

Z-buffer algorithm

for each polygon in model

project vertices of polygon onto viewing plane

for each pixel in ~~Assignment Project Exam Help~~ *calculate pixel colour*

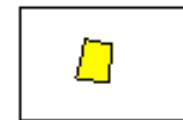
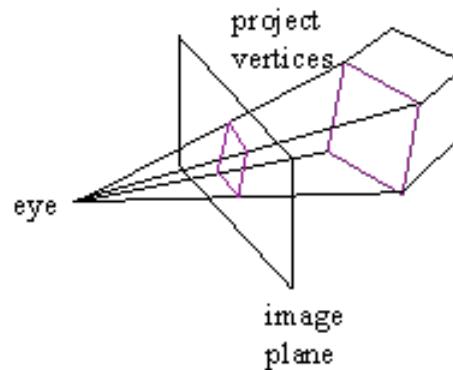
calculate pixel z-value <https://powcoder.com>

compare pixel z-value to value stored for pixel in z-buffer [Add WeChat powcoder](#)

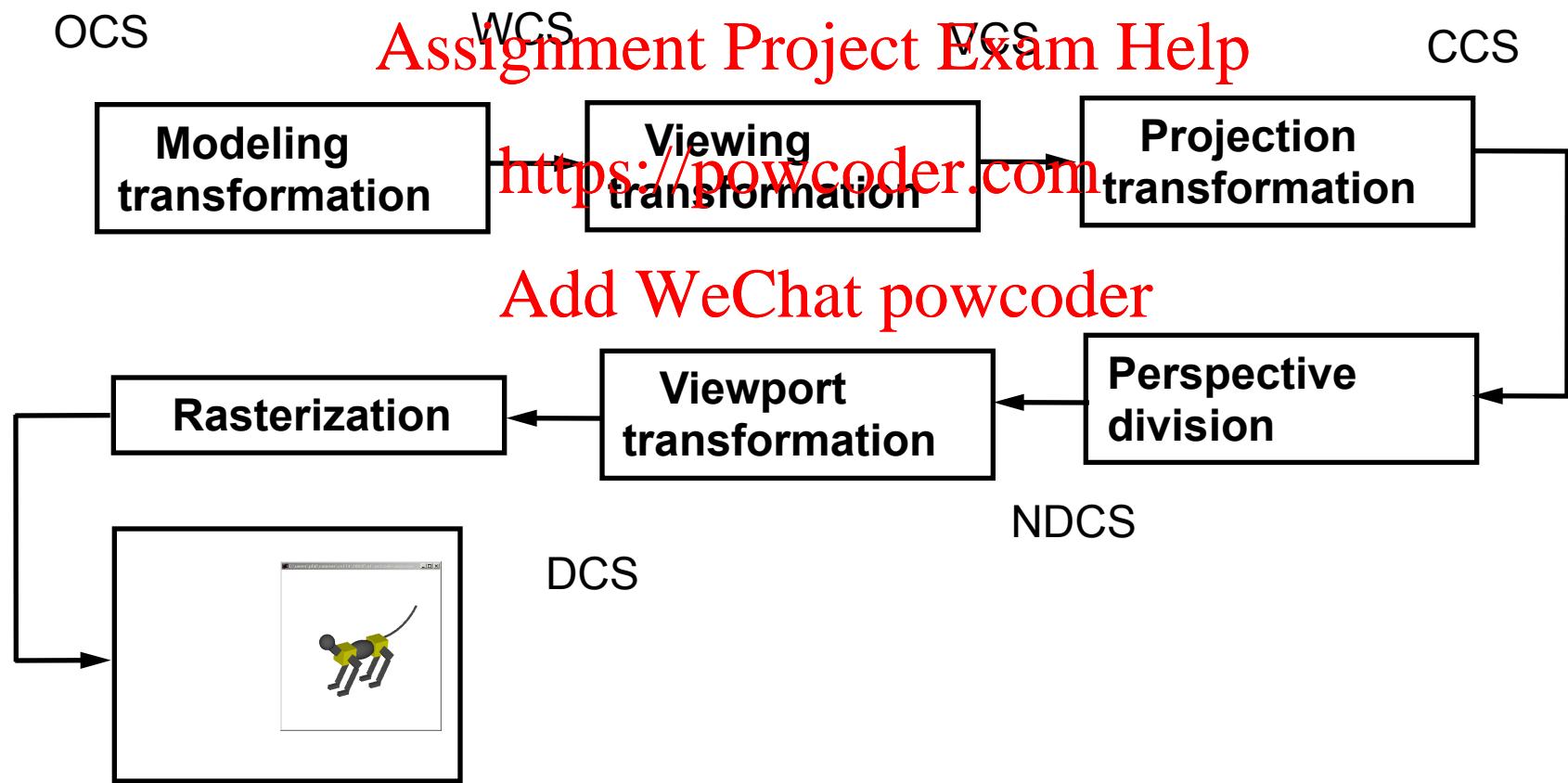
if pixel is closer, draw it in frame-buffer and z-buffer

end

end



Z-buffer Graphics Pipeline



Global Illumination

Rendered by
LuxRender

Assignment Project Exam Help

Copyright

<https://powcoder.com>

Simon Wendsche Add WeChat powcoder



Physics-based Rendering

Light Transport Equation (LTE)

$$L_o(p, \omega_0) = L_e(p, \omega_0) + \int_{\mathcal{S}^2} f(p, \omega_o, \omega_i) L_i(p, \omega_i) |\omega_i \cdot n| d\omega_i$$

<https://powcoder.com>

L_o : Outgoing radiance at p in direction ω_o

L_e : Emitted radiance at p in direction ω_o

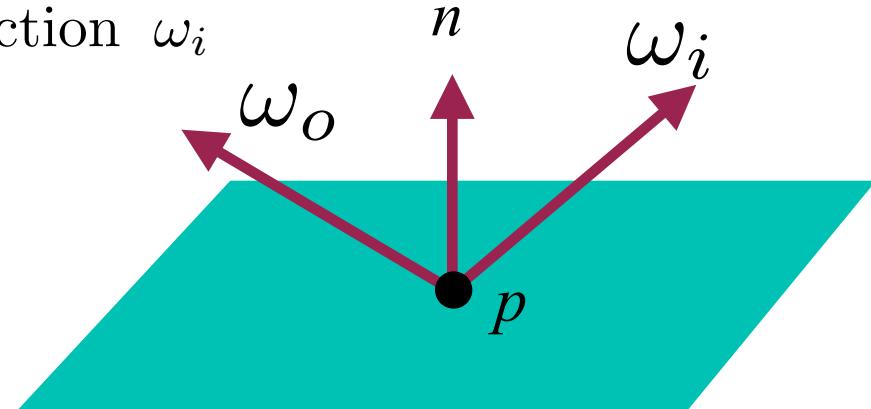
L_i : Incident radiance at p from direction ω_i

n : Unit normal at p

f : Bidirectional scattering function

\mathcal{S}^2 : The unit sphere centered at p

- Integral over directions



BSDF

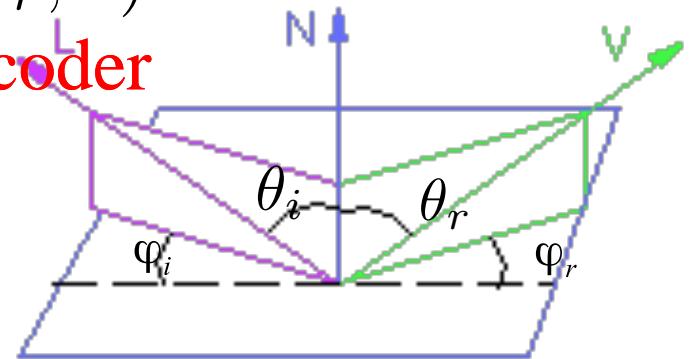
Bidirectional Scattering Distribution Function

Assignment Project Exam Help

- BRDF: Bidirectional Reflectance Distribution Function

$$f_r(p, \omega_o, \omega_i, \lambda) = f_r(p, \phi_i, \theta_i, \phi_r, \theta_r, \lambda)$$

Add WeChat powcoder



- BTDF: Bidirectional Transmittance Distribution function

Examples of a few simple diffuse BRDFs (wikipedia)

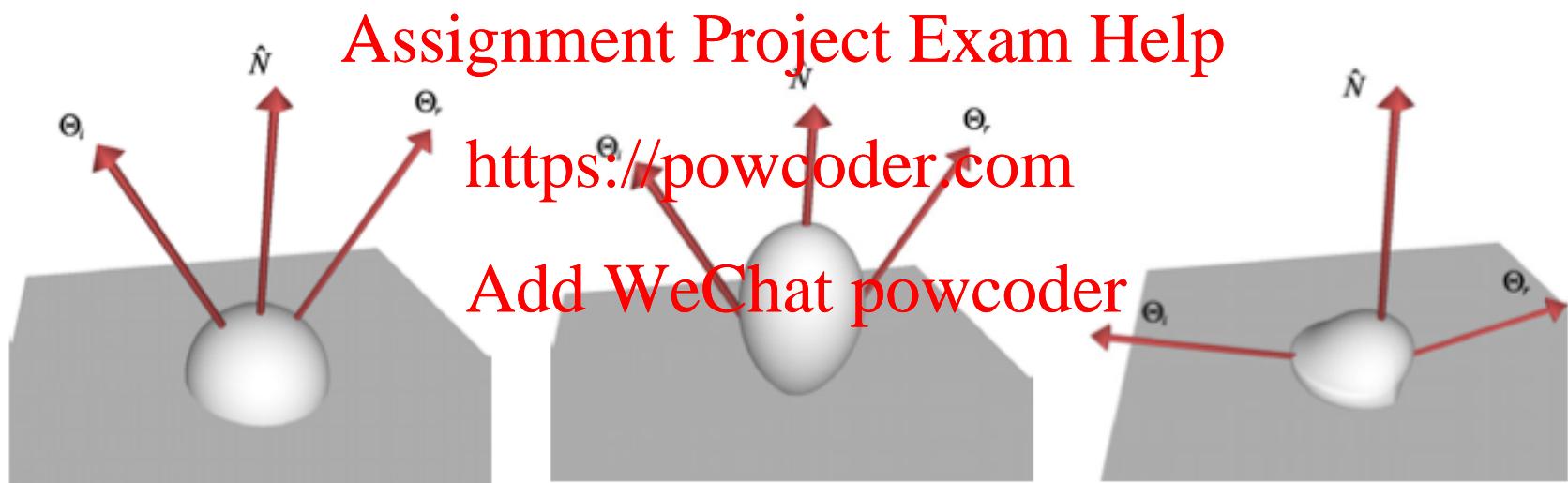
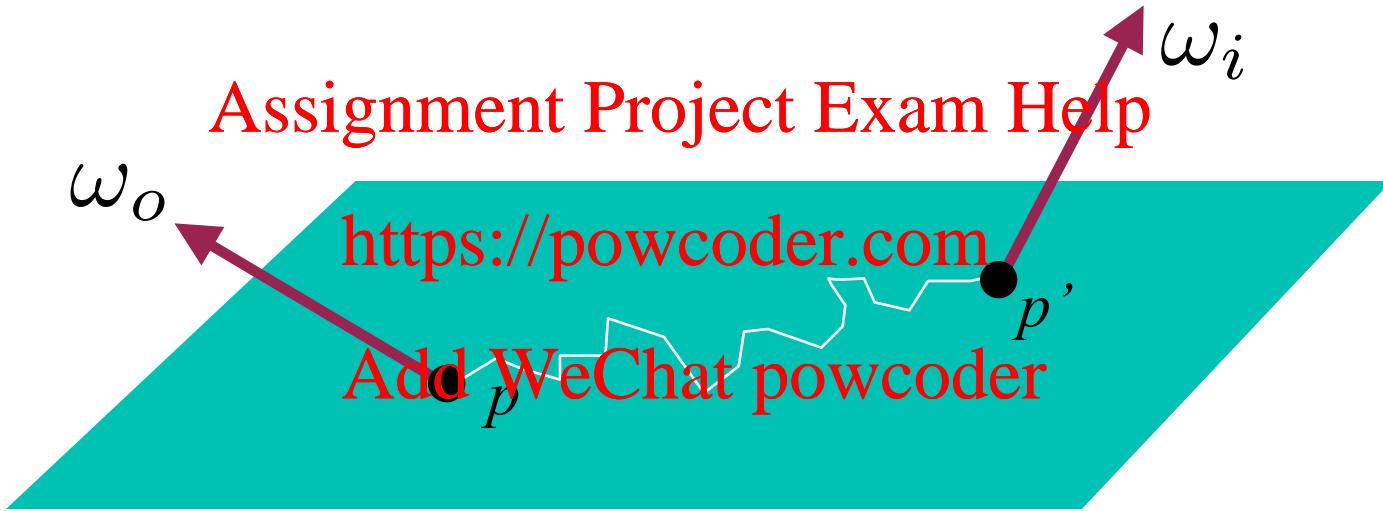


Fig. 2. Diffuse BRDF models: Lambert, Minnaert and Oren-Nayar.

BSSRF

- Bidirectional scattering-surface distribution function S()



- We now need to integrate over surface area as well

$$L(p, \omega_0) = \int_A \int_{S^2} S(p', \omega_i, p, \omega_o) \cos \theta_i d\omega_i dA$$

Rendering Equation

Simple but difficult to solve

- High dimensionality
 - b is a function of 6 parameters,
 - ρ is a function of 6 parameters
 - and we have not even used a variable for color (wavelength of light)

Solutions based on

- simplifying assumptions
- stochastic sampling
- FEM

One solution

Ray-tracing

Assignment Project Exam Help

Open source ray tracers

<https://powcoder.com>

- Povray: www.povray.org
- LuxRender, <http://www.luxrender.net>
- MegaPov

Add WeChat powcoder

Raytracing

Offline high quality rendering

- Real-time versions for limited complexity scenes

Tuned for specular and transparent objects

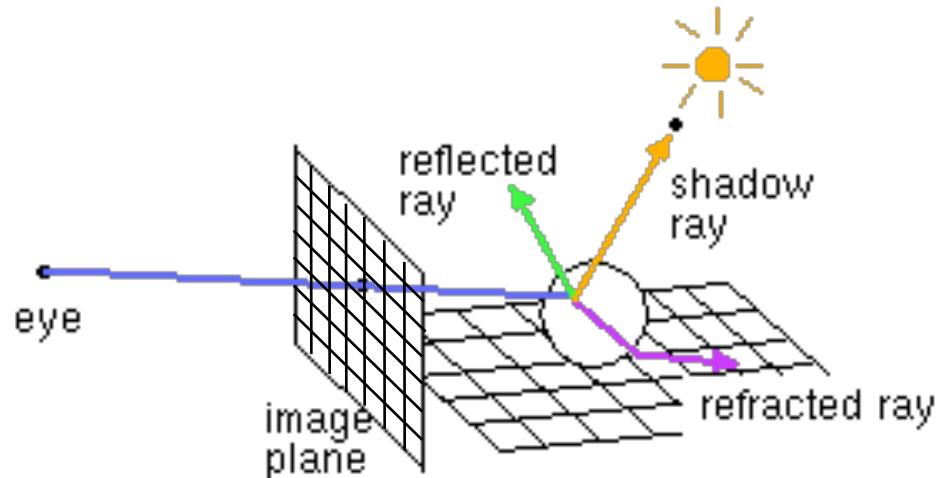
- Partly physics, optics

<https://powcoder.com>

Main idea:

Add WeChat powcoder

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



Forward and Backward methods

***Forward: from light sources
to eye***

Assignment Project Exam Help

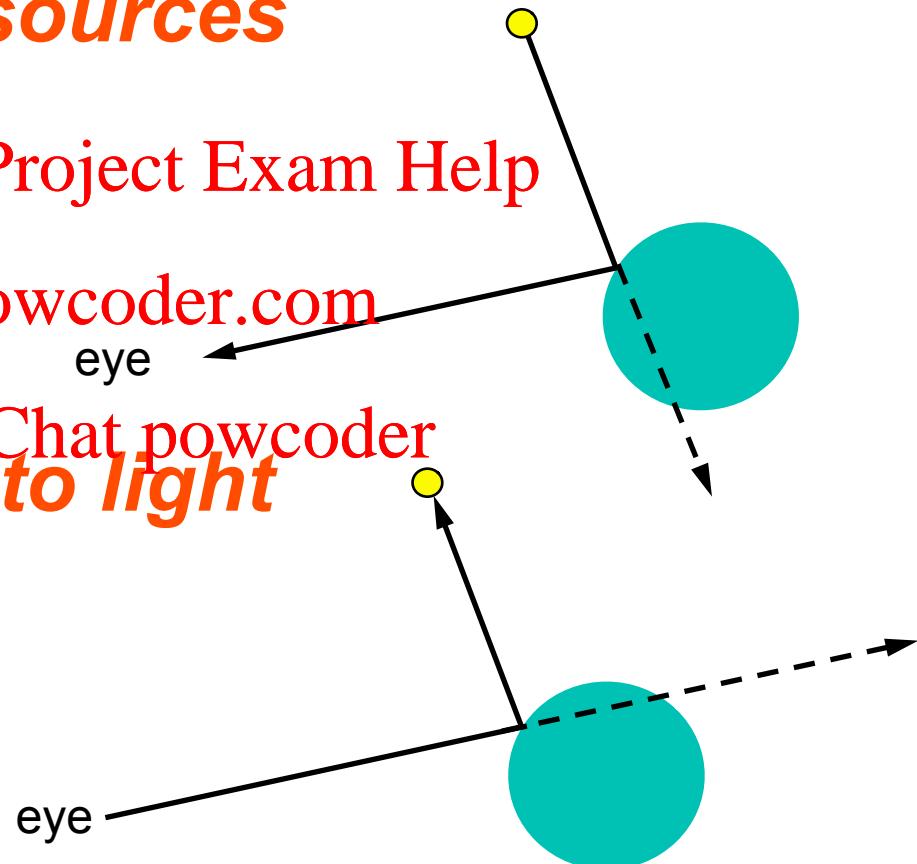
<https://powcoder.com>

eye

***Backward: from eye to light
sources***

Add WeChat powcoder

eye



Our version

There are different variants of the backwards RT method that make different assumptions and simplifications

Assignment Project Exam Help

<https://powcoder.com>

We will study a specific version that is

- Efficient
- Simple to implement
- Educational

Original Backwards Raytracing

Turner Whitted 1979

Assumptions

- Reflection and Refraction only on ideal direction cones
- Lambertian diffuse surfaces (no custom BRDFs)
- Diffuse objects do not receive reflected light

Assignment Project Exam Help

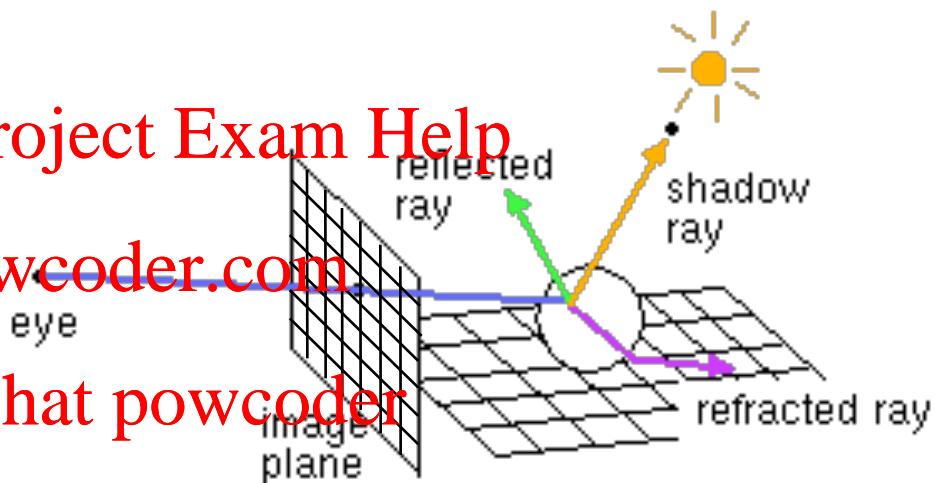
<https://powcoder.com>

Add WeChat powcoder

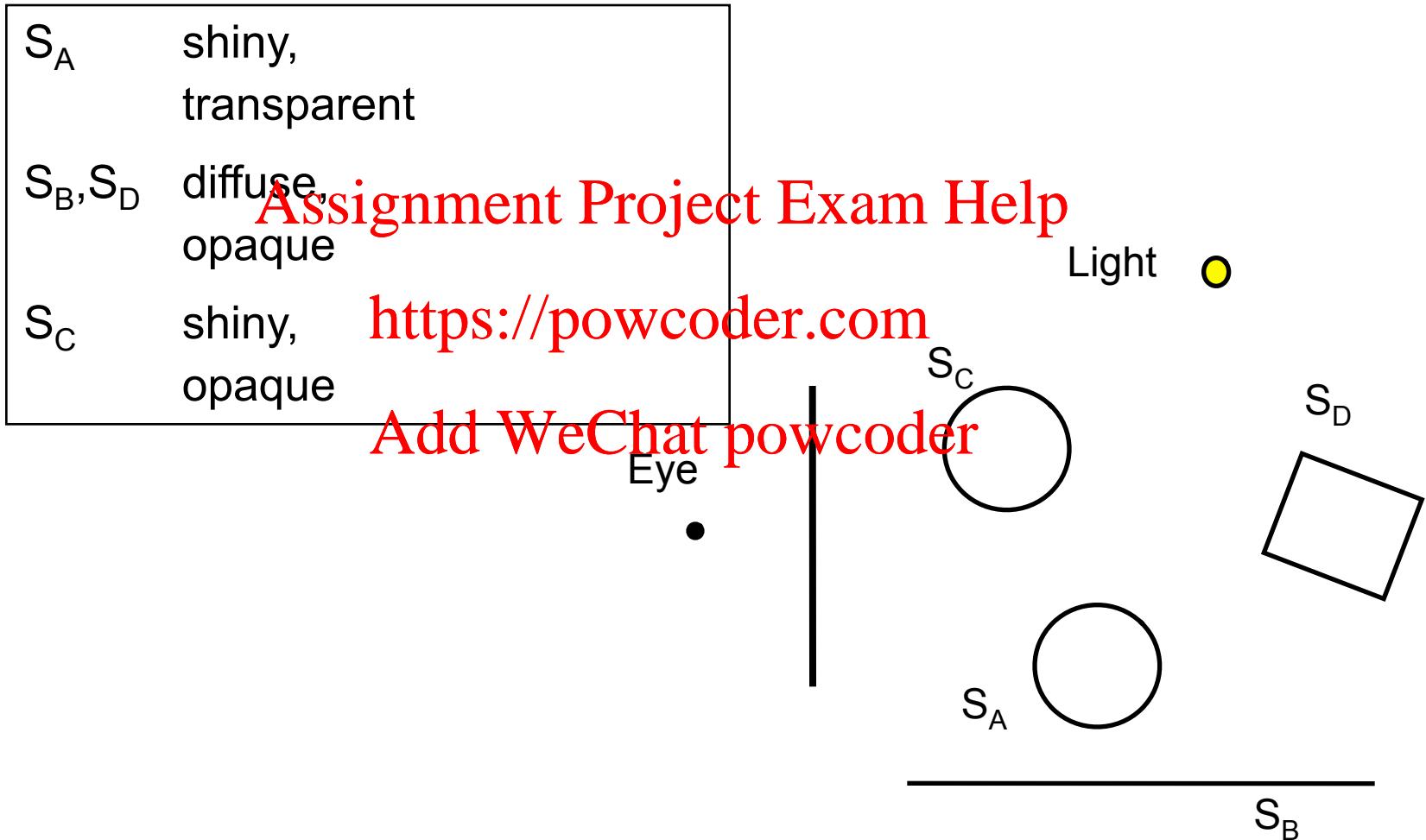
for each pixel on screen

- determine ray from eye through pixel
- find closest intersection of ray with an object
- compute direct illumination (shadow ray)
- trace reflected and refracted ray, recursively
- calculate pixel colour, draw pixel

end



Scene



Three sources of light

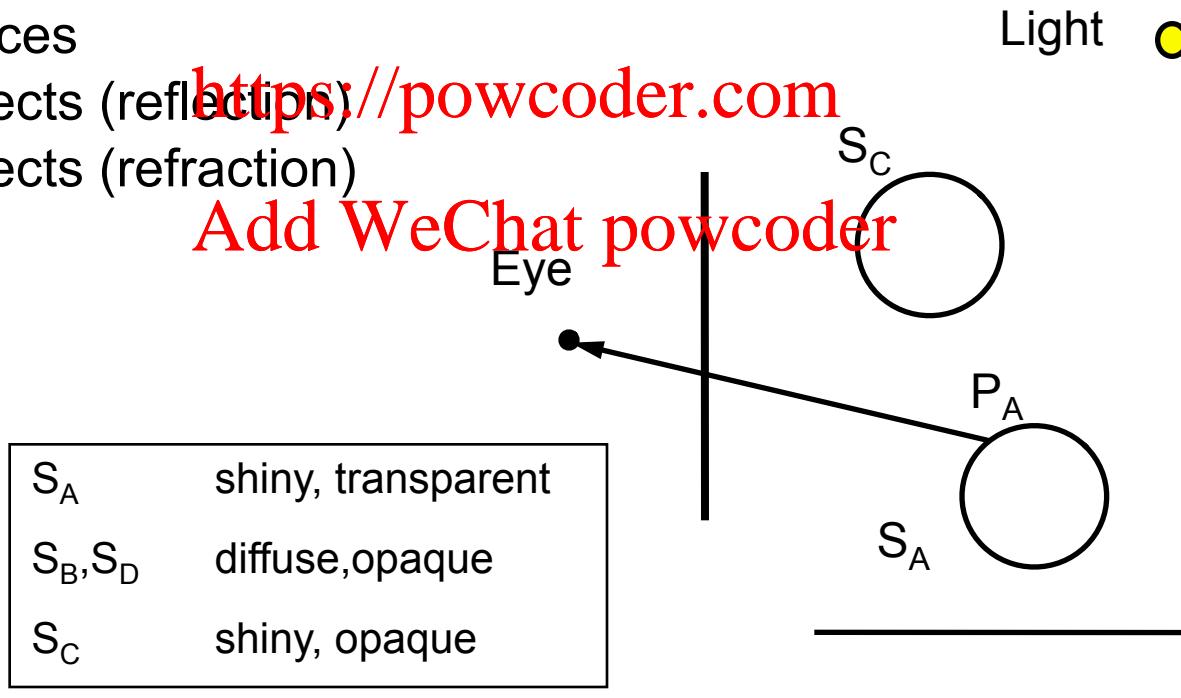
The light that point P_A emits to the eye comes from:

Assignment Project Exam Help

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

Add WeChat powcoder

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



Directly from a light source

Local illumination model:

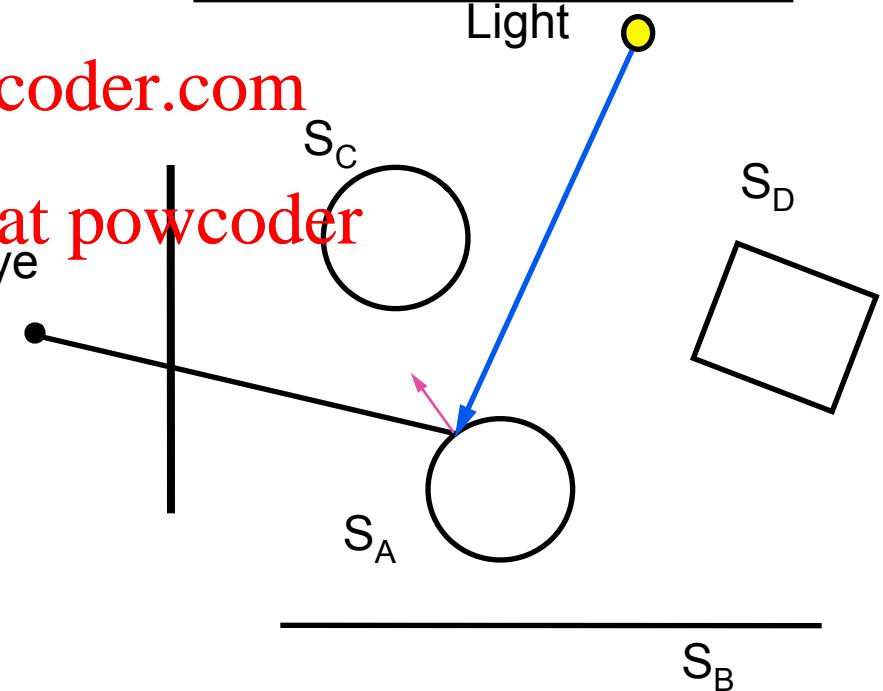
$$I = I_a + I_{diff} + I_{spec}$$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

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



Reflection

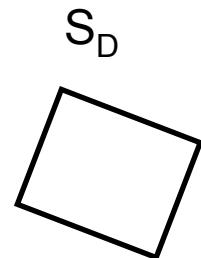
What is the color that is reflected to P_A and that P_A reflects back to the eye?

*What ever comes from the direction
that eventually reflects towards the
eye.*

Assignment Project Exam Help
<https://powcoder.com>

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

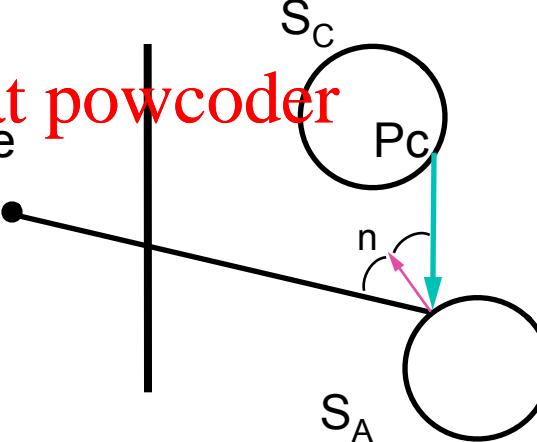
Light



Add WeChat powcoder

Eye

•



S_B

Reflection

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

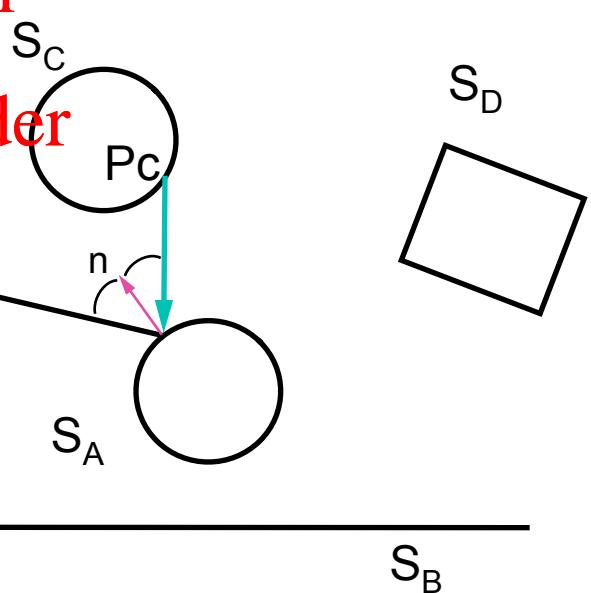
*What ever comes from the direction
that eventually reflects towards the
eye.*

<https://powcoder.com>

That direction gets light from point P_C
What is the color of P_C ?

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

Light



Reflection

What is the color of P_c ?

Just like P_A : raytrace P_c i.e compute

the three contributions from

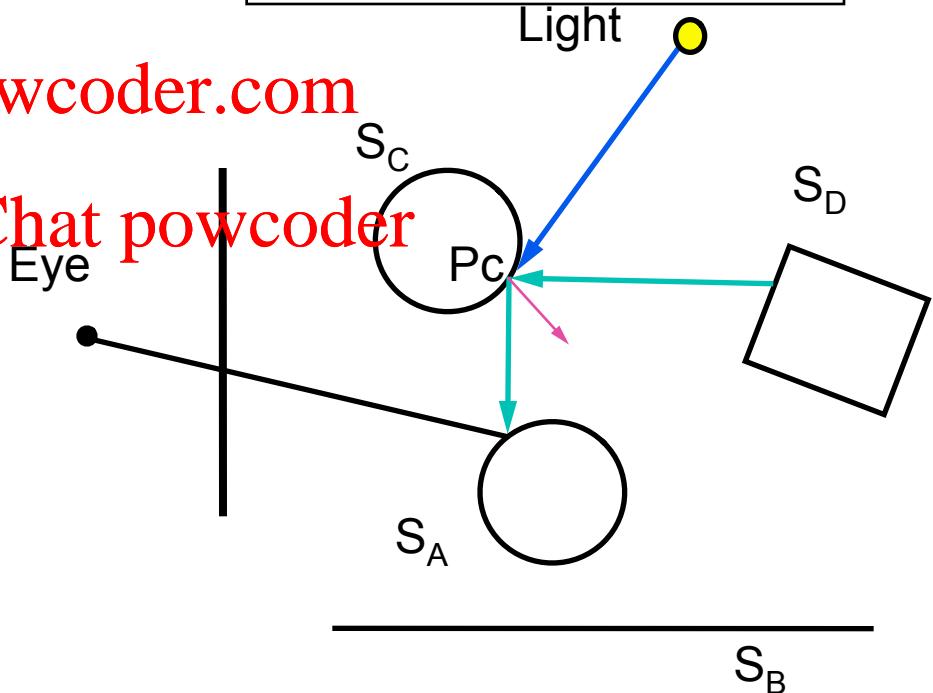
Assignment Project Exam Help

- Light sources
- Reflection
- Refraction

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

<https://powcoder.com>

Add WeChat powcoder



Refraction

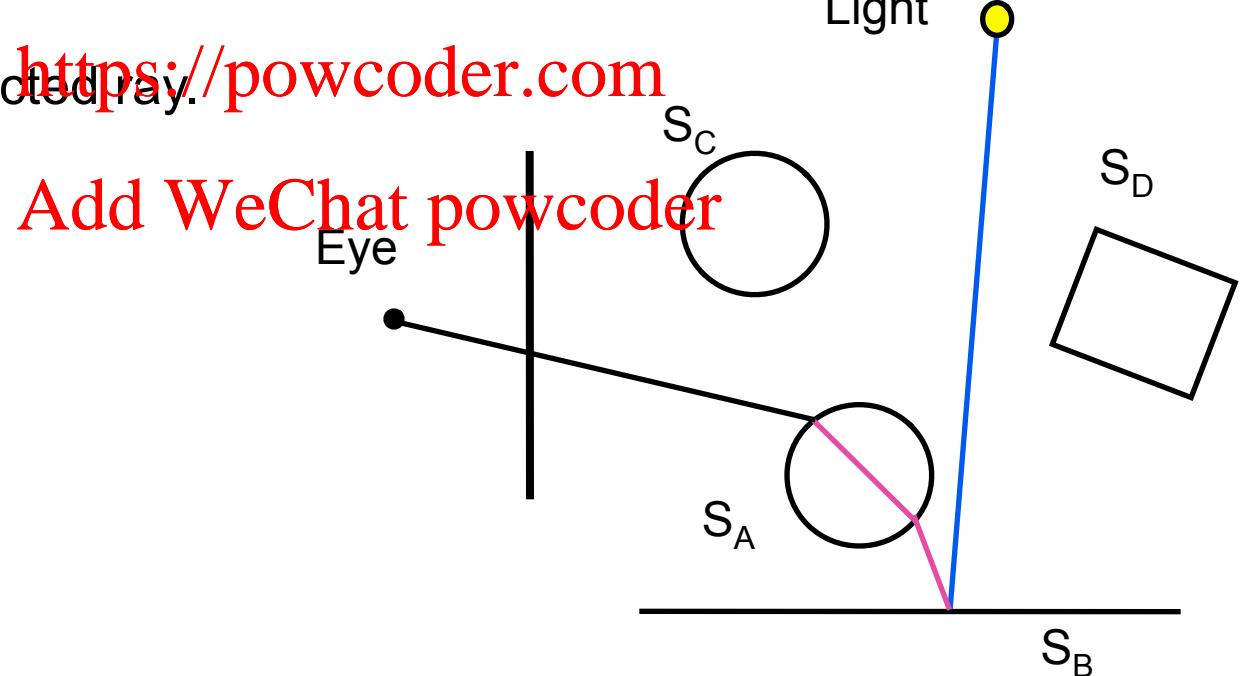
Transparent materials

How do we compute the refracted
contribution?

We 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 have we ignored ?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

What have we ignored ?

Diffuse objects do not receive light from other objects.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Three sources of light together

The color that the pixel is assigned comes from:

light sources

[Assignment Project Exam Help](#)

other objects (reflection)

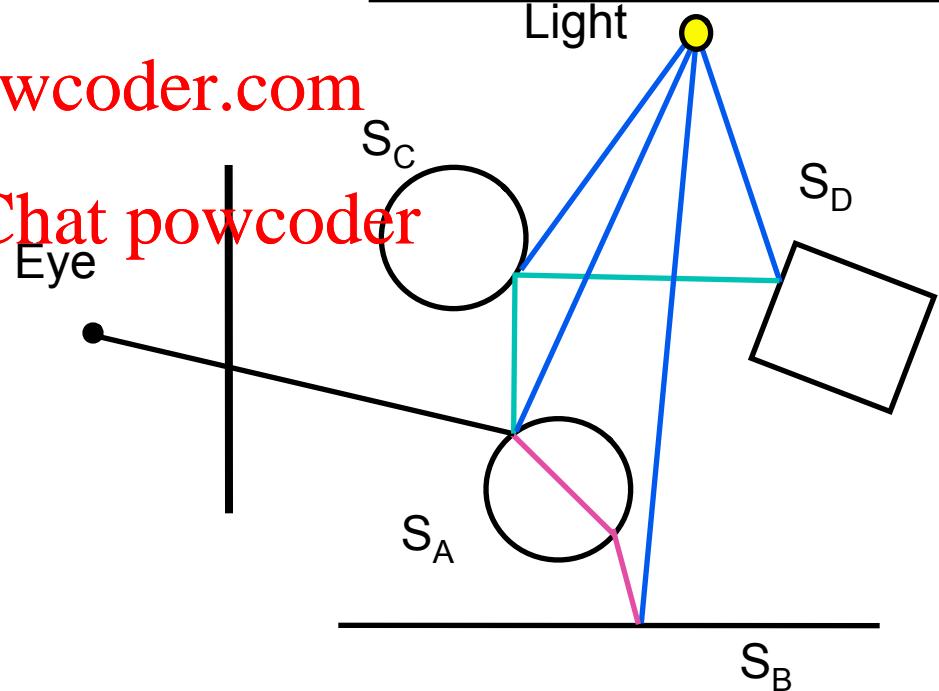
other objects (refraction)

<https://powcoder.com>

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

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

Add WeChat powcoder



Backwards Raytracing Algorithm

For each pixel construct a ray: eye \rightarrow 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
```

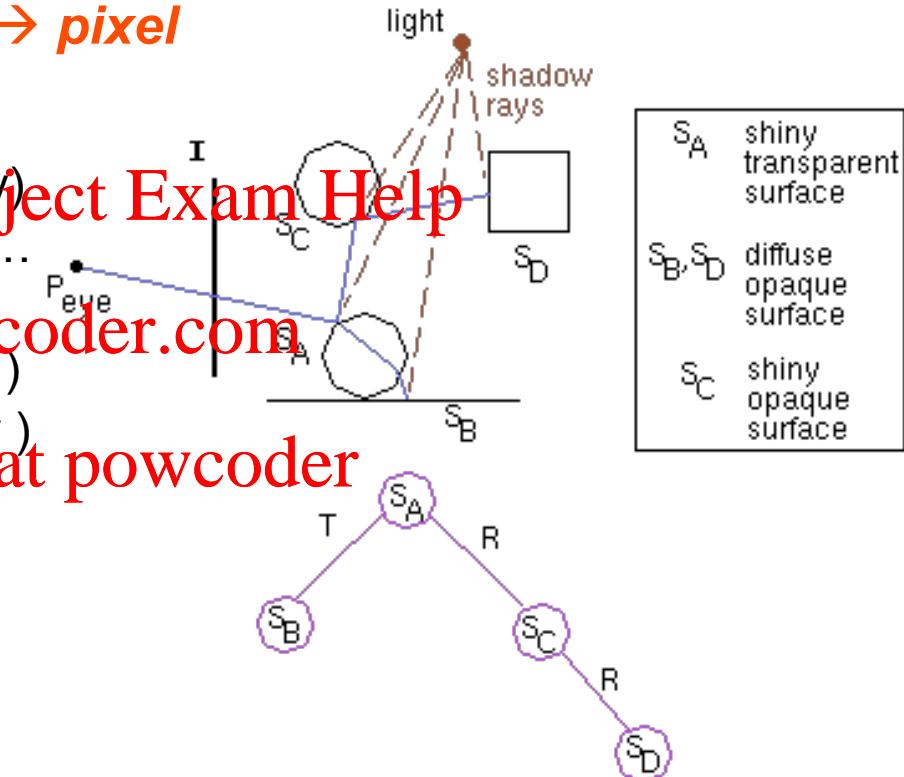
```
+ k_ra *color_refract
```

```
return( color )
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



How many levels of recursion do we use?

The more the better.

Infinite reflections at the limit.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

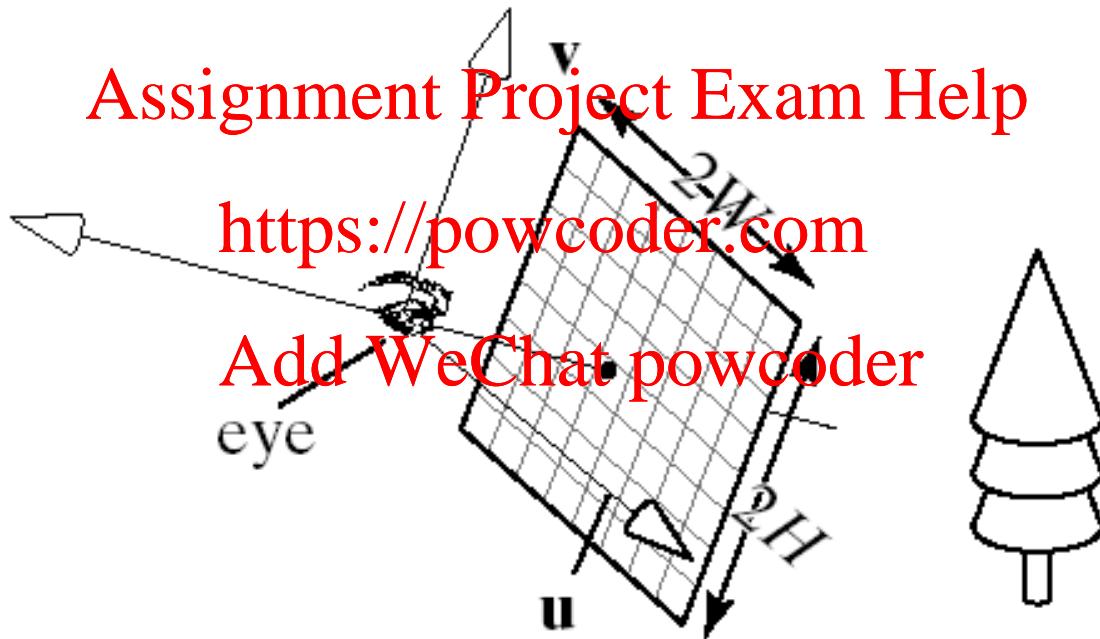


Stages of raytracing

We need to do the following:

- Set the camera and the image plane
[Assignment](#) [Project](#) [Exam](#) [Help](#)
- Compute a ray from the eye to every pixel and trace it in the scene <https://powcoder.com>
- Compute object-ray intersections
[Add](#) [WeChat](#) [powcoder](#)
- Cast shadow, reflected and refracted rays at each intersection

Setting up the camera



Set the viewport and resolution in Camera Coordinates

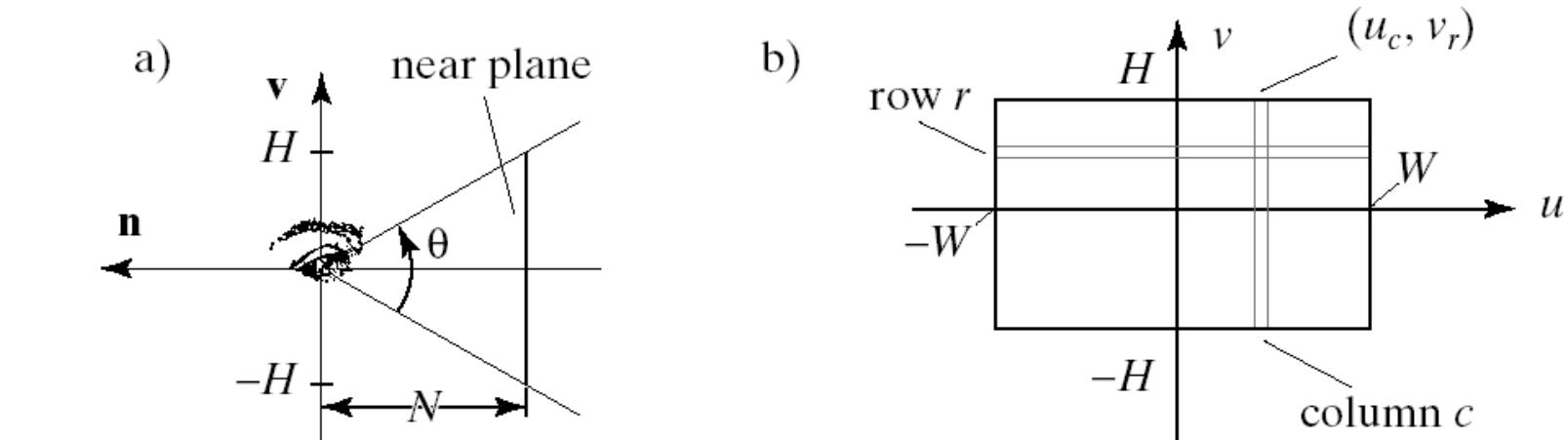
Image parameters

- Width $2W$, Height $2H$
- Number of pixels $n_{\text{Cols}} \times n_{\text{Rows}}$

Assignment Project Exam Help

Camera parameters (viewpoint.com)

- Camera coordinate system (eye , $\mathbf{u}, \mathbf{v}, \mathbf{n}$)
- Image plane at $\mathbf{n} = -N$



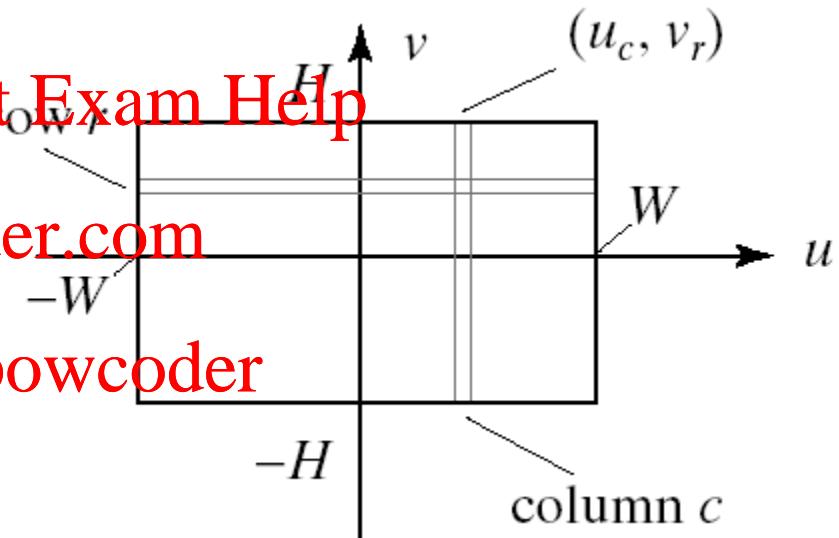
Pixel coordinates in camera coordinate system

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

Assignment Project Exam Help

<https://powcoder.com>

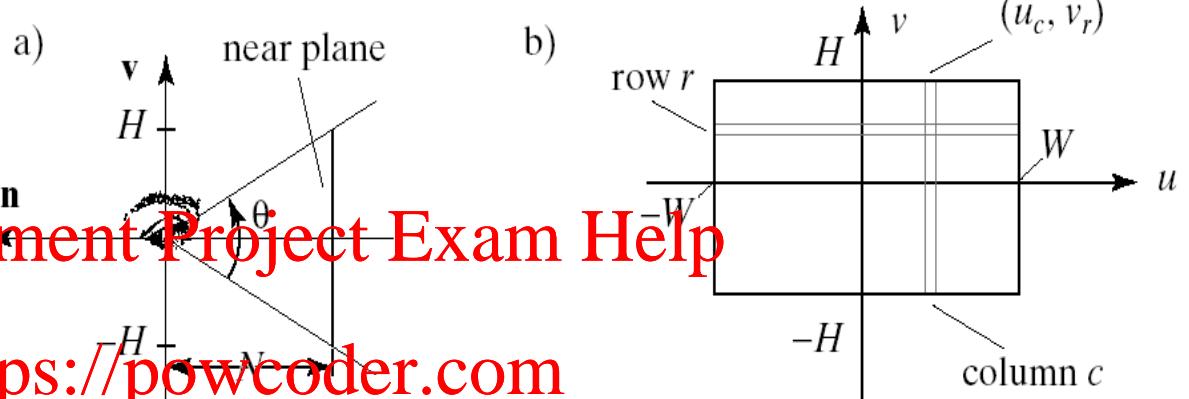
Add WeChat powcoder



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

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

Ray through pixel



Assignment Project Exam Help

<https://powcoder.com>

Lower left corner

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

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

Ray through pixel:

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

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

Ray-object intersections

Objects

- Spheres, cylinders, surfaces, etc
- Any object that we can mathematically compute intersections with lines

Assignment Project Exam Help

<https://powcoder.com>

For us

Add WeChat powcoder

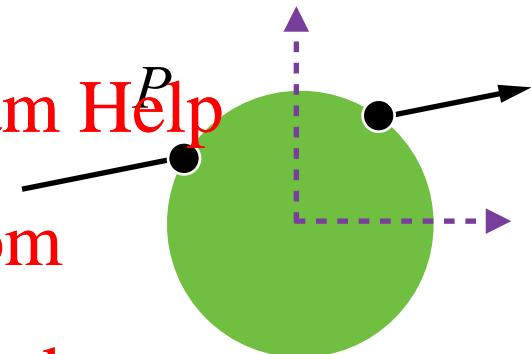
- Only affine transformed spheres

Ray - Canonical sphere intersection

Can I use homogeneous coordinates?

$$ray(t) = S + ct$$

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



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

$$|S + ct| - 1 = 0 \Rightarrow (S + ct)(S + ct) - 1 = 0 \Rightarrow$$

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

This is a quadratic equation in t

Solving a quadratic equation

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

Assignment Project Exam Help

$$t_h = \frac{-B \pm \sqrt{B^2 - AC}}{A}$$

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

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

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

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

Note!

Can I use homogeneous coordinates?

$$ray(t) = S + ct$$

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

Add WeChat powcoder

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

$$|S + ct| - 1 = 0 \Rightarrow (S + ct)(S + ct) - 1 = 0 \Rightarrow$$

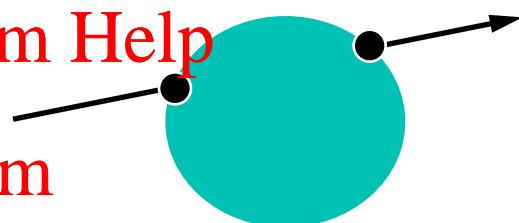
$$|\mathbf{c}|^2 t^2 + 2(S \cdot \mathbf{c})t + |S|^2 - 1 = 0$$

Note!

Can I use homogeneous coordinates?

$$ray(t) = S + ct$$

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



Add WeChat powcoder

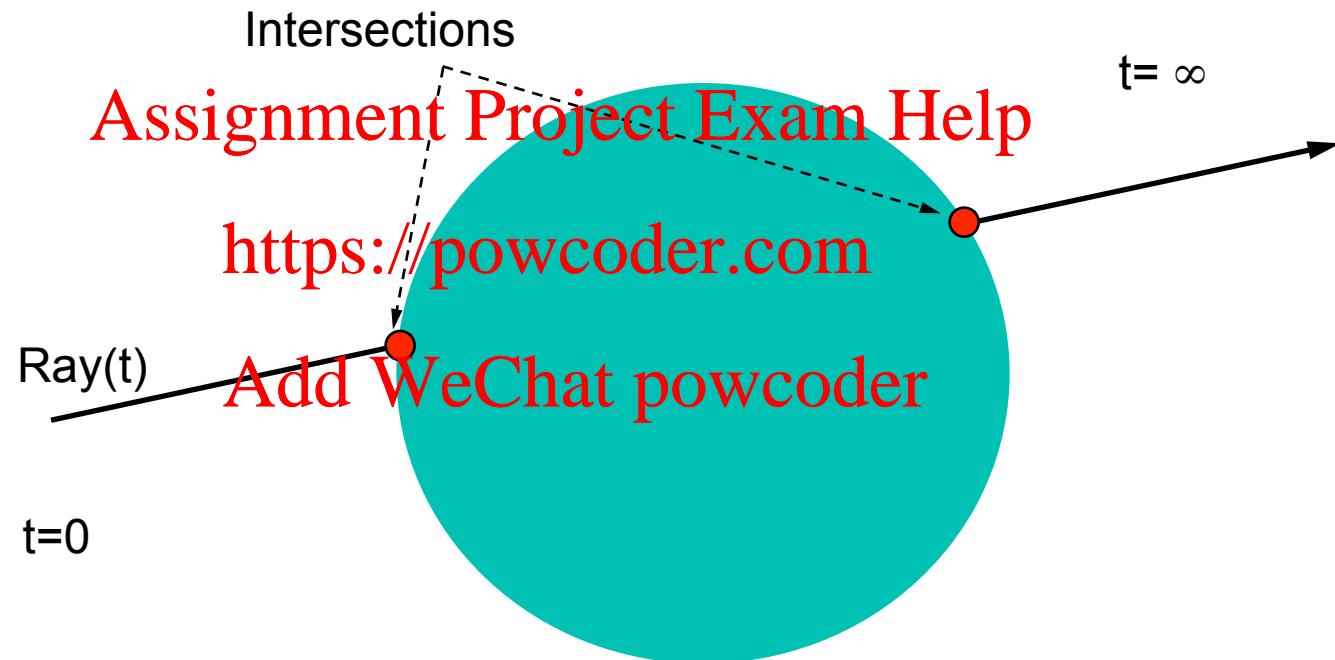
$$Sphere(ray(t)) = 0 \Rightarrow$$

$$|S + ct| - 1 = 0 \Rightarrow (S + ct)(S + ct) - 1 = 0 \Rightarrow$$

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

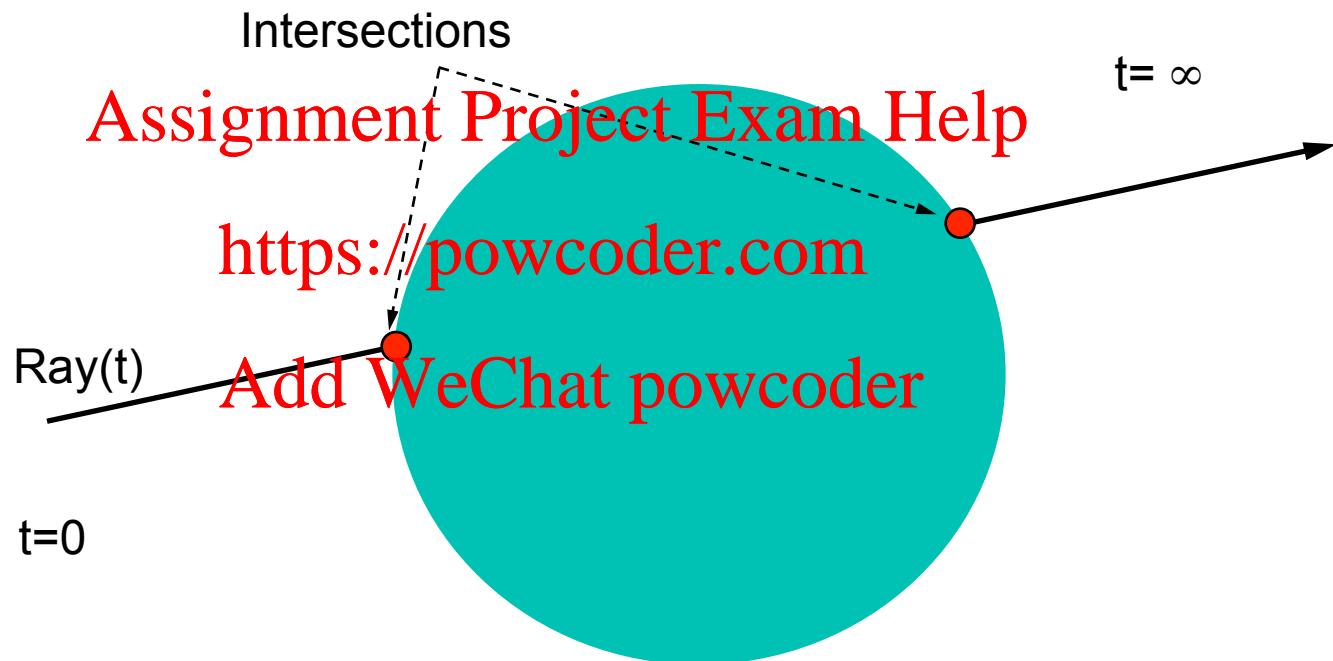
Must solve the equation in cartesian space

First intersection?



First intersection?

$t_1 < t_2$

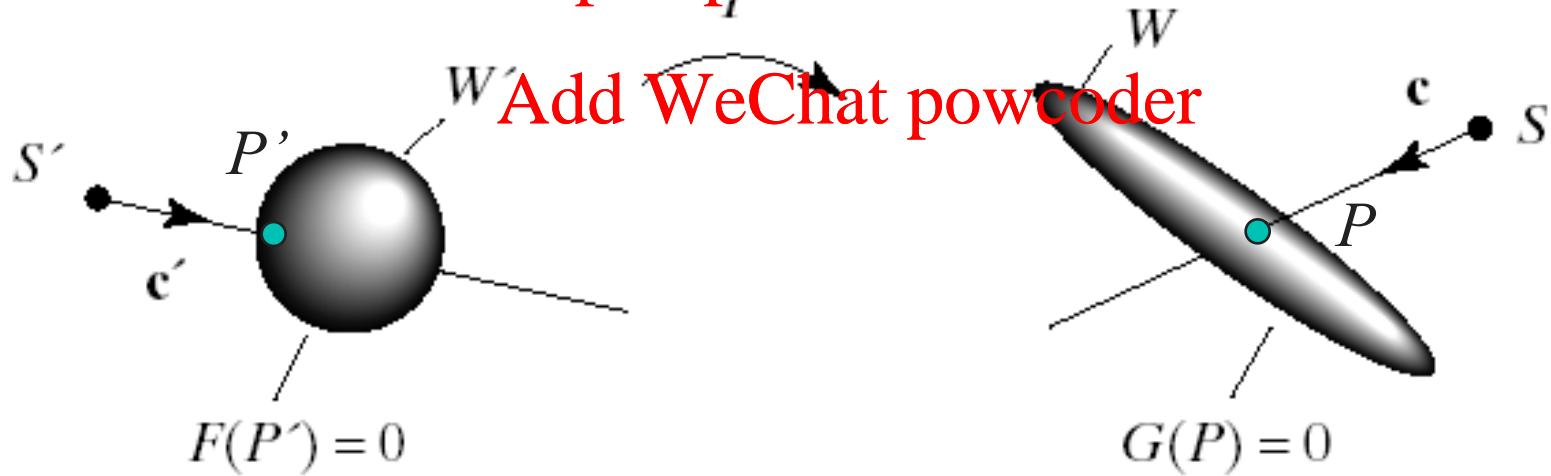


Transformed primitives?

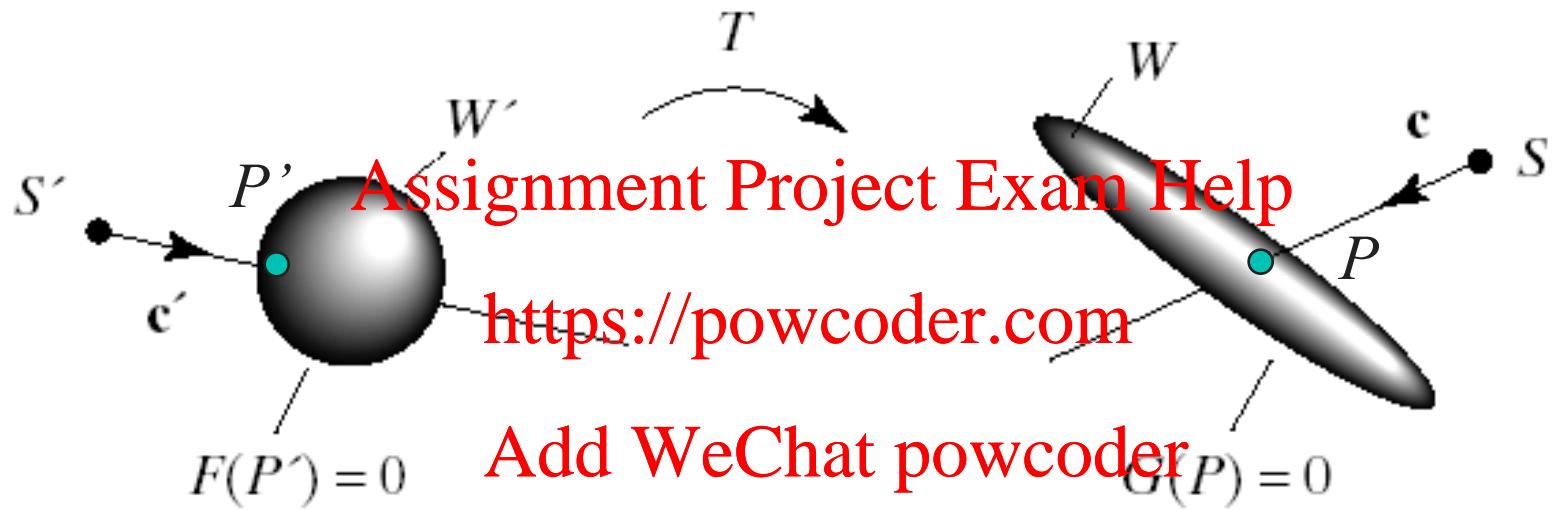
- That was a canonical sphere.
- Where does $S+ct$ hit the transformed sphere $G = T(F)$?

Assignment Project Exam Help

<https://powcoder.com>



Linear transformation

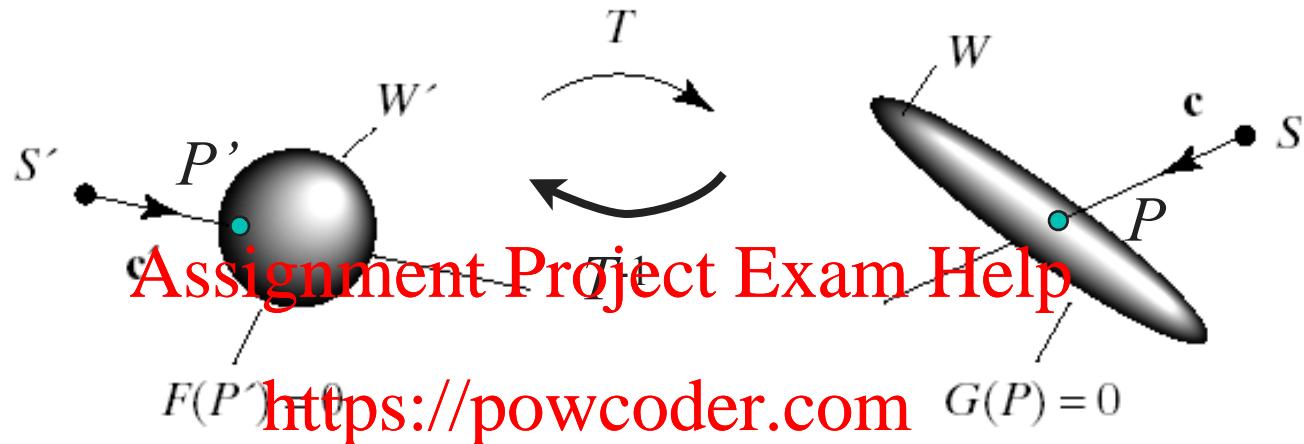


Implicit equation $G(P) = 0$.

Untransformed implicit equation $F(P') = 0$.

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

Linear transformation



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

Add WeChat powcoder

$$F(P') = F(T^{-1}(P)) = 0 \Rightarrow F(T^{-1}(P)) = 0$$

$$F(T^{-1}(S + ct)) = 0 \Rightarrow$$

$$F(T^{-1}(S) + T^{-1}(ct)) = 0$$

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

Final Intersection

Inverse transformed ray

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

Assignment Project Exam Help
<https://powcoder.com>

- Drop 1 and 0 to get $S' + c't$

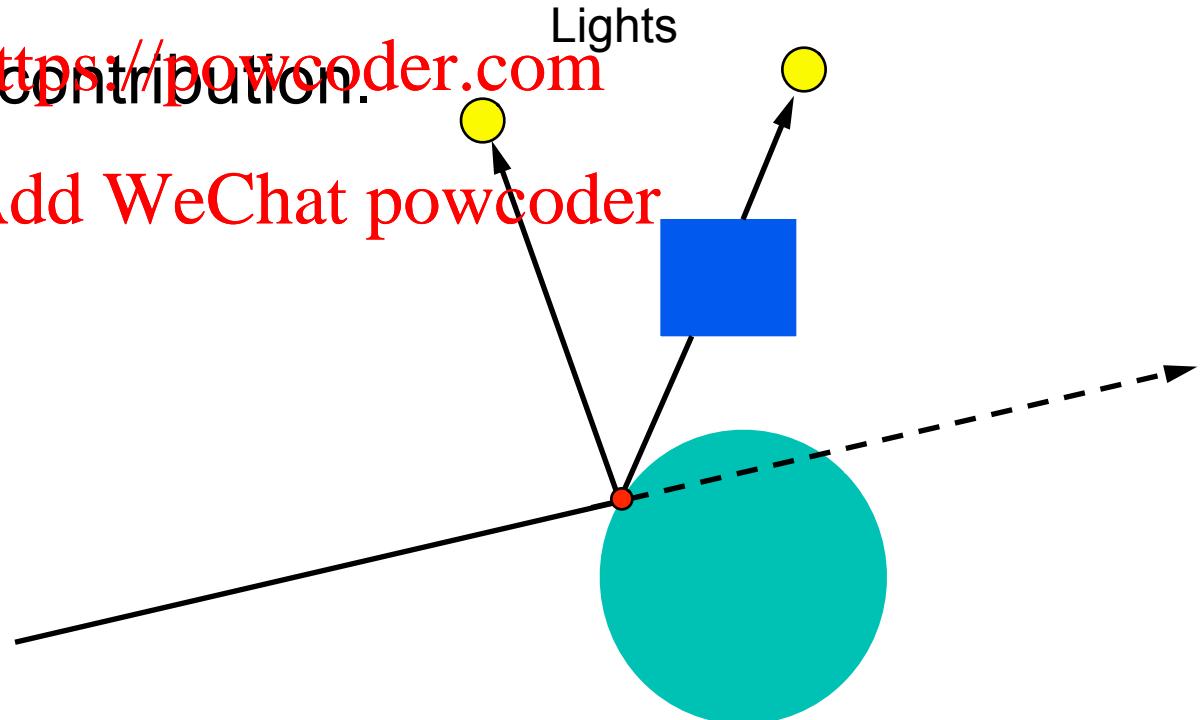
Add WeChat powcoder

So ..for each object

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

Shadow ray

- For each light intersect shadow ray with all objects.
- If no intersection is found apply local illumination
[Assignment](#) [Project](#) [Exam](#) [Help](#)
- If in shadow no <https://powcoder.com> contribution.



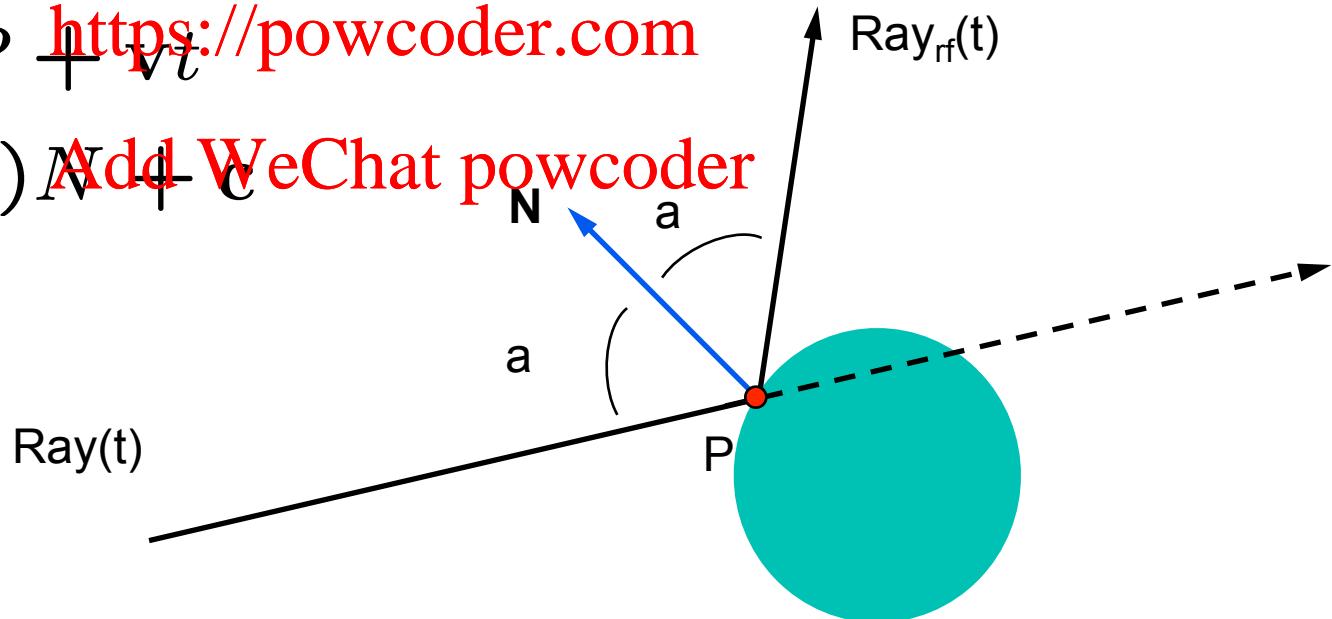
Reflected ray

Raytrace the reflected ray

$$Ray(t) = A + \text{Assignment Project Exam Help} ct$$

$$Ray_{rf}(t) = P + \text{https://powcoder.com} vt$$

$$\mathbf{v} = -2(\mathbf{N} \cdot \mathbf{c})\mathbf{N} + \text{Add WeChat powcoder} \mathbf{c}$$



Refracted ray

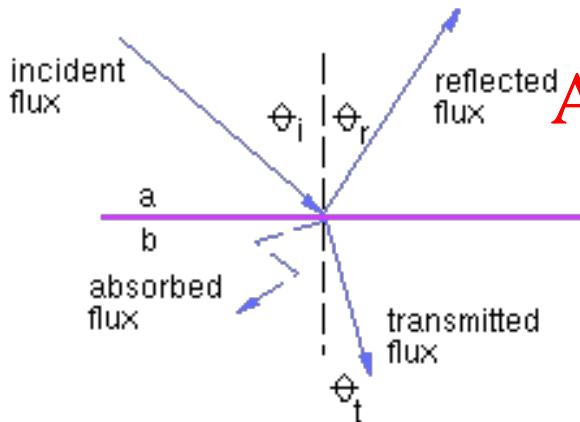
Raytrace the refracted ray

Snell's law

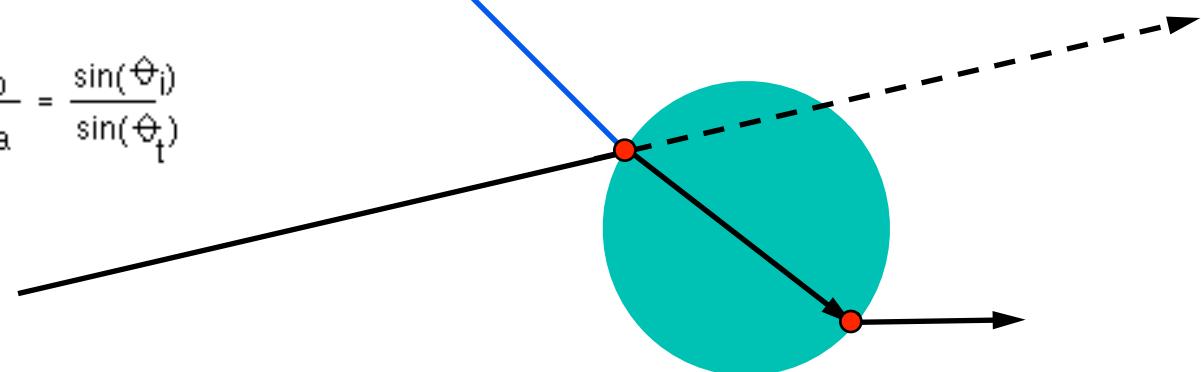
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



$$\frac{n_b}{n_a} = \frac{\sin(\theta_i)}{\sin(\theta_t)}$$



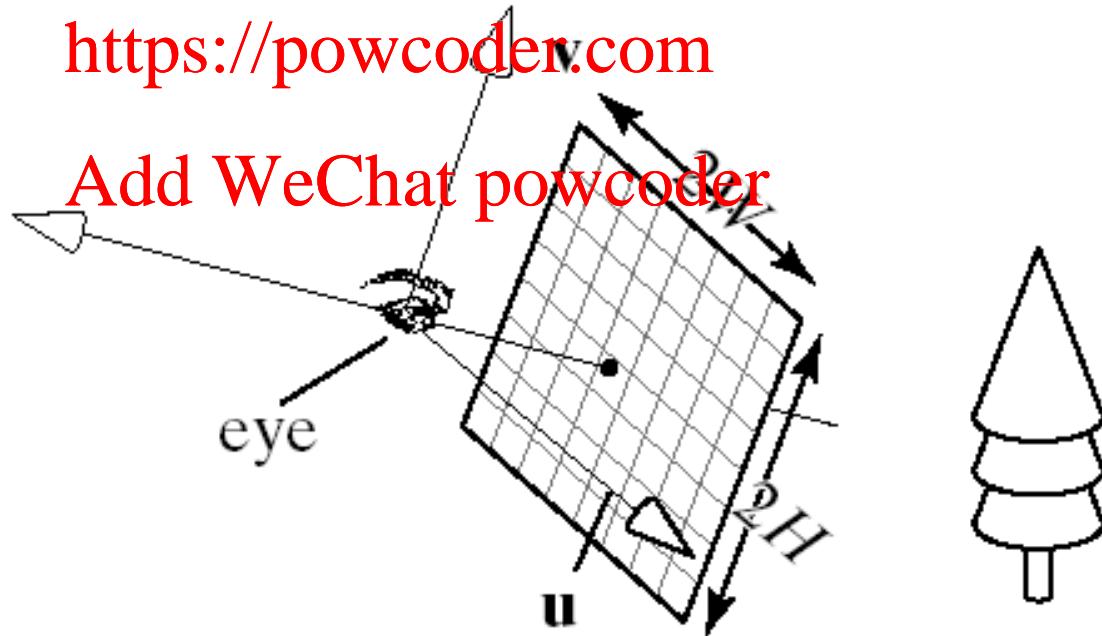
Add all together

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

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Efficiency issues

Computationally expensive

- avoid intersection calculations
 - *Voxel grids* Assignment Project Exam Help
 - *BSP trees* <https://powcoder.com>
 - *Octrees* Add WeChat powcoder
 - *Bounding volume trees*
- optimize intersection calculations
 - *try recent hit first*
 - *reuse info from numerical methods*

Summary: Raytracing

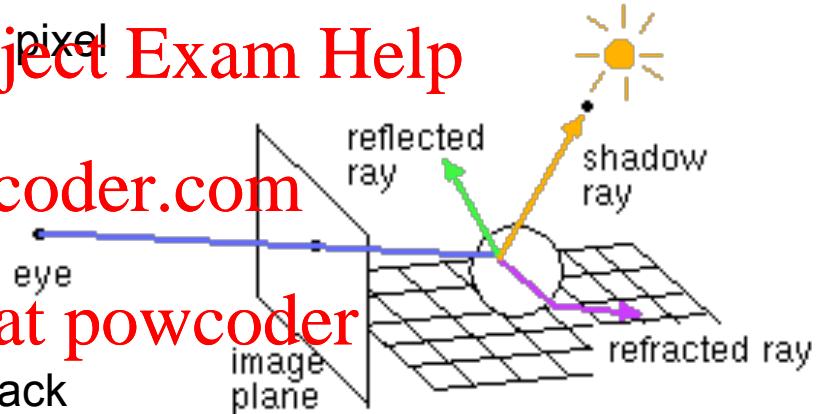
Recursive algorithm

```
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))
    cre = raytrace(rre)
    cra = raytrace(rra)
    return (clocal+kre*cre+kra*cra)
end
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Advanced concepts

Participating media

Transculency Assignment Project Exam Help

Sub-surface scattering (e.g. Human skin)
<https://powcoder.com>

Photon mapping Add WeChat powcoder

Photon Mapping

(slides courtesy of Shawn Singh)

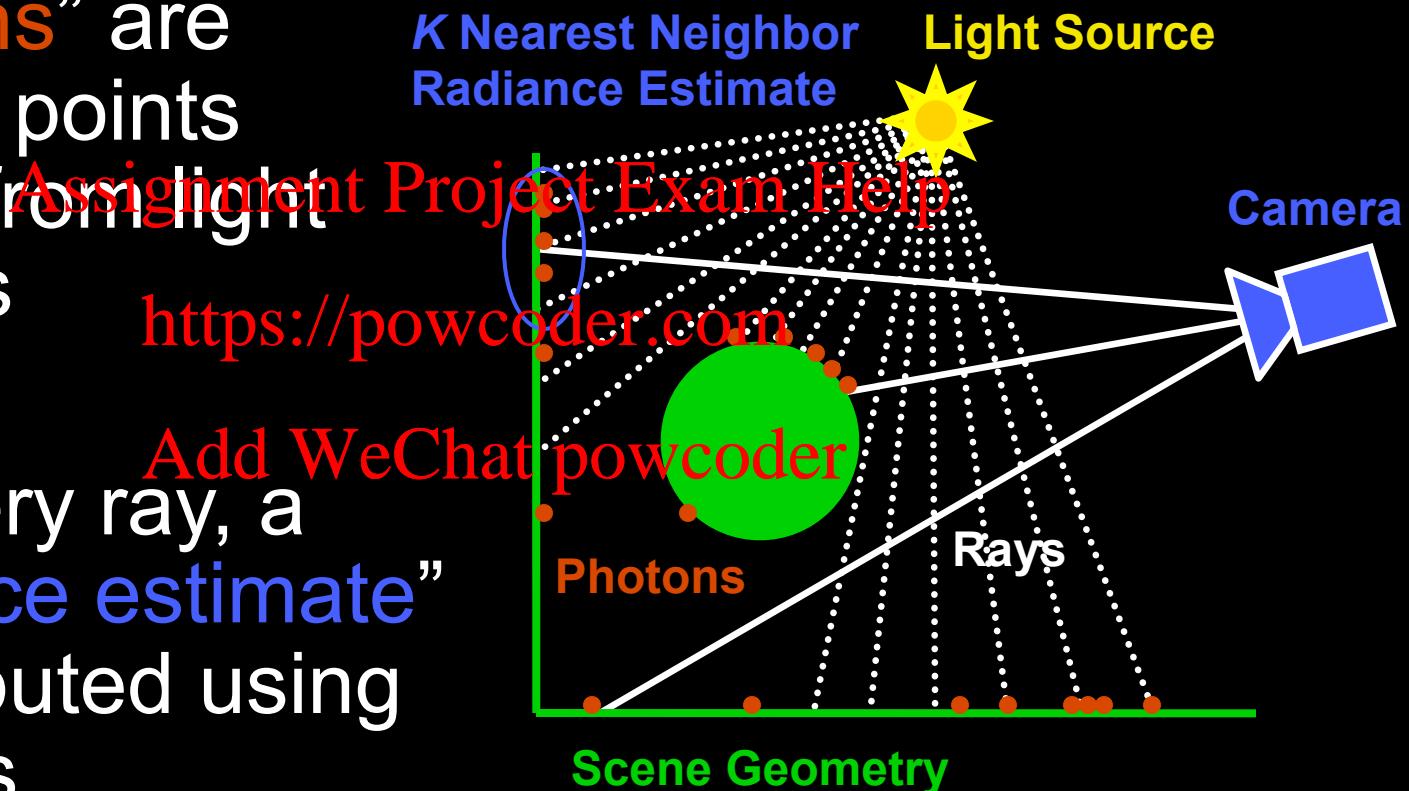
Create a sampling of the illumination

- For scene [Assignment](#) [Project](#) [Exam](#) [Help](#) pre-computed
- View independent
<https://powcoder.com>

Add WeChat powcoder

Photon Mapping

- “Photons” are sample points traced from light sources
- For every ray, a “radiance estimate” is computed using photons



Resources

- Real-time Ray Tracing
Our implementation based on [Wald 2004]
- GPU Photon Mapping
~~Assignment Project Exam Help~~
[Purcell et al 2003], [Larsen and Christensen 2004]
<https://powcoder.com>
- Specialized Hardware
[Steinhurst 2007], [Singh 2007]
Add WeChat powcoder
- Improve performance of each k-nearest neighbor query
[Ma and McCool 2002], [Gunther et al 2004], [Wald et al. 2004]
- Improve coherence of queries
[Steinhurst et al. 2005], [Havran et al. 2005]
- Reverse photon mapping
[Havran et al 2005]

A Useful Analogy

Ray Tracing

- Each ray performs a search through scene geometry

Assignment Project Exam Help
<https://powcoder.com>

Photon Mapping

- Each query point performs a search through photon map

Add WeChat powcoder
Each search finds many *<photon, camera-point>* shader operations

- Each search finds many ray-triangle intersection operations

- SIMD packets

- SIMD packets ???

Raytracing summary

View dependent

Assignment Project Exam Help

Computationally expensive

<https://powcoder.com>

Good for refraction and reflection effects

Add WeChat powcoder