- Subdivide until each cell contains no more than *n* elements, or maximum depth *d* is reached



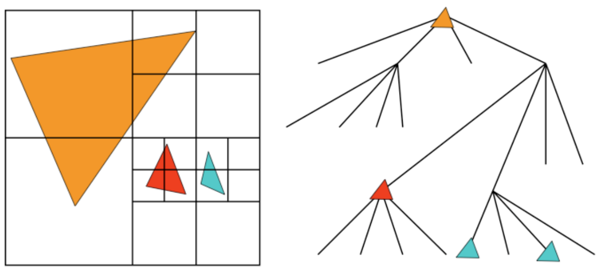
Nested Grids



Octree/(Quadtree)

Primitives in an Adaptive Grid

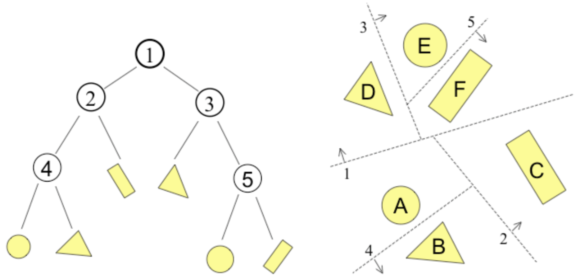
- Can live at intermediate levels, or be pushed to lowest level of grid



Octree/(Quadtree)

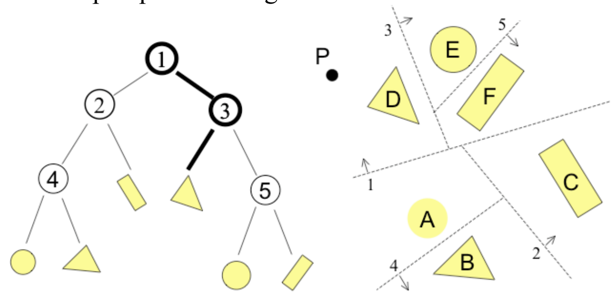
Binary Space Partition (BSP) Tree

- Recursively partition space by planes
- Every cell is a convex polyhedron



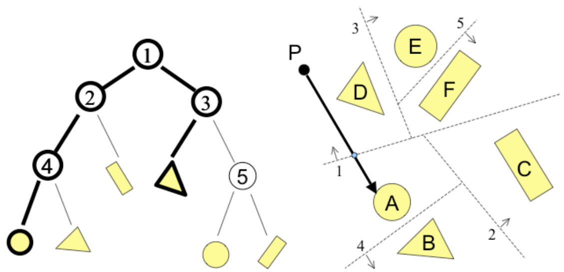
Binary Space Partition (BSP) Tree

- Simple recursive algorithms
- Example: point finding



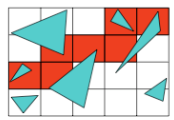
Binary Space Partition (BSP) Tree

- Trace rays by recursion on tree
  - BSP construction enables simple front-to-back traversal



Grid Discussion

- Regular
  - + easy to construct
  - + easy to traverse
  - – may be only sparsely filled
  - – geometry may still be clumped



- Adaptive
  - + grid complexity matches geometric density
  - – more expensive to traverse (especially BSP tree)