Ray Tracing

Hendrik Speleers

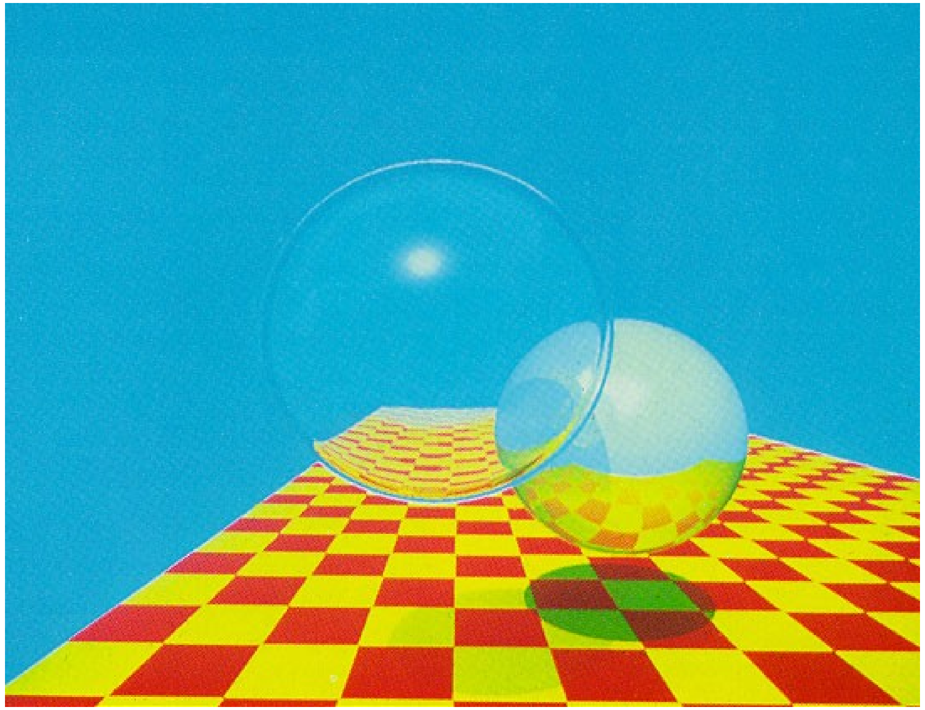
Ray Tracing

- How did we generate pictures so far? Object-based rendering
  - Modeling + viewing transform
  - Perspective transform
  - Clipping
  - Scan-conversion + *Z*-buffer for visibility
- Ray tracing ~ pixel-based rendering
  - RT is a more powerful technique to render scenes
  - Each pixel uses its own ray(s) to determine color
  - No perspective transform, no clipping, ...

Ray Tracing

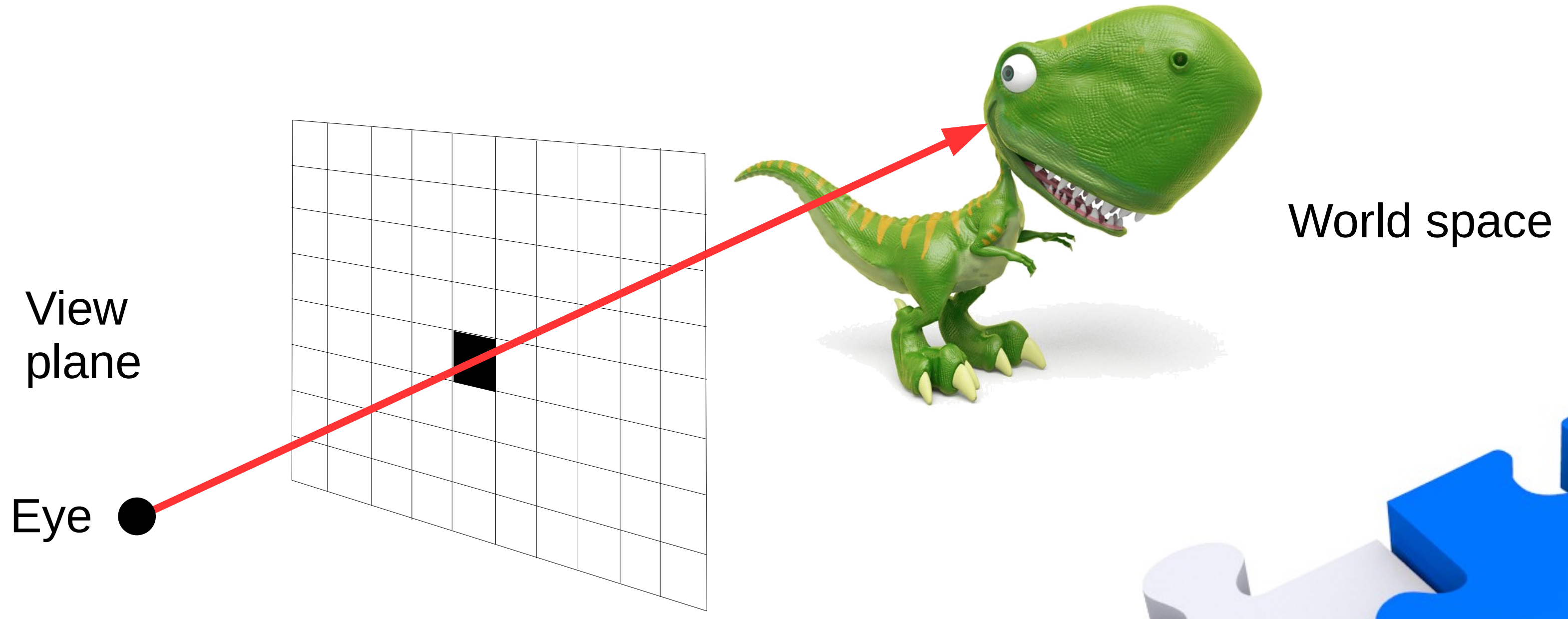
- Very first ray-traced picture

Turner Whitted
1980



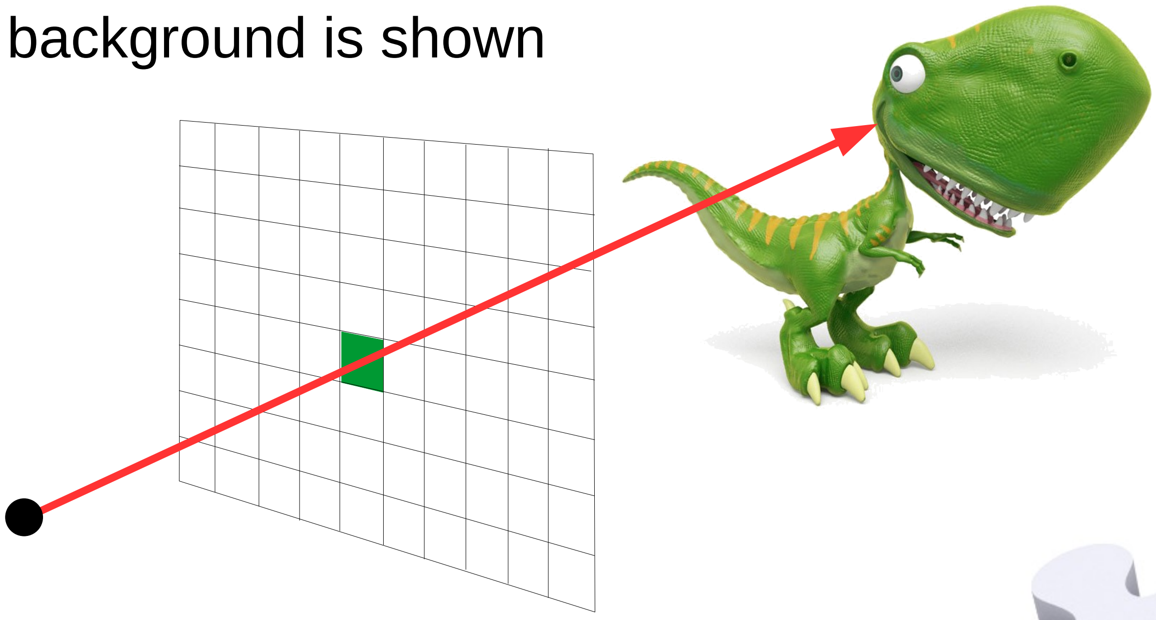
Ray Tracing

- Basic RT algorithm
  - Set-up: camera defined by eye-point and view plane in world space
  - Send a ray from the eye through each pixel



Ray Tracing

- Basic RT algorithm
  - Intersect ray with all geometry in the scene
  - If a ray hits an object, then that object is shown
  - … otherwise background is shown



Ray Tracing

Università di Roma
Tor Vergata

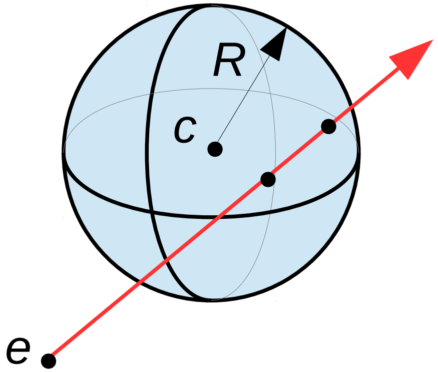
- Basic RT algorithm
  - For each pixel (*x*, *y*):
    - Construct the ray for the pixel
    - Intersect the ray with all objects in the scene
    - Pick the closest intersection in front of the eye
    - Compute the color of the hit-point considering all light sources
    - Color pixel
  - Order of pixels is not important (→ parallelization)
  - Not restricted to polygonal geometry
    - Exact normals can be used, instead of interpolation

Ray Tracing

Università di Roma
Tor Vergata

- Ray intersections
  - Hit-point on ray: $p = e + t\ dir$
  - Sphere (centered at c with radius R)
    - Solving equation for t yields 0, 1, 2 intersections

$$(e_x - c_x + t\,dir_x)^2 + (e_y - c_y + t\,dir_y)^2 + (e_z - c_z + t\,dir_z)^2 = R^2$$

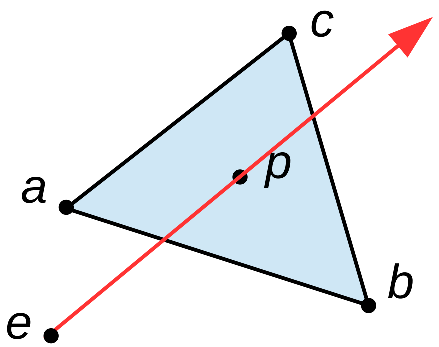


  - Triangle (vertices a, b, c)
    - Barycentric coordinates of p

$$p = \alpha a + \beta b + \gamma c \qquad \alpha + \beta + \gamma = 1$$

    - Inside triangle if

$$0 < \alpha < 1 \qquad 0 < \beta < 1 \qquad 0 < \gamma < 1$$



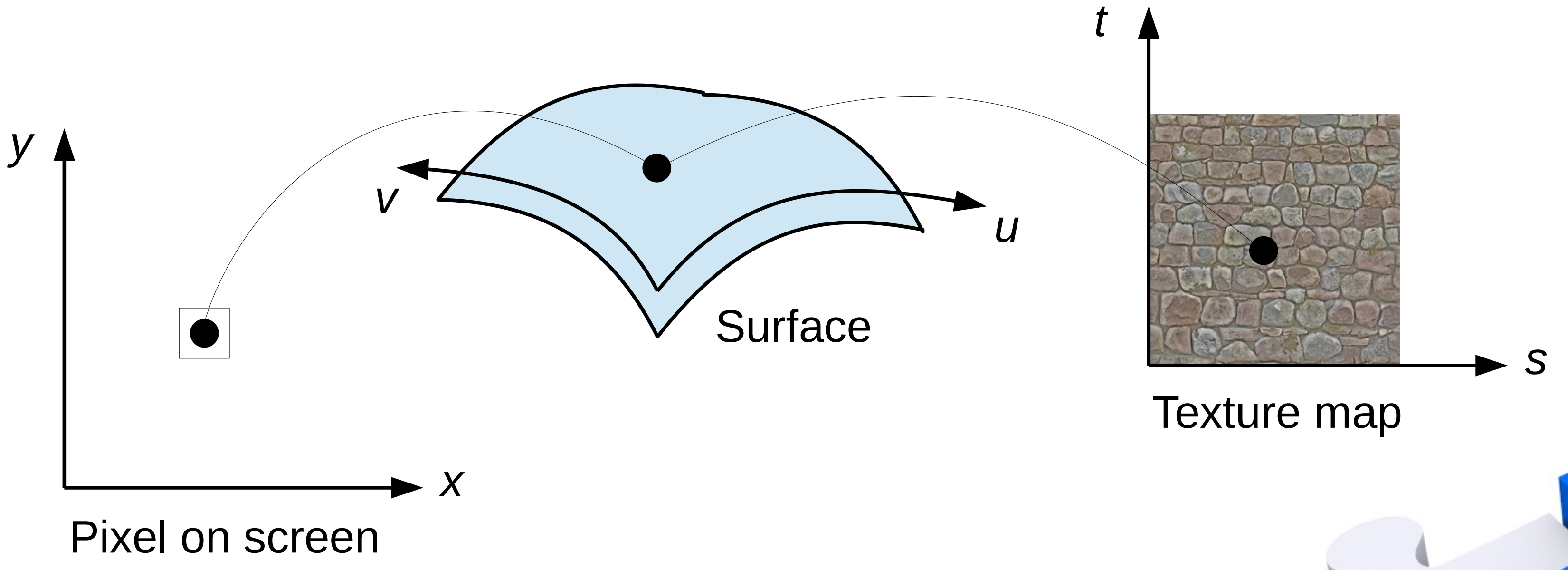
Ray Tracing

- Ray intersections
  - Transformed object
    - Apply inverse transform to ray
    - Intersect transformed ray with original object
    - Apply transform to hit-point
  - Normal vector at hit-point
    - Needed for shading
    - Transformed objects?

$$p' = Mp \qquad n' = M^{-T} n$$

Ray Tracing

- Ray intersections
  - Local parameterization (*u*, *v*) for hit-point
  - Texture mapping: used to specify color on object



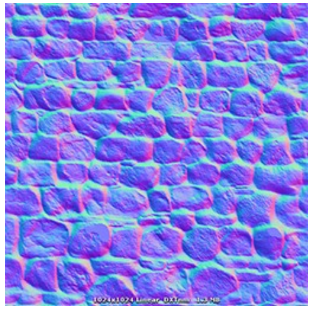
Ray Tracing

- Ray intersections
  - Local parameterization (u, v) for hit-point
  - Texture mapping: used to specify color on object

Unity – Manual

Ray Tracing

- Ray intersections
  - Local parameterization (*u*, *v*) for hit-point
  - Bump mapping: used to perturb normal vectors → shading looks different



Unity – Manual

Ray Tracing

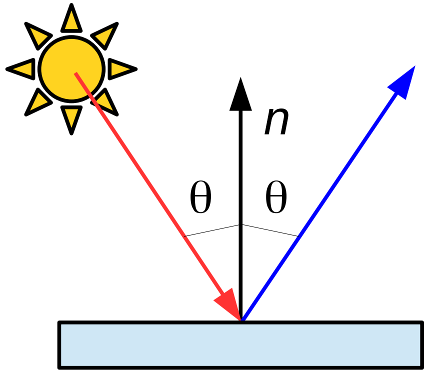
- RT and shadows
  - Trace shadow ray to light source
    - Origin: point to be shaded (= colored)
    - Direction: towards the (point) light source
  - If shadow-ray hits something, then shadow
  - Actual hit-point is not important
    - It is enough if we find any intersection point to require shadow
    - We do not need to find the closest one
      - Can be made slightly more effective

Ray Tracing

- Reflections and transparency
  - So far:
    - Direct illumination (shading model)
    - Shadows
  - What about indirect light?
    - Reflections
      - light coming from perfect specular direction
    - Refractions (Transparency)
      - light coming from perfect refractive direction (Snell's law)

Ray Tracing

- Reflections and transparency
  - Reflections
    - Shoot perfectly reflected ray, find closest hit-point
    - Add color of this hit-point to current hit-point
    - Reflected ray
      - Origin: point to be shaded
      - Direction: perfectly reflected direction
  - Refractions: idem
  - Illumination model

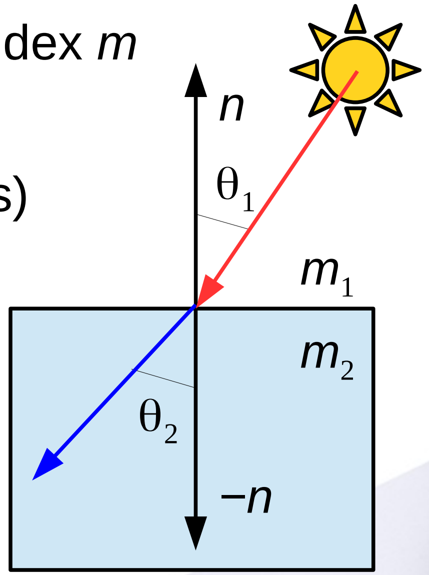


$$I = I_a k_a + \sum_{all\ lights} I_{light}\left[k_d \cos\theta + k_s(\cos\alpha)^p\right] + \overbrace{I_{refl} k_{refl} + I_{refr} k_{refr}}^{\text{Recursive RT}}$$

Ray Tracing

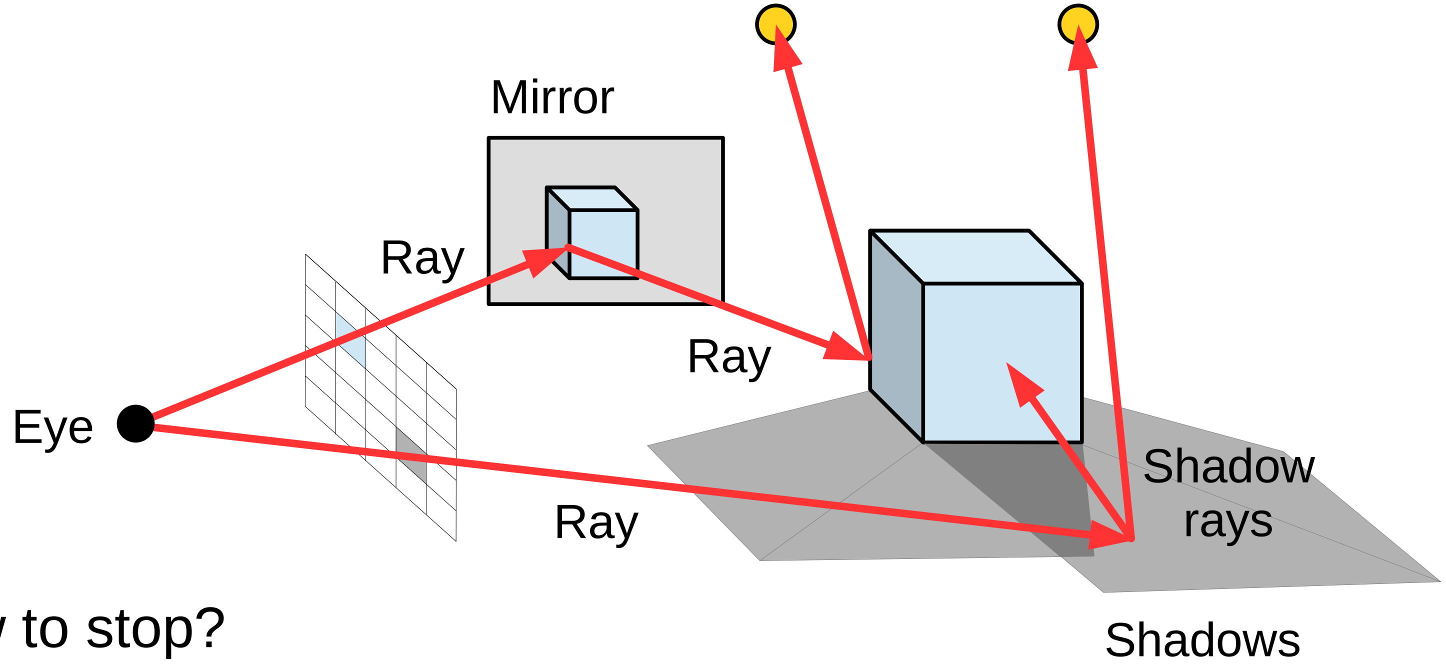
- Transmission with refraction
  - Refraction
    - The bending of light due to its different speed through different materials
  - Refractive index
    - Speed of light is *c/m* in a material of refractive index *m*
    - Speed of light is *c* in a vacuum
    - Varies with wavelength (→ rainbows and prisms)
  - Snell's law
    - Angles of refraction are related by

$$m_1 \sin\theta_1 = m_2 \sin\theta_2$$



Ray Tracing

- Recursive RT



  - How to stop?
    - Fixed depth
    - When color contribution falls below threshold

Ray Tracing

- Complexity
  - Many, many rays
    - Traditional RT is O($N \cdot P$) with N objects and P pixels
    - One shadow ray for each light source
    - Recursive rays
  - Making RT faster
    - Faster ray-object intersections
      - Bounding volumes
    - Fewer ray-object intersections
      - Hierarchies of bounding volumes
      - Spatial subdivision techniques

Ray Tracing

Università di Roma Tor Vergata

- RT examples

Glasses, by G. Tran with POV-Ray

Bolts, by J.V. Piqueres with POV-Ray