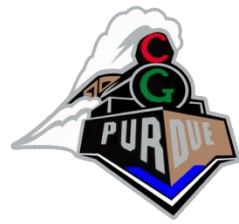# Ray Tracing (Part I)

CS334 Spring 2022

Daniel G. Aliaga

Department of Computer Science

Purdue University

Ray Casting and Ray Tracing
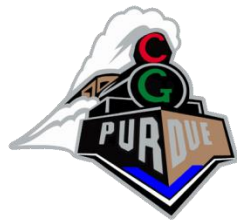
- ## Ray Casting
  - Arthur Appel, started around 1968…

- ## Ray Tracing
  - Turner Whitted, started around 1980…
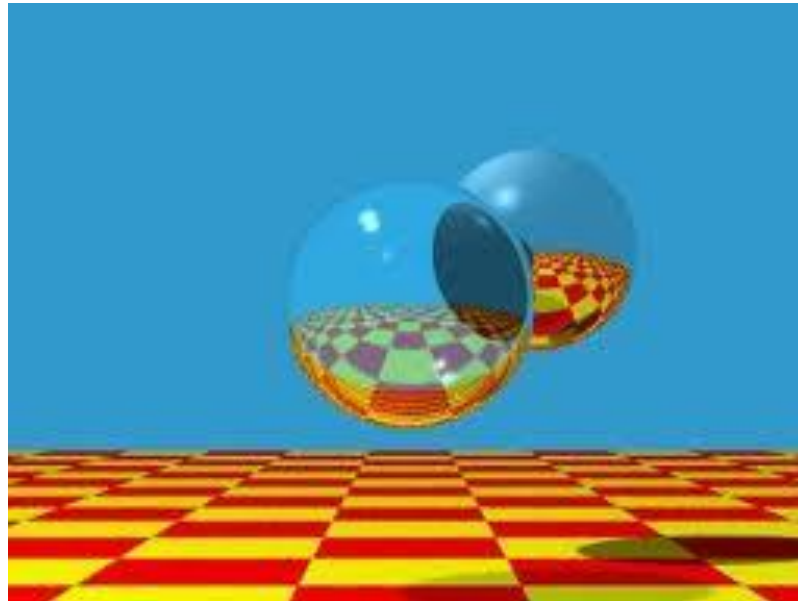
Ray Tracing

- Turner Whitted, Communications of the ACM, 23(6), 343-349, June 1980

- **ABSTRACT**
"To accurately render a two-dimensional image of a three-dimensional scene, global illumination information that affects the intensity of each pixel of the image must be known at the time the intensity is calculated. In a simplified form, this information is stored in a tree of "rays" extending from the viewer to the first surface encountered and from there to other surfaces and to the light sources. A visible surface algorithm creates this tree for each pixel of the display and passes it to the shader. The shader then traverses the tree to determine the intensity of the light received by the viewer. Consideration of all of these factors allows the shader to accurately simulate true reflection, shadows, and refraction, as well as the effects simulated by conventional shaders. Anti-aliasing is included as an integral part of the visibility calculations. Surfaces displayed include curved as well as polygonal surfaces."
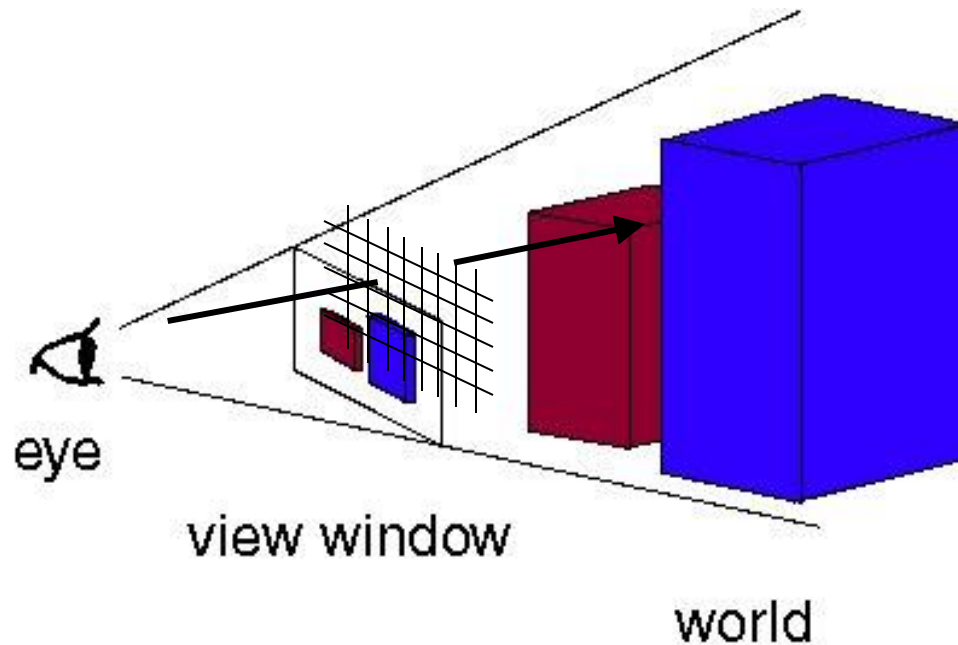
Ray Tracing

- Turner Whitted, Communications of the ACM, 23(6), 343-349, June 1980
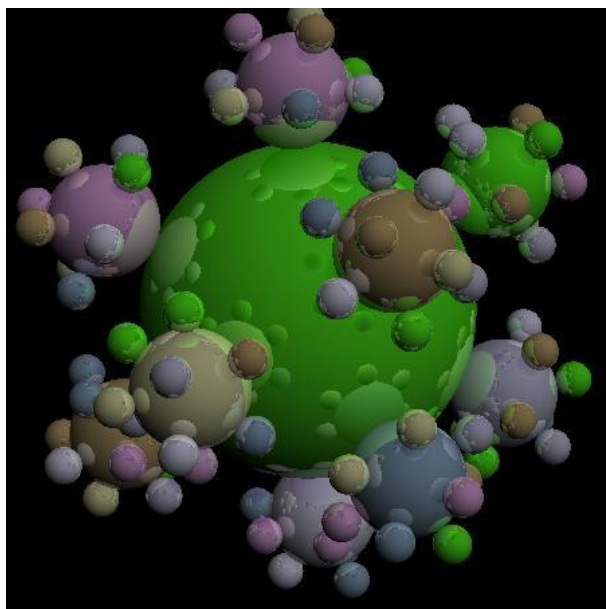
Ray Casting

- To "cast" a ray for each pixel into the scene and color the ray based on which object it "hits"
  - This can be interpreted as the inverse of the feed-forward graphics pipeline which effectively "pushes" geometry onto the screen

eye

view window

world

Ray Tracing

- In addition to ray-casting, rays can reflect off objects, reflect, refract, identify objects in shadow, and numerous other effects…
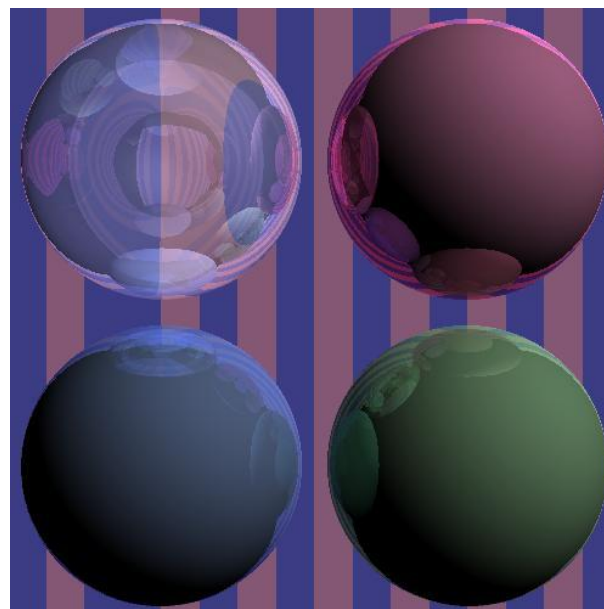
Ray Tracing

- In addition to ray-casting, rays can reflect off objects, reflect, refract, identify objects in shadow, and numerous other effects…

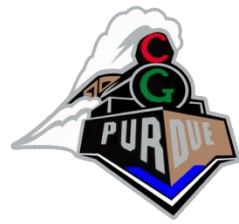ray trace example

using CUDA/GPU programming

Ray Tracing

- Fluid Animation:
  - https://www.youtube.com/watch?v=3rlbnqayGZs

- Real-time
  - https://www.youtube.com/watch?v=2ow7cbtzg1E

Ray Tracers

- POVRay
  - http://www.povray.org
- RayShade
  - http://www-graphics.stanford.edu/~cek/rayshade/rayshade.html
- NVIDIA OptiX
  - https://developer.nvidia.com/optix

Basic Ray Tracing Algorithm

- Setup image plane
  - Center-of-projection
  - Field-of-view

- Define objects
  - Functional descriptions
    - E.g., sphere, box, f(x,y,z)=0
  - Polygonal descriptions

- Foreach pixel…

Basic Ray Tracing Algorithm

Foreach pixel

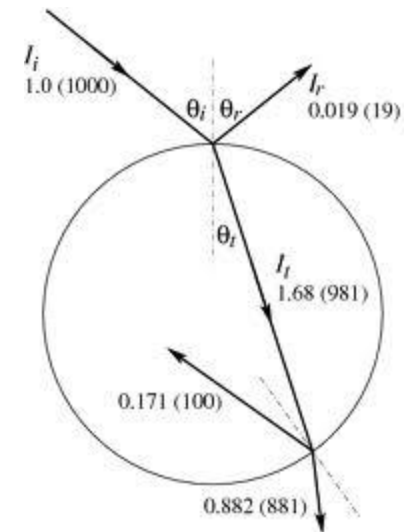   **define ray from eye through pixel and into scene**
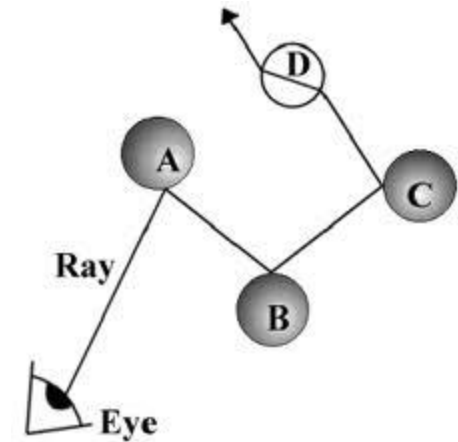
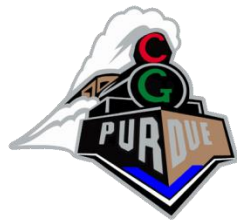   **get intersection with scene**

   trace reflected ray...

   trace refracted ray...

   trace to-light ray...

   combine resulting colors...

Ray tracing

- Ray tracing (intersection computations)
  - Sphere, plane, polygon, box, quadric
- Reflections
- Refractions
- Shadows

(more on the board…)
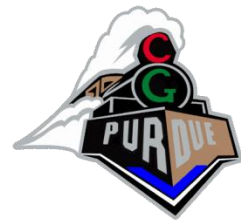
Ray Object Intersections

- A ray is defined as

$$R_0 = [x_0, y_0, z_0] \quad \text{- origin of ray}$$
$$R_d = [x_d, y_d, z_d] \quad \text{- direction of ray}$$

Then, a set of points on the ray are defined as

$$R(t) = R_0 + R_d * t \quad \text{where } t > 0$$

If R_d is normalized, then t equals the distance of the ray from origin in World Coordinates,

Ray Sphere Intersection

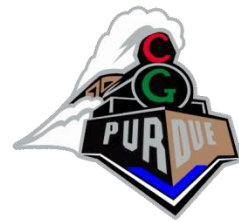- Sphere is defined by its center ($S_c = [x_c, y_c, z_c]$) and radius S_r

  Thus, the equation of sphere is

  $$(x-x_c)^2 + (y-y_c)^2 + (z-z_c)^2 = S_r^2$$

  Substitute equation of ray and solve quadratic eqn for **t**,
  - If discriminant $< 0 \Rightarrow$ No intersection
  - If discriminant $\geq 0 \Rightarrow$ Intersection. Smaller **t** gives the nearest intersection point

Slides: Vivek Puri

Ray Plane Intersection

- Plane is defined by $Ax+By+Cz+D=0$

  Substitute the equation for ray and solve for **t**,

  $$\mathbf{t} = -(AX_0 + BY_0 + CZ_0 + D) / (AX_d + BY_d + CZ_d)$$

  $$= -(\mathbf{P}_n \cdot \mathbf{R}_0 + D) / (\mathbf{P}_n \cdot \mathbf{R}_d)$$

  where $\mathbf{P_n}$ is the plane normal of unit length

  If $\mathbf{P}_n \cdot \mathbf{R}_d = 0 \Rightarrow$ Ray is parallel to the plane

  If $\mathbf{P}_n \cdot \mathbf{R}_d > 0 \Rightarrow$ Normal of plane points away from the ray

  If $\mathbf{t} < 0 \Rightarrow$ Ray intersects plane behind origin

  If $\mathbf{t} > 0 \Rightarrow$ Substitute $\mathbf{t}$ in equation for intersection point

Slides: Vivek Puri

Ray Polygon Intersection

- Check whether the ray intersects the polygon plane

- Calculate the point of intersection

- Check whether the point is within the polygon. **How?**
  - Cross products
  - Shoot a ray from the point and count edges crossed, if odd then inside
  - or one of several other approaches

Slides: Vivek Puri

Ray Box Intersection

- Transform box vertices (V_i) to align with the coordinate axes

- Solve for intersection with x_{min} and x_{max} planes; similarly solve for Y and Z.

- Check for common interval or intersection

  If,   max (t_{min}) ≤ min (t_{max})

  Then, t_0 = max (t_{min})

                  t_1 = min (t_{max})

Slides: Vivek Puri

Ray Quadric Intersection

- Class of quadrics (surfaces that can be defined by a quadratic equation) include cylinders, cones, ellipsoids, paraboloids, etc

- The general quadric surface equation is

  $Ax^2 + By^2 + Cz^2 + Dxy + Exz + Fyz + Gx + Hy + Iz + J = 0$

- Substitute the equation of ray, we get the form,

  $A_q t^2 + B_q t + C_q = 0$

- Solve for **t** using quadratic formula
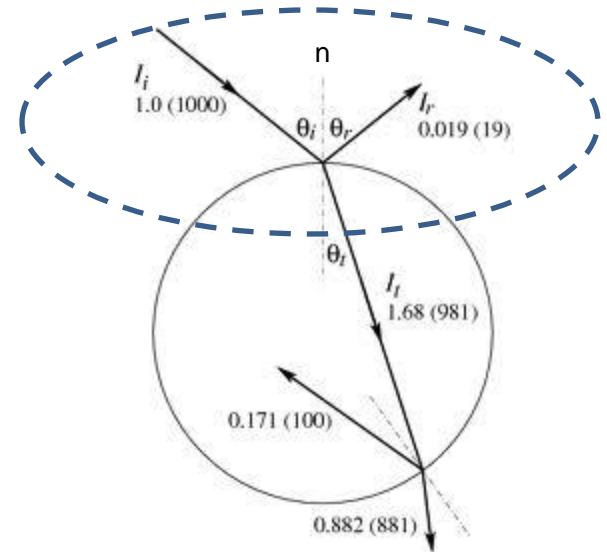
Slides: Vivek Puri

Ray tracing

- Ray tracing (intersection computations)
- Reflections
- Refractions
- Shadows

  (more on the board…)

Reflections

- Similar to what we saw for shading/illumination

- Recall $\theta_i = \theta_r$

- $l_r = l_i - 2n(n \cdot l_i)$

Refraction (or Transmission)

- Need refraction indices:

  k_1=outside object (e.g., air)

  k_2=inside object (e.g., glass)

- Snell's Law:
$$k_1 \sin(\theta_i) = k_2 \sin(\theta_t)$$

- After some rearrangement:
$$l_t = -n\cos(\theta_t) + b\sin(\theta_t)$$
$$\text{where } b = n \times (n \times l_i)$$