Z-buffer algorithm

for each polygon in model

  project vertices of polygon onto viewing plane

  for each pixel inside the projected polygon

    calculate pixel colour

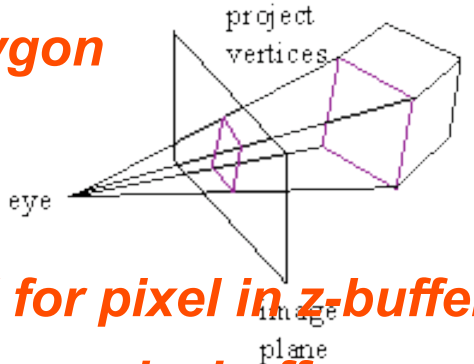
    calculate pixel z-value

    compare pixel z-value to value stored for pixel in z-buffer

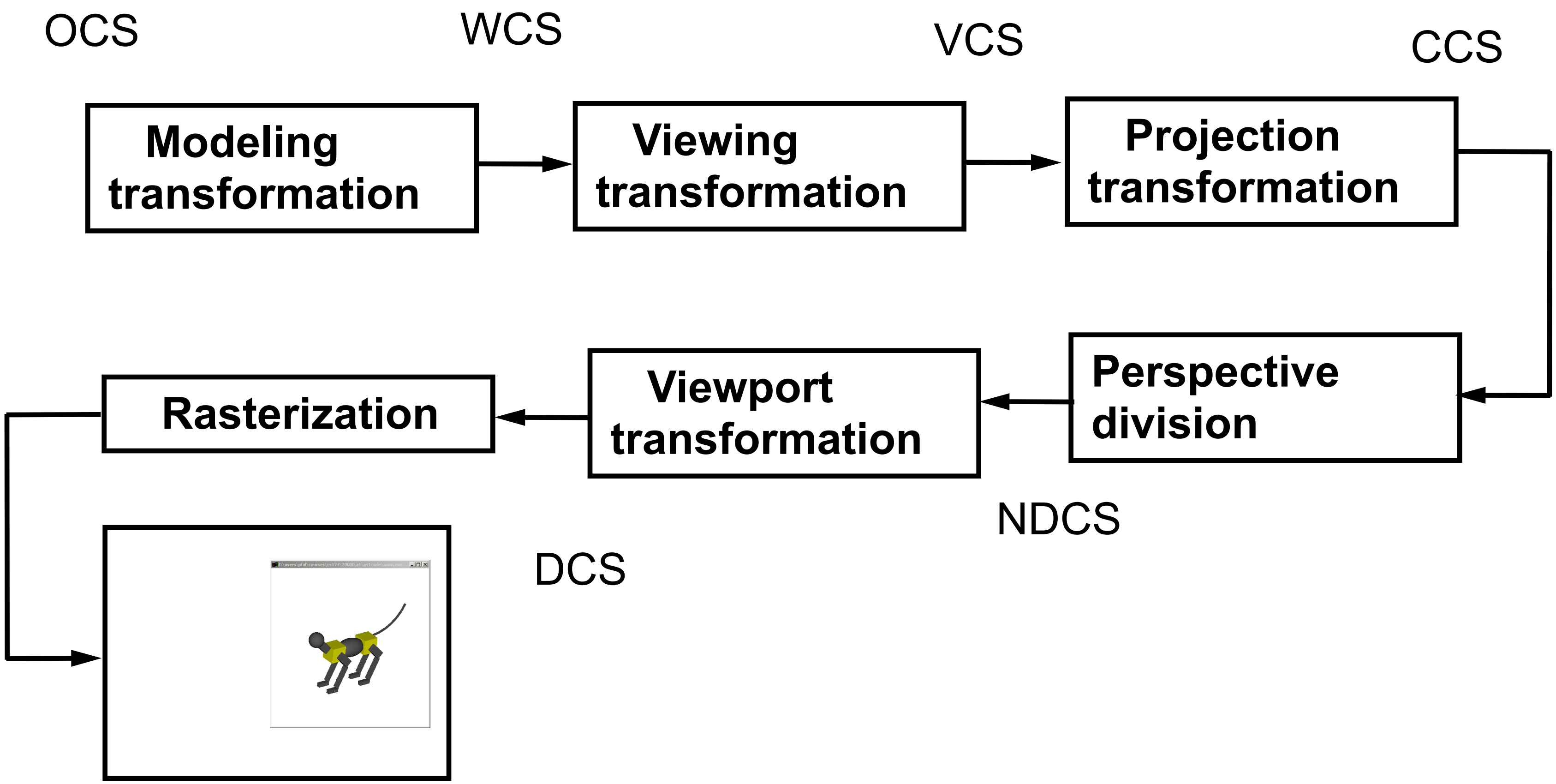
    if pixel is closer, draw it in frame-buffer and z-buffer

  end

end



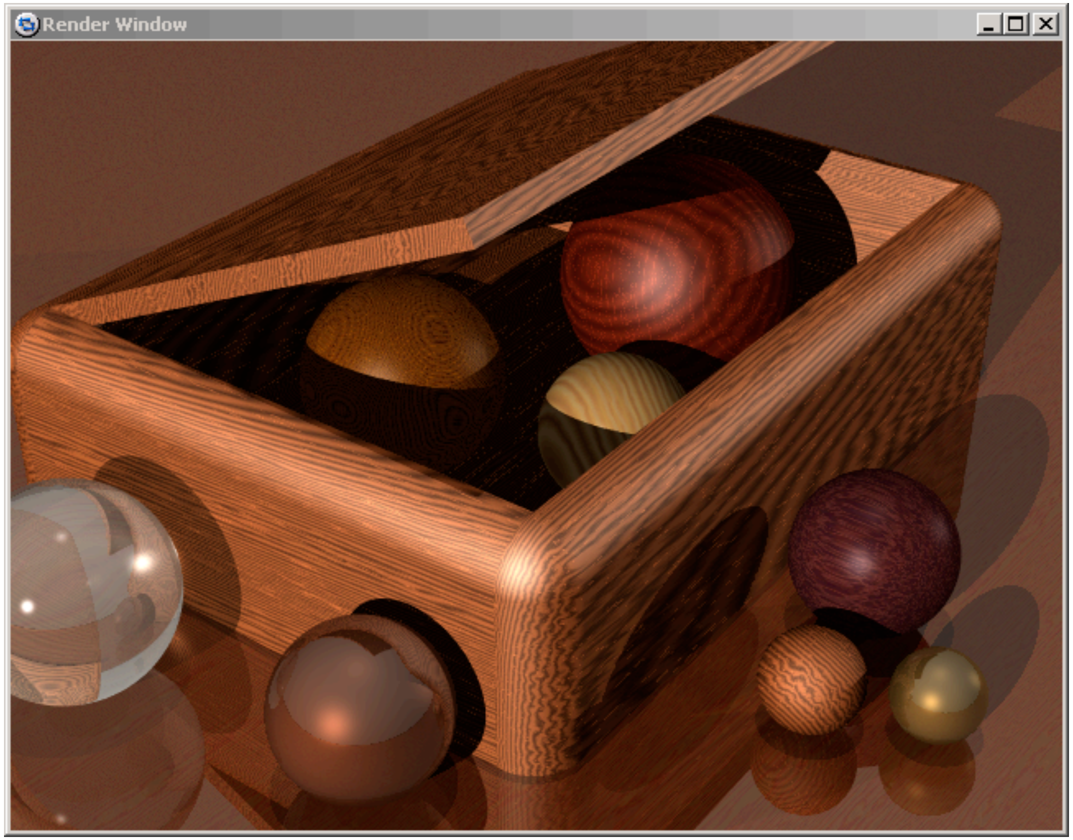
COMPLETION OF Z-buffer Graphics Pipeline



Raytracing

Rendered by PovRay 3.6

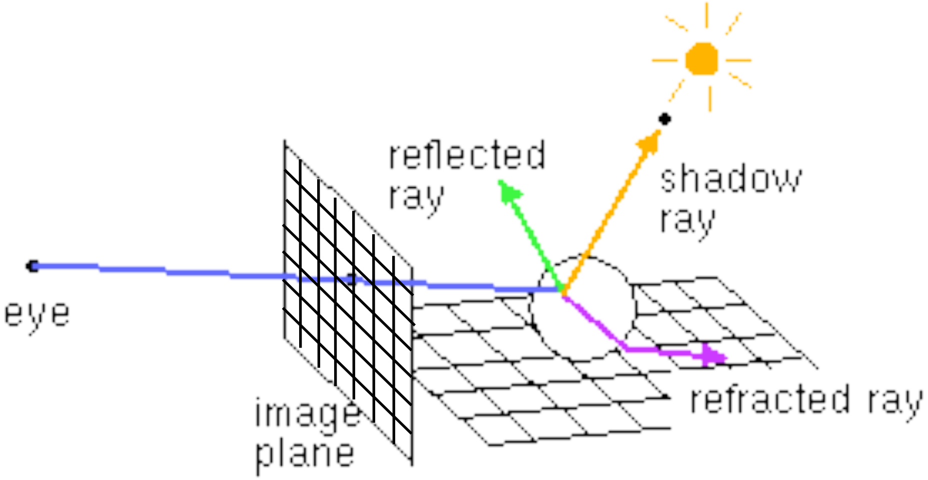
www.povray.org



Raytracing

Tuned for specular and transparent objects

- Partly physics, geometric, optics



A pixel should have the color of the object point that projects to it.

Raytracing

for each pixel on screen

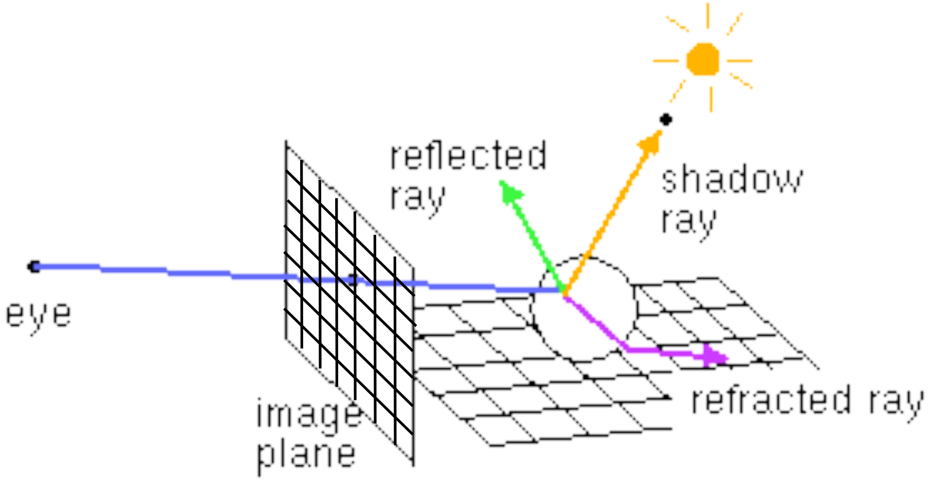
determine ray from eye through pixel

find closest intersection of ray with an object

cast off reflected and refracted ray, recursively

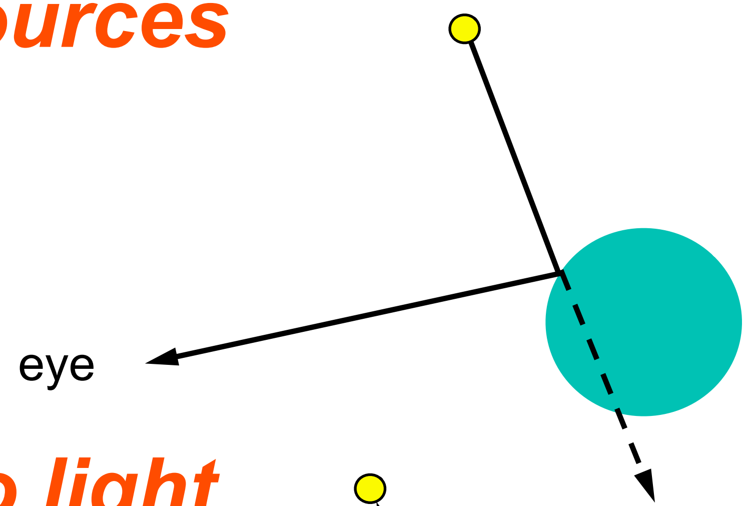
calculate pixel colour, draw pixel

end

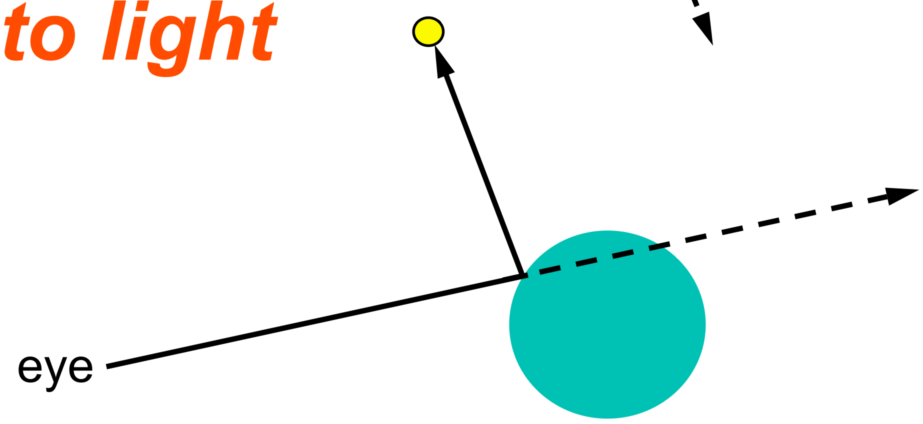


Forward and Backward methods

Forward: from light sources to eye

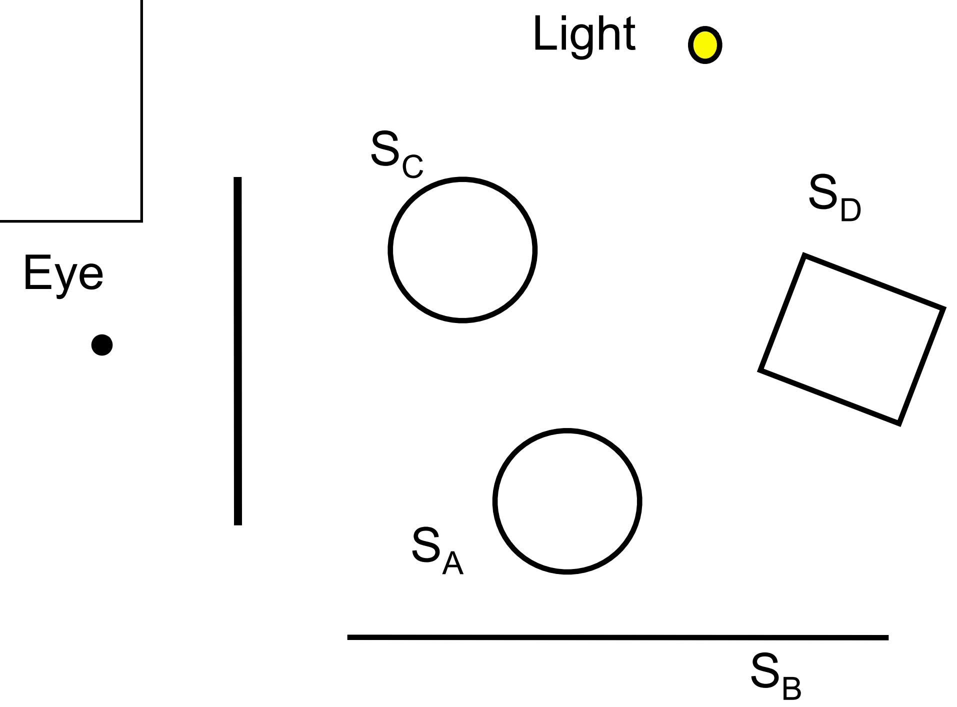


Backward: from eye to light sources



Scene

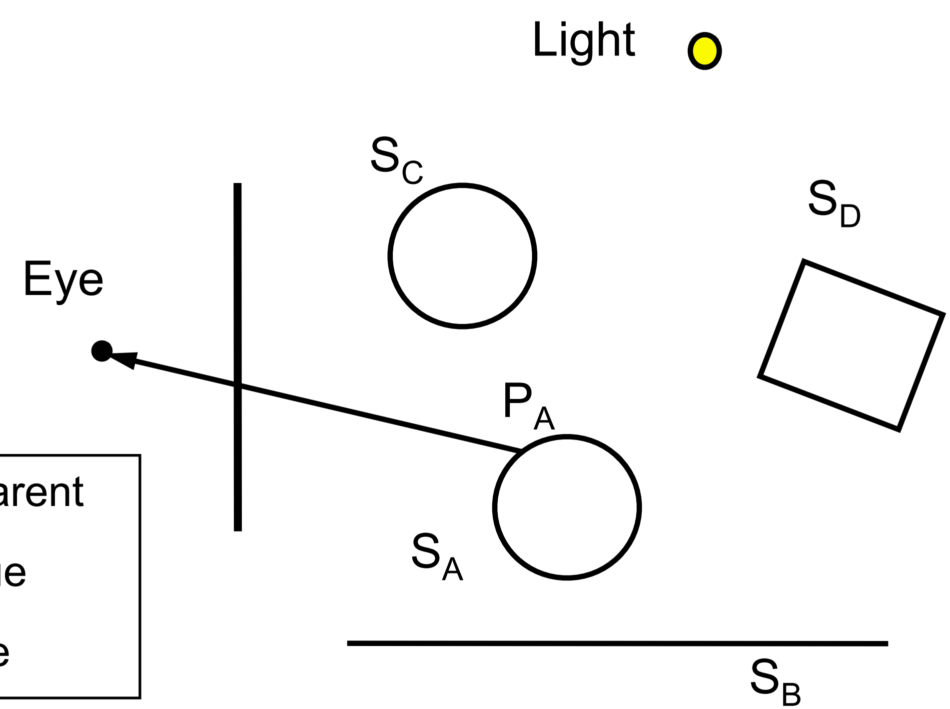
S_A	shiny, transparent
S_B, S_D	diffuse, opaque
S_C	shiny, opaque



Three sources of light

The light that point P_A emits to the eye comes from:

light sources
other objects (reflection)
other objects (refraction)



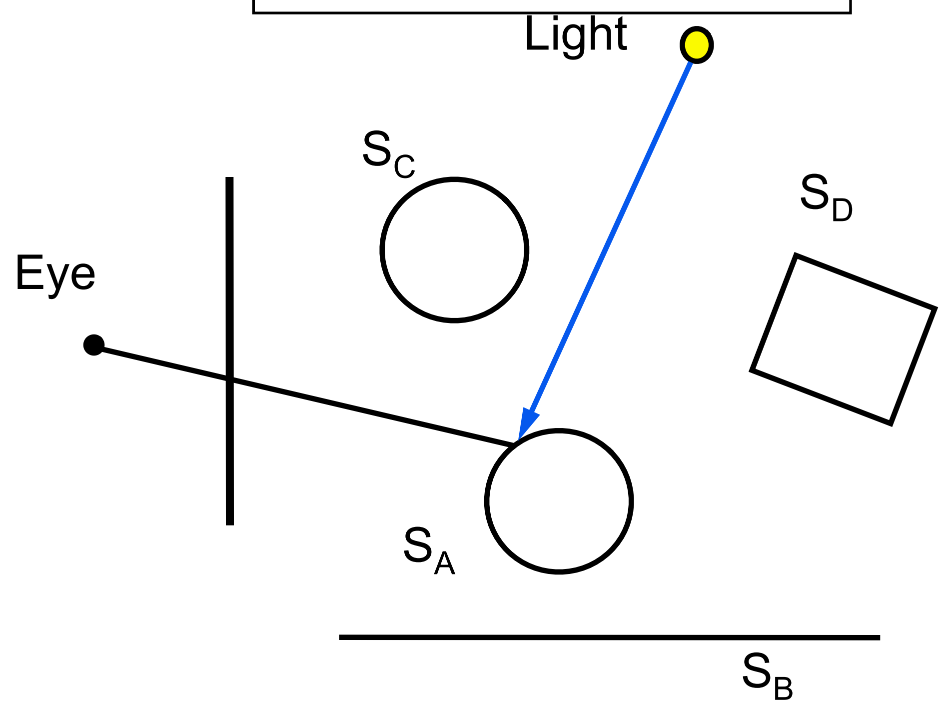
S_A	shiny, transparent
S_B,S_D	diffuse,opaque
S_C	shiny, opaque

Directly from light source

Local illumination model:

$I = Ia + Idiff + Ispec$

S_A	shiny, transparent
S_B,S_D	diffuse,opaque
S_C	shiny, opaque



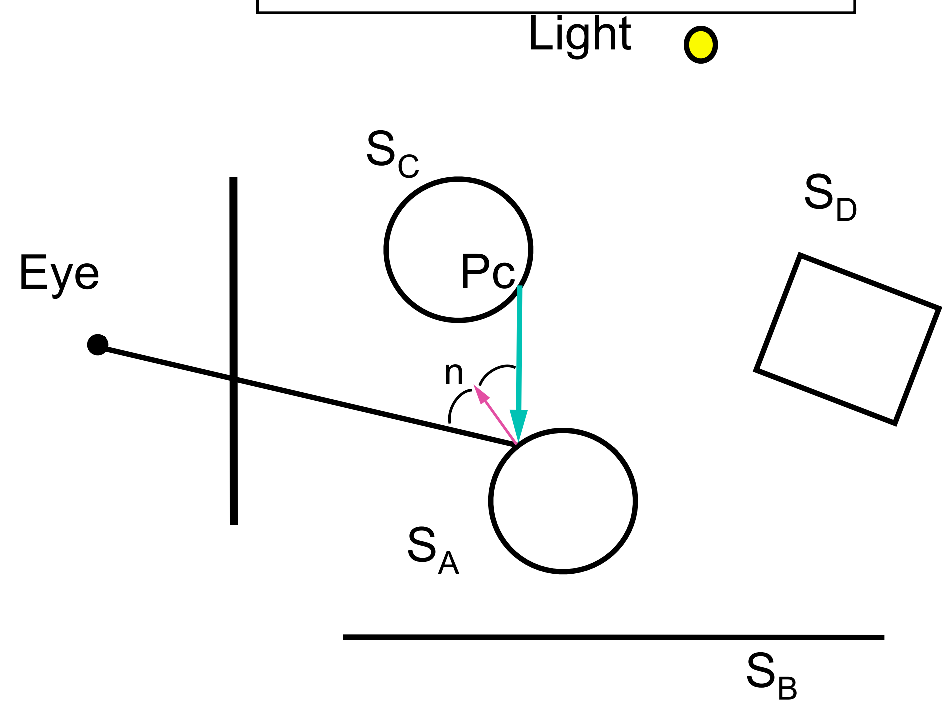
Reflection

What is the color that is reflected to P_A and that PA reflects back to the eye?

The color of P_C.

What is the color of P_C ?

S_A	shiny, transparent
S_B,S_D	diffuse,opaque
S_C	shiny, opaque



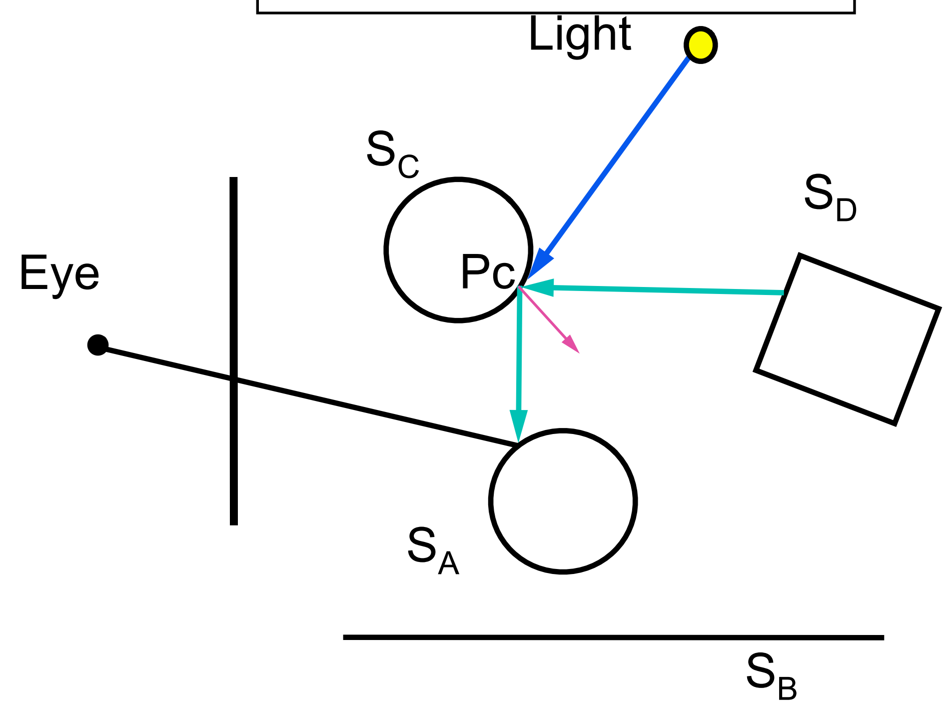
Reflection

What is the color of Pc?

Just like P_A : raytrace P_C i.e compute the three contributions from

- *Light sources*
- *Reflection*
- *Refraction*

S_A	shiny, transparent
S_B,S_D	diffuse,opaque
S_C	shiny, opaque



Refraction

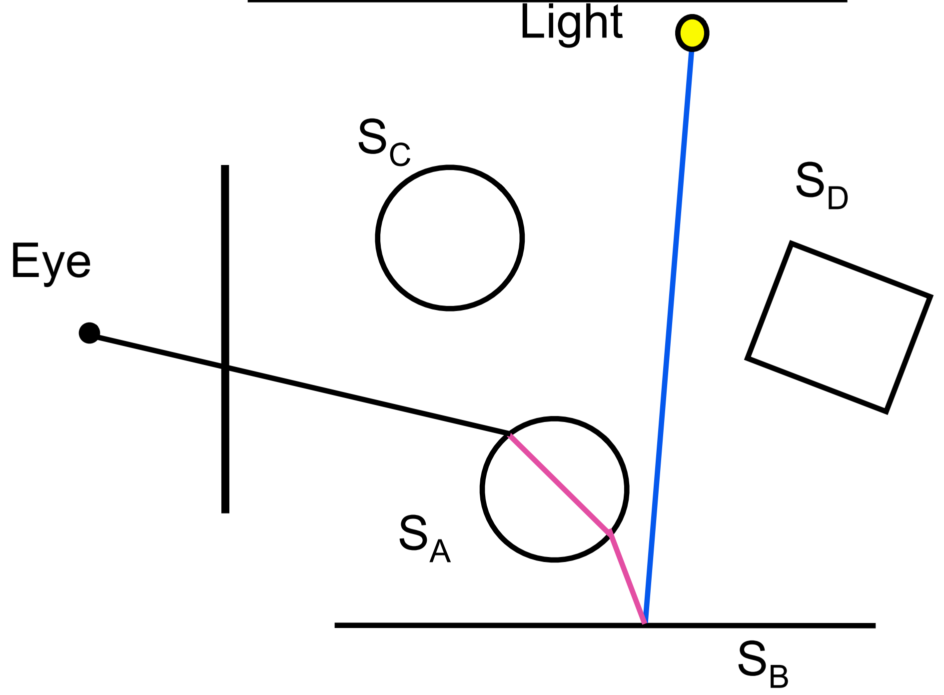
Transparent materials

How do you compute the refracted contribution?

You raytrace the refracted ray.

1. *Lights*
2. *Reflection*
3. *Refraction*

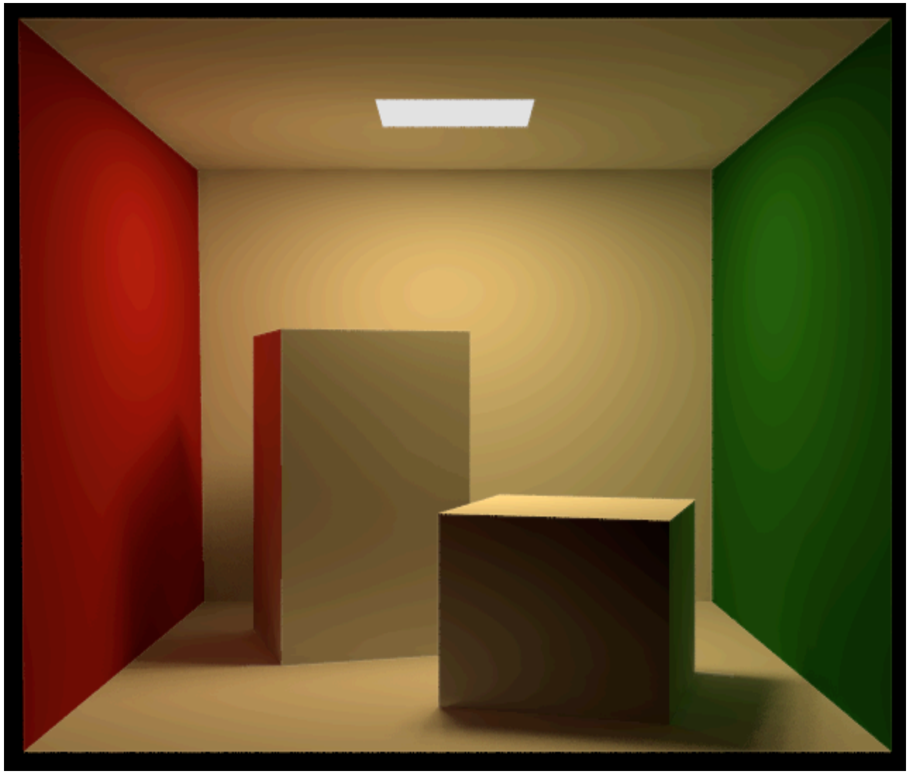
S_A	shiny, transparent
S_B,S_D	diffuse,opaque
S_C	shiny, opaque



What are we missing?

What are we missing?

Diffuse objects do not receive light from other objects.

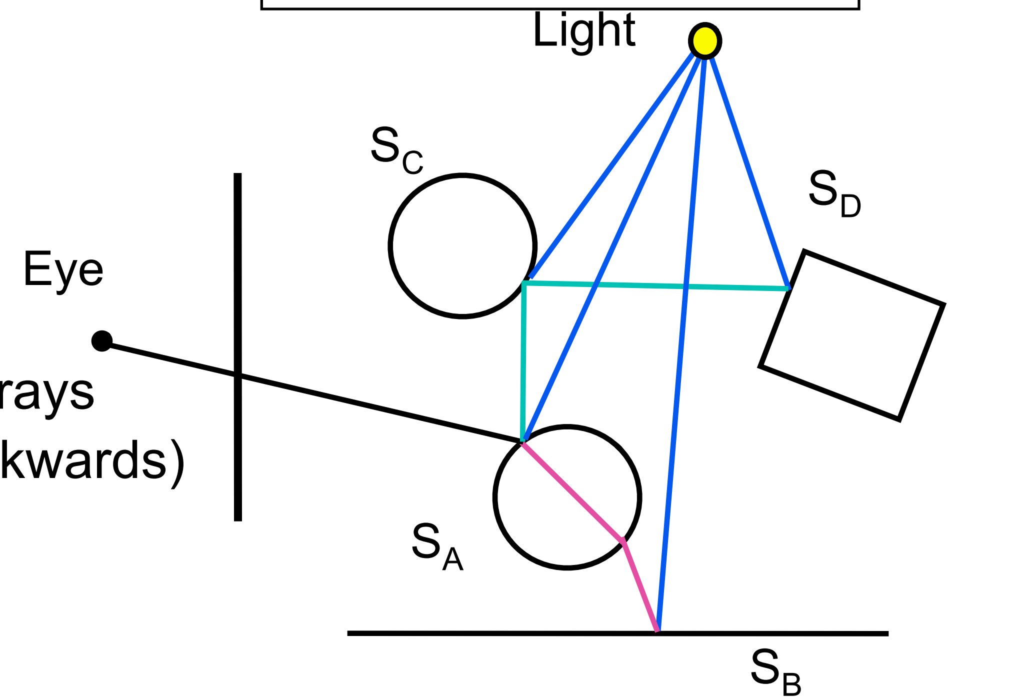


Three sources of light together

S_A	shiny, transparent
S_B,S_D	diffuse,opaque
S_C	shiny, opaque

The color that the pixel is assigned comes from:

light sources
other objects (reflection)
other objects (refraction)



It is more convenient to trace the rays from the eye to the scene (backwards)

Backwards Raytracing Algoritm

For each pixel construct a ray: eye→ pixel

raytrace(ray)

P = compute_closest_intersection(ray)

color_local = ShadowRay(light1, P)+…

+ ShadowRay(lightN, P)

color_reflect = raytrace(reflected_ray)

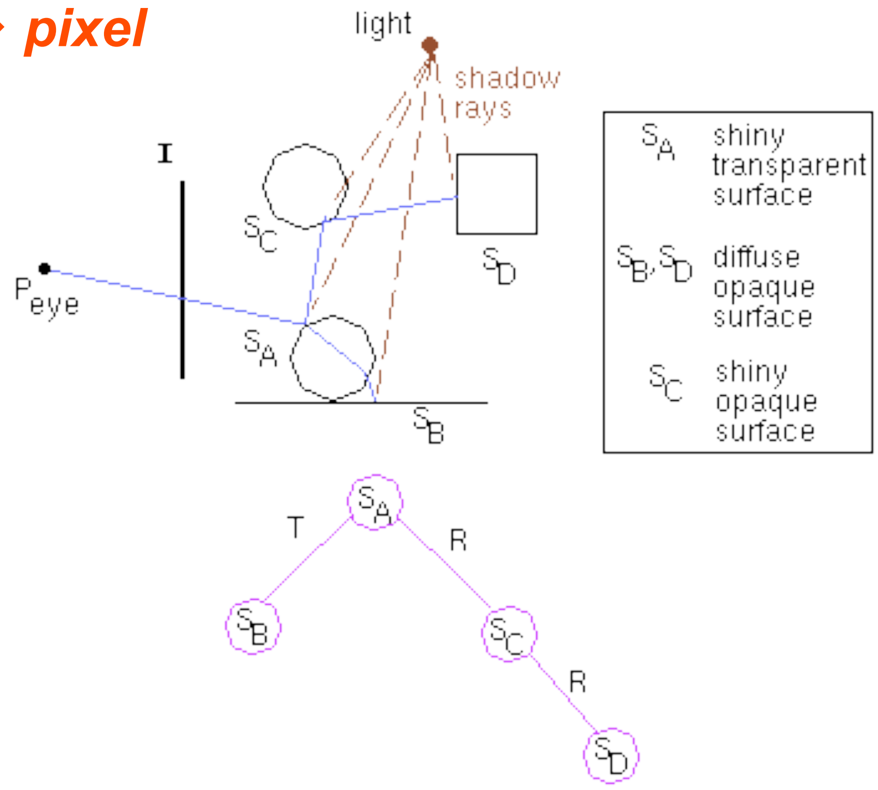
color_refract = raytrace(refracted_ray)

color = color_local

+ k_{re}*color_reflect

+ k_{ra}*color_refract

return(color)



How many levels of recursion do we use?

The more the better.

Infinite reflections at the limit.

Stages of raytracing

Setting the camera and the image plane

Computing a ray from the eye to every pixel and trace it in the scene

Object-ray intersections

Shadow, reflected and refracted ray at each intersection

Setting up the camera

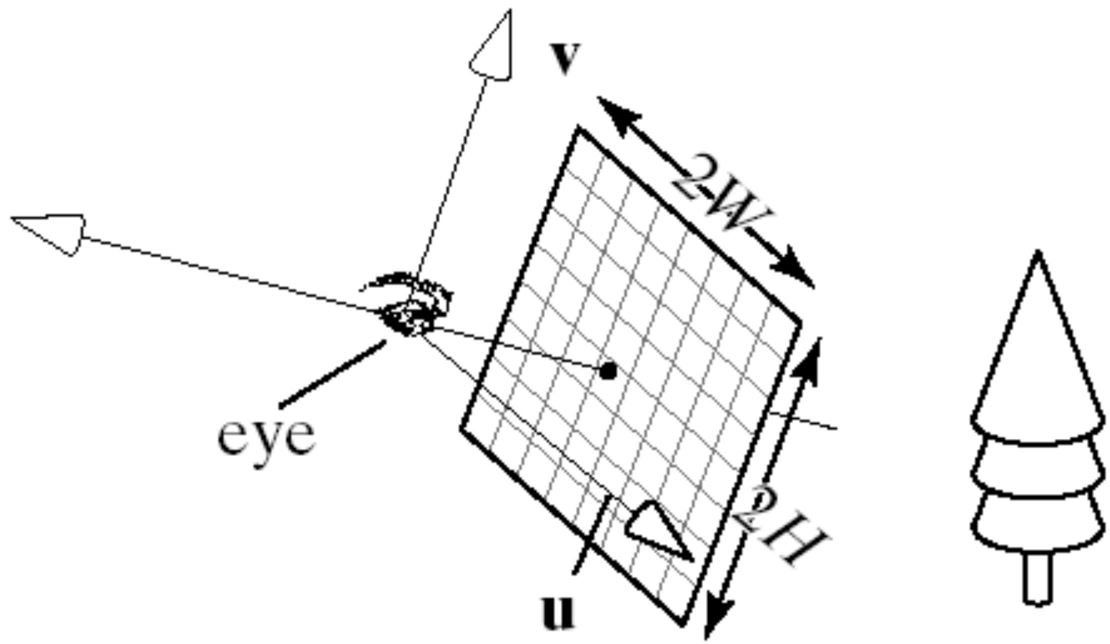


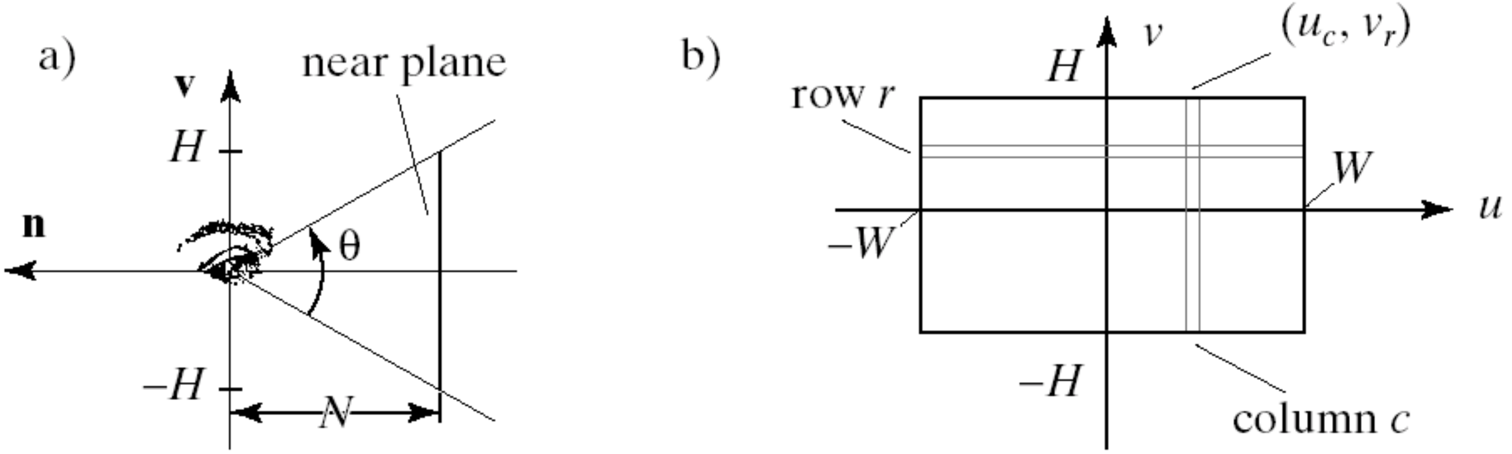
Image parameters

Width 2W, Height 2H

Number of pixels nCols x nRows

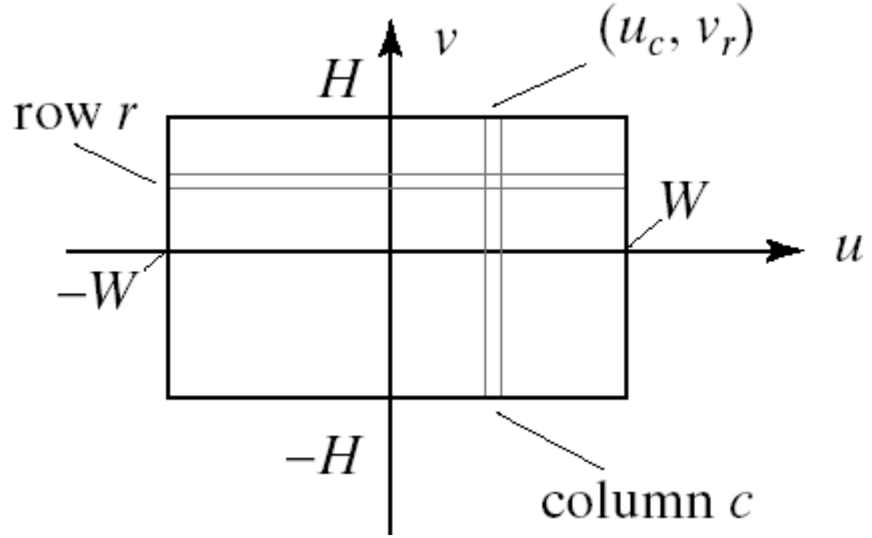
Camera coordinate system (eye, u,v,n)

Image plane at -N



Pixel coordinates in camera coordinate system

Lower left corner of pixel P(r,c) has coordinates in camera space:



$$u_c = -W + W\frac{2c}{nCols}, \quad c = 0, 1, \ldots, nCols - 1,$$

$$v_r = -H + H\frac{2r}{nRows}, \quad r = 0, 1, \ldots, nRows - 1,$$

Ray through pixel

Lower left corner

$$Camera\ \ coordinates: \quad P(r,c) = (u_c, v_r, -N)$$

$$Wolrd\ \ coordinates: \quad P(r,c) = eye - N\mathbf{n} + u_c\mathbf{u} + u_r\mathbf{v}$$

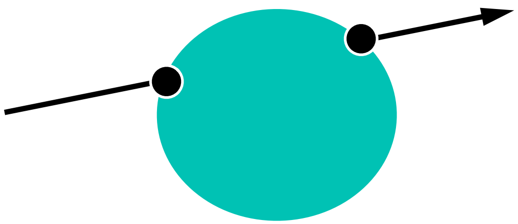
Ray through pixel:

$$ray(r,c,t) = eye + t(P(r,c) - eye)$$

$$ray(r,c,t) = eye + t(-N\mathbf{n} + W(\frac{2c}{nCols} - 1)\mathbf{u} + H(\frac{2r}{nRows} - 1)\mathbf{v}$$

Ray-object intersections

Unit sphere at origin - ray intersection:



$$ray(t) = S + \mathbf{c}t$$

$$Sphere(P) = |P| - 1 = 0$$

$$Sphere(ray(t)) = 0 \Rightarrow$$

$$|S + \mathbf{c}t| - 1 = 0 \Rightarrow (S + \mathbf{c}t)(S + \mathbf{c}t) - 1 = 0 \Rightarrow$$

$$\boxed{|\mathbf{c}|^2 t^2 + 2(S \cdot \mathbf{c})t + |S|^2 - 1 = 0}$$

That's a quadratic equation

Solving a quadratic equation

$$|\mathbf{c}|^2 t^2 + 2(S \cdot \mathbf{c})t + |S|^2 - 1 = 0$$
$$At^2 + 2Bt + C = 0$$

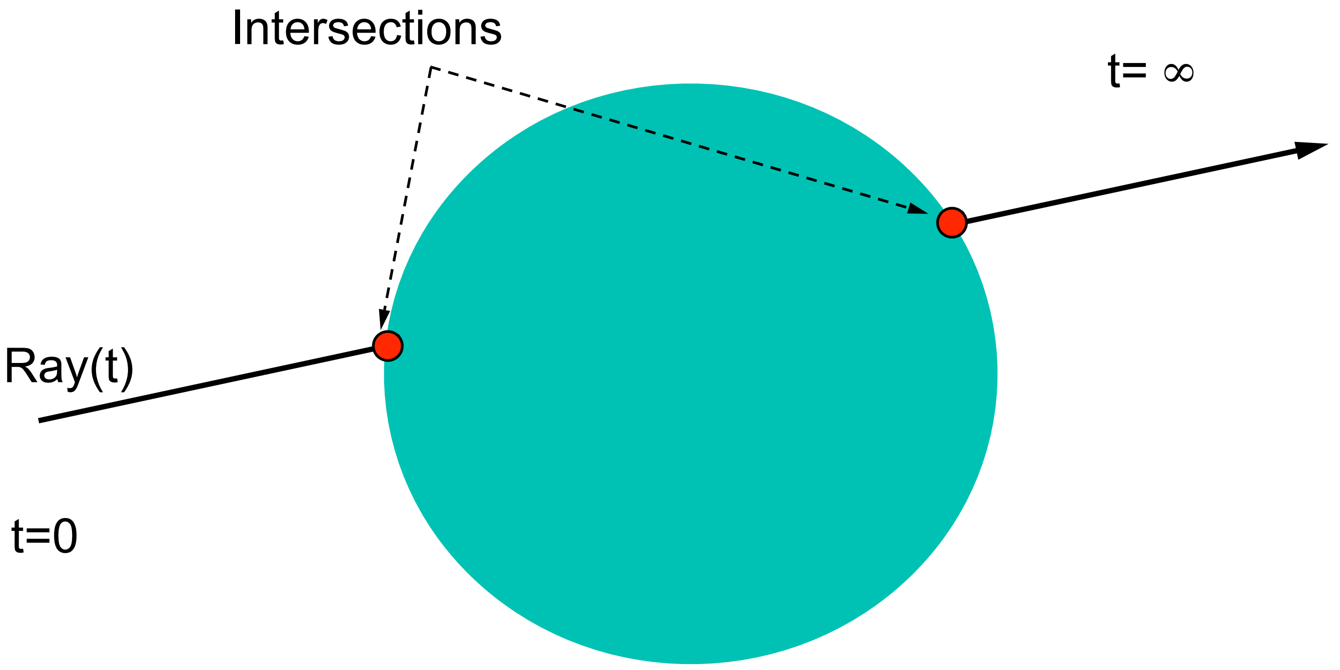
$$t_h = -\frac{B}{A} \pm \frac{\sqrt{B^2 - AC}}{A}$$
$$t_h = -\frac{S \cdot \mathbf{c}}{|\mathbf{c}|^2} \pm \frac{\sqrt{(S \cdot \mathbf{c})^2 - |\mathbf{c}|^2 (|S|^2 - 1)}}{|\mathbf{c}|^2}$$

If $(B^2 - AC) = 0$ one solution

If $(B^2 - AC) < 0$ no solution

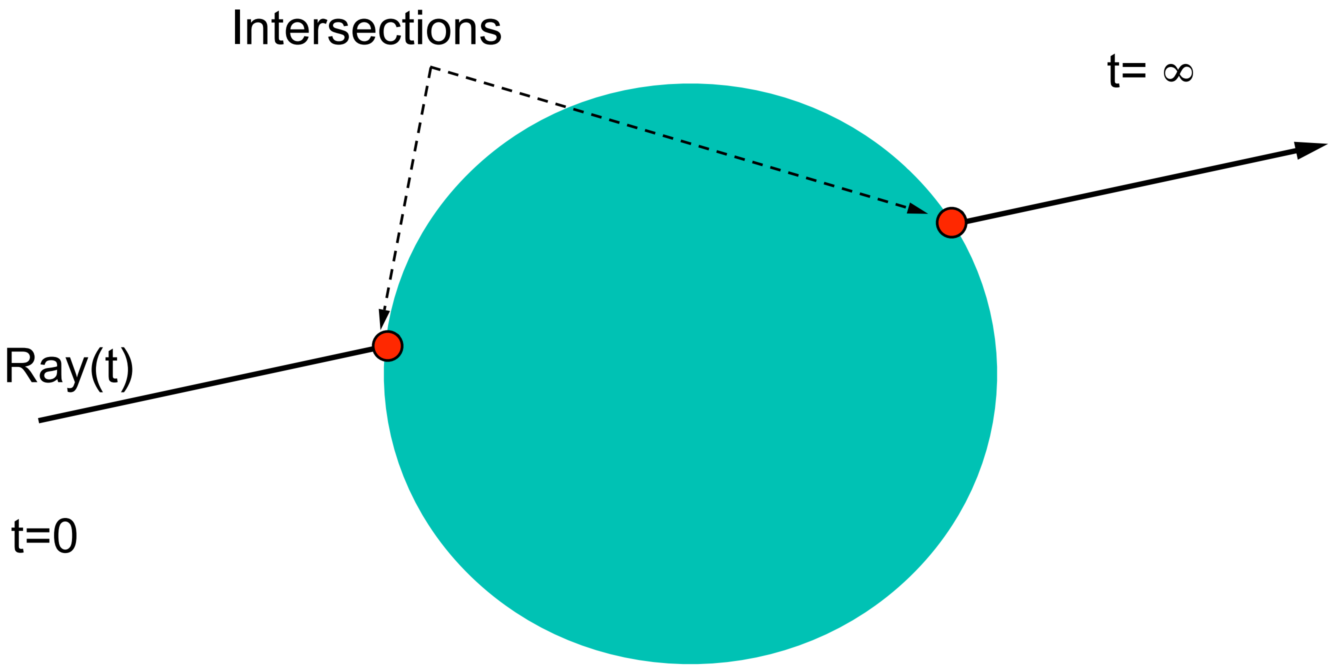
If $(B^2 - AC) > 0$ two solutions

First intersection?



First intersection?

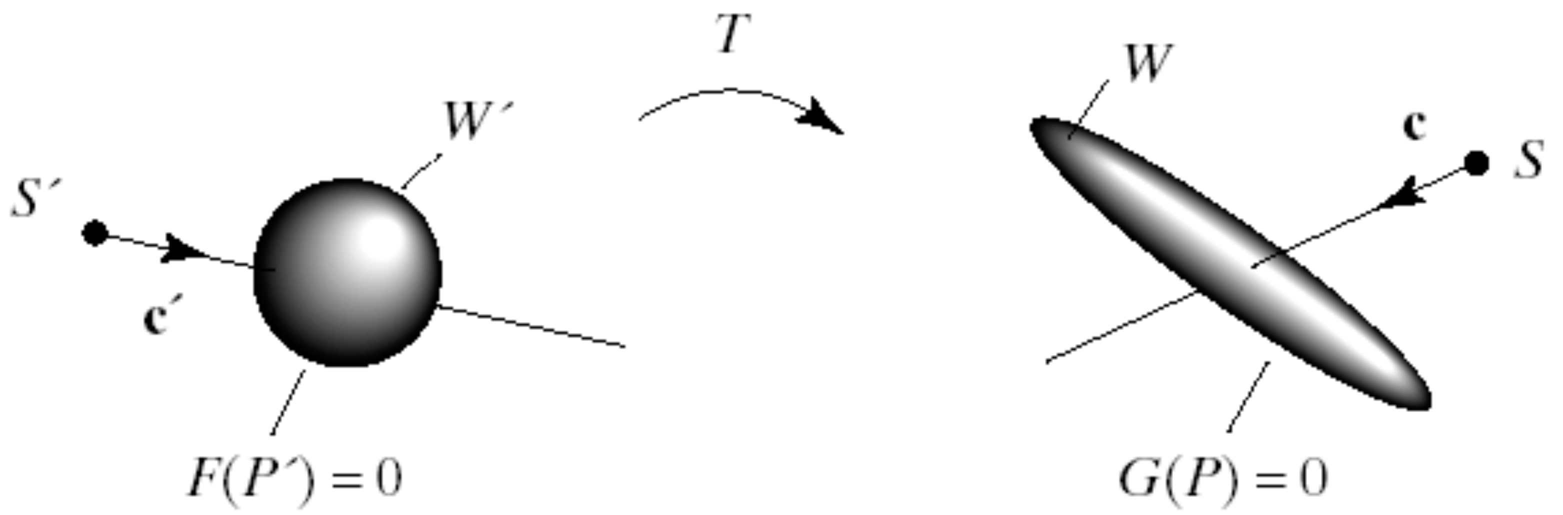
$t1 < t2$



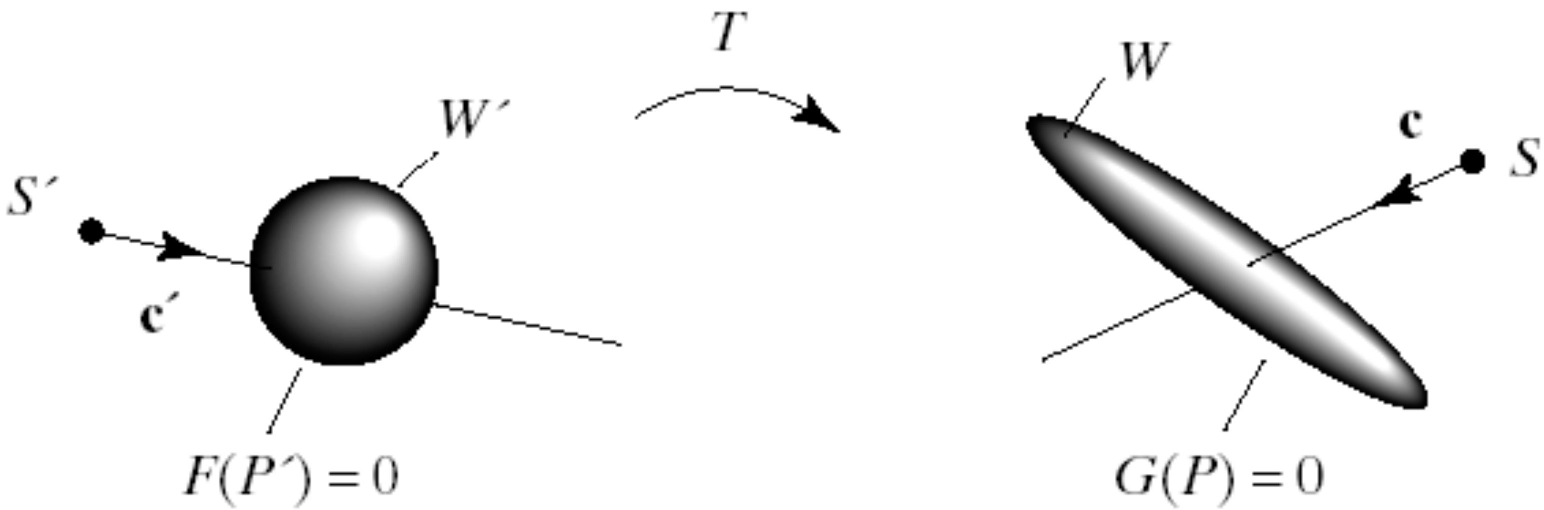
Transformed primitives?

That was a canonical sphere.

Where does S+ct hit the transformed sphere G = T(F) ?



Linear transformation

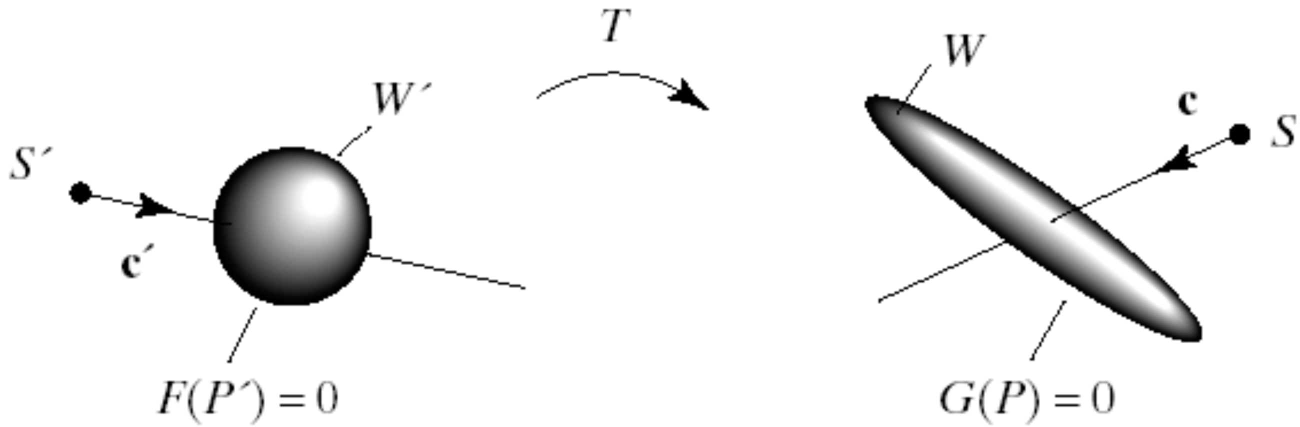


Implicit equation $G(P) = 0$.

Untransformed implicit equation F(P') = 0.

$P = MP' \Rightarrow P' = M^{-1}P$

Linear transformation



$$P = MP' \Rightarrow P' = M^{-1}P$$

$$F(P') = F(T^{-1}(P)) = 0 \Rightarrow F(T^{-1}(P)) = 0$$
$$F(T^{-1}(S + \mathbf{c}t)) = 0 \Rightarrow$$
$$F(T^{-1}(S) + T^{-1}(\mathbf{c}t)) = 0$$

Which means that we can intersect the inverse transformed ray with the untransformed primitive.

Final Intersection

Inverse transformed ray

$$\tilde{r}(t) = M^{-1}\begin{pmatrix} S_x \\ S_y \\ S_z \\ 1 \end{pmatrix} + M^{-1}\begin{pmatrix} c_x \\ c_y \\ c_z \\ 0 \end{pmatrix} = \tilde{S}' + \ \tilde{\mathbf{c}}' t$$

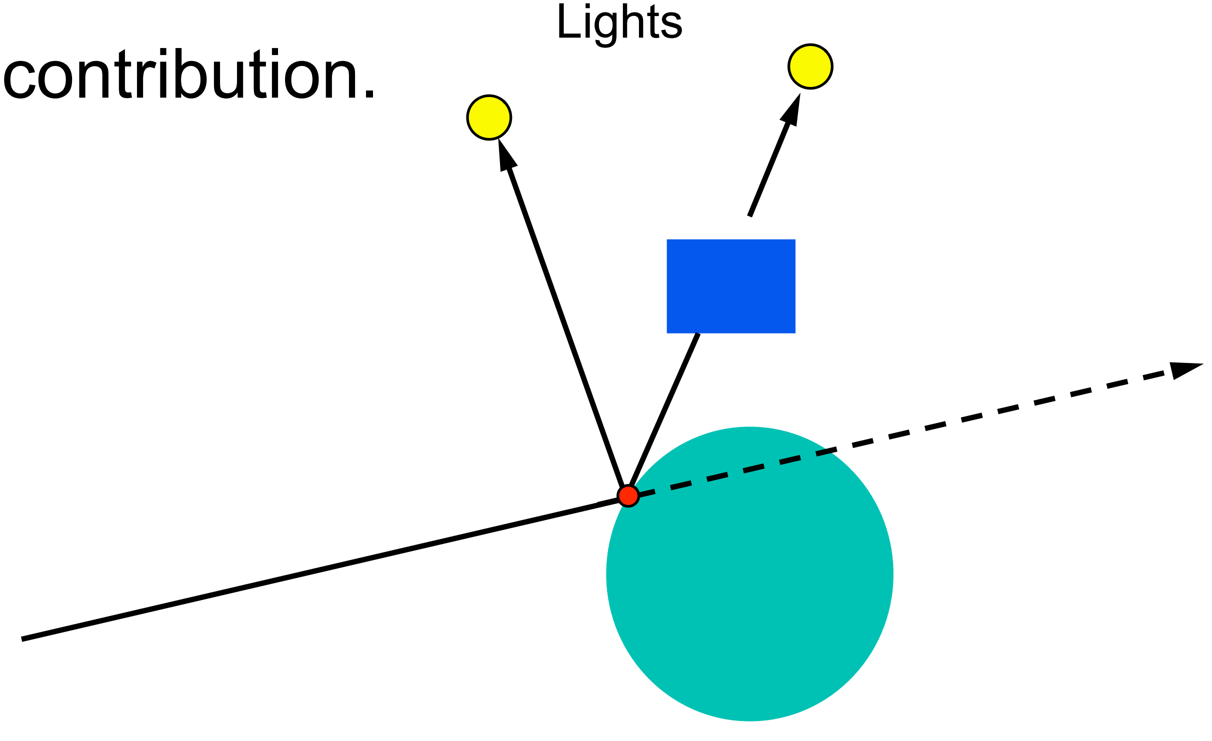
- Drop 1 and O to get *S'*+c'*t.*

So ..for each object

- Inverse transform ray and get *S'*+c'*t.*
- Find the intersection *t*, t_h, between inv-ray and canonical sphere.
- Use t_h in the untransformed ray *S*+c*t* to find the intersection.

Shadow ray

- For each light intersect shadow ray with all objects.
- If no intersection is found apply local illumination at intersection.
- If in shadow no contribution.



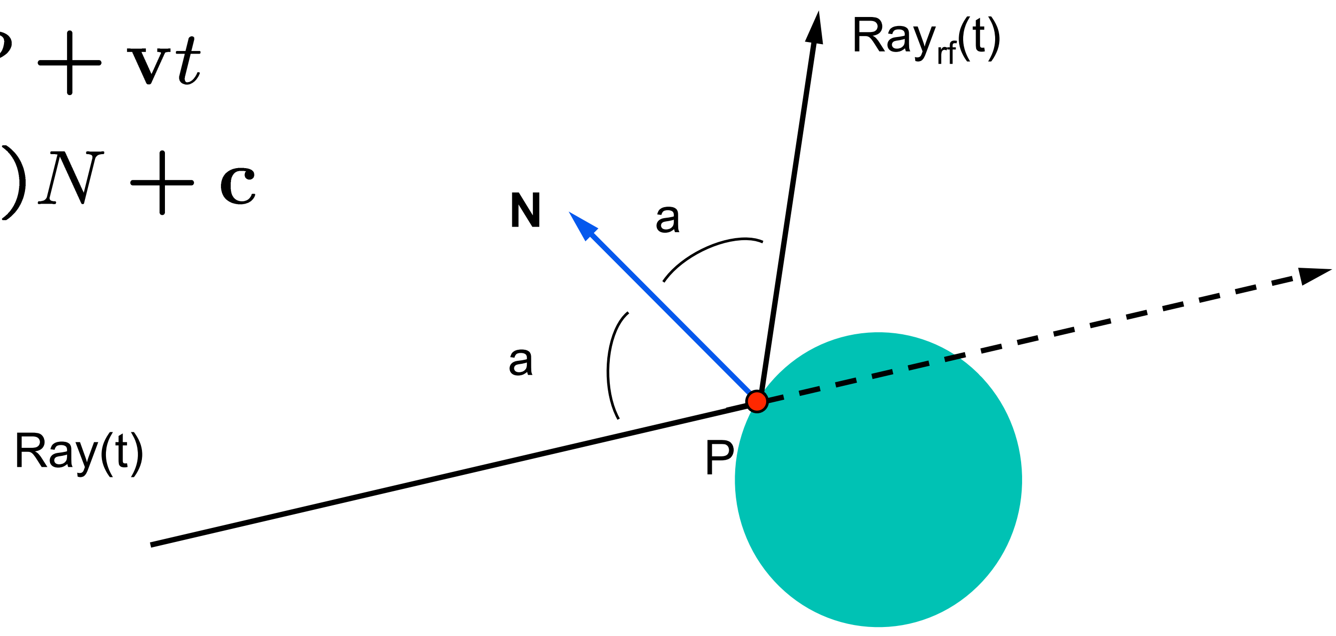
Reflected ray

Raytrace the reflected ray

$$Ray(t) = A + \mathbf{c}t$$

$$Ray_{rf}(t) = P + \mathbf{v}t$$

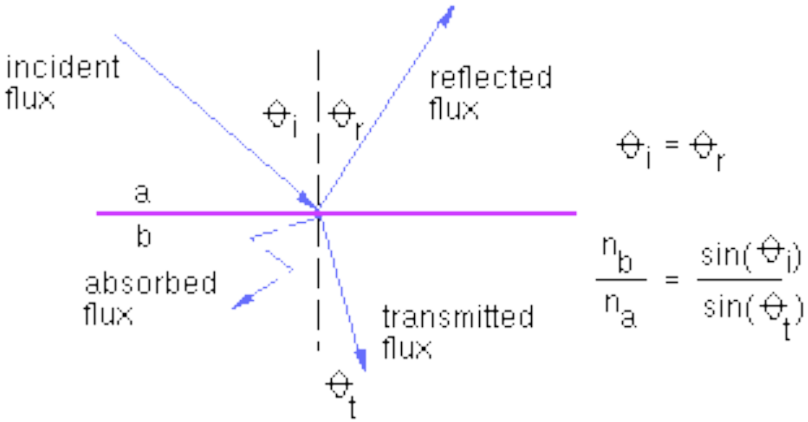
$$\mathbf{v} = -2(N \cdot \mathbf{c})N + \mathbf{c}$$

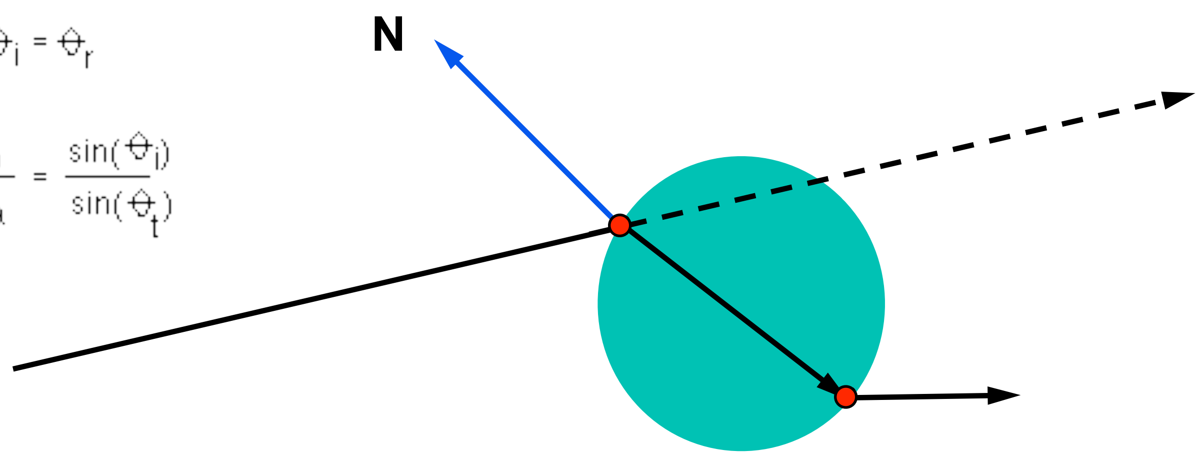


Refracted ray

Raytrace the refracted ray

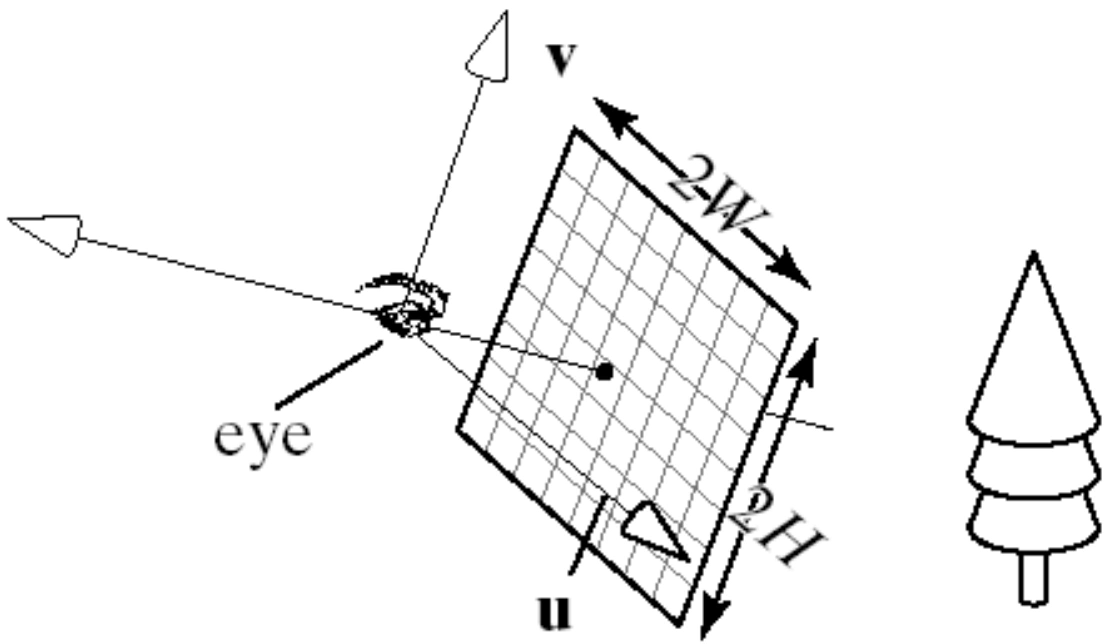
Snell's law





Add all together

- color(r,c) = color_shadow_ray + K_{re}*$color_{re}$ + K_{ra}*$color_{ra}$



Efficiency issues

Computationally expensive

- avoid intersection calculations
  - *Voxel grids*
  - *BSP trees*
  - *Octrees*
  - *Bounding volume trees*
- optimize intersection calculations
  - *try recent hit first*
  - *reuse info from numerical methods*

Summary: Raytracing

Recursive algorithm

Function Main
for each pixel (c,r) on screen
determine ray $r_{c,r}$ from eye through pixel
ray.setDepth(1)
color(c,r) = raytrace($r_{c,r}$)

end for

end

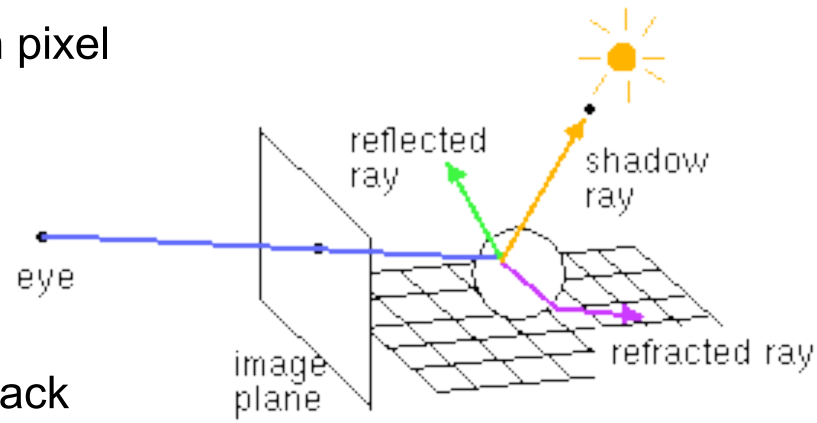
function raytrace(r)
if (ray.depth() > MAX_DEPTH) return black
P = closest intersection of ray with all objects
if(no intersection) return backgroundColor
clocal = Sum(shadowRays(P,Lighti))

c_{re} = raytrace(r_{re})

c_{ra} = raytrace(r_{ra})

return c = clocal+k_{re}*c_{re}+k_{ra}*c_{ra}

end



Advanced concepts

Participating media

Transculency

Sub-surface scattering (e.g. Human skin)

Photon mapping

Raytracing summary

View dependent

Computationally expensive

Good for refraction and reflection effects