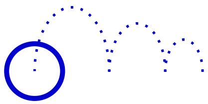
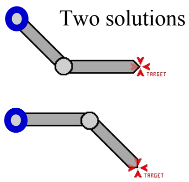
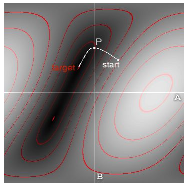
Ray Tracing

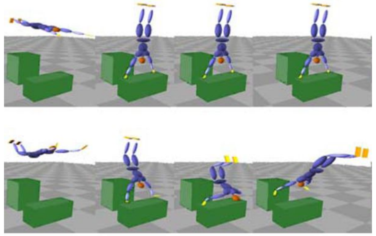
Last Time?

- Keyframing
- Procedural Animation
- Physically-Based Animation
- Forward and Inverse Kinematics
- Motion Capture









Reading for Today

"Artist-Directed Dynamics for 2D Animation", Bai, Kaufman, Liu, & Popović, SIGGRAPH 2016

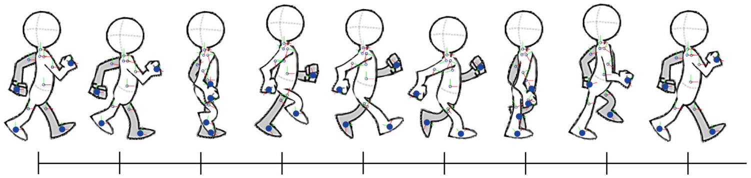


Figure 6: ***Keyframes used in the articulated character walk example.*** *The artist only specifies keyframes for a subset of handles (handles at hands and feet) which are shown as blue dots. Nine keyframes are used to create a walking cycle. Their timing is visualized by the black lines at the bottom. The artworks are adapted from Angryanimator.com (http://www.angryanimator.com/)*

Reading for Today

"Articulated Swimming Creatures" Jie Tan, Yuting Gu, Greg Turk, and C. Karen Liu, SIGGRAPH 2011

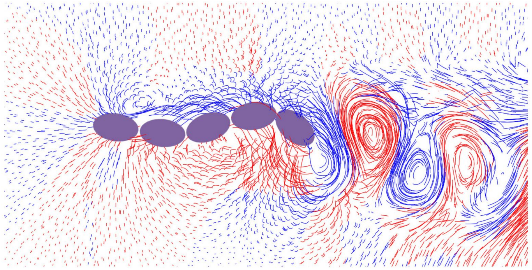


Figure 8: *A five-link eel swims in a 2D fluid environment. In contrast to the simulation in 3D, an eel swimming in 2D fluid sheds only one single vortex street. Red traces show the counter-clockwise vortices while blue traces show the clockwise vortices.*

http://www.cc.gatech.edu/~jtan34/project/articulatedSwimmingCreatures.html

Reading for Today

"Flexible Muscle-Based Locomotion for Bipedal Creatures", Geijtenbeek, van de Panne, van der Stappen, SIGGRAPH Asia 2013

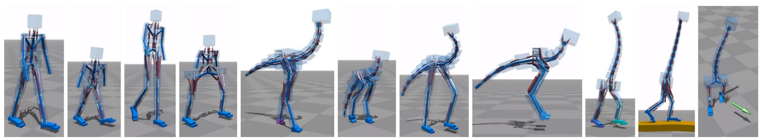


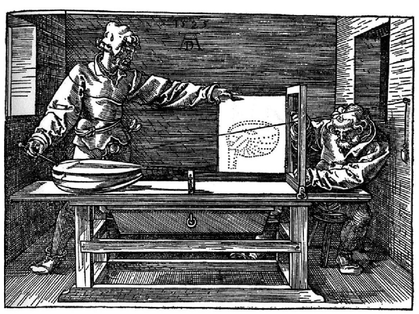
Figure 1: *Physics-based simulation of locomotion for a variety of creatures driven by 3D muscle-based control. The synthesized controllers can locomote in real time at a range of speeds, be steered to a target heading, and can traverse variable terrain.*

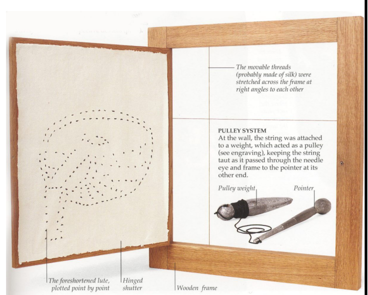
Today

- Ray Casting
  - Ray-Plane Intersection
  - Ray-Sphere Intersection
  - Point in Polygon
- Ray Tracing
- Recursive Ray Tracing
- Distribution Ray Tracing

Durer's Ray Casting Machine

- Albrecht Durer, 16th century

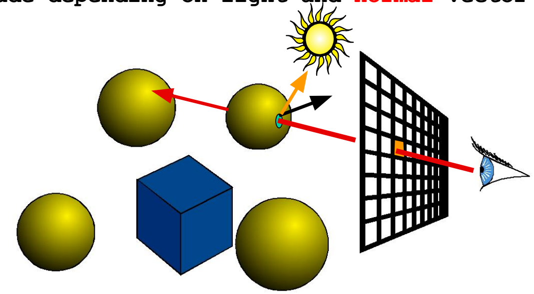




Ray Casting

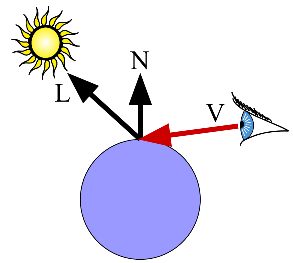
```
For every pixel
   Construct a ray from the eye
   For every object in the scene
        Find intersection with the ray
        Keep if closest
   Shade depending on light and normal vector
```

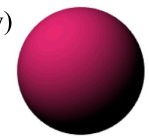
Finding the intersection and normal is the central part of ray casting



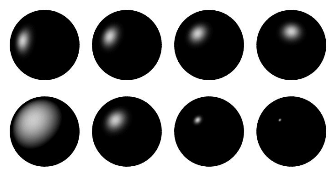
A Note on *Local* Shading

- Surface/Scene Characteristics:
  - surface normal
  - direction to light
  - viewpoint
- Material Properties
  - color/texture
  - diffuse (matte)
  - specular (shiny)
  - …
- More later!





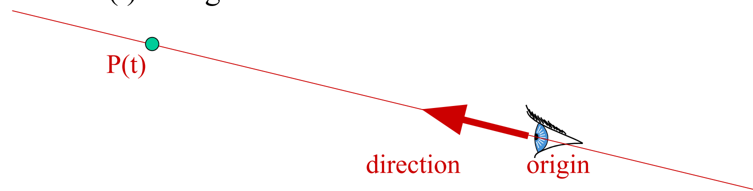
Diffuse sphere



Specular spheres

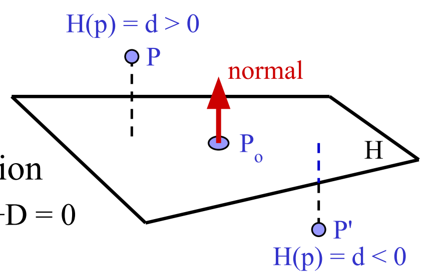
Ray Representation?

- Two vectors:
  - Origin
  - Direction (normalized is better)
- Parametric line (*explicit* representation)
  - P(t) = origin + t * direction



3D Plane Representation?

- Plane defined by
  - $P_o = (x,y,z)$
  - $n = (A,B,C)$
- *Implicit* plane equation
  - $H(P) = Ax+By+Cz+D = 0$
    $= n \cdot P + D = 0$
- Point-Plane distance?
  - If n is normalized, distance to plane, d = H(P)
  - d is the *signed distance*!



Explicit vs. Implicit?

- Ray equation is explicit $P(t) = R_o + t * R_d$
  - Parametric
  - Generates points
  - Harder to verify that a point is on the ray
- Plane equation is implicit $H(P) = n \cdot P + D = 0$
  - Solution of an equation
  - Does not generate points
  - Verifies that a point is on the plane

Ray-Plane Intersection

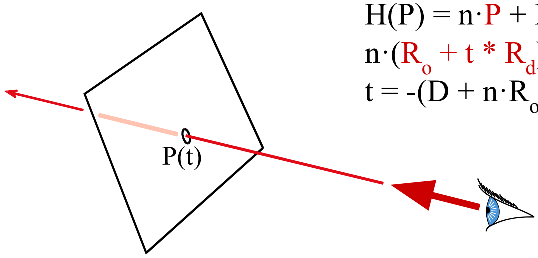
- Intersection means both are satisfied
- So, insert explicit equation of ray into implicit equation of plane & solve for t

$P(t) = R_o + t * R_d$

$H(P) = n \cdot P + D = 0$

$n \cdot (R_o + t * R_d) + D = 0$

$t = -(D + n \cdot R_o) / n \cdot R_d$



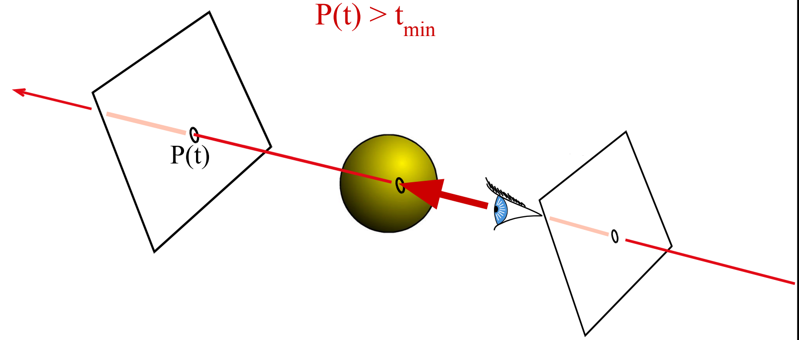
Additional Housekeeping

- Verify that intersection is closer than previous

$P(t) < t_{current}$

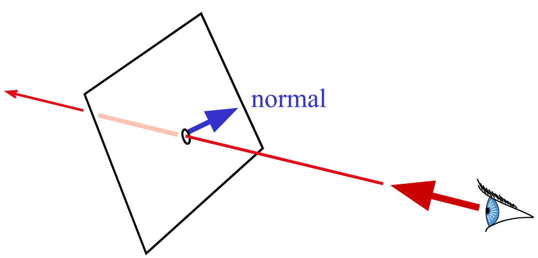
- Verify that it is not out of range (behind eye)

$P(t) > t_{min}$



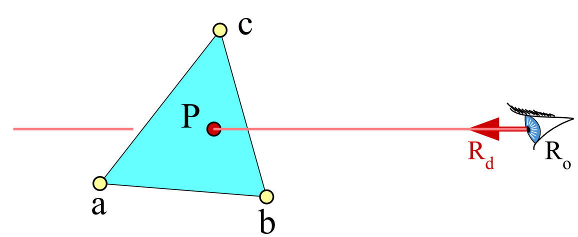
Normal at Surface Intersection

- Needed for shading
  - diffuse: dot product between light and normal
- Normal of a plane is constant!



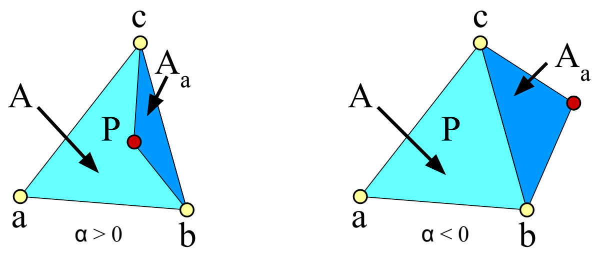
Ray-Triangle Intersection

- Intersect with the plane…
- Then use barycentric coordinates:
  - $P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$ with $\alpha + \beta + \gamma = 1$
  - If $0 < \alpha < 1$ & $0 < \beta < 1$ & $0 < \gamma < 1$ then the point is inside the triangle!



How Do We Compute α, β, γ ?

- Ratio of opposite sub-triangle area to total area
  - $\alpha = A_a/A \quad \beta = A_b/A \quad \gamma = A_c/A$
- Use signed areas for points outside the triangle



But how do I know if the point is outside the triangle? That's what I was trying to determine!

Using Cramer's Rule…

- Used to solve for one variable at a time in system of equations

$$\beta = \frac{\begin{vmatrix} a_x - R_{ox} & a_x - c_x & R_{dx} \\ a_y - R_{oy} & a_y - c_y & R_{dy} \\ a_z - R_{oz} & a_z - c_z & R_{dz} \end{vmatrix}}{|A|} \qquad \gamma = \frac{\begin{vmatrix} a_x - b_x & a_x - R_{ox} & R_{dx} \\ a_y - b_y & a_y - R_{oy} & R_{dy} \\ a_z - b_z & a_z - R_{oz} & R_{dz} \end{vmatrix}}{|A|}$$

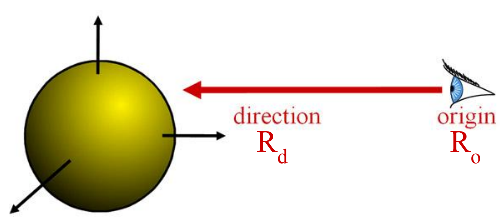
$$t = \frac{\begin{vmatrix} a_x - b_x & a_x - c_x & a_x - R_{ox} \\ a_y - b_y & a_y - c_y & a_y - R_{oy} \\ a_z - b_z & a_z - c_z & a_z - R_{oz} \end{vmatrix}}{|A|}$$

| | denotes the determinant

Can be copied mechanically into code

Sphere Representation?

- Implicit sphere equation
  - Assume centered at origin (easy to translate)
  - $H(P) = P \cdot P - r^2 = 0$



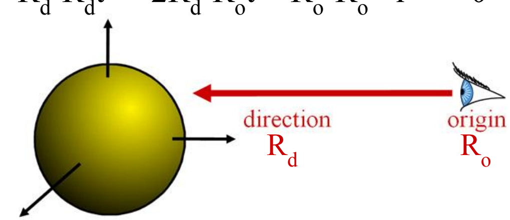
Ray-Sphere Intersection

- Insert explicit equation of ray into implicit equation of sphere & solve for t

$$P(t) = R_o + t*R_d \qquad H(P) = P \cdot P - r^2 = 0$$

$$(R_o + tR_d) \cdot (R_o + tR_d) - r^2 = 0$$

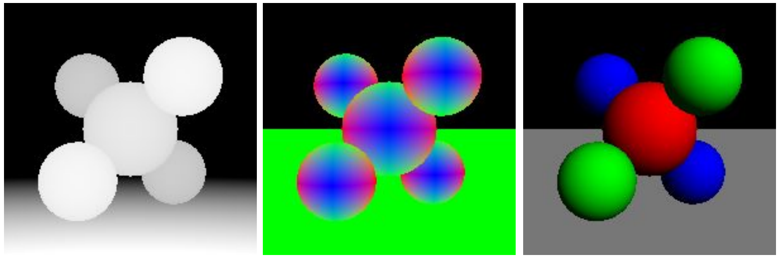
$$R_d \cdot R_d t^2 + 2R_d \cdot R_o t + R_o \cdot R_o - r^2 = 0$$



Ray-Sphere Intersection

- Quadratic: $at^2 + bt + c = 0$
  - $a = 1$ (remember, $\|R_d\| = 1$)
  - $b = 2R_d \cdot R_o$
  - $c = R_o \cdot R_o - r^2$
- with discriminant $d = \sqrt{b^2 - 4ac}$
- and solutions $t_\pm = \frac{-b \pm d}{2a}$
- What does it mean if there are no solutions, 1 solution, or 2 solutions?

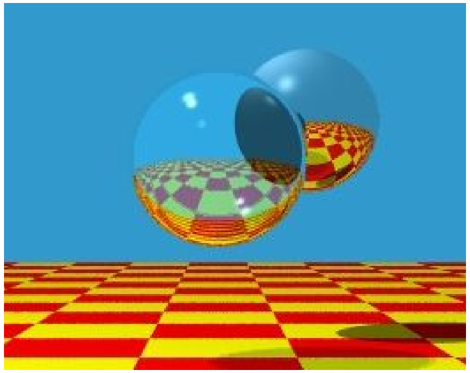
Questions?



depth normals local shading

Reading for Next Friday

- "An improved illumination model for shaded display" Turner Whitted, 1980.

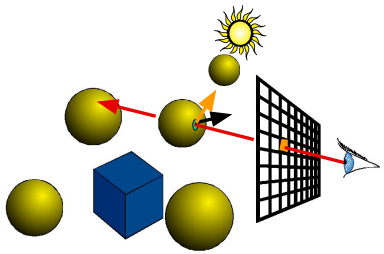


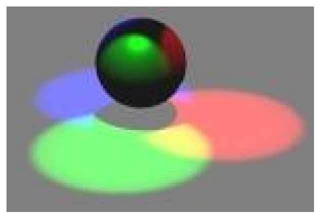
Today

- Ray Casting
- Ray Tracing
  - Shadows
  - Reflection
  - Refraction
- Recursive Ray Tracing
- Distribution Ray Tracing

How Can We Add Shadows?

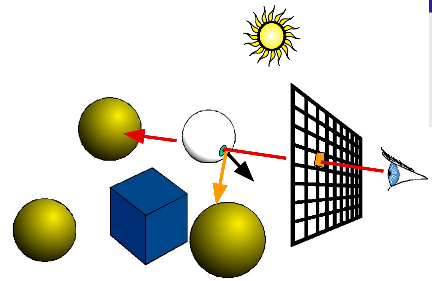
```
Find the point to be shaded
For every light,
  Construct ray from point to light
  For every object
    find intersection of ray with object
  If no objects between point and light
    Add contribution from light
```

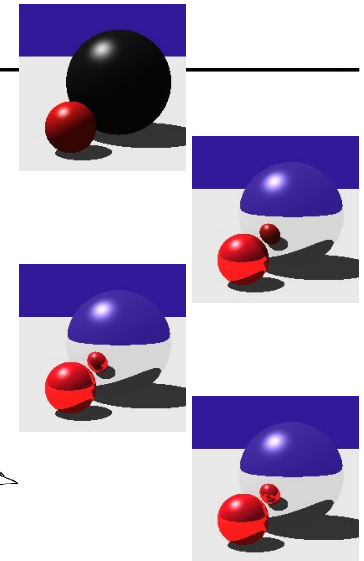




Mirror Reflection

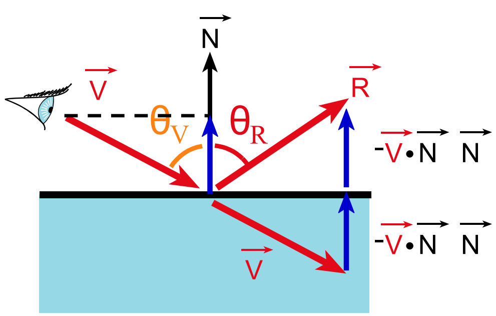
- Cast ray symmetric with respect to the normal
- Multiply by reflection coefficient (color)





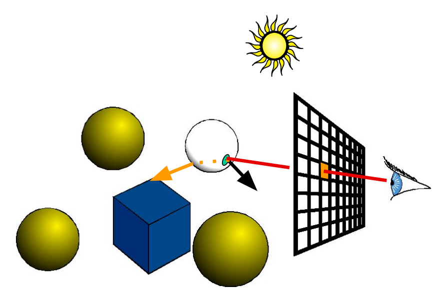
Reflection

- Reflection angle = view angle
- $\mathbf{R} = \mathbf{V} - 2\,(\mathbf{V} \cdot \mathbf{N})\,\mathbf{N}$

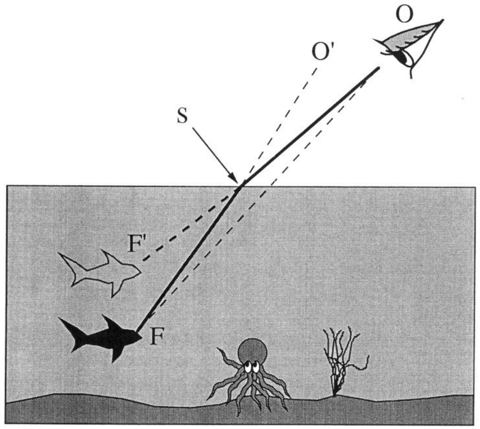


Transparency

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



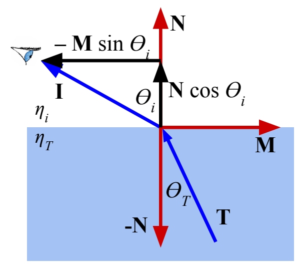
Qualitative Refraction



From "Color and Light in Nature" by Lynch and Livingston

Refraction

Note: The math works the same tracing the ray either "forwards" or "backwards", but it's really easy to get confused and have get a sign error in the direction.



$$\mathbf{I} = \mathbf{N}\cos\Theta_i - \mathbf{M}\sin\Theta_i$$
$$\mathbf{M} = (\mathbf{N}\cos\Theta_i - \mathbf{I}) / \sin\Theta_i$$

$$\begin{aligned}
\mathbf{T} &= -\mathbf{N}\cos\Theta_T + \mathbf{M}\sin\Theta_T \\
&= -\mathbf{N}\cos\Theta_T + (\mathbf{N}\cos\Theta_i - \mathbf{I})\sin\Theta_T / \sin\Theta_i \\
&= -\mathbf{N}\cos\Theta_T + (\mathbf{N}\cos\Theta_i - \mathbf{I})\,\eta_r \\
&= [\,\eta_r\cos\Theta_i - \cos\Theta_T]\,\mathbf{N} - \eta_r\,\mathbf{I} \\
&= [\,\eta_r\cos\Theta_i - \sqrt{(1 - \sin^2\Theta_T)}]\,\mathbf{N} - \eta_r\,\mathbf{I} \\
&= [\,\eta_r\cos\Theta_i - \sqrt{(1 - \eta_r^2\sin^2\Theta_i)}]\,\mathbf{N} - \eta_r\,\mathbf{I} \\
&= [\,\eta_r\cos\Theta_i - \sqrt{(1 - \eta_r^2(1 - \cos^2\Theta_i))}]\,\mathbf{N} - \eta_r\,\mathbf{I} \\
&= [\,\eta_r(\mathbf{N}\cdot\mathbf{I}) - \sqrt{(1 - \eta_r^2(1 - (\mathbf{N}\cdot\mathbf{I})^2))}]\,\mathbf{N} - \eta_r\,\mathbf{I}
\end{aligned}$$

Snell-Descartes Law:

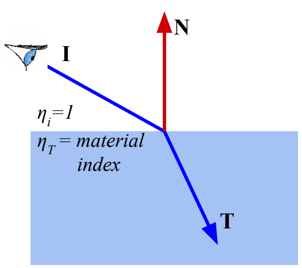
$$\eta_i \sin\Theta_i = \eta_T \sin\Theta_T$$

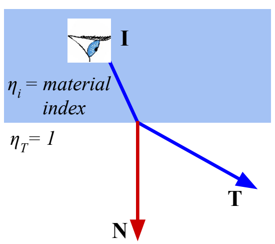
$$\frac{\sin\Theta_T}{\sin\Theta_i} = \frac{\eta_i}{\eta_T} = \eta_r$$

- **Total internal reflection when the square root is imaginary**
- **Don't forget to normalize!**

Refraction & the Sidedness of Objects

- Make sure you know whether you're entering or leaving the transmissive material:





- What about intersecting transparent objects?

Total Internal Reflection

Fig. 3.7A The optical manhole. From under water, the entire celestial hemisphere is compressed into a circle only 97.2° across. The dark boundary defining the edges of the manhole is not sharp due to surface waves. The rays are analogous to the crepuscular type seen in hazy air, Section 1.9. (Photo by D. Granger)

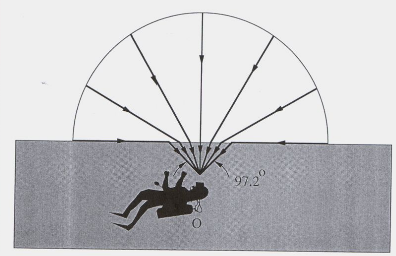


Fig. 3.7B The optical manhole. Light from the horizon (angle of incidence = 90°) is refracted downward at an angle of 48.6°. This compresses the sky into a circle with a diameter of 97.2° instead of its usual 180°.

From "Color and Light in Nature" by Lynch and Livingston

Questions?

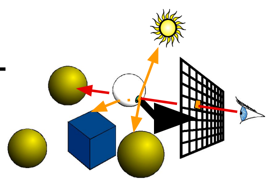
Today

- Ray Casting
- Ray Tracing
- Recursive Ray Tracing
- Distribution Ray Tracing

Ray Tracing

```
trace ray
    Intersect all objects
    color = ambient term
    For every light
        cast shadow ray
        color += local shading term
    If mirror
        color += color_refl *
            trace reflected ray
    If transparent
        color += color_trans *
            trace transmitted ray
```

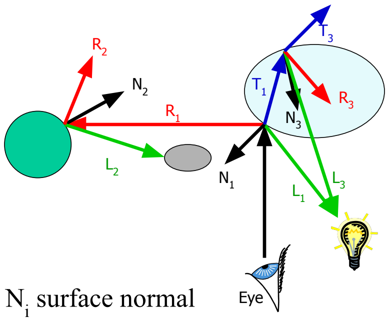
- ***Does it ever end?***

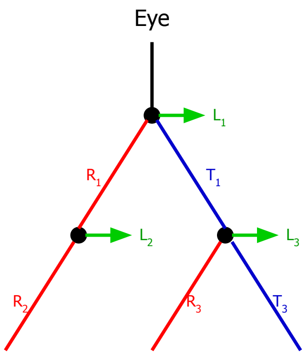


Stopping criteria:

- Recursion depth
  - Stop after a number of bounces
- Ray contribution
  - Stop if reflected / transmitted contribution becomes too small

The Ray Tree





N_i surface normal

R_i reflected ray

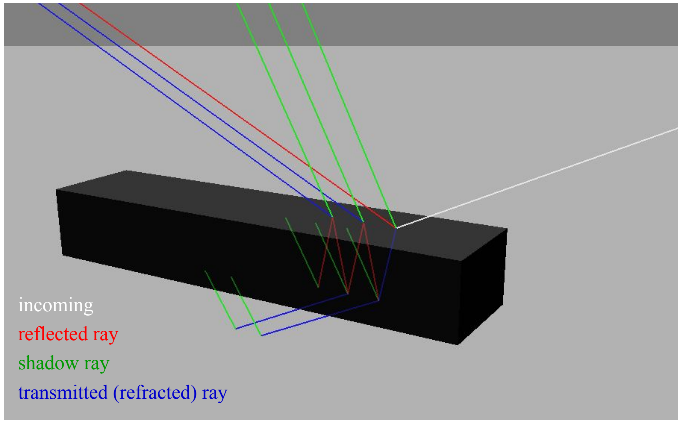
L_i shadow ray

T_i transmitted (refracted) ray

Complexity?

Ray Debugging

- Visualize the ray tree for single image pixel



Other Reading for Next Friday

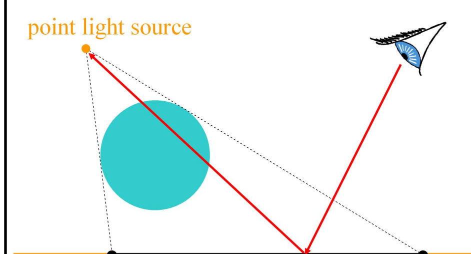
- "Distributed Ray Tracing", Cook, Porter, & Carpenter, SIGGRAPH 1984.

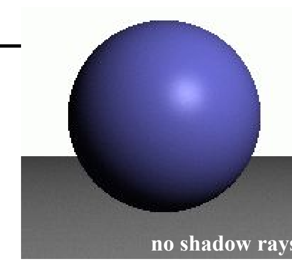
Today

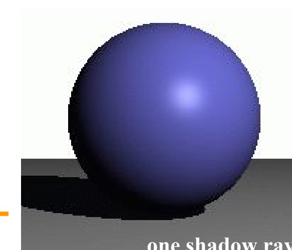
- Ray Casting
- Ray Tracing
- Recursive Ray Tracing
- Distribution Ray Tracing
  - Soft shadows
  - Antialiasing (getting rid of jaggies)
  - Glossy reflection
  - Motion blur
  - Depth of field (focus)

Shadows

- one shadow ray per intersection per point light source



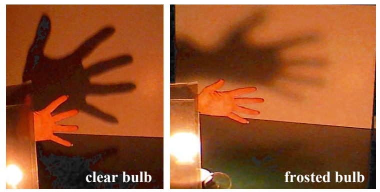




Shadows & Light Sources

http://3media.initialized.org/photos/2000-10-18/index_gall.htm

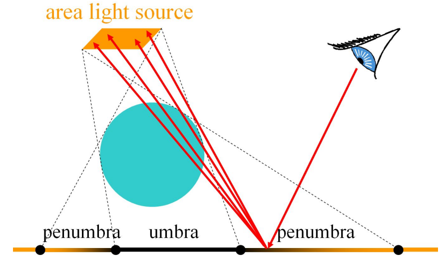
http://www.davidfay.com/index.php

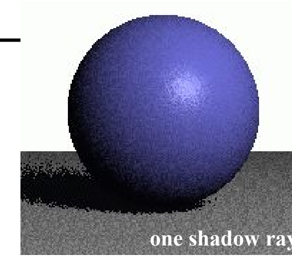


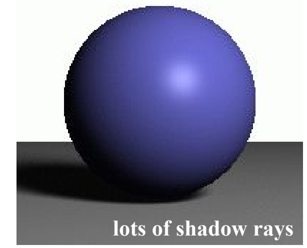
http://www.pa.uky.edu/~sciworks/light/preview/bulb2.htm

Soft Shadows

- multiple shadow rays to sample area light source







Antialiasing – Supersampling

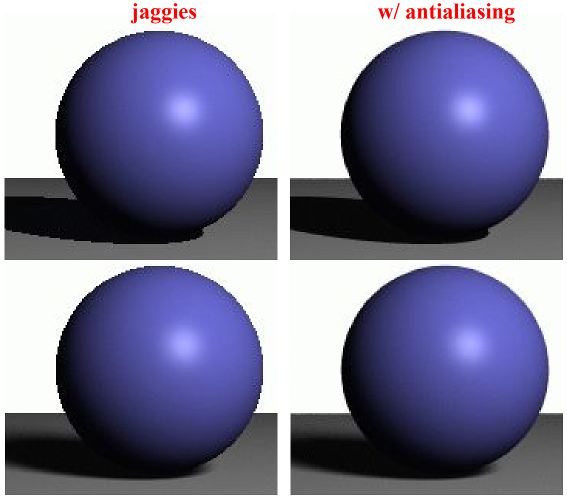
- multiple rays per pixel

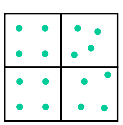
jaggies

w/ antialiasing

point light

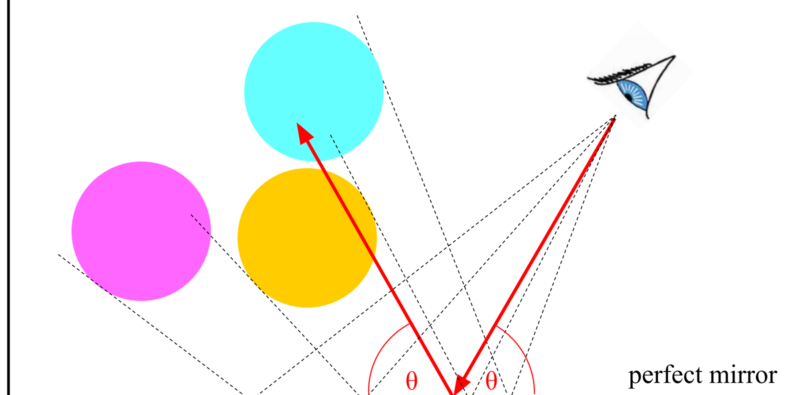
area light





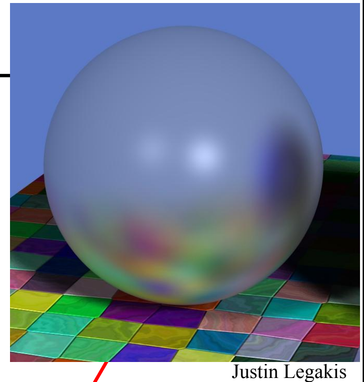
Reflection

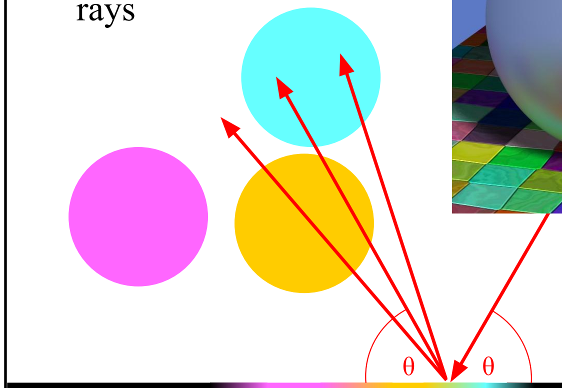
- one reflection ray per intersection



Glossy Reflection

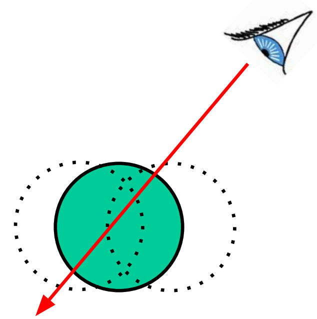
- multiple reflection rays





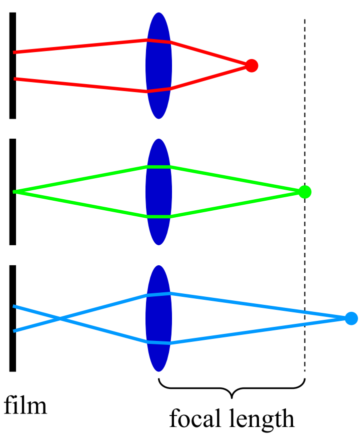
Motion Blur

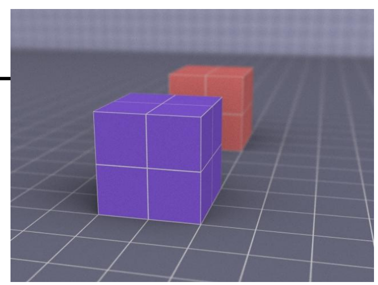
- Sample objects temporally

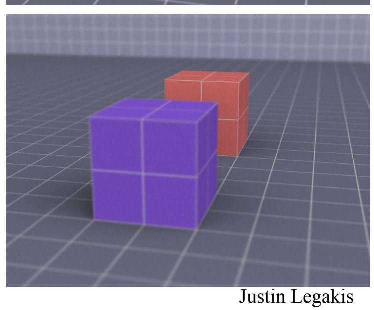


Depth of Field

- multiple rays per pixel







Ray Tracing Algorithm Analysis

- Ray casting
- Lots of primitives
- Recursive
- Distributed Ray Tracing Effects
  - – Soft shadows
  - – Anti-aliasing
  - – Glossy reflection
  - – Motion blur
  - – Depth of field

cost ≈ height * width *
num primitives *
intersection cost *
size of recursive ray tree *
num shadow rays *
num supersamples *
num glossy rays *
num temporal samples *
num focal samples *
. . .

can we reduce this?

these can serve double duty