CompSci 373 Tutorial

Ray Tracing

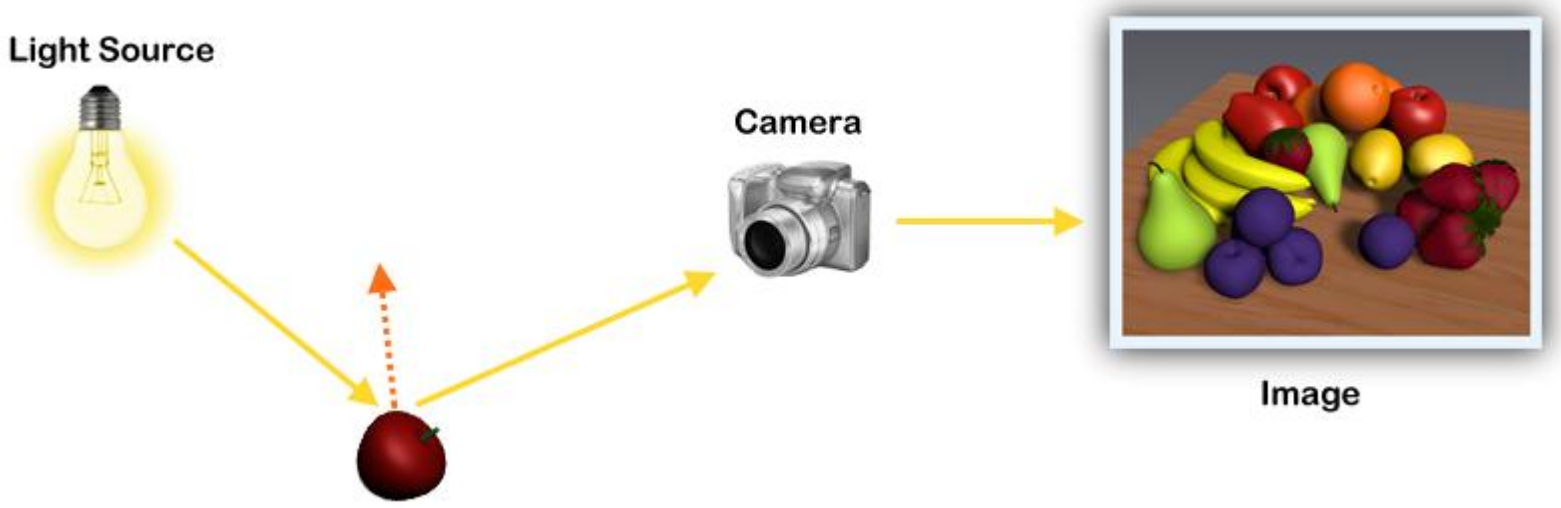
What is a Ray Tracer?

- What does it do?
  - "Simulate" image capturing process done by a camera
- What does it take as inputs and what does it produce as output(s)?
  - Take: shape and location of objects in surrounding environment, and where you are
  - Return: a single realistic 2D image

What is a Ray Tracer?

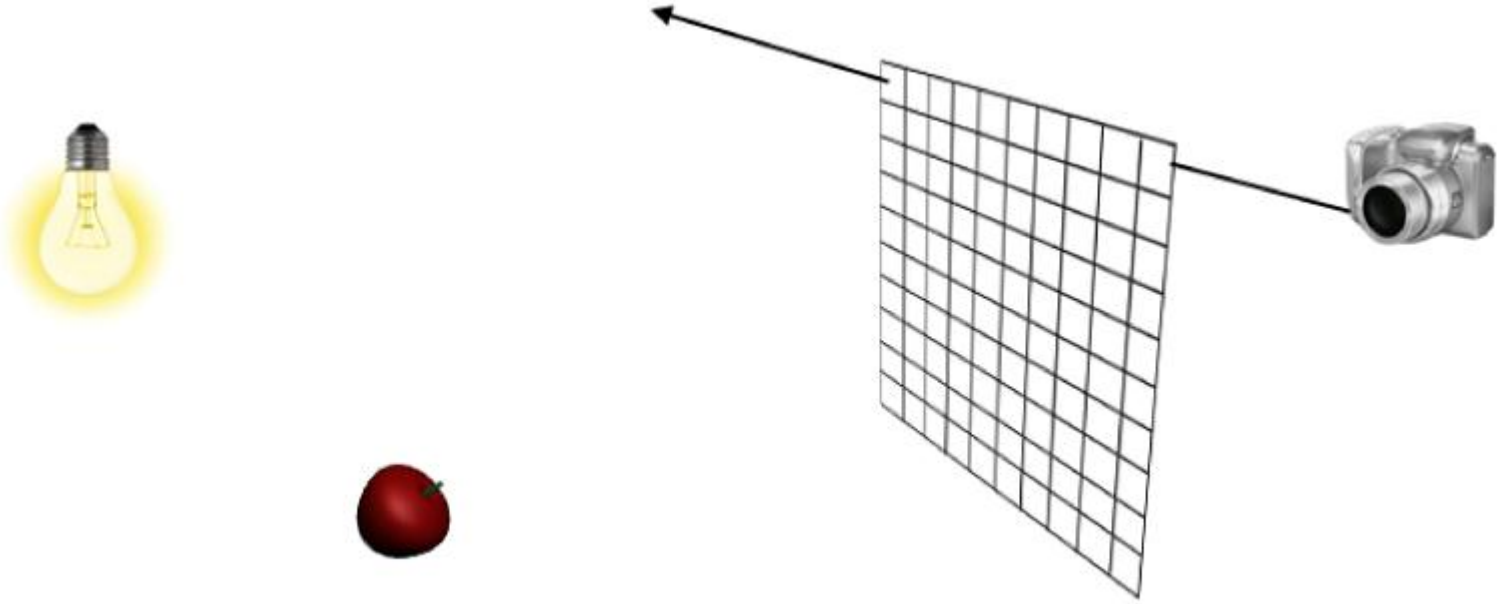
- Why is it important?

How does a camera work?

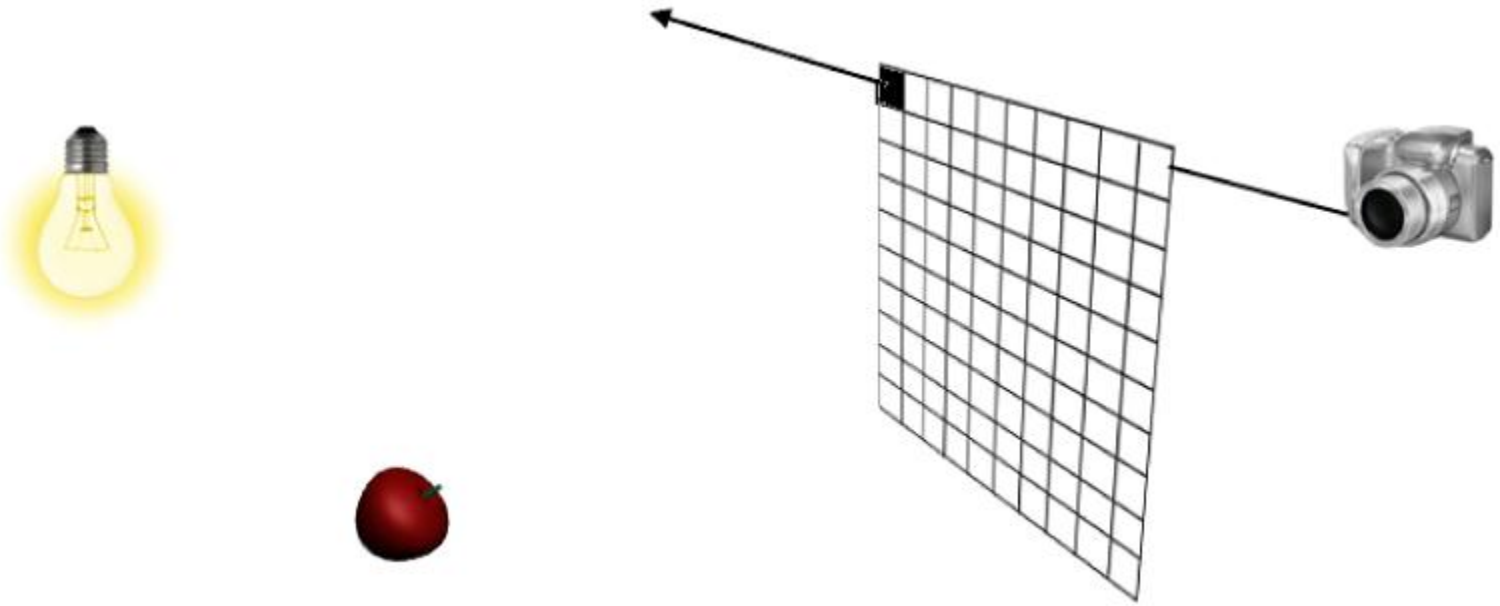


How does a ray tracer work?

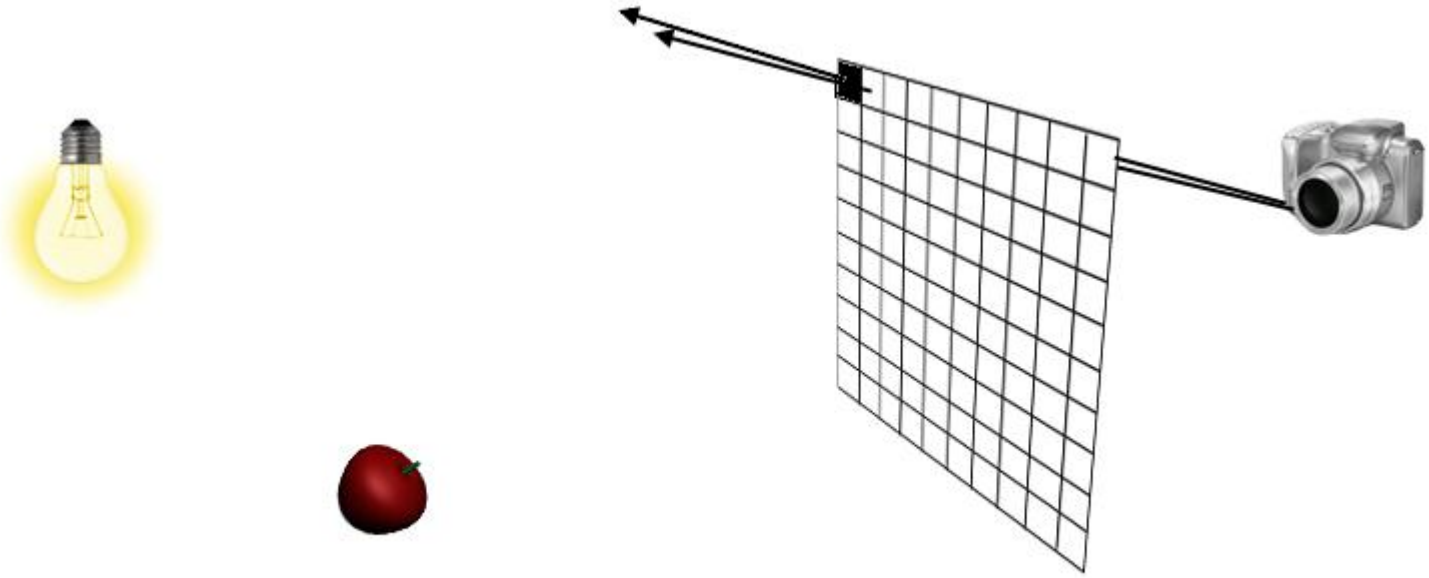
How does a ray tracer work?



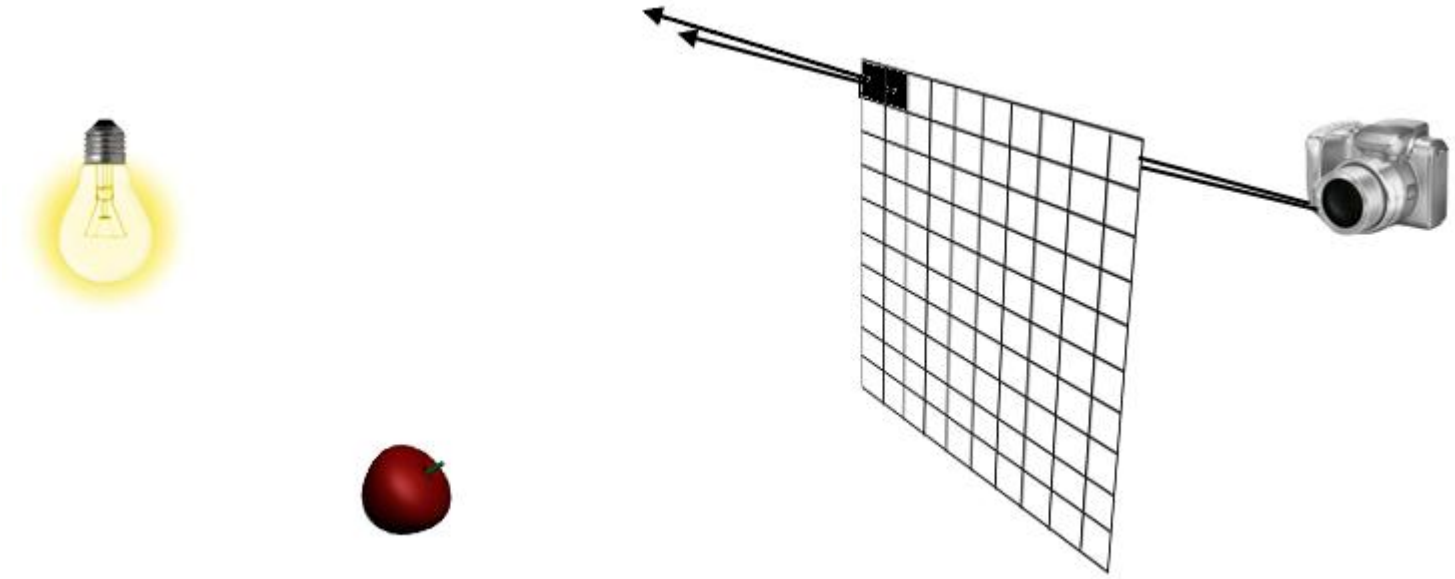
How does a ray tracer work?



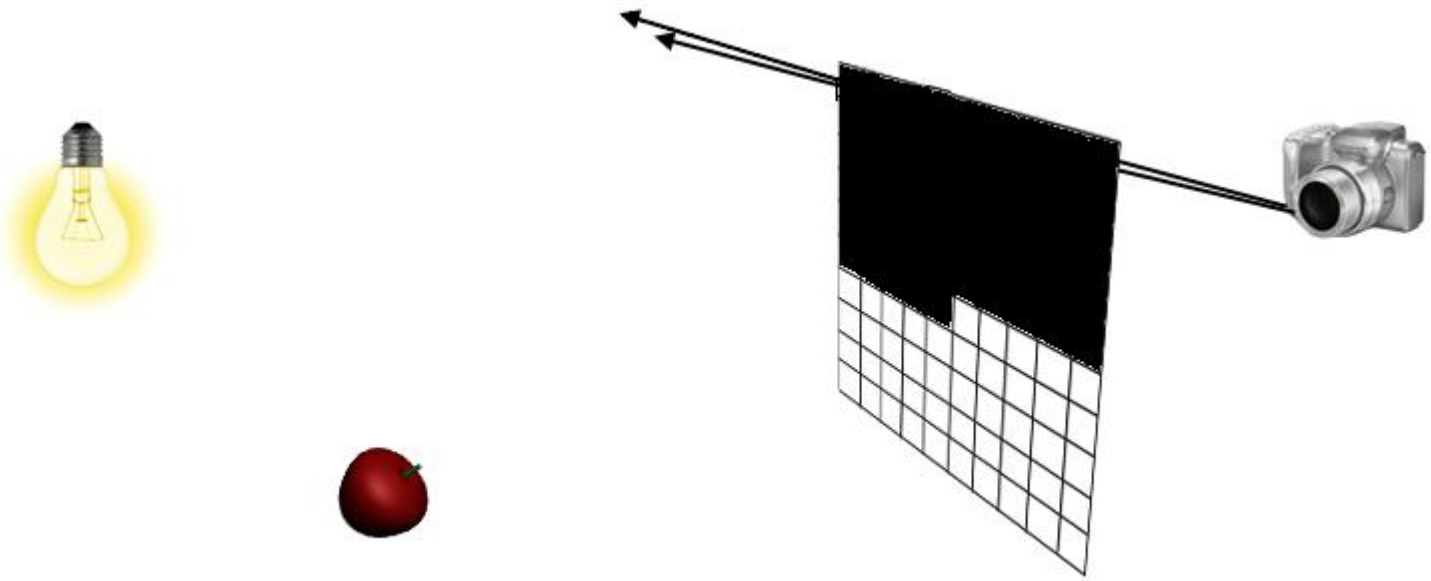
How does a ray tracer work?



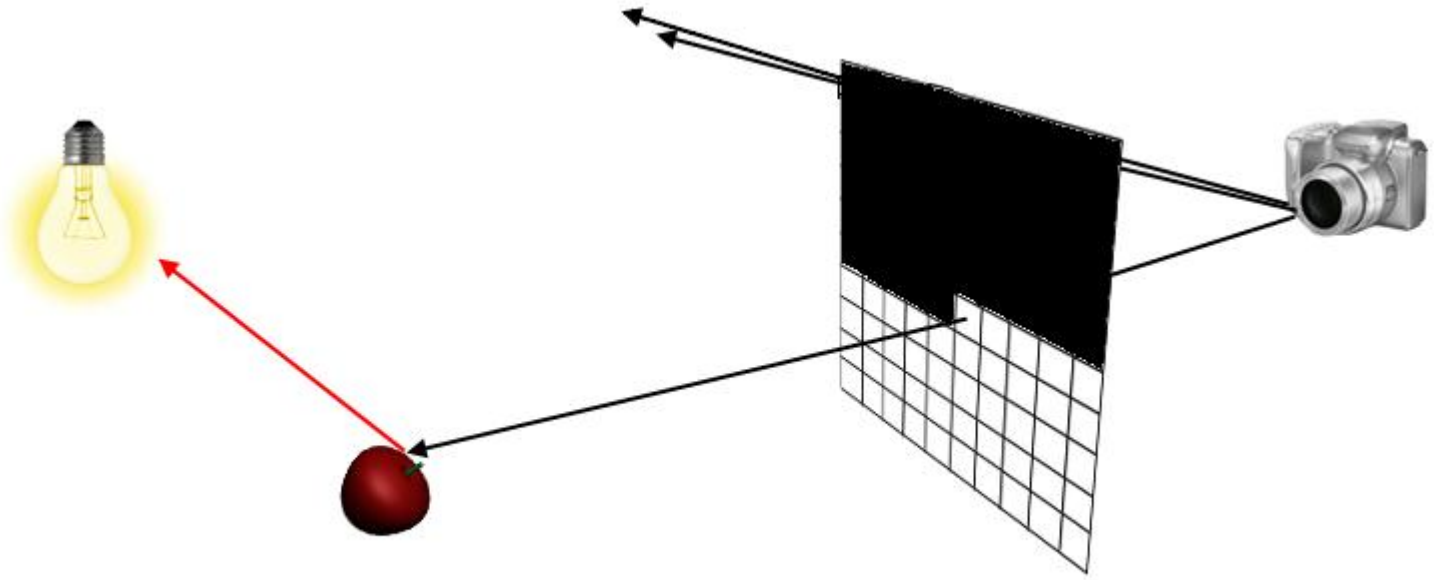
How does a ray tracer work?



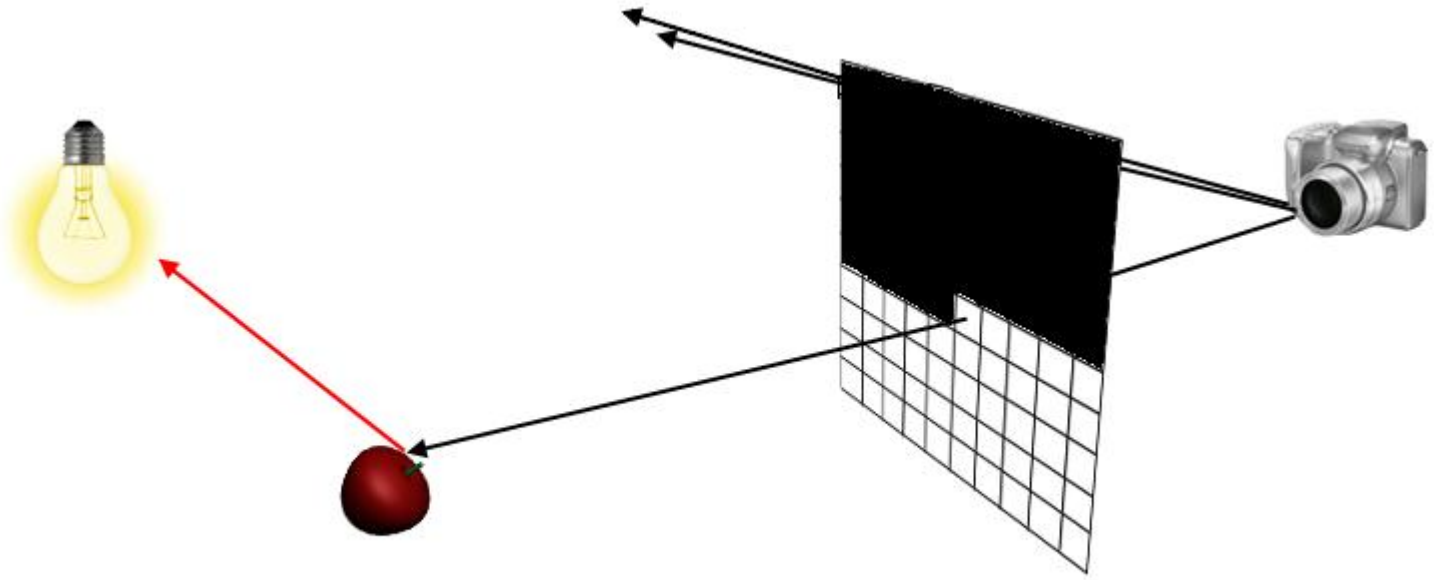
How does a ray tracer work?



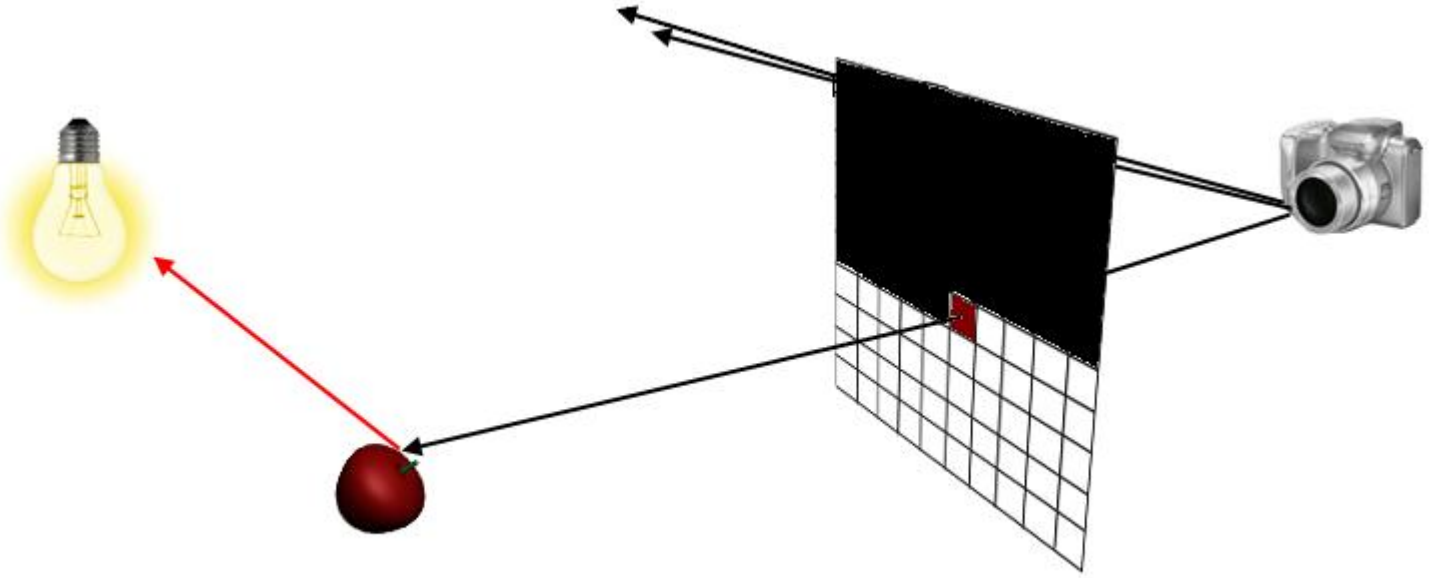
How does a ray tracer work?



How does a ray tracer work?



How does a ray tracer work?



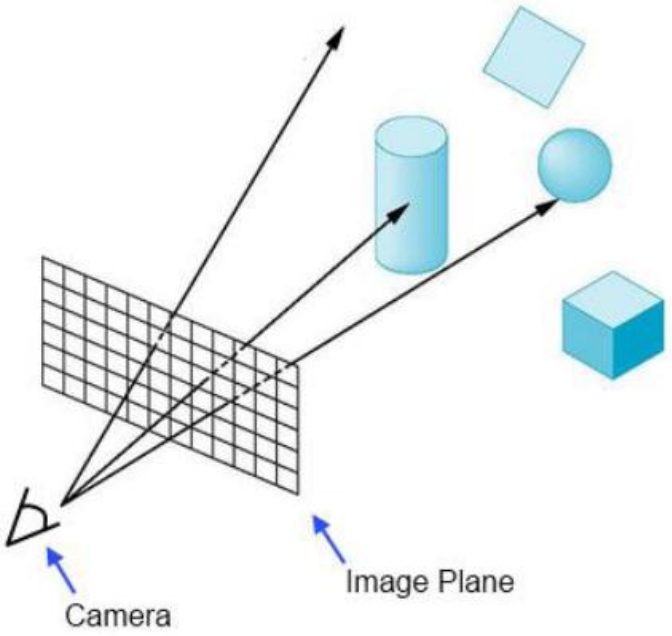
Image

Structure of a Ray Tracer

- What do we have to start with?
  - Where we're standing? (**the location of the camera** – a 3D point)
  - **Height** and **Width** of the Image we want to produce
  - The **distance** between the camera and the image
  - A collection of objects in the scene (shape & colour, position)
  - A collection of light sources (colour & location)
- What do we want?
  - A 2D image of how the scene looked from where we stand

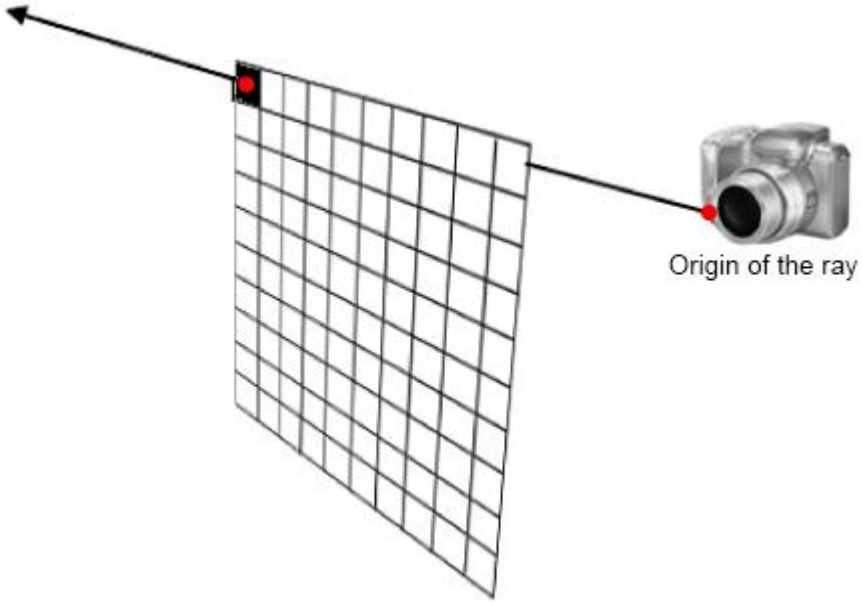
Structure of a Ray Tracer

```
foreach pixel in the output image
{
        1. Construct a ray starting from the camera through the pixel

        2. Find the first object hit by this ray

        3. Compute the color at the intersection point

        4. Assign the color to the pixel
}
```

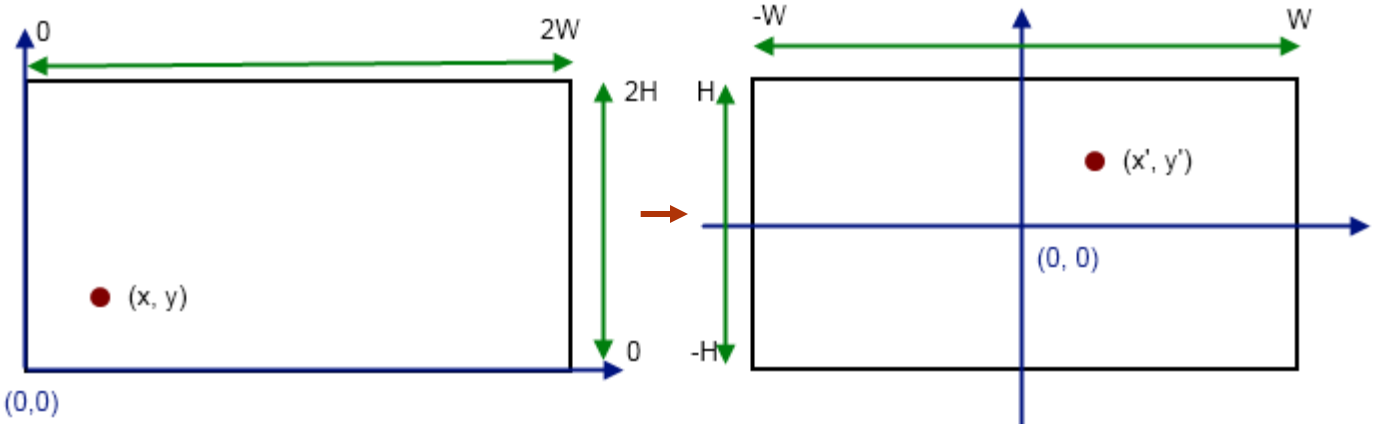


1. Ray Construction

- A ray is defined by **an origin** and **a direction**.
  - Origin (3D point): where was the ray shot from?
  - Direction (3D vector): where does it go?



1. Ray Construction



$$x' = W\left(\frac{2x}{n_{cols}} - 1\right) \qquad y' = H\left(\frac{2y}{n_{rows}} - 1\right)$$

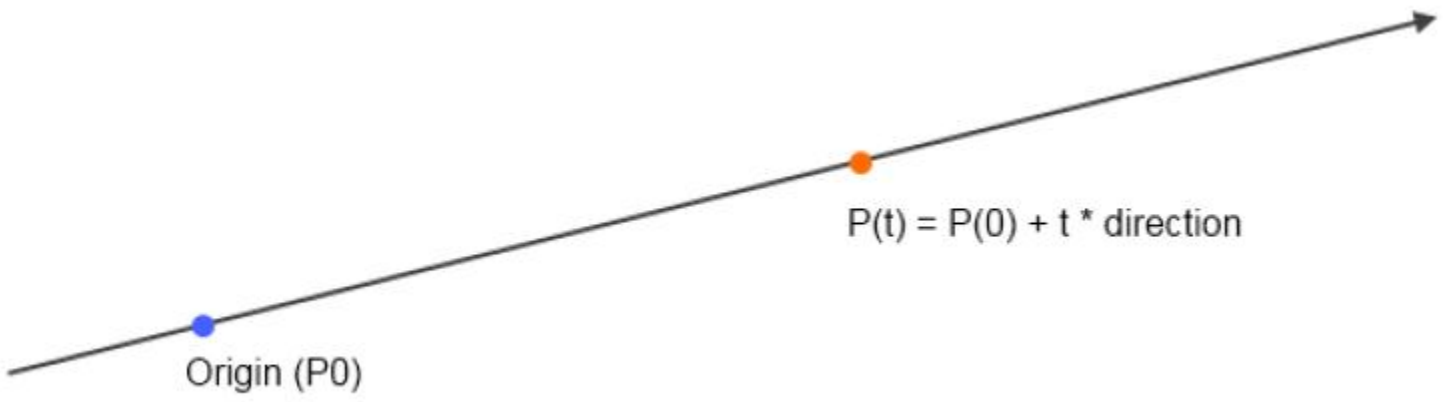
- Now we have:
  - The origin
  - We can compute the direction of a ray (x' – camera.x, y' – camera.y, -N)

2. Closest Object Intersection

- A ray is parameterised by a arithmetic distance **t**. It denotes the distance of a point on the ray from the origin.

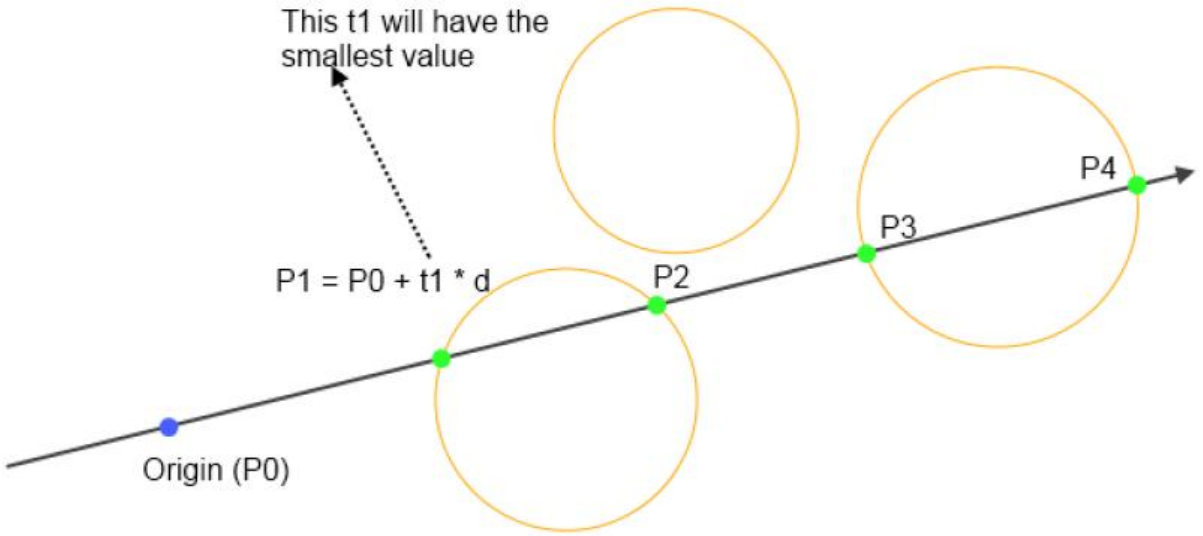
$$p = O + td$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_o \\ y_o \\ z_o \end{pmatrix} + t \begin{pmatrix} x_d \\ y_d \\ z_d \end{pmatrix}$$



2. Closest Object Intersection

- To find the closest intersected object, we iterate through each object and find if there is an intersection point. If there is, retrieve the **t** value for that point. The hit point with smallest **t** tells us which object is nearest.



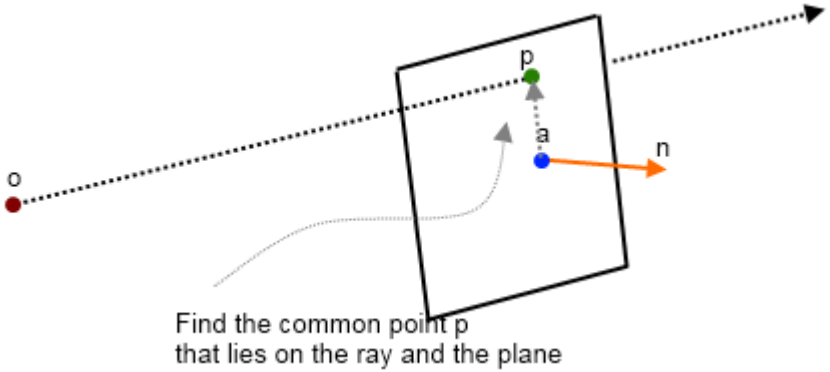
2. Object Intersection - Plane

- Plane equation

$$n \bullet p = a$$

- Ray equation

$$p = o + td$$



- Find the point p

$$n \bullet (o + td) = a$$

$$n \bullet o + n \bullet td = a$$

$$n \bullet td = a - (n \bullet o)$$

$$t = \frac{a - (n \bullet o)}{n \bullet d}$$

- Only accept t > 0

2. Object Intersection - Sphere

A sphere with center c = (x_c, y_c, z_c) and radius **r** can be represented by the implicit equation

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

- Rewrite in vector form

$$(p-c)\bullet(p-c)=r^2$$

- Ray equation

$$p=o+td$$

2. Object Intersection - Sphere

- Find the common point p

$$(o + td - c) \bullet (o + td - c) = r^2$$

$$(d \bullet d)t^2 + \left[2(o - c) \bullet d\right]t + (o - c) \bullet (o - c) - r^2 = 0$$

$$at^2 + bt + c = 0$$

2. Object Intersection - Sphere

- Find the common point p

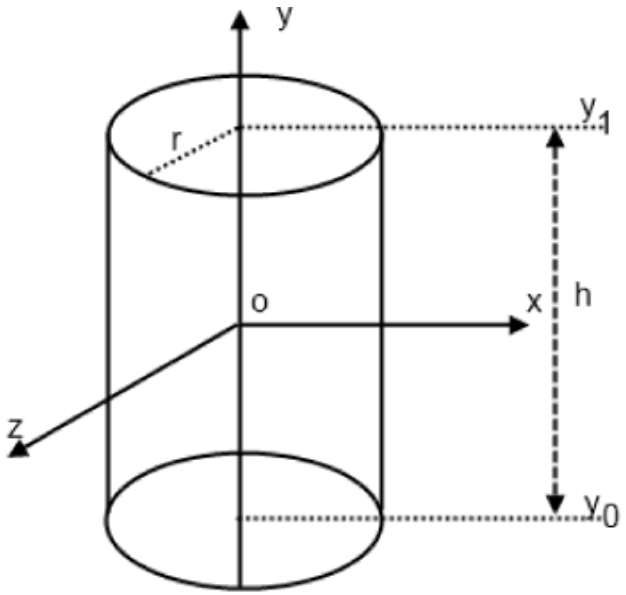
$$(o + td - c) \bullet (o + td - c) = r^2$$

$$(d \bullet d)t^2 + [2(o - c) \bullet d]t + (o - c) \bullet (o - c) - r^2 = 0$$

$$at^2 + bt + c = 0$$

- Solve for t: 3 cases
  - 0 solution: ray missed the sphere
  - 1 solution: ray hits the sphere at 1 point
  - 2 solutions: ray hits the sphere at 2 points – **get the one with smaller t**

2. Object Intersection – Open Cylinder



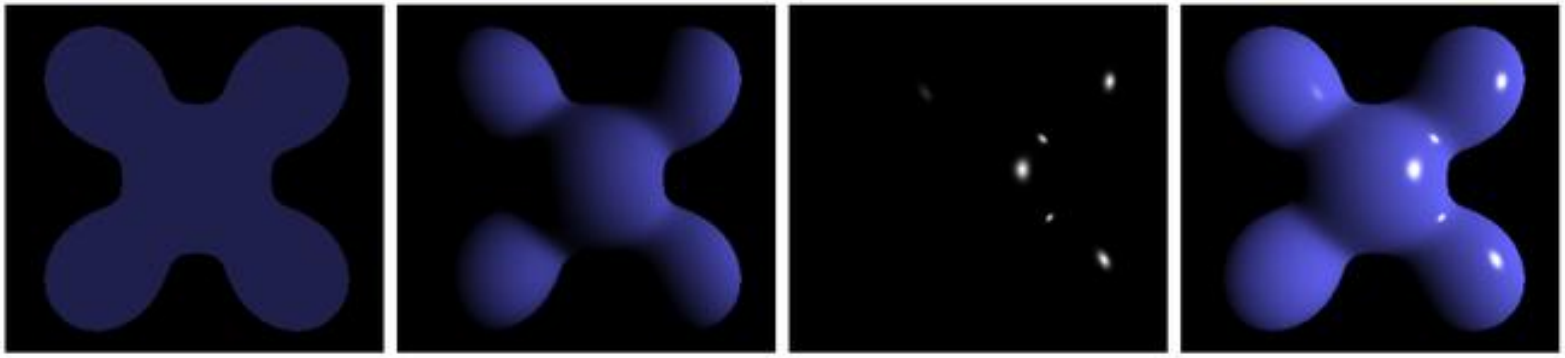
- A unit cylinder with radius **r** with infinite high can be represented by the implicit equation

$$x^2 + z^2 = r^2$$

- To determine if a given ray intersect a sphere with a certain height.
  - Find out if the ray intersects the infinite cylinder
  - Find out if the y coordinate of the hit point is $> y_0$ and $< y_1$

3. Objects and Colors

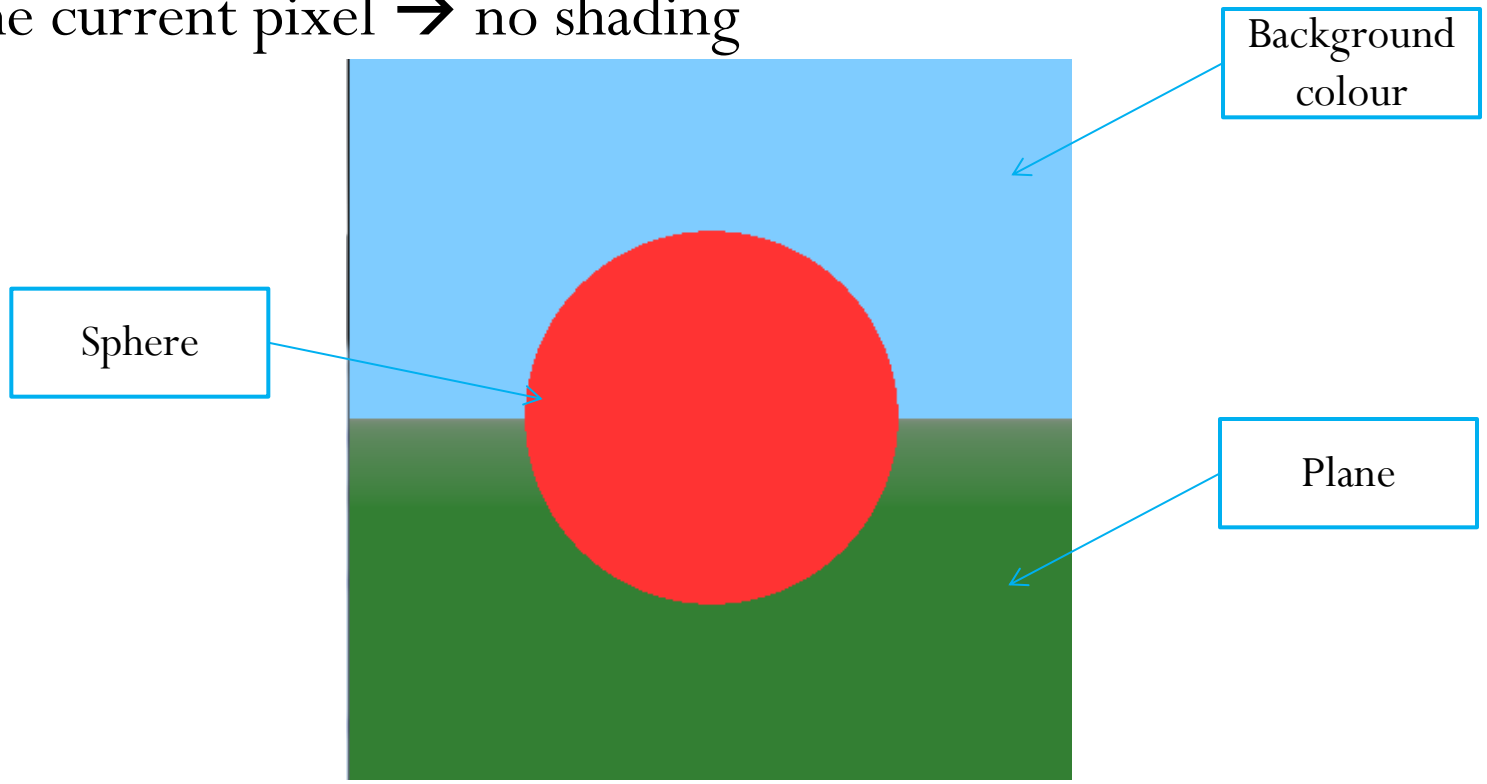
- Each object has 3 colour components (ambient, diffuse, specular), which once combined in a certain way produce the final colour.
- The colour of an object is also influenced by the number of light sources and light colours. Each light source will also have 3 colour components.



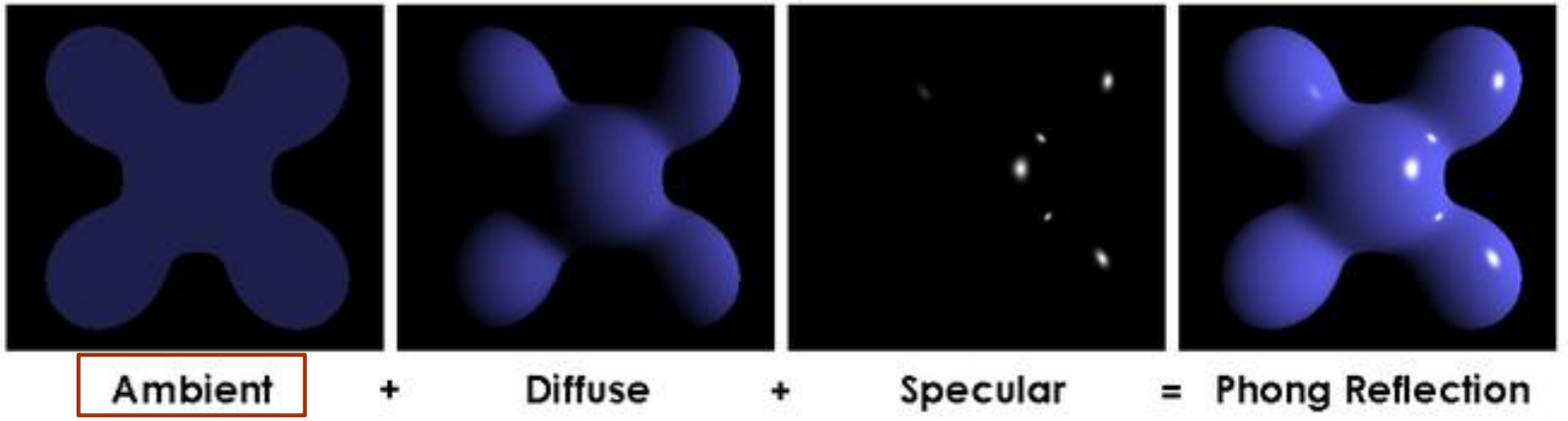
3. Color at intersection Point

Lets ignore the illumination, and only use the ambient colour of the object.

If a ray hit an object, assign the ambient colour of that object to the current pixel → no shading

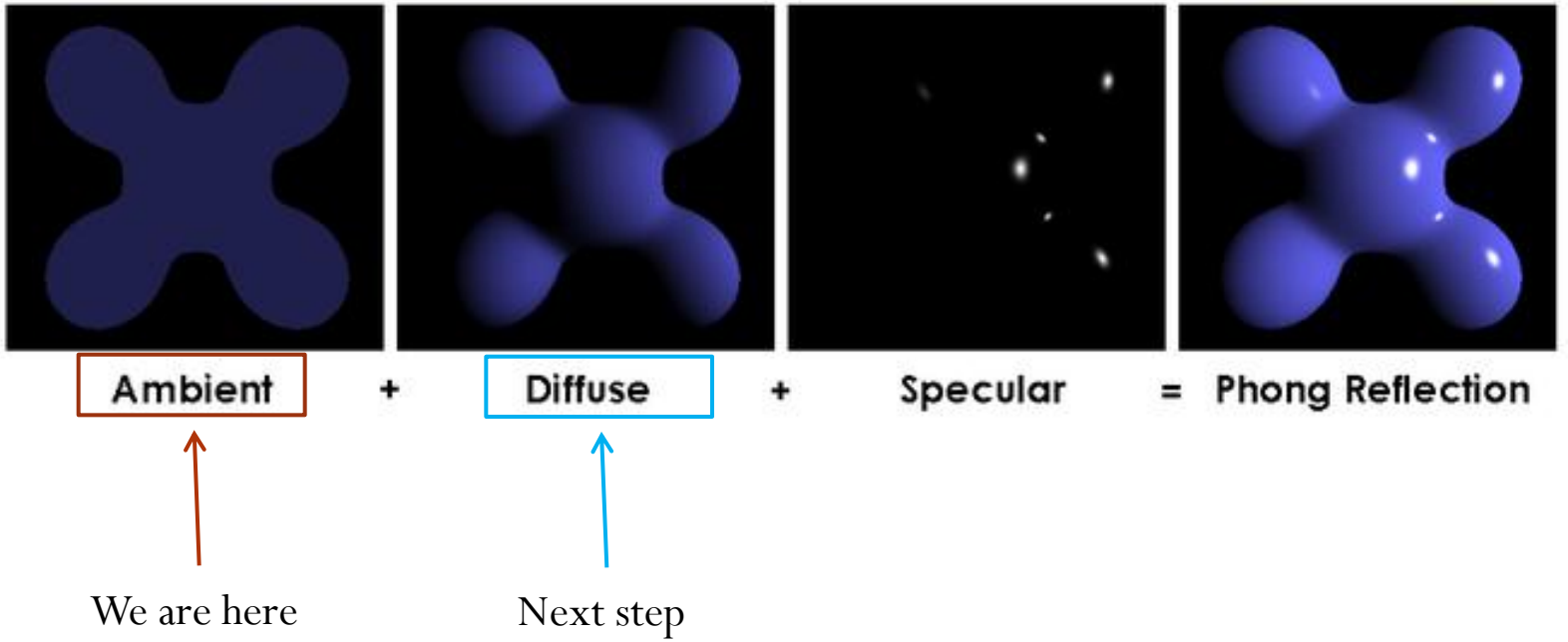


Where are we?



We are here

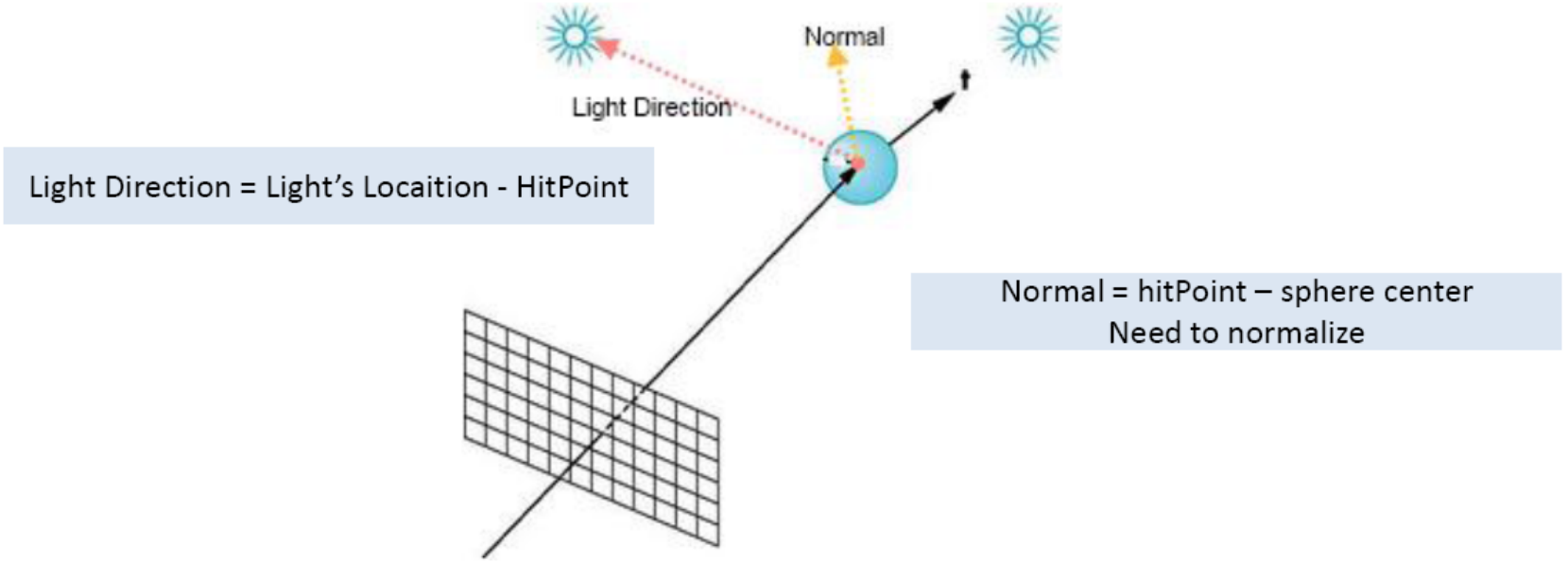
Where are we?


Ambient
+
Diffuse
+
Specular
=
Phong Reflection
We are here
Next step

Add Diffuse Illumination

The amount of light at a pixel is the sum of the contribution of each individual light source.

We should not take into account any light that is not visible from our side of the surface. That is if the normal to the sphere and the light direction, point toward opposite direction



Add Diffuse Illumination

So do what we do?

1. For each light source
2. Construct a ray starting at the hit point, heading toward the current light
3. Make sure the light and the hit point is on the same side (use dot product)
4. Compute the diffuse colour at the current pixel (must take into account light's contribution)

$$Diffuse = I_d \rho_d * L_d \rho_d * \frac{s \cdot m}{|s||m|}$$

5. Add the diffuse colour to the ambient colour earlier

With **s** the light direction and **m** the normal of the object at the hit point

Add Diffuse Illumination

So do what we do?

1. For each light source
2. Construct a ray starting at the hit point, heading toward the current light
3. Make sure the light and the hit point is on the same side (use dot product)
4. Compute the diffuse colour at the current pixel (must take into account light's contribution)

$$Diffuse = I_d \rho_d * L_d \rho_d * \frac{s \cdot m}{|s||m|}$$

5. Add the diffuse colour to the ambient colour earlier

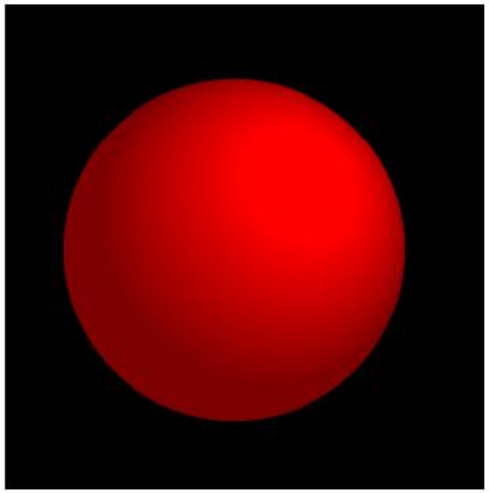
With **s** the light direction and **m** the normal of the object at the hit point

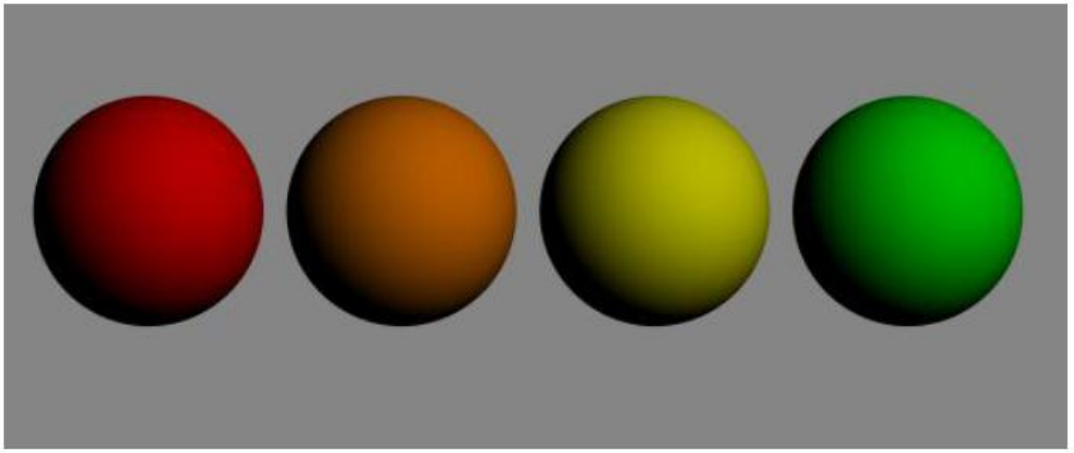
Diffuse colour of the object

Diffuse scale factor

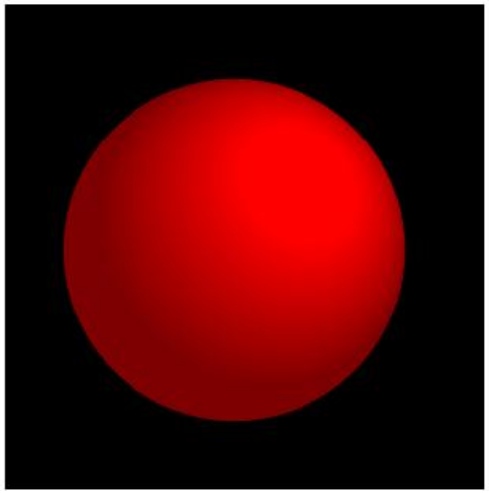
Diffuse colour of the current light

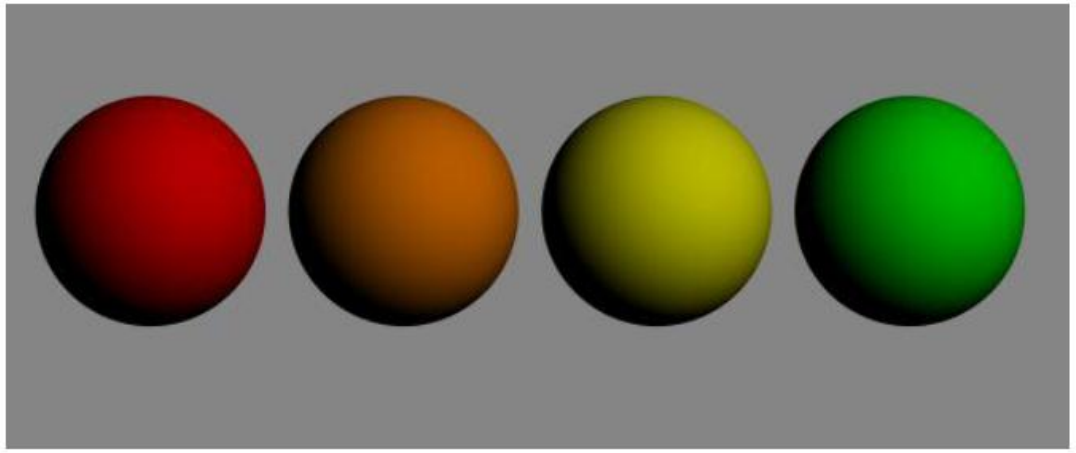
What do we have?





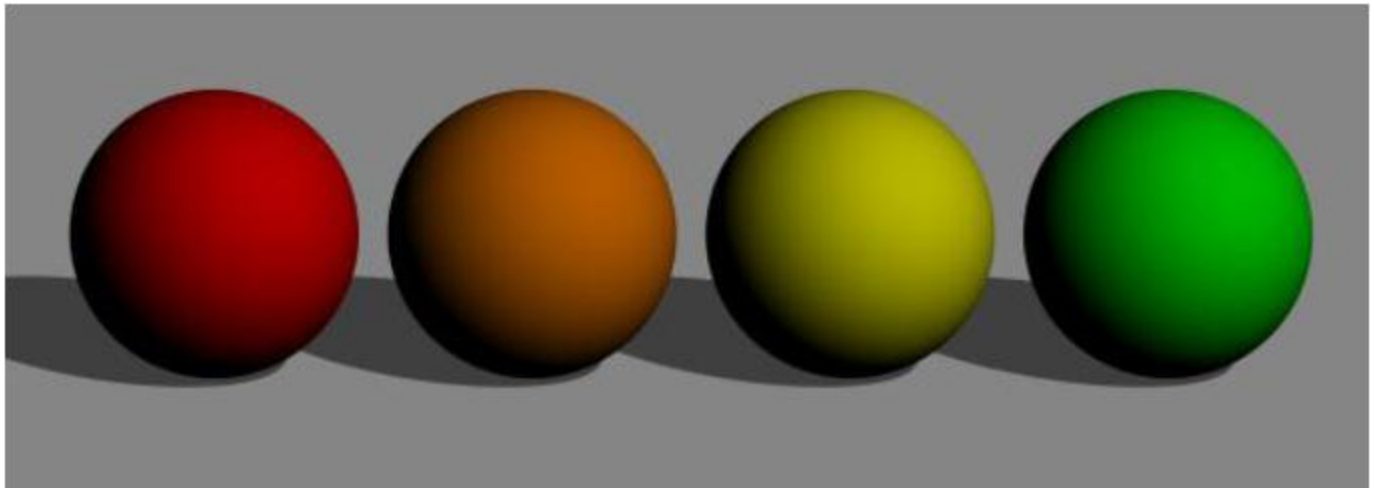
What do we have?





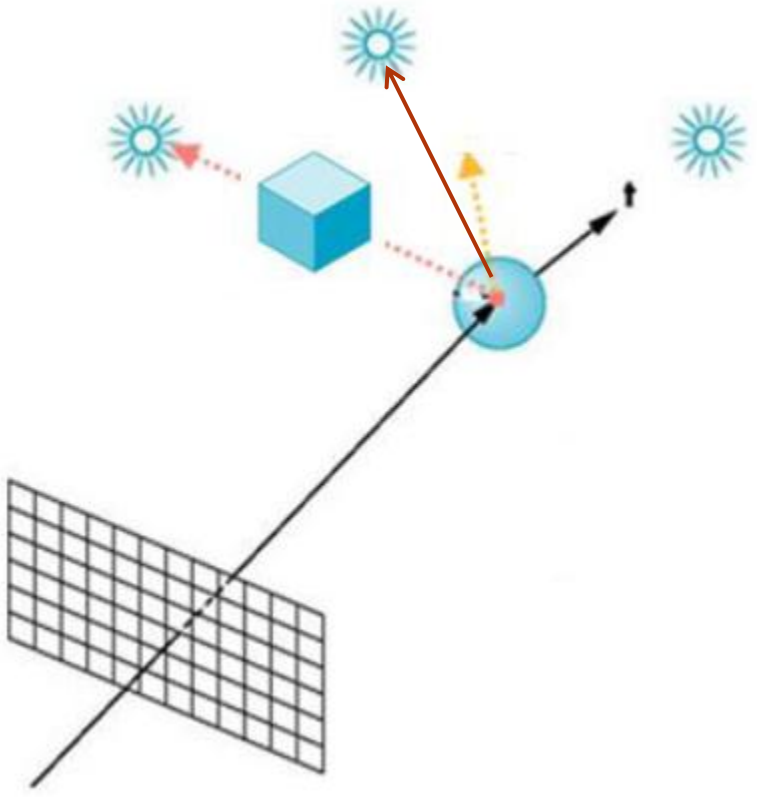
How height above the ground these spheres lie?

Real world objects often cast shadow.



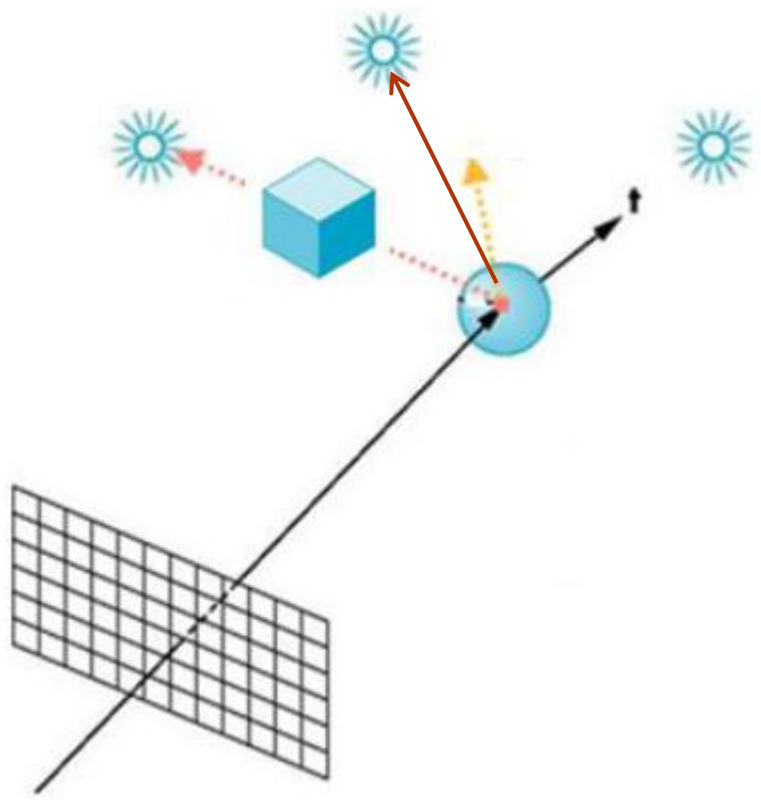
Add Shadow

The amount of light at a pixel is the sum of the contribution of each individual light source EXCEPT light sources that are blocked by other objects



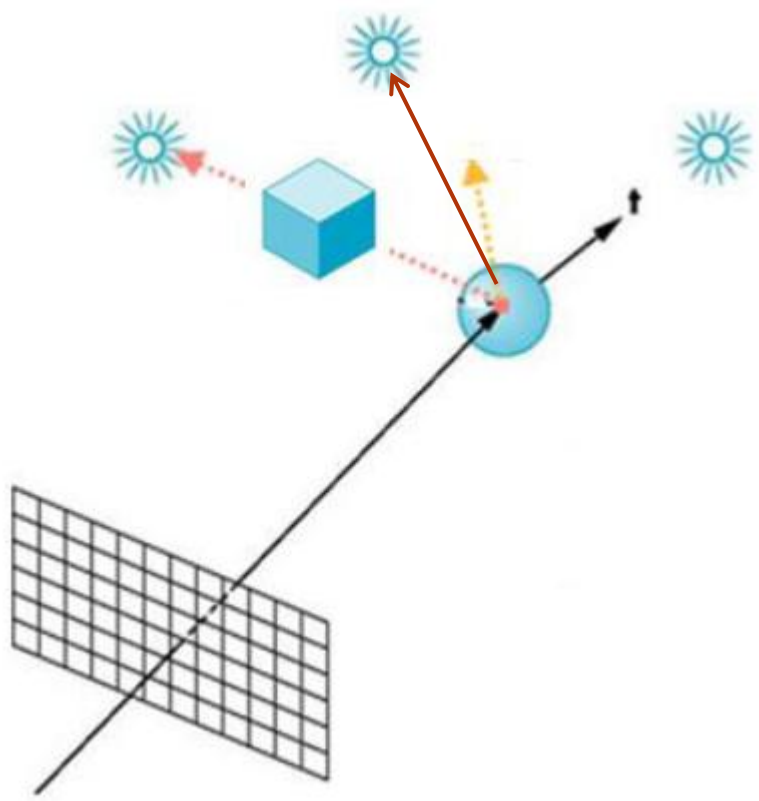
Add Shadow

Given an intersection point and a light source, how could we know if that light source is blocked from the point's perspective?



Add Shadow

Given an intersection point and a light source, how could we know if that light source is blocked from the point's perspective?



Solution: Shoot another ray from the intersection point toward the light source

Check if the new ray intersects with any object?

If it does, the light is blocked, ignore that light

Add Shadow

So do what we do?

1. For each light source
2. Construct a ray starting at the hit point, heading toward the current light
3. Make sure the light and the hit point is on the same side (use dot product)

//shadow

4. Construct a new ray originates from the hit point, going to the current light.
5. Check if that ray intersects any object? If yes, ignore this light.

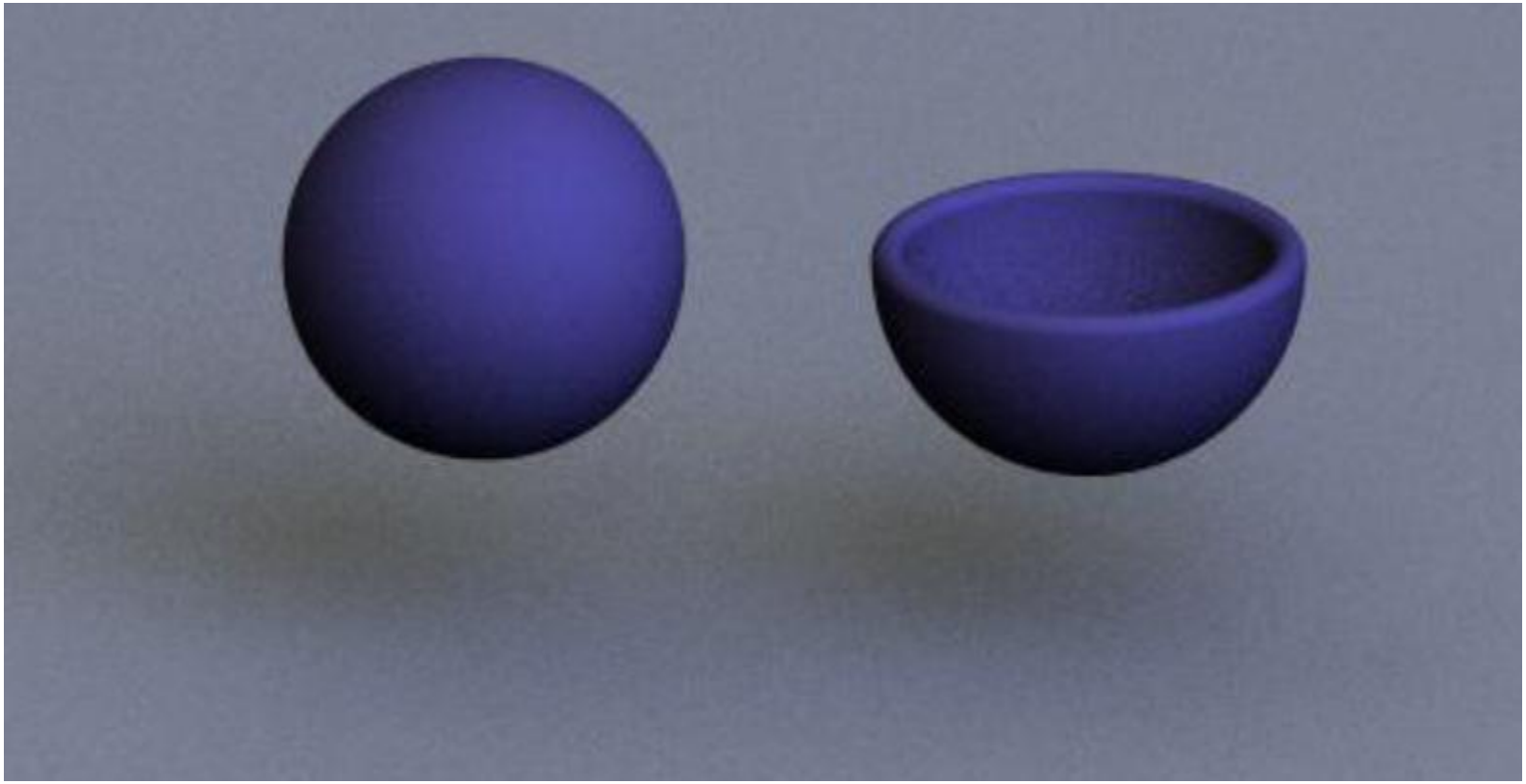
//diffuse

6. Compute the diffuse colour at the current pixel (must take into account light's contribution)

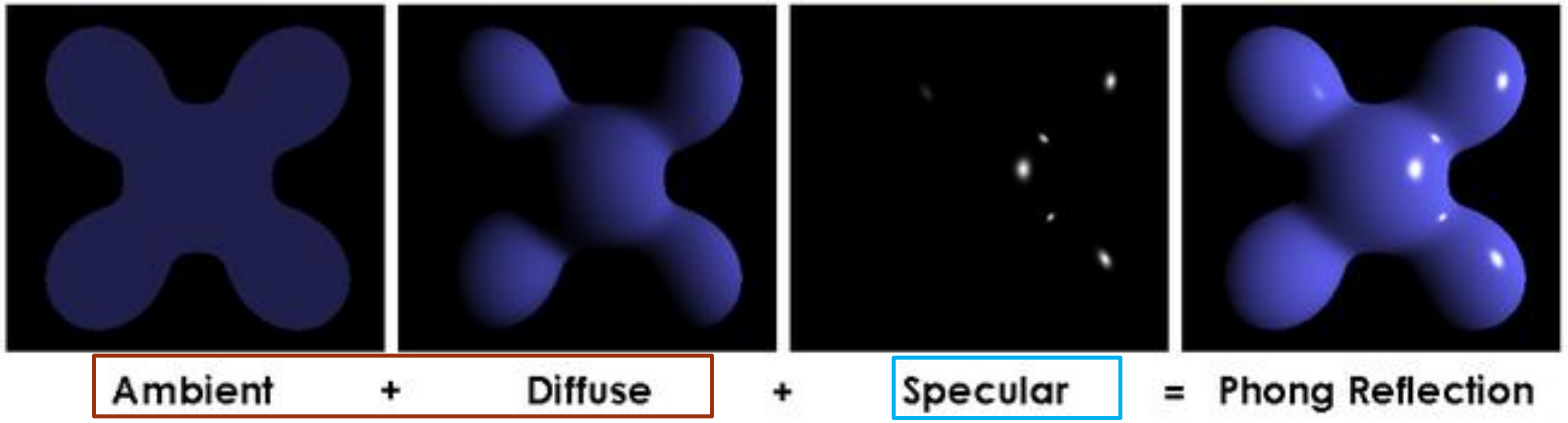
$$Diffuse = I_d \rho_d * L_d \rho_d * \frac{s \cdot m}{|s||m|}$$

7. Add the diffuse colour to the ambient colour earlier

What does it look like?

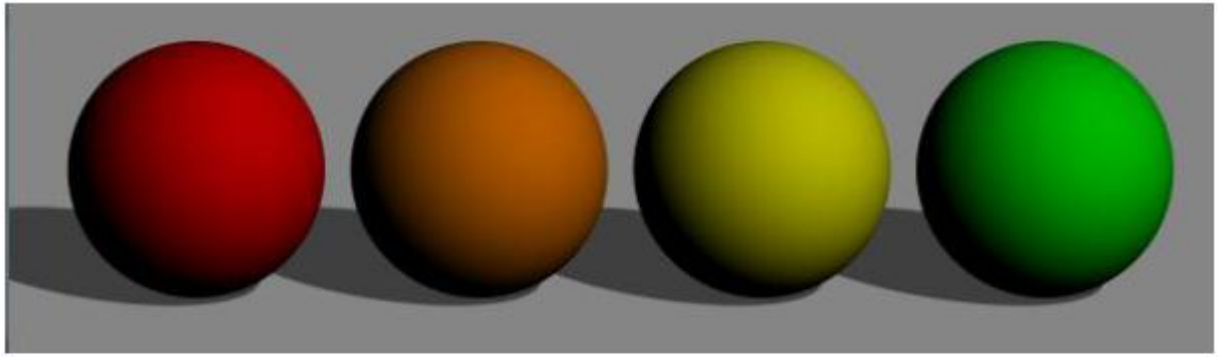


Where are we?

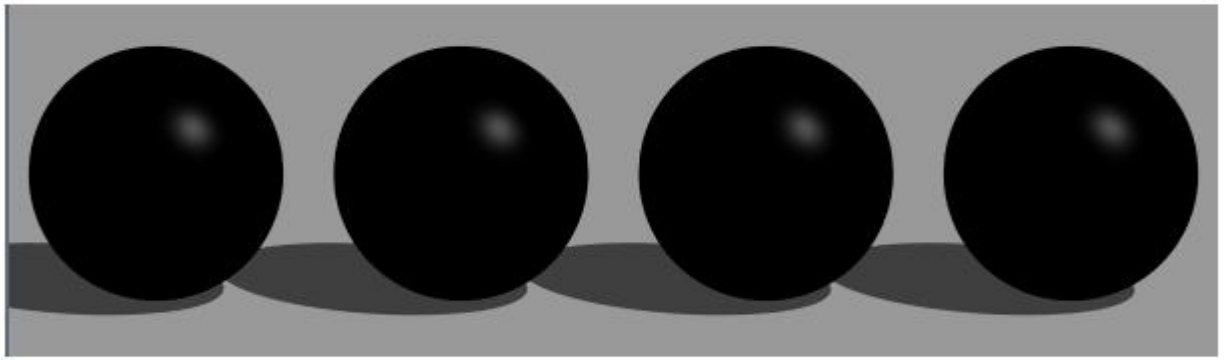

Ambient
+
Diffuse
+
Specular
=
Phong Reflection
We are here
Next step

Add Specular Light

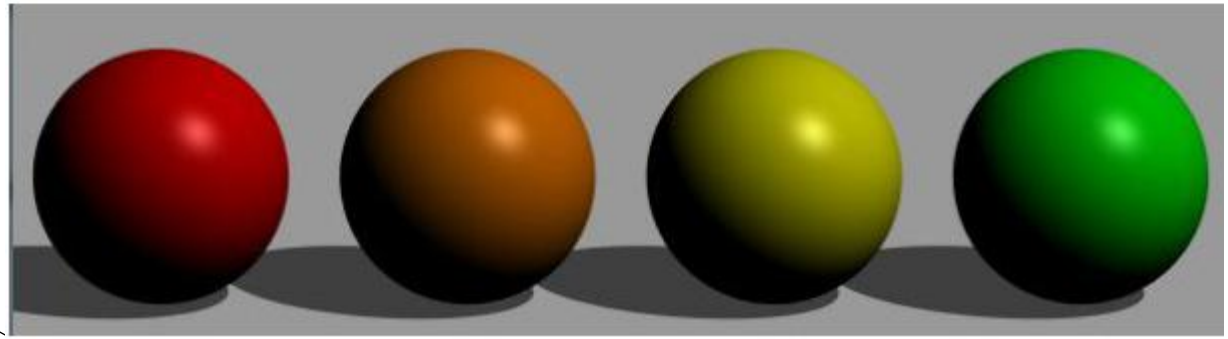
Model shiny surface by adding additional specular reflection



$$Ambient + Diffuse = Ambient + I_d \rho_d \frac{s \cdot m}{|s||m|}$$



$$Specular = I_s \rho_s \left(\frac{h \cdot m}{|h||m|} \right)^{\alpha}$$



$$Ambient + Diffuse + Specular$$

Add Specular Light

Compute Specular component

$$Specular = I_s \rho_s * L_s \rho_s * \left(\frac{h \cdot m}{|h||m|} \right)^{\alpha}$$

Specular colour of the object

Specular scale factor

h is the half-way vector between light direction vector **s** and **v** (viewing direction point from hit point toward camera)

alpha is the specular scaling

Add Specular Illumination

So do what we do?

1. For each light source
2. Construct a ray starting at the hit point, heading toward the current light
3. Make sure the light and the hit point is on the same side (use dot product)

//shadow

4. Construct a new ray originates from the hit point, going to the current light.
5. Check if that ray intersects any object? If yes, ignore this light.

//diffuse

6. Compute the diffuse colour at the current pixel (must take into account light's contribution)

$$Diffuse = I_d \rho_d * L_d \rho_d * \frac{s \cdot m}{|s||m|}$$

7. Add the diffuse colour to the ambient colour earlier

//specular

8. Compute Specular colour

$$Specular = I_s \rho_s * L_s \rho_s * \left(\frac{h \cdot m}{|h||m|} \right)^{\alpha}$$

9. Add the computed specular colour to the colour we have so far
10. Assign the colour to the current pixel

What do we have?