

Ray Tracing

Introduction to Computer Graphics

Andrzej Szymczak

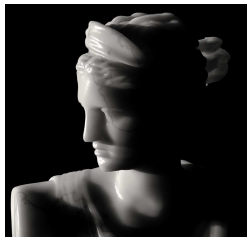
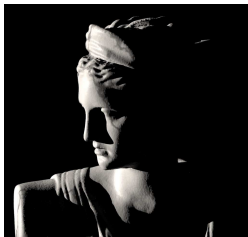
August 23, 2011

Introduction

- ▶ We'll focus on a simple variant of ray tracing:
 - ▶ Point light source(s)
 - ▶ Direct illumination (will only care about light coming directly from the light source and scattered toward the viewpoint)
 - ▶ Medium = vacuum (good enough approximation for clean dry air)
 - ▶ All surfaces are opaque (no subsurface scattering, transparency etc.)
 - ▶ Static scenes only (no motion blur/shutter speed)

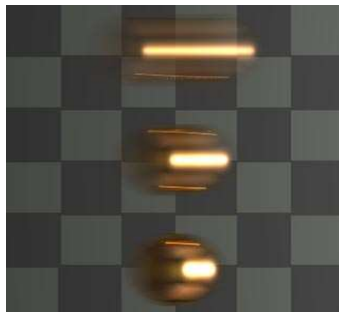
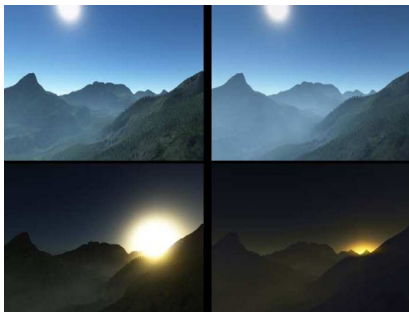
Subsurface scattering

- ▶ Diana the Huntress (marble); courtesy Henrik Wann Jensen
- ▶ Left: no subsurface scattering (what you would get using the simple ray tracing algorithm we'll focus on)



Fog and motion blur

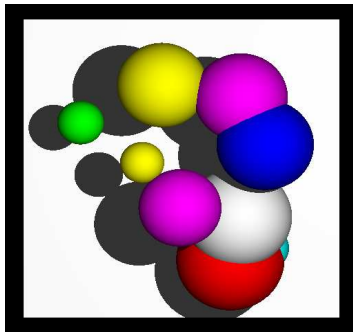
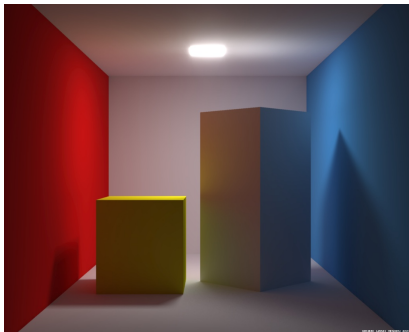
Two other things we won't be covering



Multiple scattering and area light sources vs single scattering and point light sources

What we won't be able to get:

- ▶ Color bleeding
- ▶ Soft shadows



Input and output

Input: We want the screen to be the window to the virtual world; what do we need?

- ▶ A mathematical description of the virtual world
 - ▶ Geometric primitives (triangles, spheres...) that form the objects we'd like to see
 - ▶ Location, intensity, color of each light source
 - ▶ Surface properties (color; matte or shiny? more generally: how does it scatter light); for now, we'll assume that the color is specified on per primitive basis
- ▶ A mathematical description of the view
 - ▶ Viewpoint (location of the viewer/user in the virtual world)
 - ▶ Location and size of the window in the virtual world
 - ▶ Can be thought of as the location, direction, aspect ratio and focal length of the camera

Output: A good-looking image

What is available?

- ▶ We can color pixels!
- ▶ How to compute the right color for a pixel?
 - 1 What is visible through the pixel
 - 2 How much light is scattered there toward the viewpoint?
- ▶ 1: Follow a ray from the viewpoint through the (center of) the pixel and find **the closest intersection** point (call it p) with a primitive in the virtual world
- ▶ 2: Is p in shadow or not? What is the primitive made of?
- ▶ p is in shadow if and only if...
 - ▶ it cannot see the light source
 - ▶ follow the ray starting at p in the direction of the light source; is the **closest intersection point** of that ray and a primitive between p and the light source? p is in shadow if and only if it is.

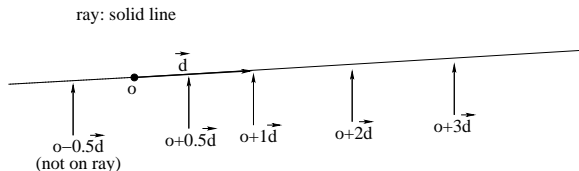
Ray-Primitive Intersections

Andrzej Szymczak

August 23, 2011

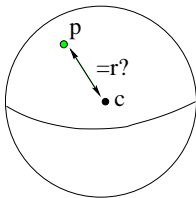
Ray

- ▶ Defined by:
 - ▶ origin o
 - ▶ direction \vec{d}
- ▶ A point p is on the ray if and only if $p = o + t\vec{d}$ for some nonnegative number t



Sphere

- ▶ Defined by:
 - ▶ center $c = (c_x, c_y, c_z)$
 - ▶ radius r
- ▶ A point p is on the sphere if and only if $|p - c| = r$, where $|\cdot|$ is the Euclidean norm ($|\vec{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}$)
- ▶ A point $p = (p_x, p_y, p_z)$ is on the sphere if and only if $(p_x - c_x)^2 + (p_y - c_y)^2 + (p_z - c_z)^2 = r^2$



Ray-sphere intersection

- ▶ We want a procedure that:
 - ▶ Takes ray (o, \vec{d}) and sphere (c, r) as the input
 - ▶ Returns the intersection point closest to the ray's origin if an intersection point exists or reports there is no intersection
- ▶ An intersection point is a point p that is both on the ray and on the sphere
 - (i) $p = o + t\vec{d}$
 - (ii) $|p - c|^2 = r^2$
- ▶ $0 = |p - c|^2 - r^2 = |o + t\vec{d} - c|^2 - r^2 =$
 $= |\vec{co} + t\vec{d}|^2 - r^2 = |\vec{co}|^2 - r^2 + 2(\vec{co} \cdot \vec{d})t + |\vec{d}|^2 t^2 =$
 $= C + Bt + At^2$

where:

$$C = |\vec{co}|^2 - r^2$$

$$B = 2\vec{co} \cdot \vec{d}$$

$$A = |\vec{d}|^2$$

Ray-sphere intersection

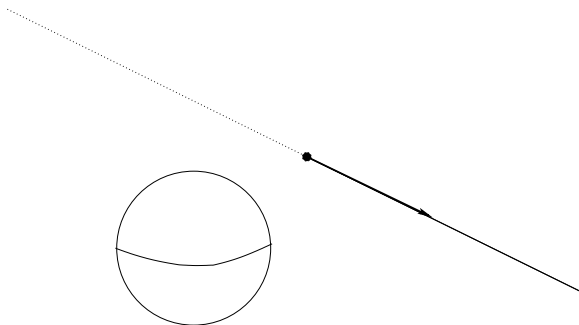
- ▶ $At^2 + Bt + C = 0$
- ▶ Let's not worry whether $t \geq 0$ or not for a moment...
- ▶ The number of solutions depends on its discriminant $\Delta = B^2 - 4AC$; we have the following three cases:
 - (1) if $\Delta < 0$, no solution exists
 - (2) if $\Delta = 0$, there is only one solution
 - (3) if $\Delta > 0$, there are two solutions.
- ▶ Case (2): the solution is given by $t = -B/(2A)$
- ▶ Case (3): the two solutions are given by $t_{\pm} = (-B \pm \sqrt{\Delta})/(2A)$.
- ▶ If all solutions t are negative, we have no intersection points (report 'no intersection'); this includes case (1)
- ▶ If there is a positive solution t , let t_* be the smallest positive solution (sometimes – in case (3) – there may be two t 's and both can be positive; in this case discard the larger one)
- ▶ The intersection point is given by $p_* = o + t_* \vec{d}$

Ray-sphere intersection: Geometry

Case (1): $\Delta < 0$, no t

Ray-sphere intersection: Geometry

Case (1): $\Delta < 0$, no t

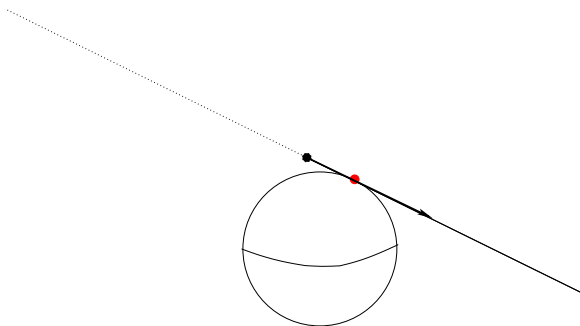


Ray-sphere intersection: Geometry

Case (2): $\Delta = 0$, one positive t , one intersection point (ray tangent to sphere)

Ray-sphere intersection: Geometry

Case (2): $\Delta = 0$, one positive t , one intersection point (ray tangent to sphere)

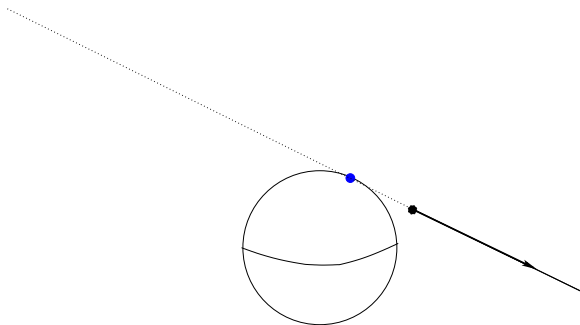


Ray-sphere intersection: Geometry

Case (2): $\Delta = 0$, one negative t , no intersection point (line containing ray tangent to sphere but tangency point not on the ray)

Ray-sphere intersection: Geometry

Case (2): $\Delta = 0$, one negative t , no intersection point (line containing ray tangent to sphere but tangency point not on the ray)

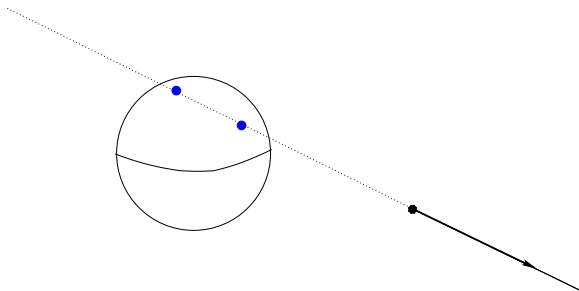


Ray-sphere intersection: Geometry

Case (3): $\Delta > 0$, two ts , both negative: no intersection point

Ray-sphere intersection: Geometry

Case (3): $\Delta > 0$, two ts , both negative: no intersection point

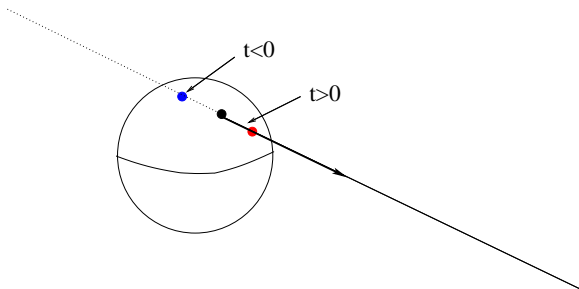


Ray-sphere intersection: Geometry

Case (3): $\Delta > 0$, two ts , one positive and one negative:
intersection point corresponds to the positive t

Ray-sphere intersection: Geometry

Case (3): $\Delta > 0$, two t s, one positive and one negative:
intersection point corresponds to the positive t

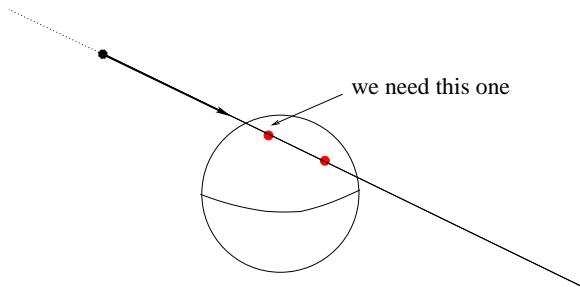


Ray-sphere intersection: Geometry

Case (3): $\Delta > 0$, two ts , both positive; two intersection points; we care about the one corresponding to the smaller t value

Ray-sphere intersection: Geometry

Case (3): $\Delta > 0$, two ts , both positive; two intersection points; we care about the one corresponding to the smaller t value



Triangle

- ▶ Triangle is defined by three vertices, a_1 , a_2 and a_3
- ▶ A point p is in the triangle if and only if p is a *weighted average* of a_1 , a_2 and a_3 , i.e. there exist nonnegative numbers α , β and γ such that $\alpha + \beta + \gamma = 1$ and

$$p = \alpha a_1 + \beta a_2 + \gamma a_3.$$

Ray-Triangle Intersection I

- ▶ Thus, we need to find t and the weights such that

$$o + t\vec{d} = \alpha a_1 + \beta a_2 + \gamma a_3.$$

- ▶ We have 4 equations:

(1) $o_x + d_x t = a_{1x}\alpha + a_{2x}\beta + a_{3x}\gamma$

(2) $o_y + d_y t = a_{1y}\alpha + a_{2y}\beta + a_{3y}\gamma$

(3) $o_z + d_z t = a_{1z}\alpha + a_{2z}\beta + a_{3z}\gamma$

(4) $\alpha + \beta + \gamma = 1$

- ▶ In the matrix form:

$$\begin{bmatrix} a_{1x} & a_{2x} & a_{3x} & -d_x \\ a_{1y} & a_{2y} & a_{3y} & -d_y \\ a_{1z} & a_{2z} & a_{3z} & -d_z \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} o_x \\ o_y \\ o_z \\ 1 \end{bmatrix}$$

- ▶ Easy to eliminate one of the variables (say, γ) and turn this system into 3 equations with 3 unknowns

Ray-Triangle Intersection I

- ▶ Let's assume it has a unique solution t_* , α_* , β_* , γ_*
- ▶ If t_* , α_* , β_* and γ_* are all nonnegative, then there is an intersection point and it is given by $o + t_* \vec{d}$
- ▶ Otherwise, there is no intersection point
- ▶ If there is no solution or it is not unique, the ray is parallel to the triangle's plane; in practice, we can just report no intersection

Ray-Triangle Intersection II (probably easier)

Two-step process:

- Step 1** : Compute the intersection point p of the triangle's plane and the ray; if there is no intersection, report 'no intersection' and stop.
- Step 2** : Decide whether p is inside or outside the triangle; if it is outside, report 'no intersection', otherwise, the intersection point is p .

Ray-Triangle Intersection II

Step 1: intersect the ray and the triangle's plane

- ▶ Determine the equation of the triangle's plane
 - ▶ Compute the triangle's normal $n := a_1\vec{a}_2 \times a_1\vec{a}_3$
 - ▶ p is in the triangle's plane if and only if

$$a_1\vec{p} \cdot \vec{n} = 0, \text{ or } p \cdot \vec{n} = a_1 \cdot \vec{n}.$$

- ▶ Substitute $p := o + t\vec{d}$ and solve:

$$(o + t\vec{d}) \cdot \vec{n} = a_1 \cdot \vec{n} \Rightarrow t = \frac{(a_1 - o) \cdot \vec{n}}{\vec{d} \cdot \vec{n}}.$$

- ▶ Report no intersection if $t < 0$
- ▶ If $t \geq 0$, the ray-plane intersection point is $p = o + t\vec{d}$.

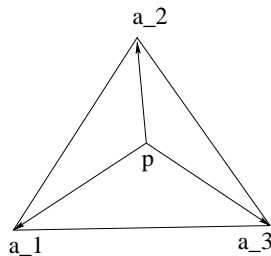
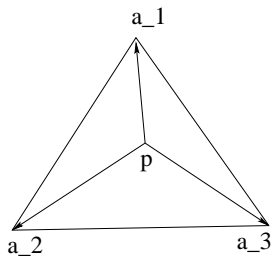
Ray-Triangle Intersection II

Step 2: Test if the intersection point is inside the triangle

Ray-Triangle Intersection II

Step 2: Test if the intersection point is inside the triangle

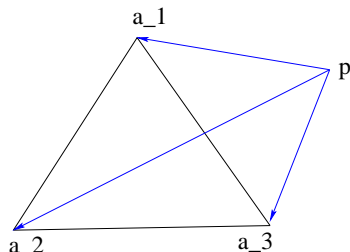
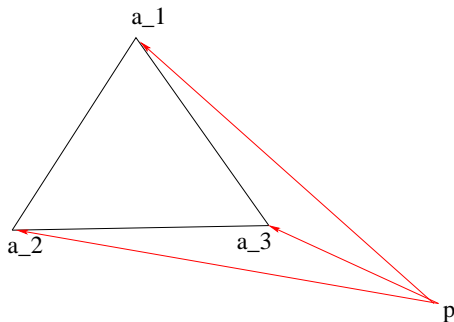
- ▶ Basic idea: in which direction do we need to rotate pa_i to make it point along pa_{i+1} (for $i = 1, 2, 3$)?



Ray-Triangle Intersection II

Step 2: Test if the intersection point is inside the triangle

- ▶ Basic idea: in which direction do we need to rotate pa_i to make it point along pa_{i+1} (for $i = 1, 2, 3$)?



Ray-Triangle Intersection II

Step 2: Test if the intersection point is inside the triangle

- ▶ Compute the cross products $\vec{v}_1 := p\vec{a}_1 \times p\vec{a}_2$, $\vec{v}_2 := p\vec{a}_2 \times p\vec{a}_3$ and $\vec{v}_3 := p\vec{a}_3 \times p\vec{a}_1$. Note that all of them are perpendicular to the triangle's plane.
- ▶ p is inside the triangle if and only if v_1 , v_2 and v_3 point in the same direction, i.e. the three dot products $\vec{v}_1 \cdot \vec{v}_2$, $\vec{v}_2 \cdot \vec{v}_3$, $\vec{v}_3 \cdot \vec{v}_1$ are all nonnegative.

Ray-primitive intersections: accuracy

- ▶ Ray parallel (or very close to parallel) to the triangle's plane
 - ▶ we might have no solution of our system of equations or infinitely many solutions
 - ▶ the exact and numerically computed values of the determinant of the matrix of the system, i.e.

$$\begin{bmatrix} a_{1x} & a_{2x} & a_{3x} & -d_x \\ a_{1y} & a_{2y} & a_{3y} & -d_y \\ a_{1z} & a_{2z} & a_{3z} & -d_z \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

may be of opposite signs, or the numerically computed value may be zero → weird solution or problems with arithmetic (division by zero)

- ▶ Ray may intersect the triangle's plane very close to the boundary of the triangle
 - ▶ signs of α , β or γ may come out wrong because of numerical precision issues
 - ▶ you may report no intersection when there is one or the other way around

Ray-primitive intersections: accuracy

- ▶ Similar problems for ray-sphere intersection (for rays close to tangent to the sphere or if origin is close to the sphere's surface)
- ▶ Not a big deal in a simple ray tracer described here (although it could make your program crash or generate a bad image if the input is really nasty, depending on implementation)

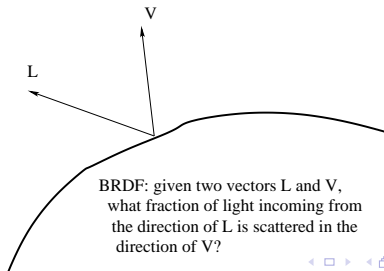
Local Illumination Models

Andrzej Szymczak

August 23, 2011

Illumination

- ▶ How do we compute the amount of light scattered at a point p on a primitive toward the viewpoint?
- ▶ It depends on surface properties (material the primitive is made of)
- ▶ Need a model describing how a material scatters light
- ▶ The most general form of such model is a BRDF (Bidirectional Reflectance Distribution Function), but we'll keep things simple here and focus on one special form of the BRDF (Phong model)

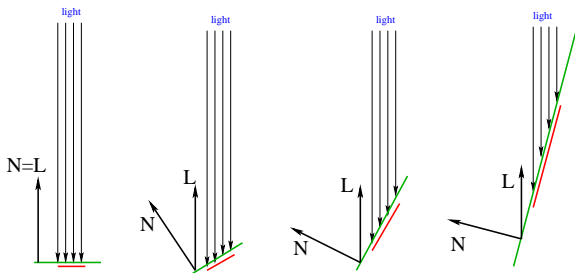


Diffuse surfaces

- ▶ The brightness of a point on a diffuse surface (under fixed lighting conditions) does not depend on where you look at it from.
- ▶ Good model for matte surfaces
- ▶ What happens at a point on a diffuse surface?
 - ▶ Light is arriving at this point, some fraction k of this light is scattered back into the environment and the rest is absorbed (turned into heat)
 - ▶ Brightness of the surface is proportional to the *amount of light per unit surface area* received by the surface and to k

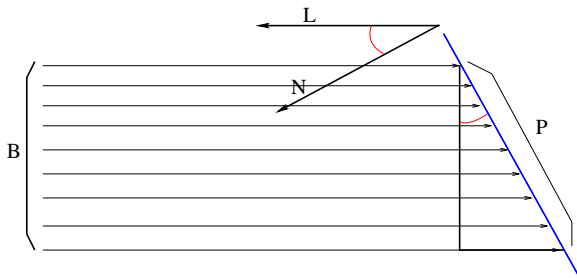
Diffuse surfaces

- ▶ Take a beam of light hitting a surface
- ▶ Amount of light hitting a unit surface area is proportional to $\cos \angle(N, L) = N \cdot L$
 - N is the unit vector normal to surface
 - L is the unit vector pointing toward the light source



Diffuse surfaces

- ▶ P (area hit by the beam) is proportional inversely proportional to $N \cdot L$
- ▶ Energy per unit area received from the beam is proportional to $N \cdot L$



$$B:P = L \cdot N, \text{ so } P \sim 1/(L \cdot N)$$

► Diffuse illumination term

$$I_d = k_d I (N \cdot L)$$

k_d diffuse coefficient

N unit normal

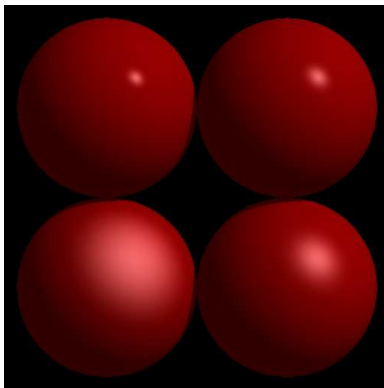
L unit vector pointing toward light source

I intensity of the light source

I_d amount of light scattered in the diffuse manner (same in any direction that makes sense, i.e. from which the point of interest is visible – by definition of diffuse surface)

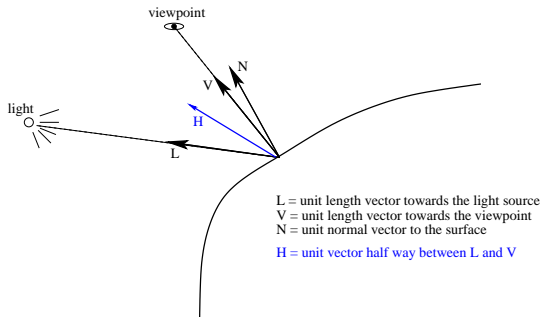
Specular term

- ▶ Diffuse term is nice, but not all surfaces are matte
- ▶ Shiny surfaces typically scatter more light in the direction symmetric to L about N than in other directions



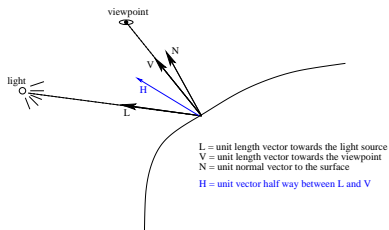
Specular term

- ▶ The standard specular term is not physics-based (just a well-working hack)



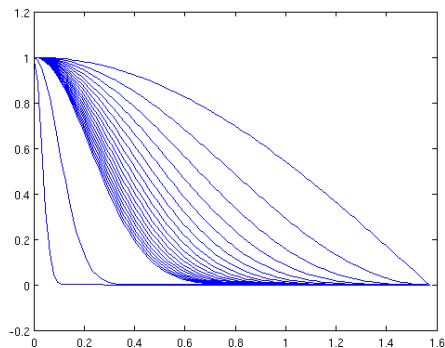
- ▶ Want to maximize the amount of light scattered toward viewpoint if $H = N$.
- ▶ Make it proportional to $(H \cdot N)^n$, where $n \geq 0$ is a parameter (called *specular exponent*)

Specular term



- ▶ The half-way vector is $H = \frac{L+V}{|L+V|}$
- ▶ Specular exponent controls how fast the amount of scattered light decreases as H moves away from N (faster if n is larger)

Specular term



- $(H \cdot N)^n$ as a function of $\angle(H, N)$; plot shows graphs for $n = 1, 2 \dots 20, 100$ and 1000.

Specular term: formula

$$I_s = k_s I (N \cdot H)^n$$

where:

k_s is the specular coefficient (roughly: how much light is scattered in specular manner?)

N is the unit normal vector

I is the intensity of the light source

n is the specular exponent (nonnegative number)

H is the half-way vector $H = \frac{V+L}{|V+L|}$, where

L is the unit vector pointing toward light source

V is the unit vector pointing toward the viewpoint

Ambient term

- ▶ What about multiply scattered light?
- ▶ **Very** hard to compute exactly; thus use another hack: make it $k_a I_a$, where
 - k_a is the ambient term (surface property)
 - I_a is the ambient light intensity (global constant)
- ▶ In particular, this means that if a point is in shadow then its intensity is equal to $k_a I_a$

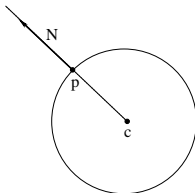
Illumination formula

$$I_{\text{total}} = I(k_d(N \cdot L) + k_s(H \cdot N)^n) + k_a I_a$$

- ▶ k_d, k_a, k_s, n are *surface* properties
- ▶ I_a is a global constant (ambient light intensity)
- ▶ I is a light source property (intensity)
- ▶ N, L and H are derived from geometry (viewpoint and light source locations and the primitives)

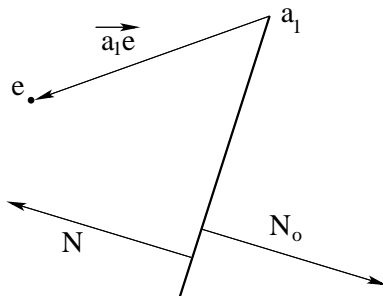
Normal vector: sphere

- ▶ Unit normal at a point p is equal to $\frac{\vec{cp}}{|\vec{cp}|}$ where c is the center of the sphere



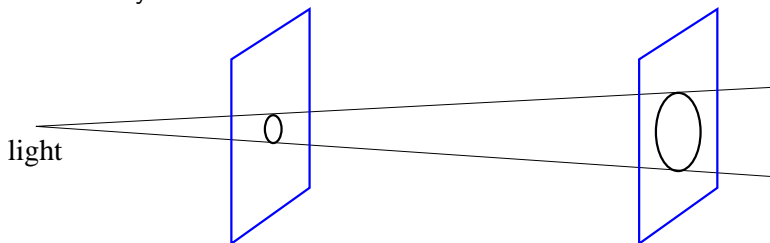
Normal vector: triangle

- ▶ The vector $\vec{N}_0 = \frac{\vec{a}_1 \vec{a}_2 \times \vec{a}_1 \vec{a}_3}{|\vec{a}_1 \vec{a}_2 \times \vec{a}_1 \vec{a}_3|}$ is a unit vector perpendicular to the triangle's surface, but so is $-\vec{N}_0$; Which of them should be used as \vec{N} ?
- ▶ It depends on which side of the triangle we are looking at
- ▶ \vec{N} should point to the side containing the viewpoint (below, e stands for the viewpoint location)
- ▶ If $\vec{N}_0 \cdot \vec{a}_1 e \geq 0$ then set $\vec{N} := \vec{N}_0$; otherwise use $\vec{N} := -\vec{N}_0$



Light source attenuation

- ▶ Things get darker as they move away from the light source
- ▶ Here is why:



- ▶ Energy emitted into the cone is distributed over a larger area for the surface farther from the light source
- ▶ Area is proportional to the square of the distance d from the surface to the light source \Rightarrow brightness of the surface is *inversely* proportional to d^2

Illumination formula with attenuation

- ▶ Including the attenuation term yields

$$I_{\text{total}} = \frac{I}{d^2} (k_d(N \cdot L) + k_s(H \cdot N)^n) + k_a I_a$$

where d is the distance from the light source to the surface

- ▶ In practice this term decays too fast:
 - ▶ Take two surfaces A and B of the same properties and roughly the same orientation with respect to light source, one 1 and one 20 units away from the light
 - ▶ A is about 400 times brighter than B ; how to display the colors correctly if intensities are to fall between 0 and 255?
 - ▶ Not impossible to do (HRD=high dynamic range images are becoming increasingly popular), but tricky
- ▶ Usually, attenuation term of the form $\frac{1}{c_2 d^2 + c_1 d + c_0}$ is used instead of $\frac{1}{d^2}$; for example, one may make light intensity decay linearly rather than quadratically in d by setting $c_2 = 0$.

- ▶ To get color images, evaluate the illumination formula for three color components (red, green, blue)
- ▶ Not much new here...

$$I_{\text{total,red}} = \frac{I_{\text{red}}}{c_2 d^2 + c_1 d + c_0} (k_{d,\text{red}}(N \cdot L) + k_{s,\text{red}}(H \cdot N)^n) + k_{a,\text{red}} I_{a,\text{red}}$$

$$I_{\text{total,green}} = \frac{I_{\text{green}}}{c_2 d^2 + c_1 d + c_0} (k_{d,\text{green}}(N \cdot L) + k_{s,\text{green}}(H \cdot N)^n) + k_{a,\text{green}} I_{a,\text{green}}$$

$$I_{\text{total,blue}} = \frac{I_{\text{blue}}}{c_2 d^2 + c_1 d + c_0} (k_{d,\text{blue}}(N \cdot L) + k_{s,\text{blue}}(H \cdot N)^n) + k_{a,\text{blue}} I_{a,\text{blue}}$$

Multiple lights

- ▶ Just sum the diffuse and specular terms over all light sources:

$$I_{\text{total}} =$$

$$\left[\sum_{l \in \text{LightSources}} \frac{I_l}{c_{2,l}d^2 + c_{1,l}d + c_{0,l}} (k_d(N \cdot L_l) + k_s(H_l \cdot N)^n) \right] +$$
$$+ k_a I_a$$

Ray tracing: overview

Andrzej Szymczak

August 23, 2011

Input and output, structure

► Input

- View: viewpoint coordinates, screen
- Virtual world: light source('s) location(s) and intensity(ies), geometric primitives and their material properties (k_d, k_a, k_s and specular exponent n), ambient light intensity I_a , attenuation coefficients c_0, c_1, c_2

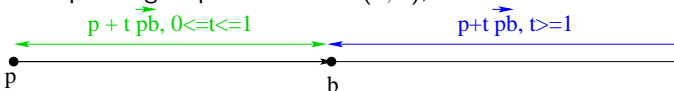
► Output

- A nice image
- Run the same procedure for each pixel to determine its color:
 - Find closest intersection p point of the eye ray and the primitives (what is visible through the pixel?)
 - Figure out if p is in shadow or not (find out if the closest intersection of the shadow ray is between the light source and p)
 - If p is in shadow, use just the ambient term to compute color for the pixel; otherwise, evaluate the full illumination formula

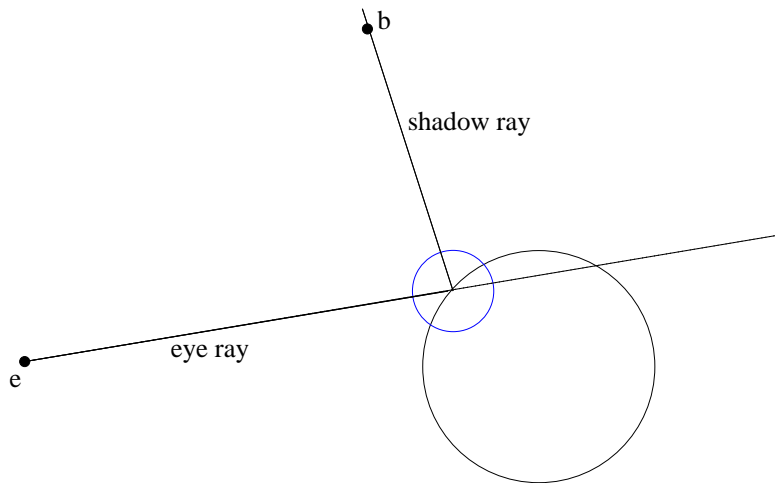
What is visible through the pixel?

- ▶ Eye ray:
 - ▶ The origin is at the viewpoint (e)
 - ▶ The direction vector is the vector \vec{eq} , where q is the center of the pixel
- ▶ Find closest intersection of the eye ray and the primitives
 - ▶ For every primitive, calculate the t value corresponding to the closest intersection point with that primitive
 - ▶ Keep track of the smallest t
 - ▶ For some primitives, there may be no intersection points (ray misses the primitive)
 - ▶ If t_* is the smallest t value, the intersection point p is given by $e + t_*\vec{eq}$
 - ▶ Sometimes, there is no intersection at all; in this case, color the pixel with some “background” color

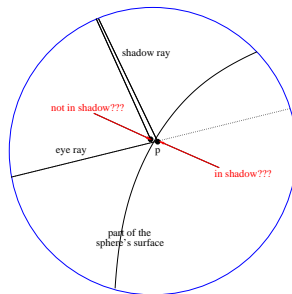
Is p in shadow?

- ▶ Shadow ray:
 - ▶ The origin is at p
 - ▶ The direction vector is \vec{pb} where b is the location of the light source
 - ▶ Is there an intersection point of the shadow ray and any of the primitives between p and b ?
 - ▶ Loop over all primitives
 - ▶ If the shadow ray intersects any primitive at a point corresponding to parameter $t \in (0, 1)$, return **'in shadow'**
- 
- ▶ Return **'not in shadow'** if no such intersection exists
- ▶ There is a problem to take care of here: p is on a primitive, but numerically computed p may not be!

Is p in shadow? Example



Is p in shadow? Example



- ▶ The numerically computed p may be slightly inside or outside the sphere
- ▶ If it is inside, the shadow ray intersects the sphere for some $t \in (0, 1)$; if it is outside, it doesn't
- ▶ This is BAD: some points on lit part of sphere are going to be treated as if in shadow \rightarrow noisy image (some pixels bright, some dark)

Is p in shadow? A way around roundoff errors

- ▶ Treat the primitive P containing p differently!
- ▶ Is p in shadow? Proceed in two steps.
 - (i) Is p in shadow of P ? If yes, return 'in shadow'.
 - (ii) Is p in shadow of any of the other primitives? If yes, return 'in shadow'; otherwise, return 'not in shadow'.
- ▶ We know how to deal with (ii) – use ray-primitive intersection procedure as described in the “Is p in shadow?” slide
- ▶ p is in shadow of its own primitive if and only if $N \cdot \vec{pb} < 0$ (angle between normal and vector toward the light source is greater than 90 degrees); note that there is no need to use normalized normal here