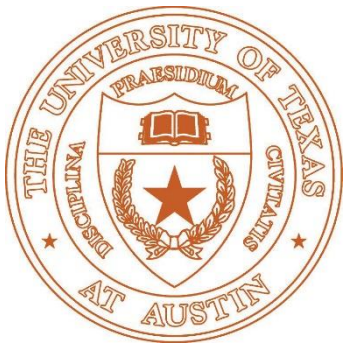
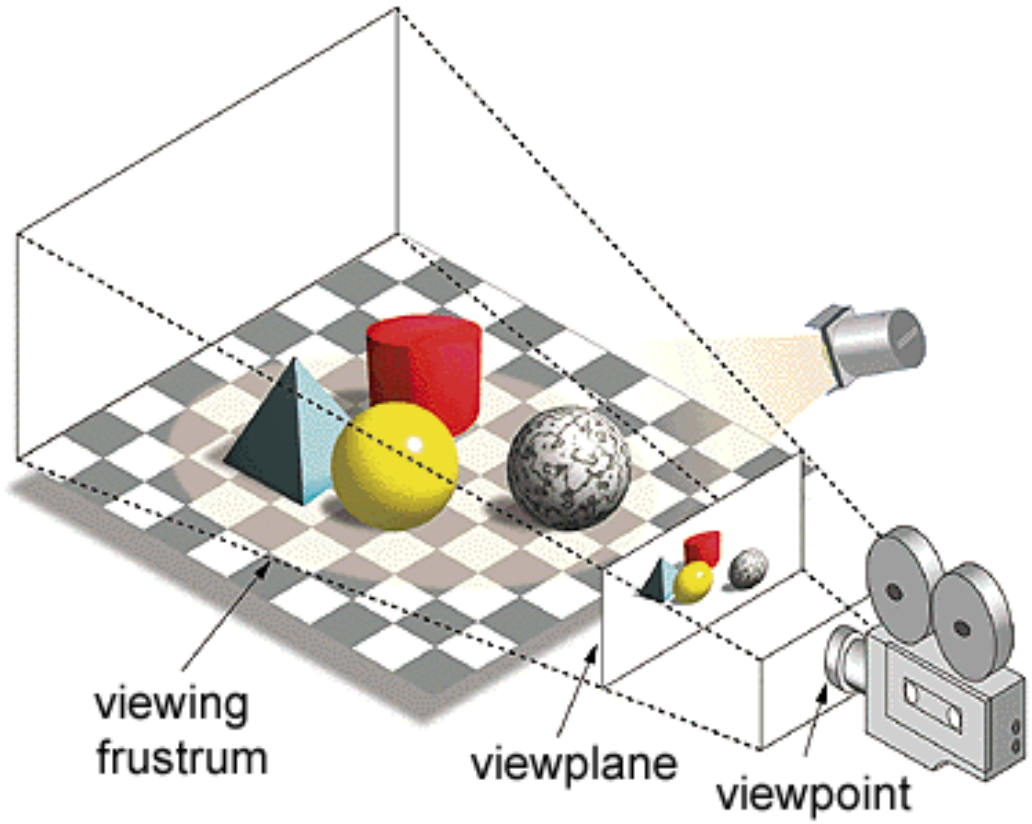
CS354 Computer Graphics
Ray Tracing II

Qixing Huang

Januray 29th 2017



Graphics Pipeline



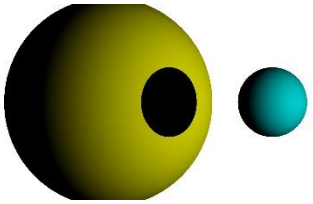
Today’s Topic

- Review Ray Tracing

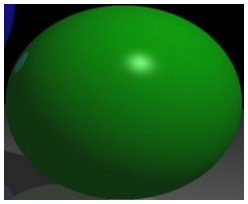
- Ray-Box Intersection

Elements of Ray Tracing

- Shadows

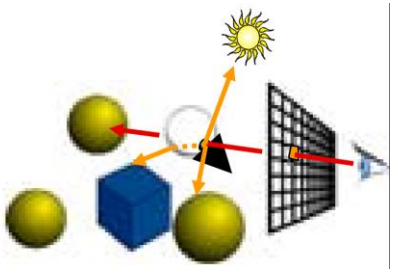


- Reflection



- Refraction

- Recursive Ray Tracing



How Can We Add Shadows?

- For every pixel
  - Construct a ray from the eye
  - For every object in the scene
    - Find intersection with the ray
    - Keep if closest
      - Shade

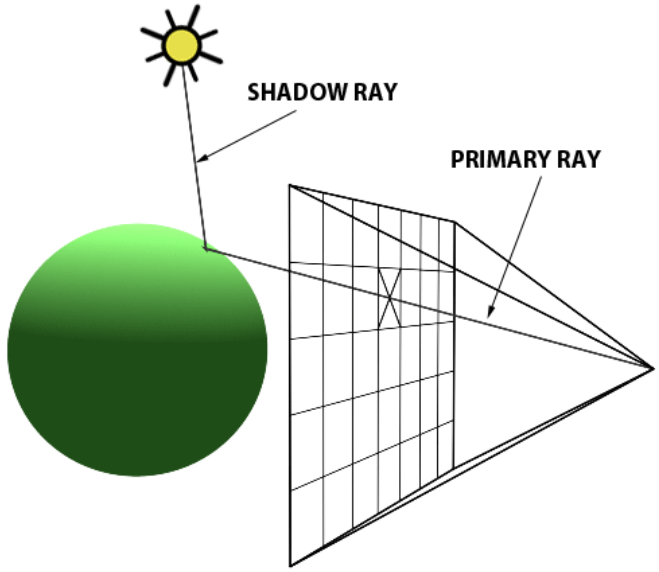
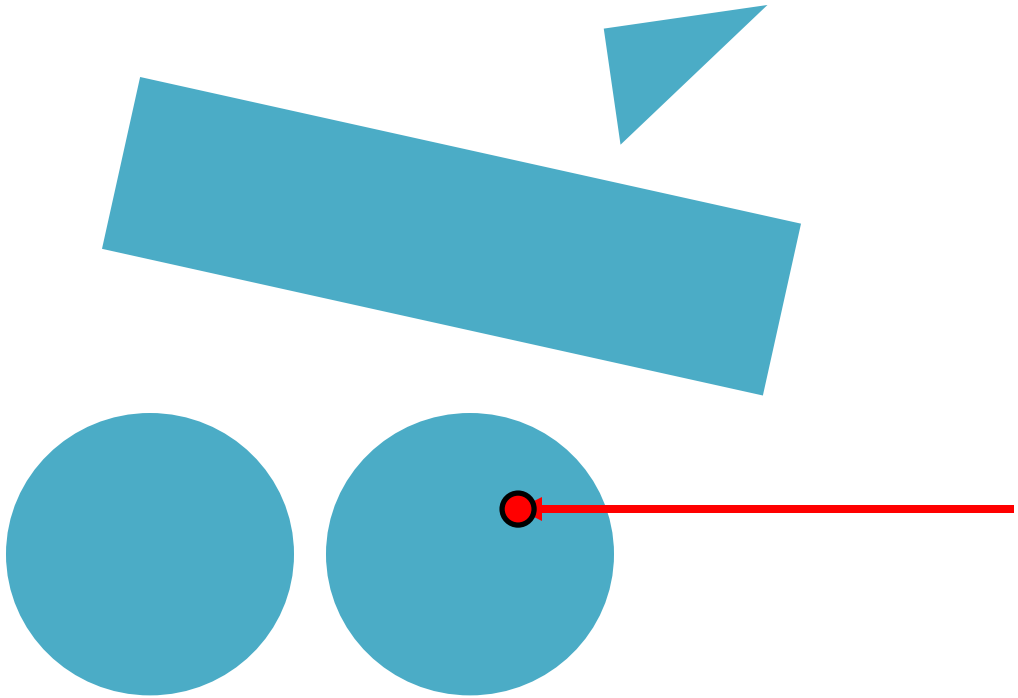


Image credit: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/implementing-the-raytracing-algorithm

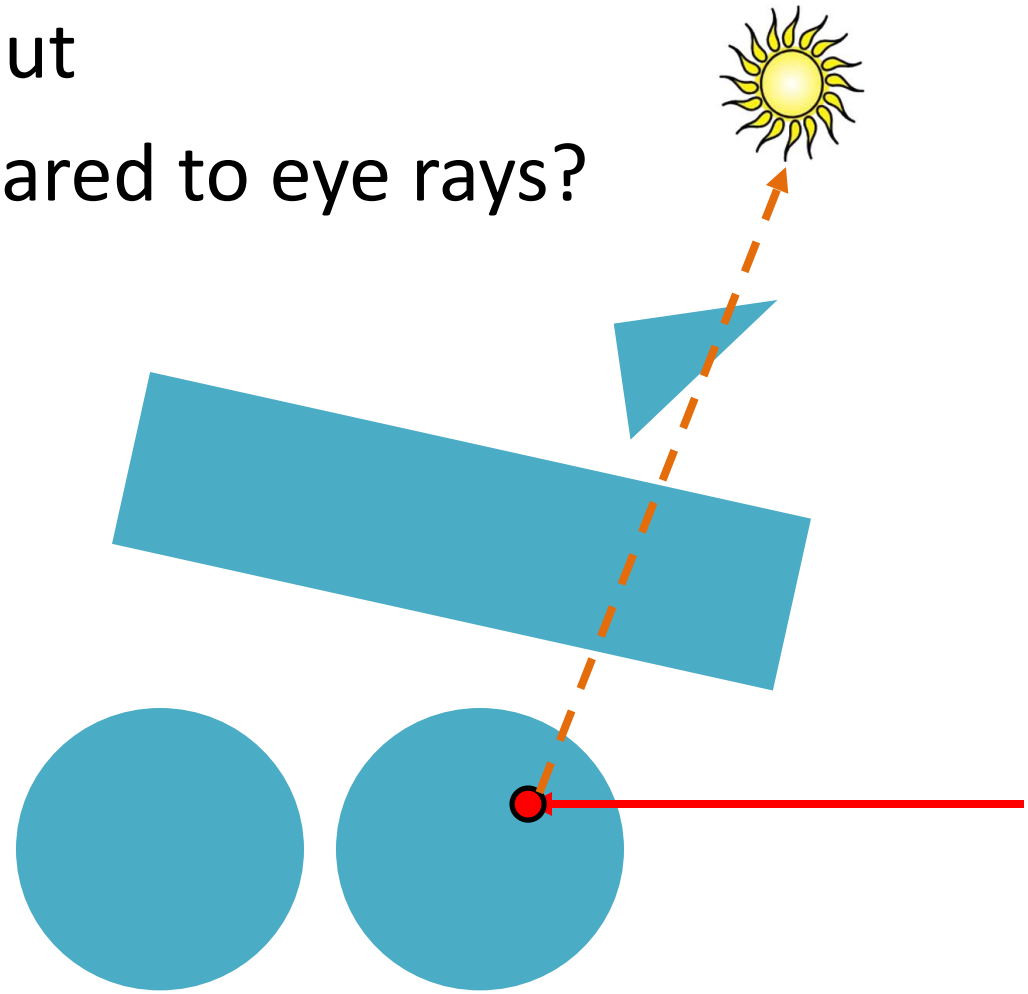
Let us Think About Shadow Rays

- What's special about shadow rays compared to eye rays?



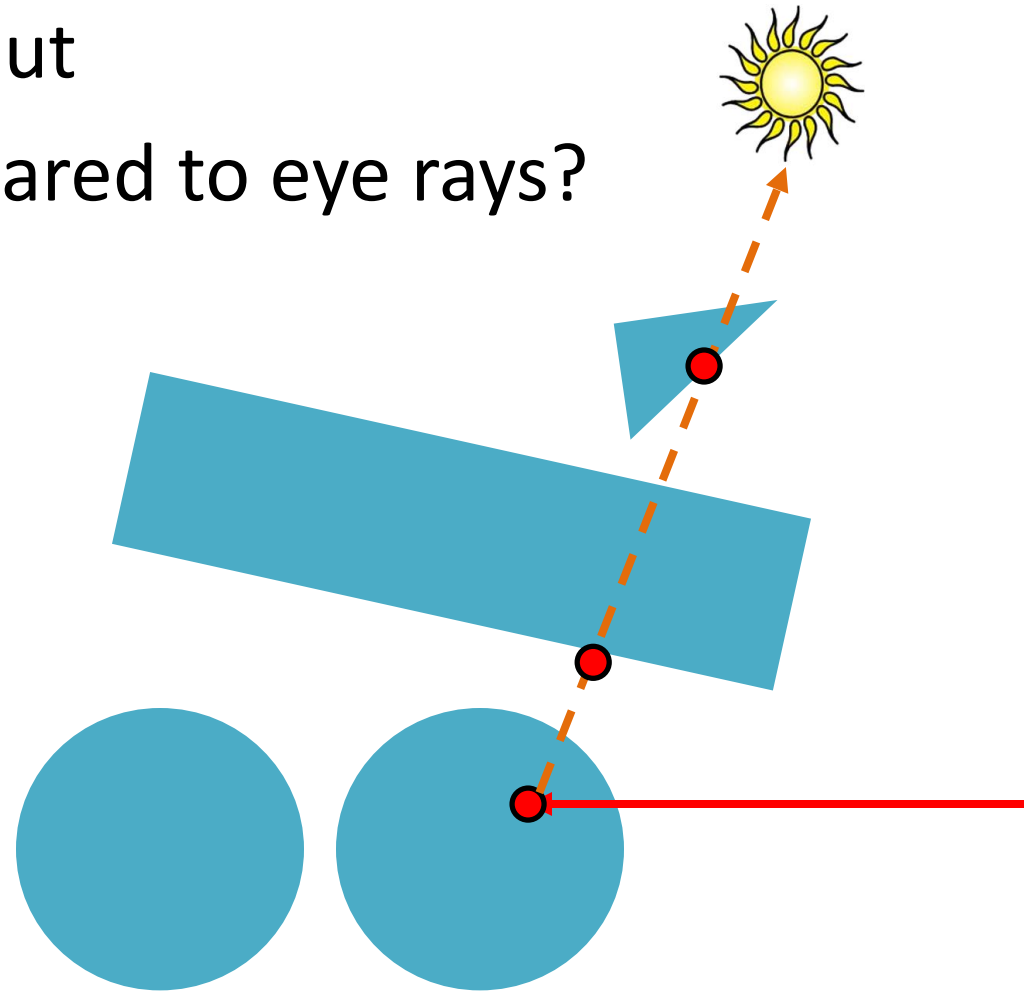
Shadow Ray Optimization

- What's special about shadow rays compared to eye rays?



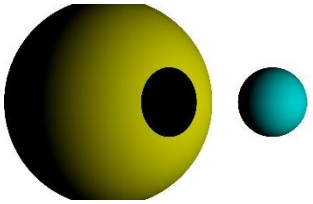
Shadow Ray Optimization

- What's special about shadow rays compared to eye rays?

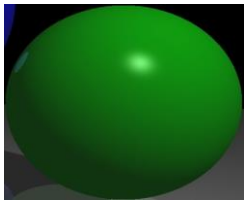


Elements of Ray Tracing

- Shadows

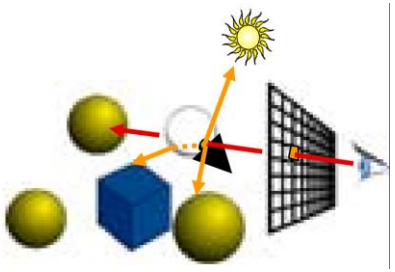


- Reflection



- Refraction

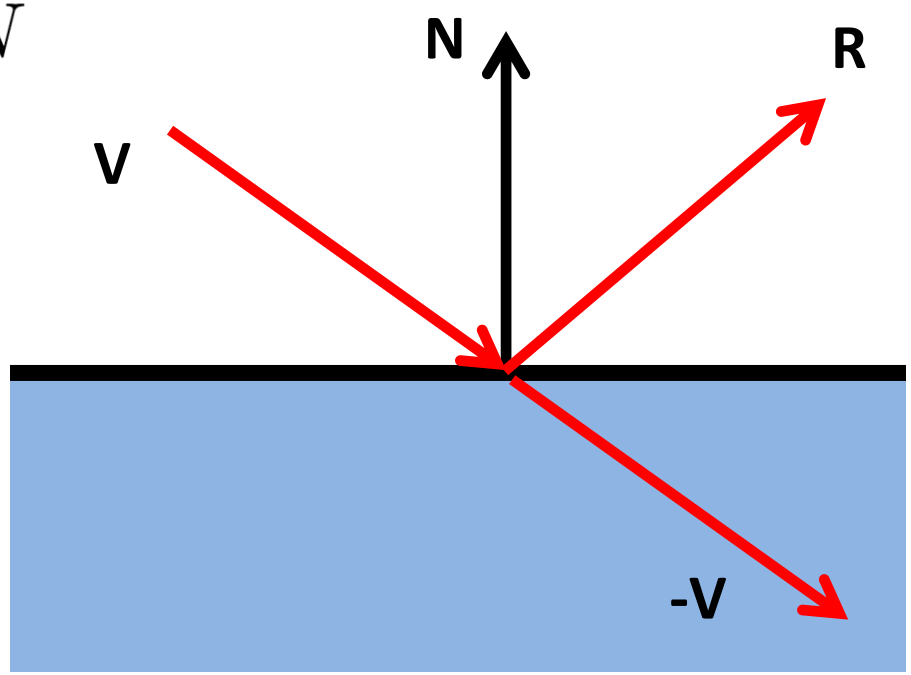
- Recursive Ray Tracing



Perfect Mirror Reflection

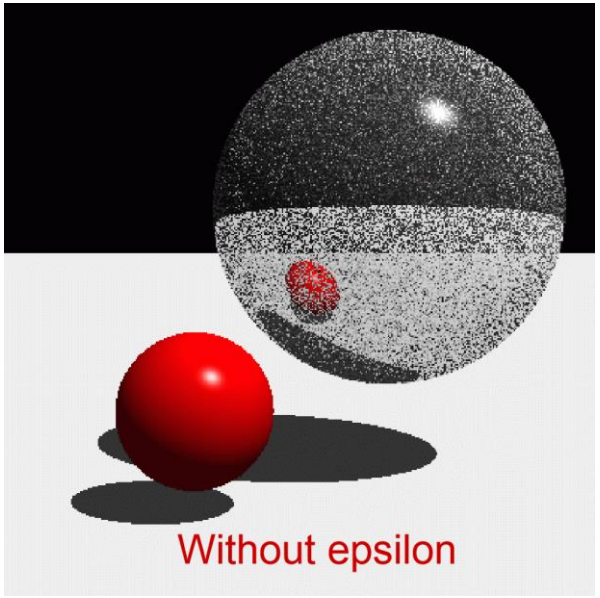
- Reflection angle = view angle
  - Normal component is negated

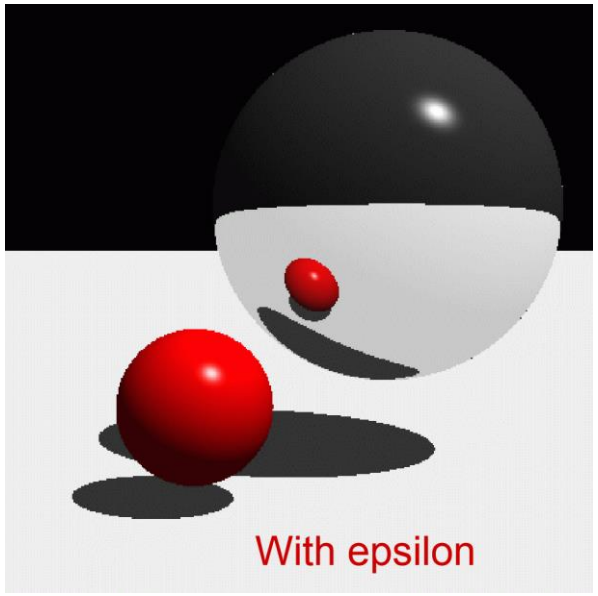
$$R = V - 2(V^T N) \cdot N$$



Mirror Reflection

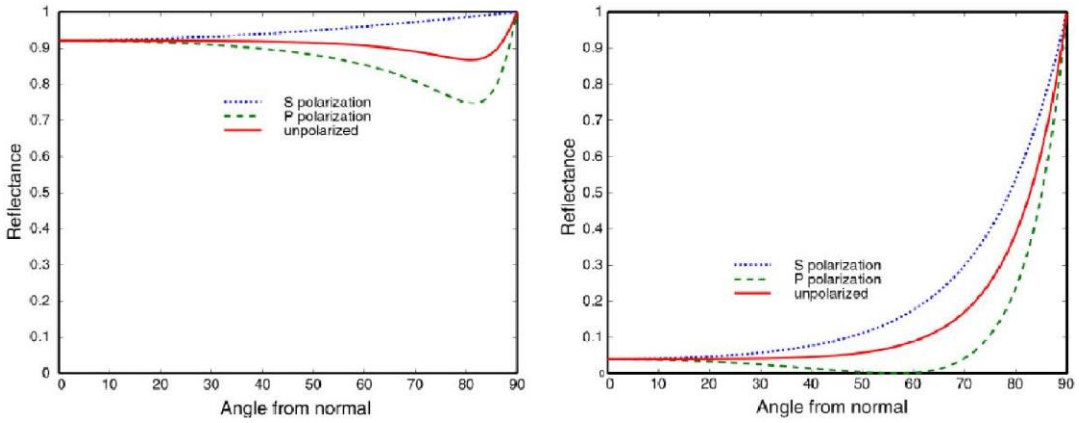
- Cast ray symmetric with respect to the normal
- Multiply by reflection coefficient k_s (color)
- Don’t forget to add epsilon to the ray!





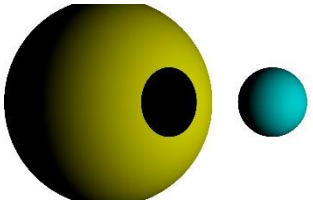
Amount of Reflection

- A typical setting
  - Constant k_s
- More realistic (we' ll do this later):
  - Fresnel reflection term (more reflection at grazing angle)
  - Schlick's approximation: $R(\theta)=R_0+(1-R_0)(1-\cos\theta)^5$
- Fresnel makes a big difference!

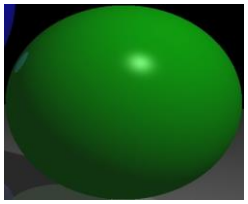


Elements of Ray Tracing

- Shadows

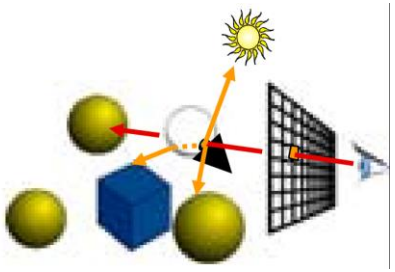


- Reflection



- Refraction

- Recursive Ray Tracing

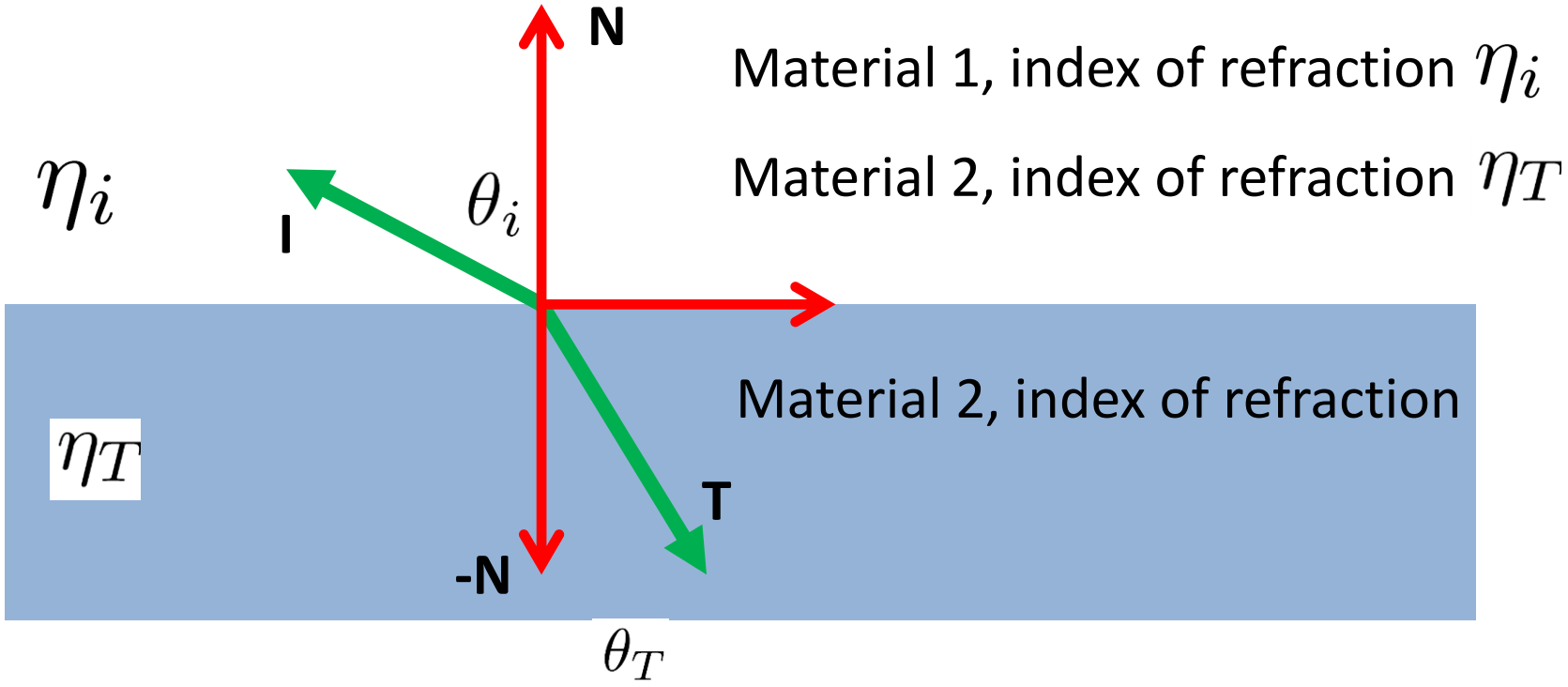


Transparency (Refraction)

- Cast ray in refracted direction
- Multiply by transparency coefficient k_t (color)

Image credit: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/reflection-refraction-fresnel

Refraction



Snell-Descartes Law:

$$\eta_i \sin\theta_i = \eta_T \sin\theta_T$$

Total Internal Reflection

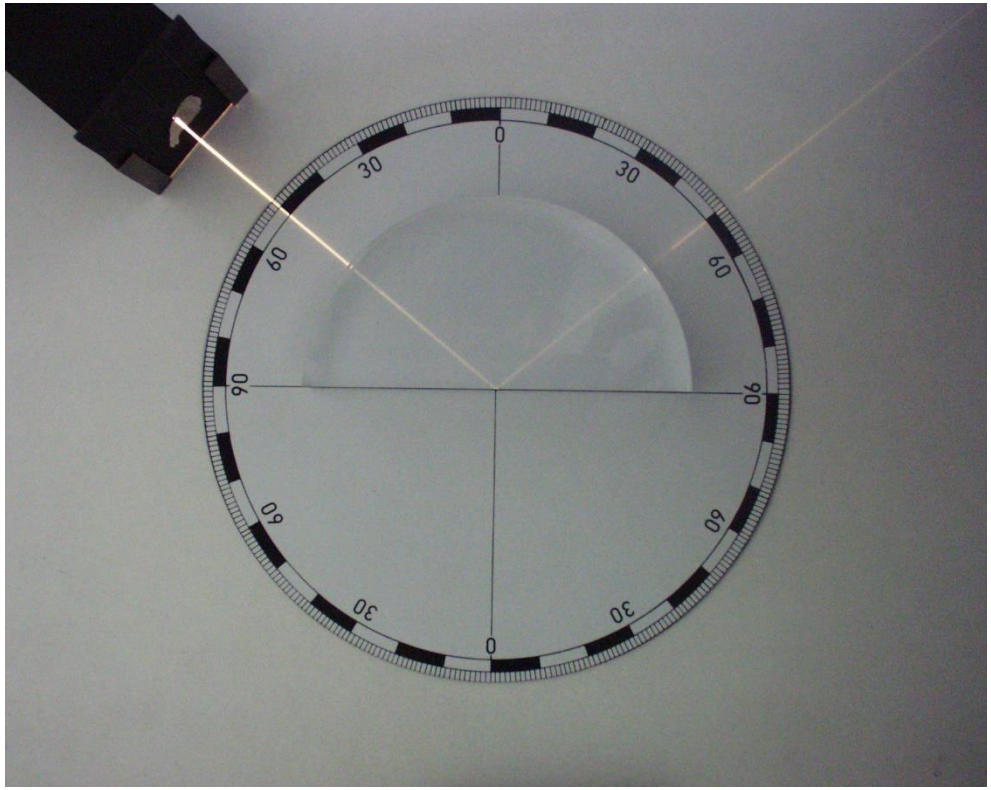


Image credit: https://en.wikipedia.org/wiki/Total_internal_reflection

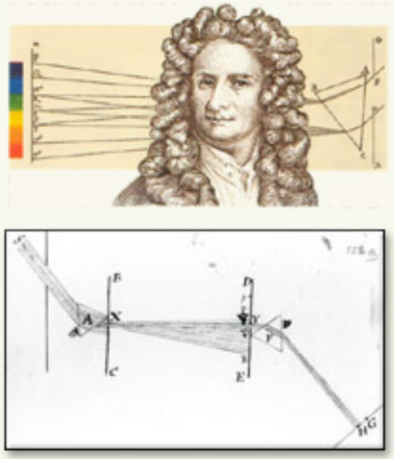
Refraction Demo

[Enright, Enright, Fedkiw, SIGGRAPH’ 02]

Wavelength

- Refraction is wavelength-dependent
- Newton's experiment
- Usually ignored in graphics

Our modern understanding of light and color begins with Isaac Newton (1642-1726) and a series of experiments that he publishes in 1672. He is the first to understand the rainbow — he refracts white light with a prism, resolving it into its component colors: red, orange, yellow, green, blue and violet.



ENLARGE

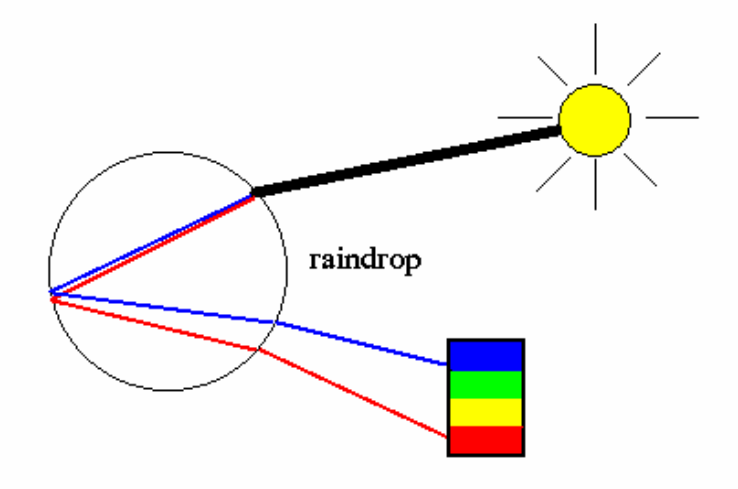
The diagram from Sir Isaac Newton's crucial experiment, 1666-72. A ray of light is divided into its constituent colors by the first prism (left), and the resulting bundle of colred rays is reconstituted into white light by the second.

In the late 1660s, Newton starts experimenting with his *'celebrated phenomenon of colors.'* At the time, people thought that color was a mixture of light and darkness, and that prisms *colored* light. Hooke was a proponent of this theory of color, and had a scale that went from brilliant red, which was pure white light with the least amount of darkness added, to dull blue, the last step before black, which was the complete extinction of light by darkness. Newton realizes this theory was false.

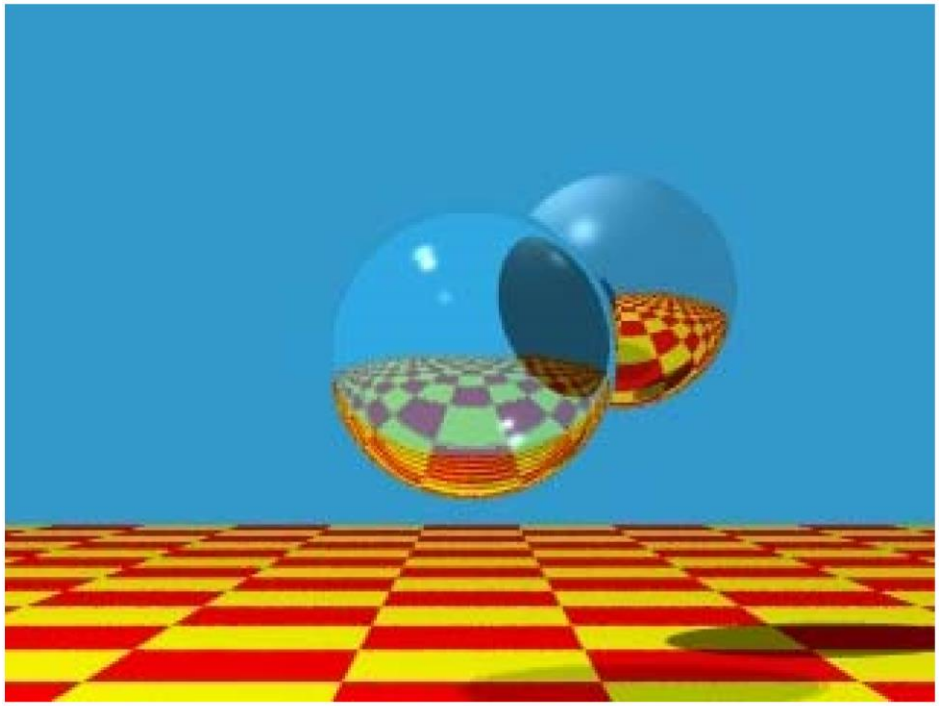
Rainbow

- Refraction depends on wavelength
- Rainbow is caused by

  refraction+internal reflection+refraction
- Maximum for angle around 42 degrees



Ray Tracing

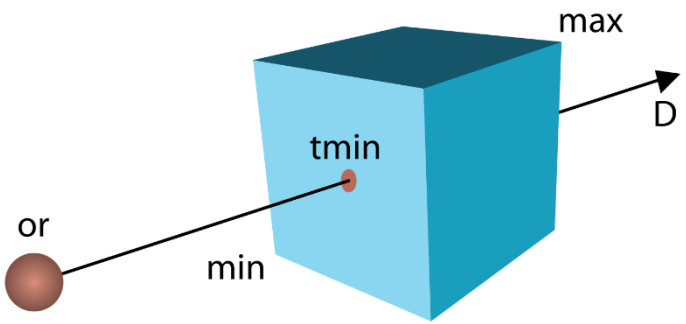


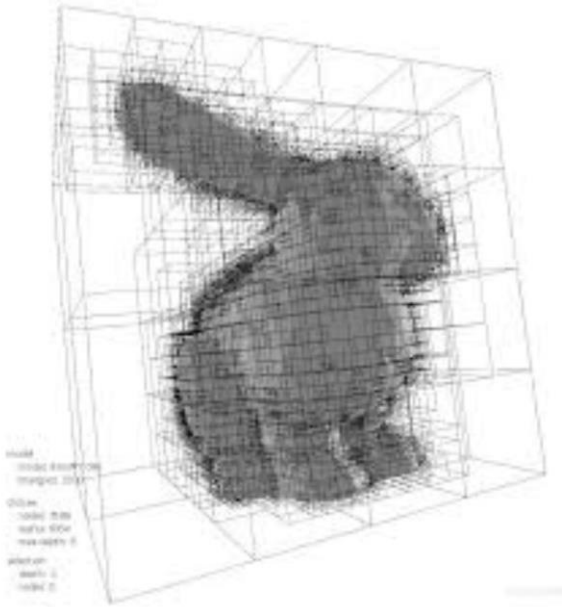
Recursive ray tracing: Turner Whitted, 1980

What are the rendering effects in this image?

Ray-Box Intersection

Why Ray-Box Intersection?





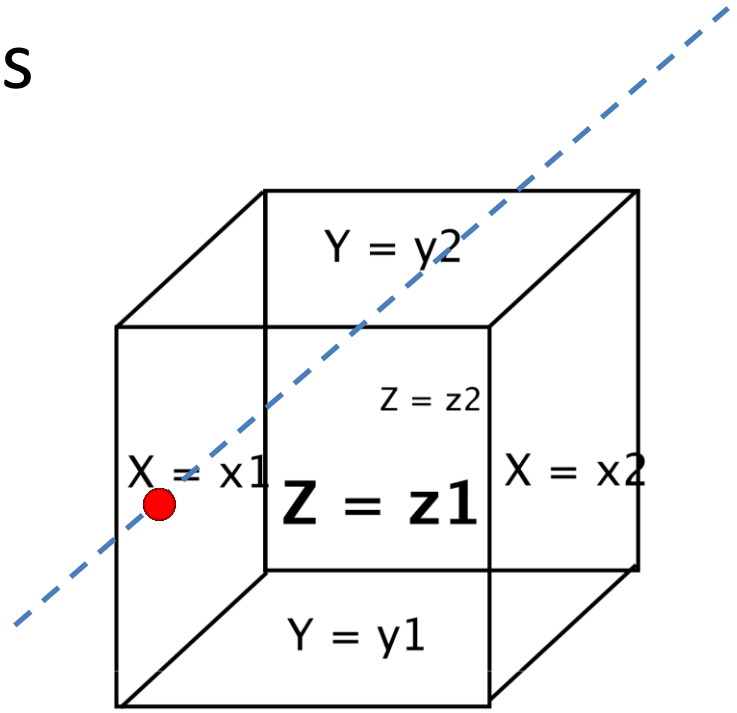
Ray-Box Intersection Test

- Intersect ray with each plane
- Box is the union of 6 planes

$$x = x_1,\ x = x_2$$

$$y = y_1,\ y = y_2$$

$$z = z_1,\ z = z_2$$



- Ray/axis-aligned plane is easy:

E.g., solve x component: $e_x + tD_x = x_1$

Ray-Box Intersection Test

- Intersect ray with each plane
- Sort the intersections
- Choose intersection

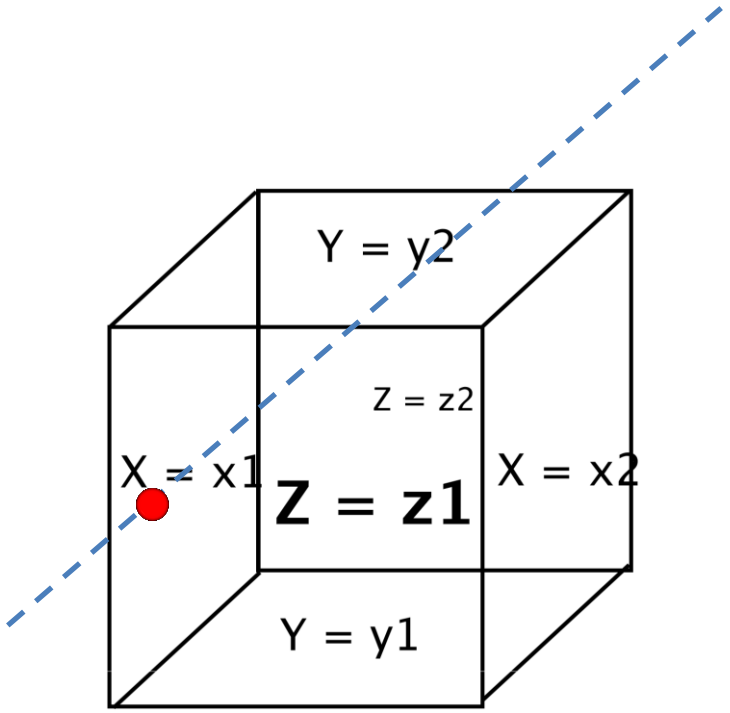
  with the smallest t > 0

  that is within the range

  of the box

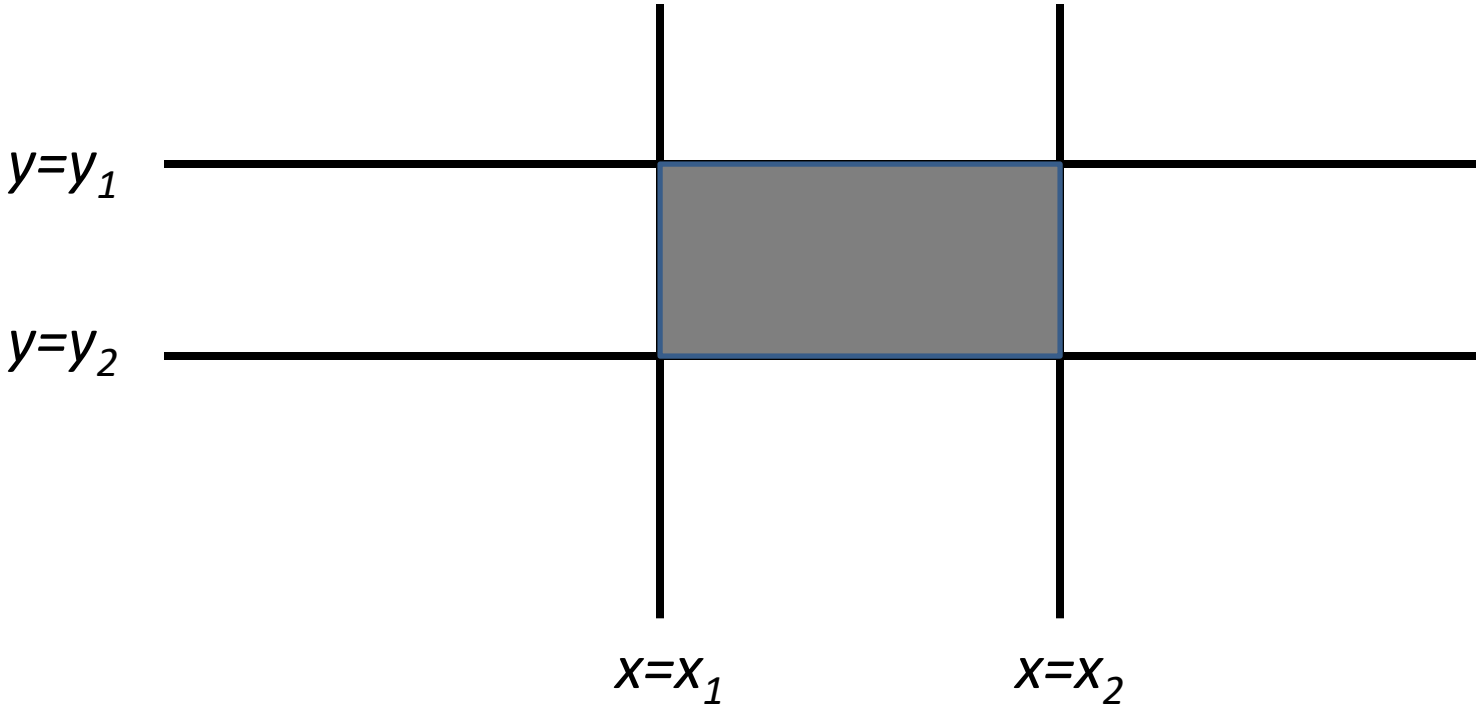
- We can do more

  efficiently



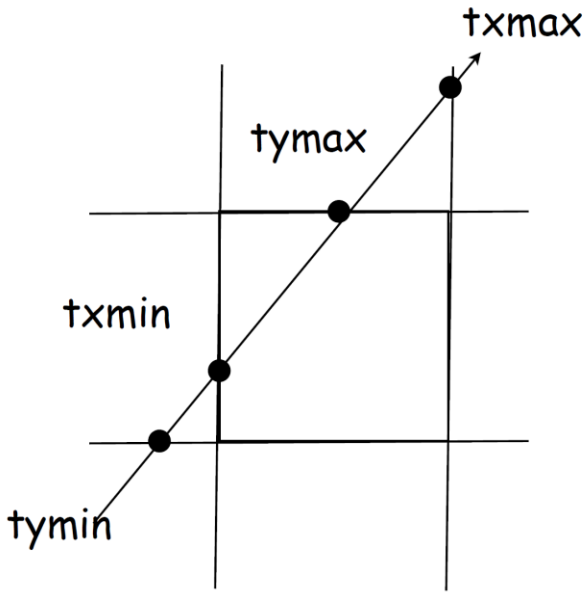
Only Consider 2D for Now

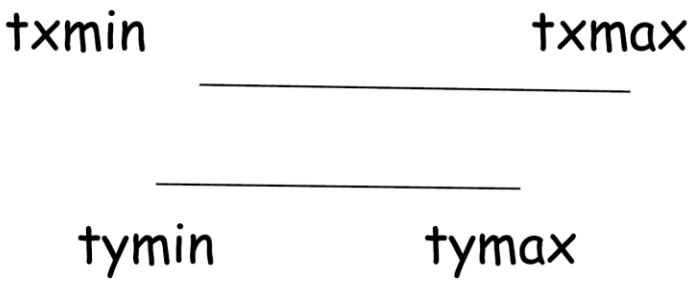
- if a point (x,y) is in the box, then (x,y) in $[x_1,x_2]$ x $[y_1, y_2]$



The Principle

- Assuming the ray hits the box boundary lines at intervals [txmin, txmax], [tymin, tymax], the ray hits the box if and only if the intersection of the two intervals is not empty





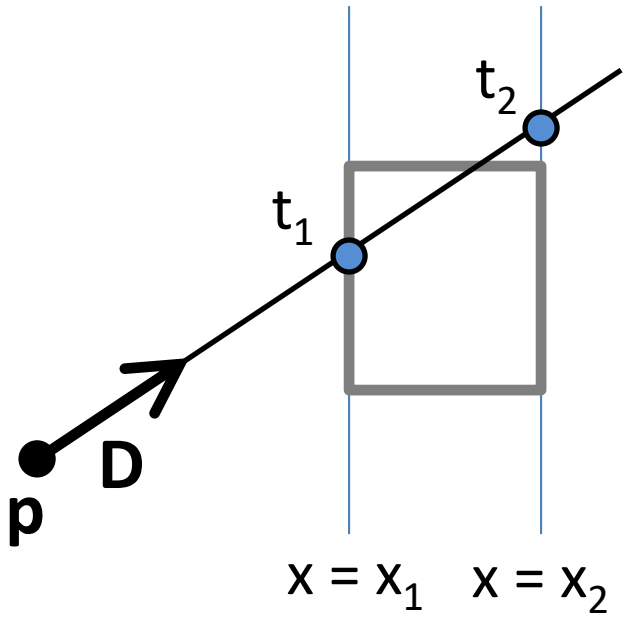
Pseudo Code

```
txmin = (x1 - ex) / Dx;  // Assume Dx > 0
txmax = (x2 - ex) / Dx;
tymin = (y1 - ey) / Dy;  // Assume Dy > 0
tymax = (y2 - ey) / Dy;
if (txmin > tymax || tymin > txmax)
  return false;
else
  return true;

txmin = (x2 - ex) / Dx;  // Assume Dx < 0
txmax = (x1 - ex) / Dx;
tymin = (y2 - ey) / Dy;  // Assume Dy < 0
tymax = (y1 - ey) / Dy;
if (txmin > tymax || tymin > txmax)
  return false;
else
  return true;
```

Update $[t_{near}, t_{far}]$

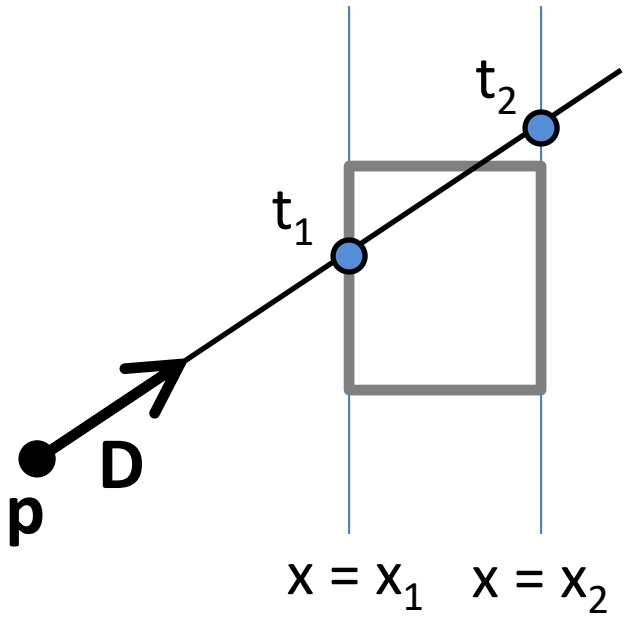
- Set t_{near} = - and t_{far} = +
- For each axis, compute t_1 and t_2
  - Make sure $t_1 < t_2$
  - If $t_1 > t_{near}$, $t_{near} = t_1$
  - If $t_2 < t_{far}$, $t_{far} = t_2$
- If $t_{near} > t_{far}$, box is missed



Now Consider All Axis

- We will calculate t_1 and t_2 for each axis (x, y, and z)
- Update the intersection interval as we compute t_1 and t_2 for each axis
- remember

```
t1 = (x1 - px) / Dx;
t2 = (x2 - px) / Dx;
```



Algorithm

Set $t_{near} = -\infty$, $t_{far} = \infty$

$R(t) = p + t * \mathbf{D}$

For each pair of planes P associated with X, Y, and Z do: (example uses X planes)

if direction $\mathbf{D}_x = 0$ then

if ($p_x < x_1$ or $p_x > x_2$)

return FALSE

else

begin

$t_1 = (x_l - p_x) / \mathbf{D}_x$

$t_2 = (x_h - p_x) / \mathbf{D}_x$

if $t_1 > t_2$ then swap (t_1, t_2)

if $t_1 > t_{near}$ then $t_{near} = t_1$

if $t_2 < t_{far}$ then $t_{far} = t_2$

if $t_{near} > t_{far}$ return FALSE

if $t_{far} < 0$ return FALSE

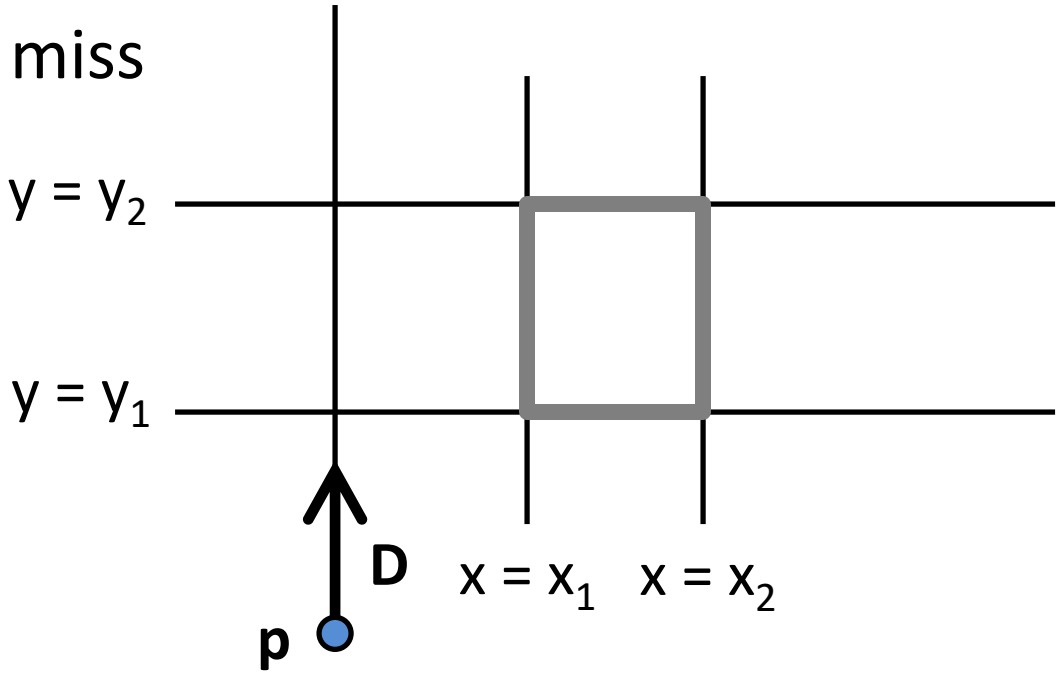
end

Return t_{near}

Special Case

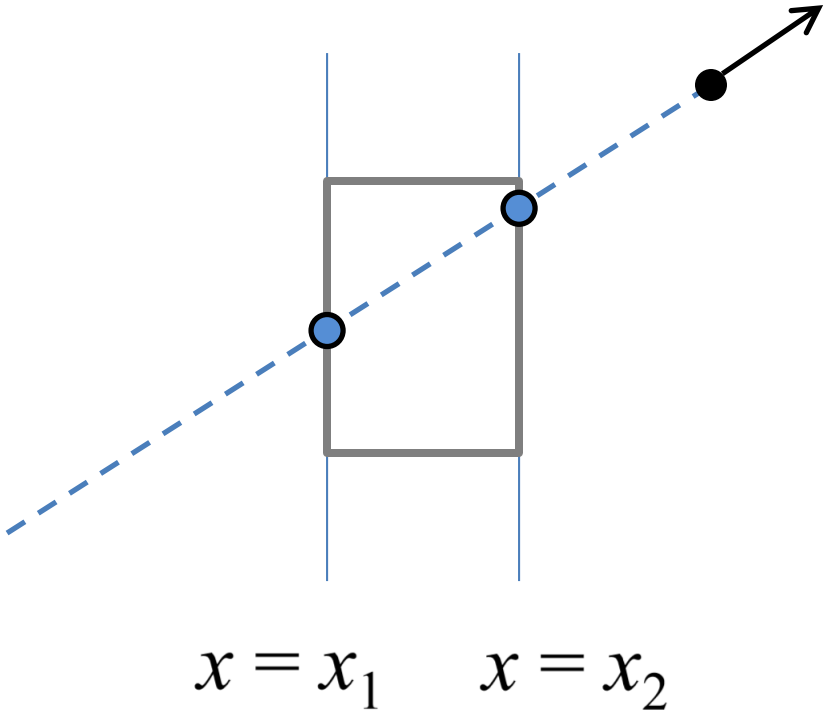
- Ray is parallel to an axis
  - If $D_x = 0$ or $D_y = 0$ or $D_z = 0$

- $p_x < x_1$ or $p_x < x_2$ then miss

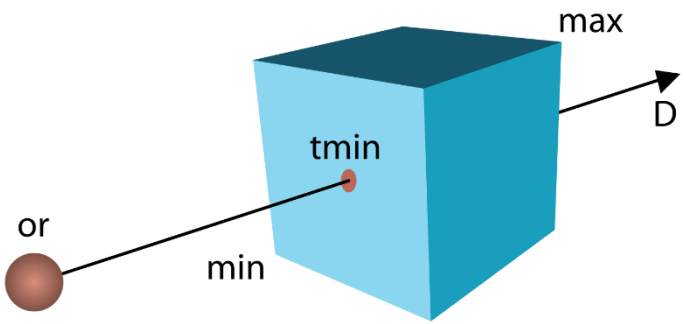


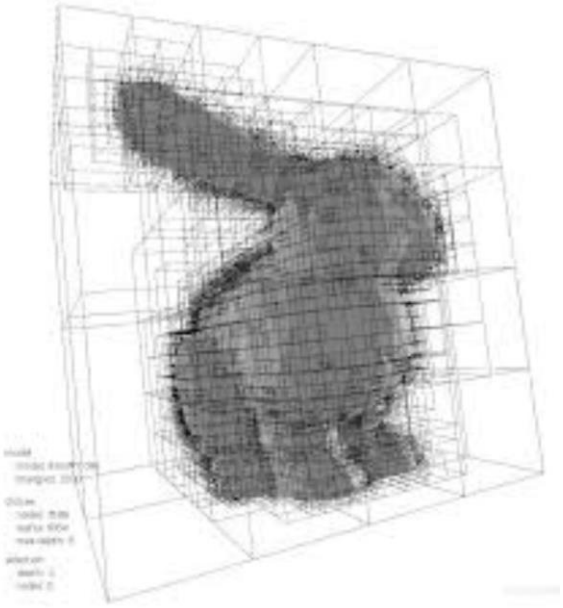
Special Case

- Box is behind the eye
  - If $t_{far} < 0$ box is behind



Ray-Mesh Intersection





If a ray does not intersect with a box, then we can pass all the triangles inside that box

Q: What about triangles that intersect with box boundaries?

Questions?