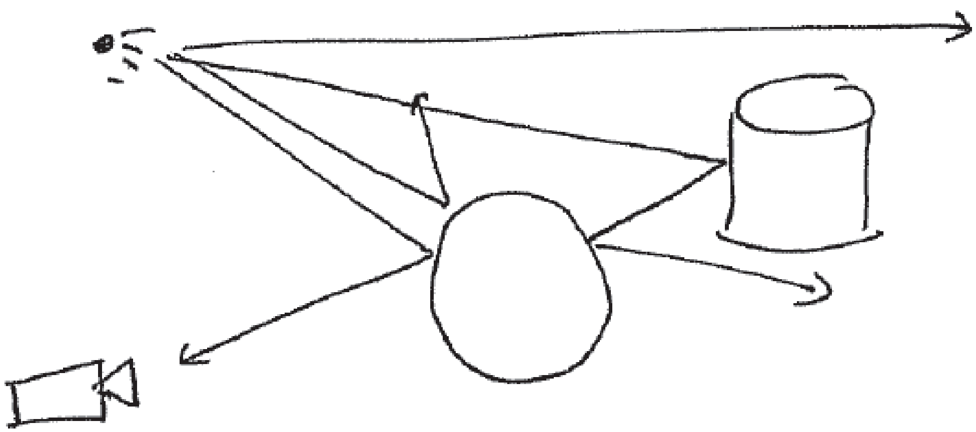
CS 428: Fall 2009

Introduction to Computer Graphics

Raytracing

“Forward” ray tracing

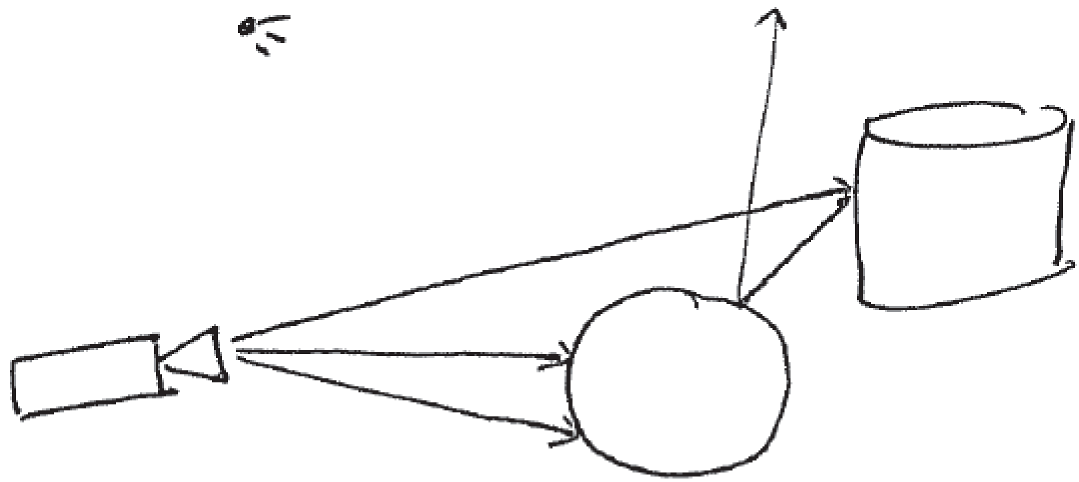
From the light sources



- Simulate light transport one ray at a time
  - Rays start from lights + bounce around the scene
  - Some hit the camera (extremely few!)
- Very expensive to compute
  - Under-sampling! So: nobody does this ☺

“Backward” ray tracing

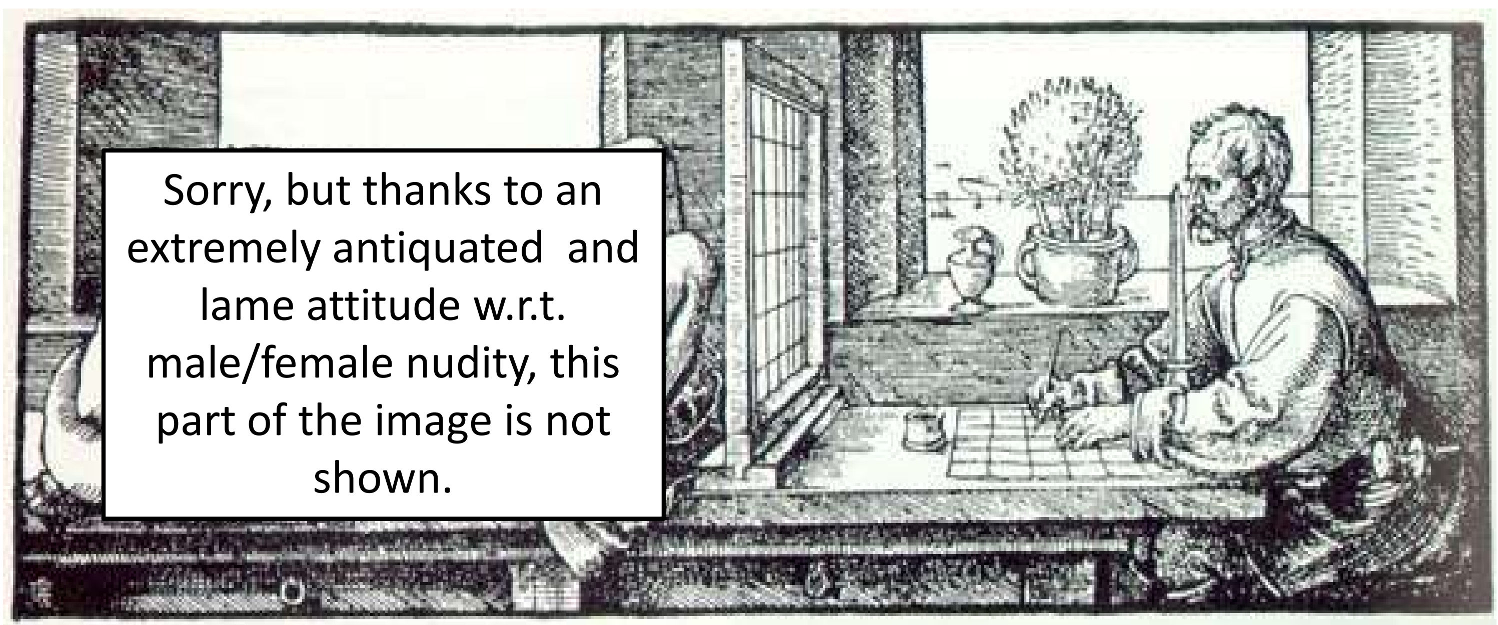
From the eye



- Use ray casting from camera to scene
  - Much faster (all rays say something about image)
- An approximation
  - Use known lighting models (diffuse, specular, etc)
  - No caustics, inter-object reflection
  - (Can be fixed/combined with other techniques…)

Raytracing

- Albrecht Dürer „Der zeichner des liegenden weibes“, 1538
  Impression of Velo von Alberti (1404 –1472)



Recursive raytracing

- Turner Whitted 1979: Model for Integrating
  - Reflection
  - Refraction
  - Visibility
  - Shadows
- (Basic) raytracing simulates the light transport and adheres to the rules of **ideal reflection** and **refraction**

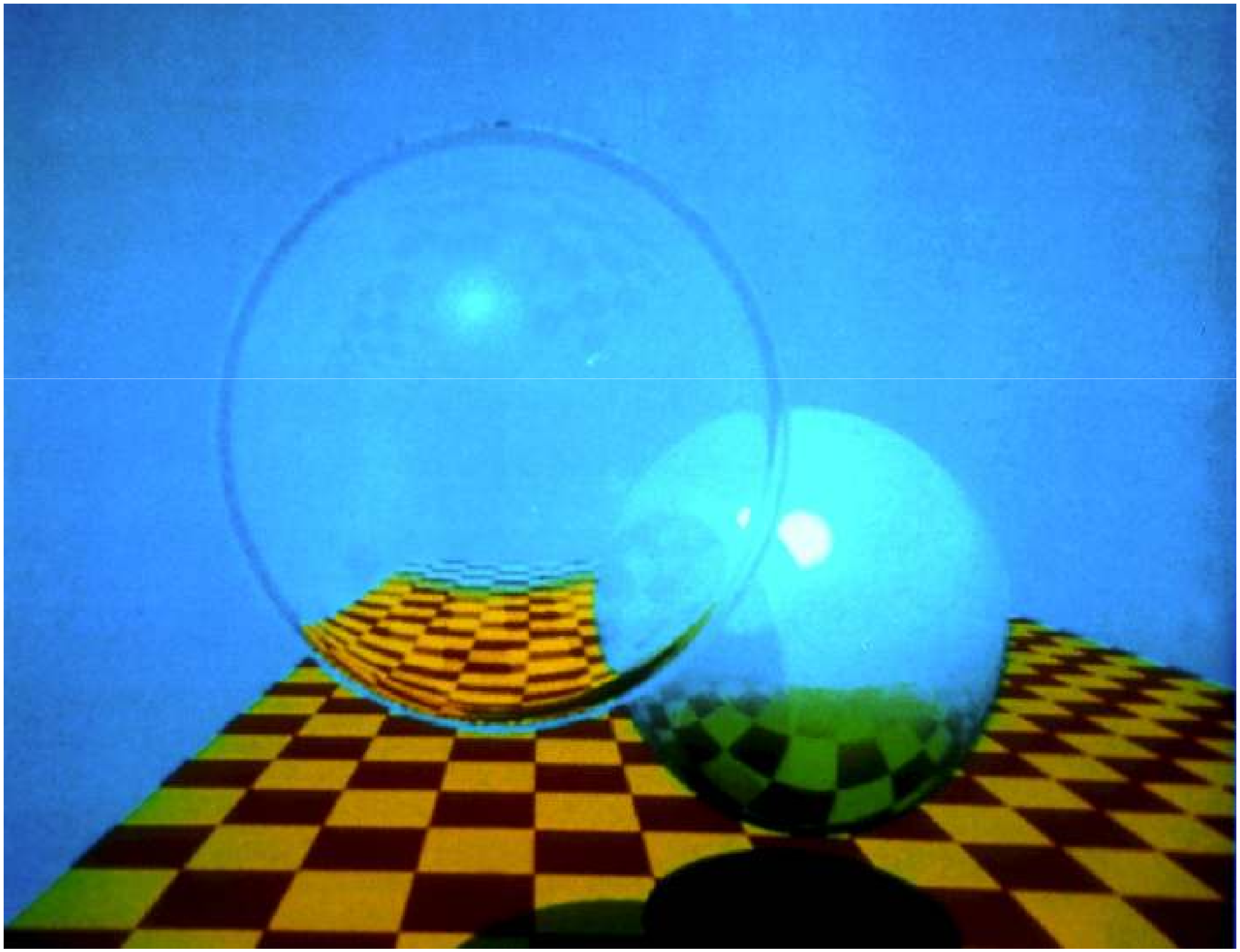
Recursive raytracing

Assumptions

- Point light sources
- Materials
  - Diffuse mit specular component (Phong model)
- Light transport
  - Occluding objects (Umbras, but no penumbras)
  - No light attenuation
  - Only specular light transport between surfaces (Rays are only followed along directions of ideal reflection)

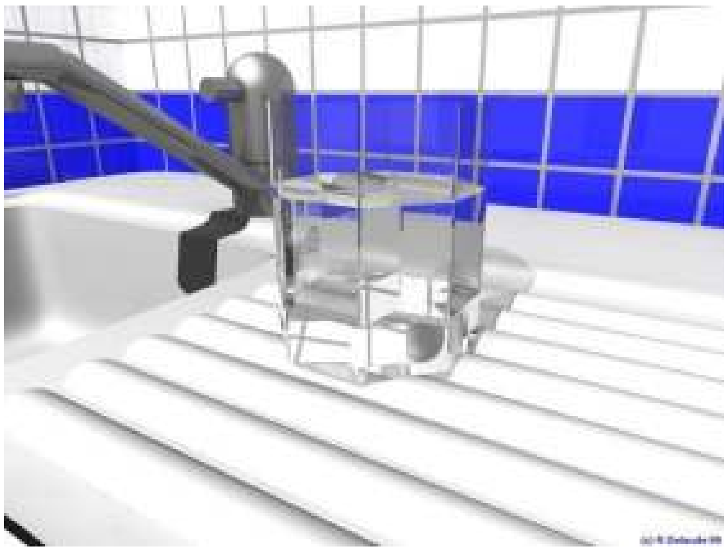
Recursive raytracing

Turner Whitted [1979]



Recursive raytracing

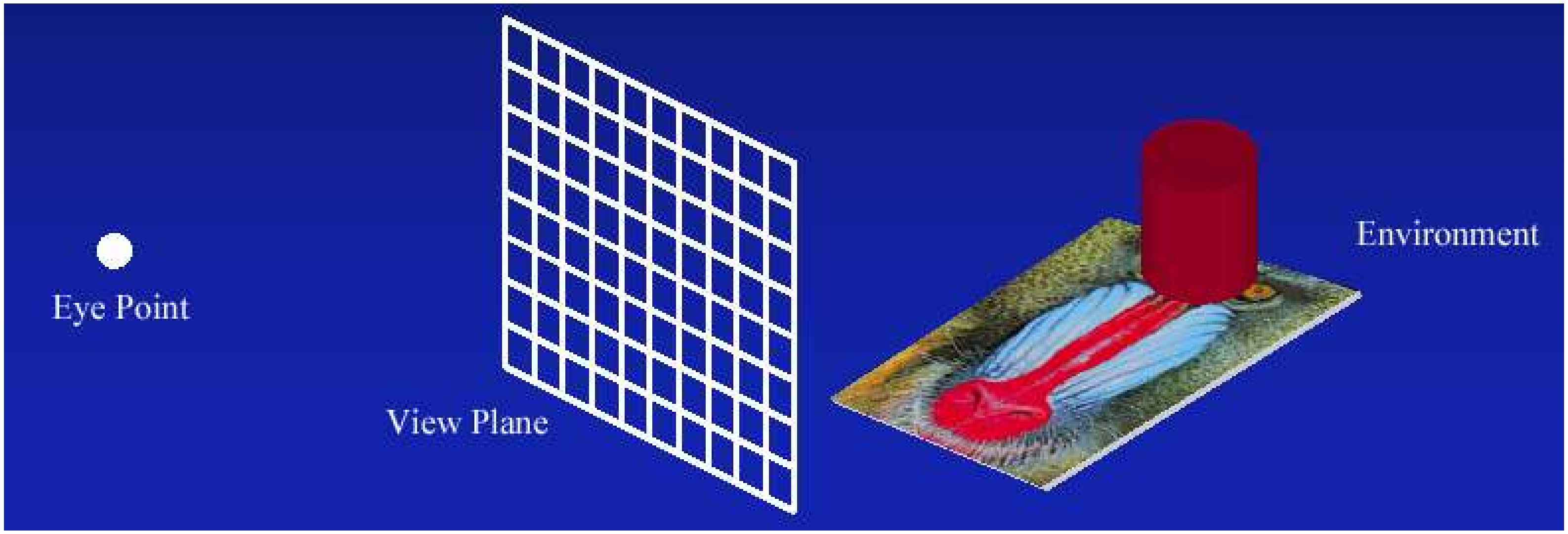
Examples



- Raytracing ist extremely suitable for scenes with many mirroring and transparent (refracting) surfaces

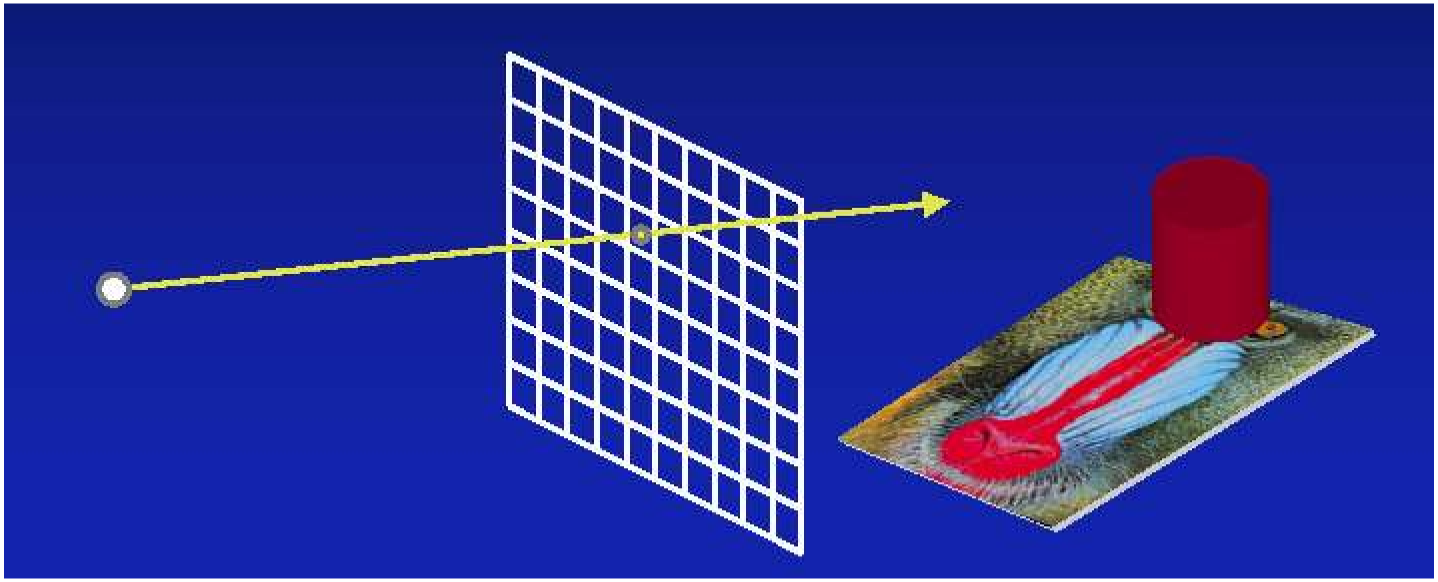
Recursive raytracing

- Synthetic camera
  - Defined by an eye point and image (view) plane in world coordinates
  - The image plane is an array of pixels, with the same resolution as the resulting image



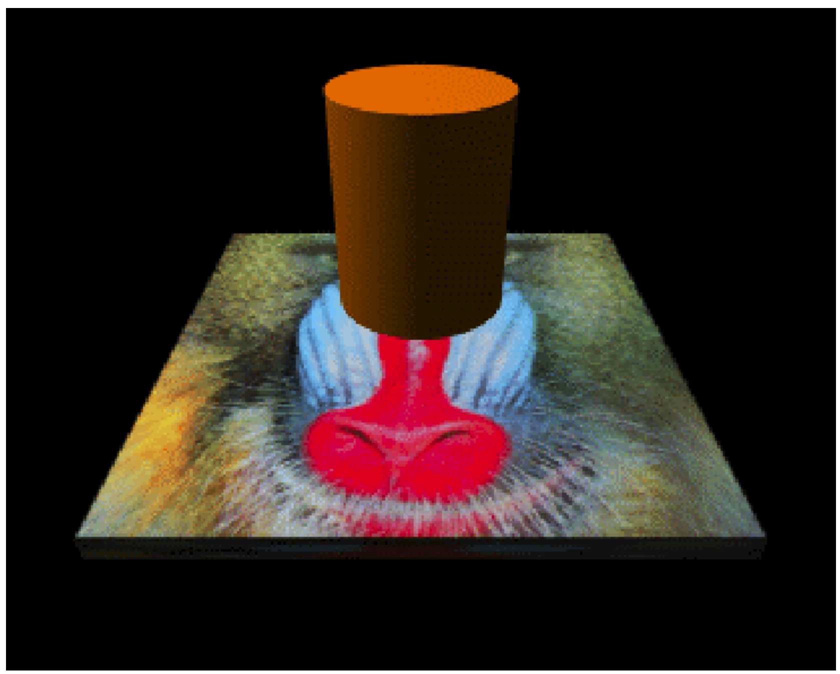
Recursive raytracing

- Rays are cast into the scene from the eye point through the pixels



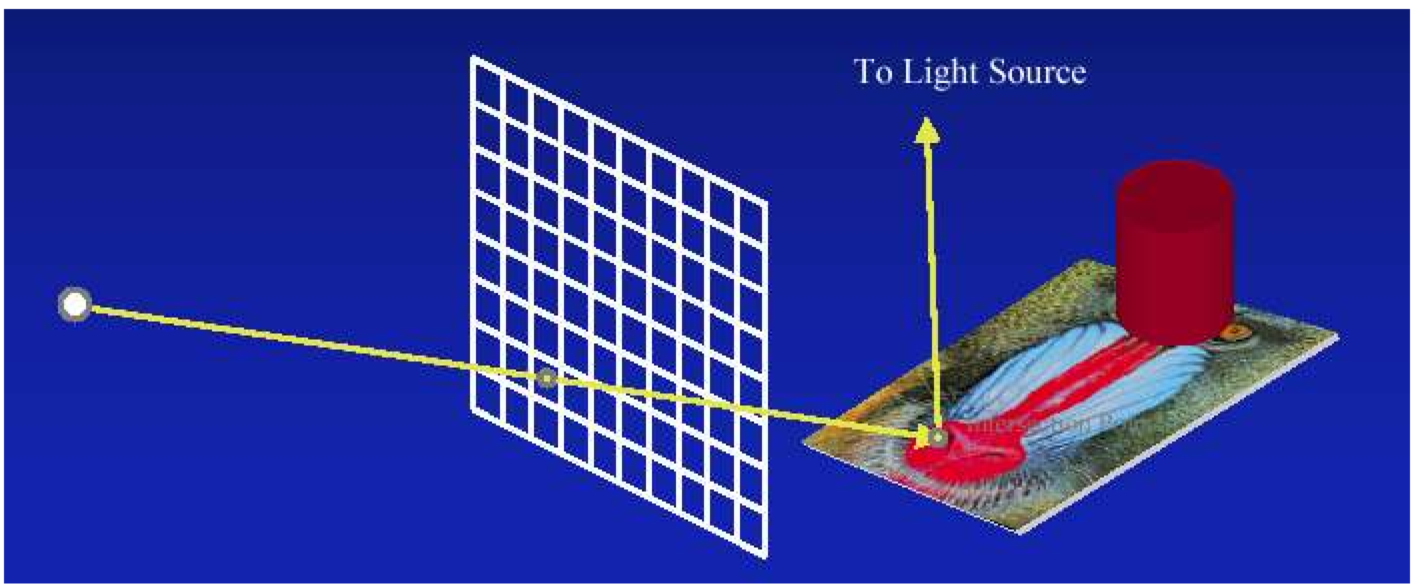
Recursive raytracing

- If the ray intersects with more than one object, then the nearest intersection is drawn
- Otherwise, draw the background color



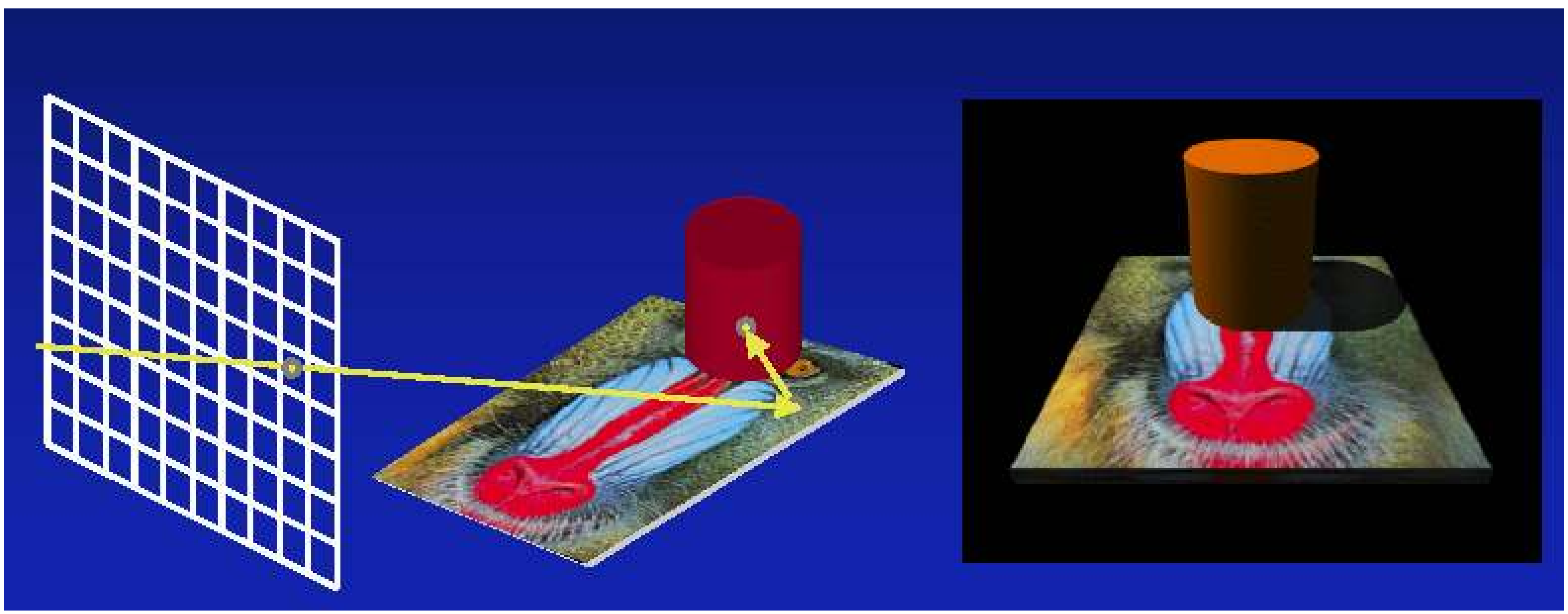
Recursive raytracing

- If a ray intersects with an object, then additional rays are cast from the point of intersection to all light sources



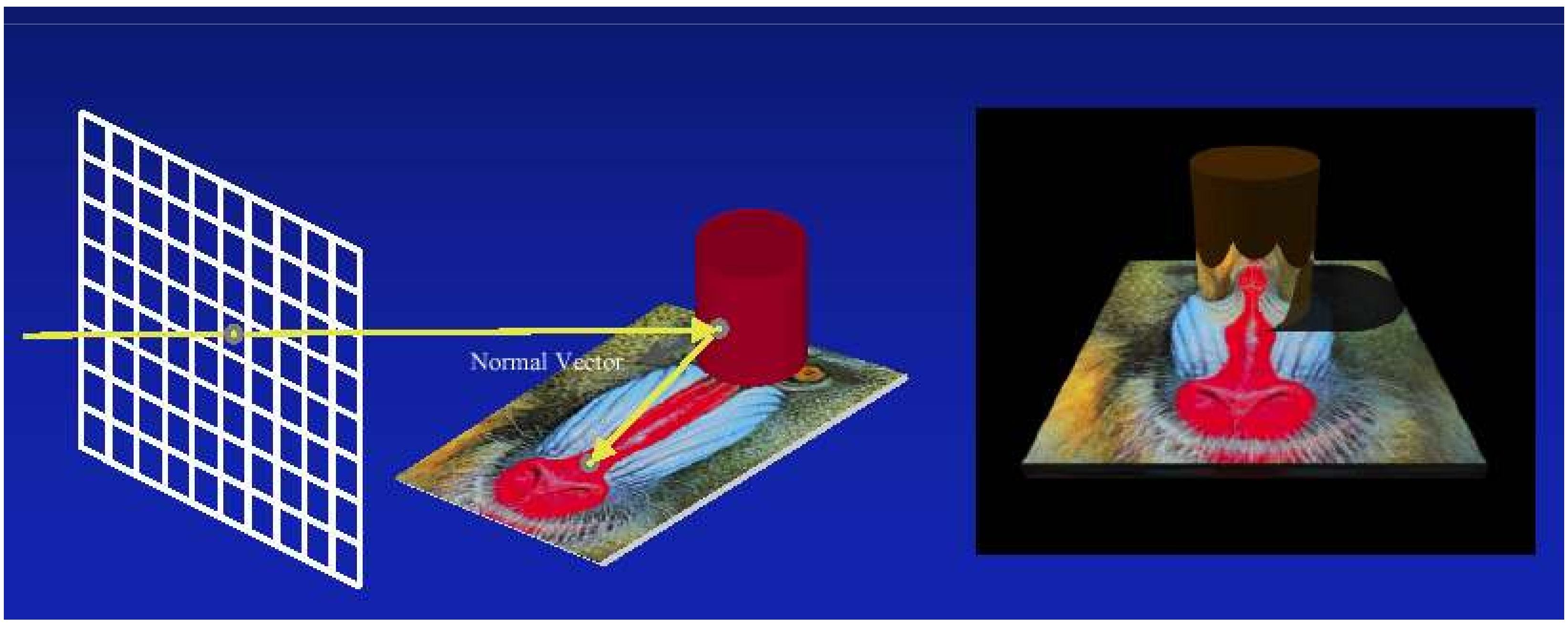
Recursive raytracing

- If these “shadow feelers“ intersect with an object, then the first intersection point is in shadow



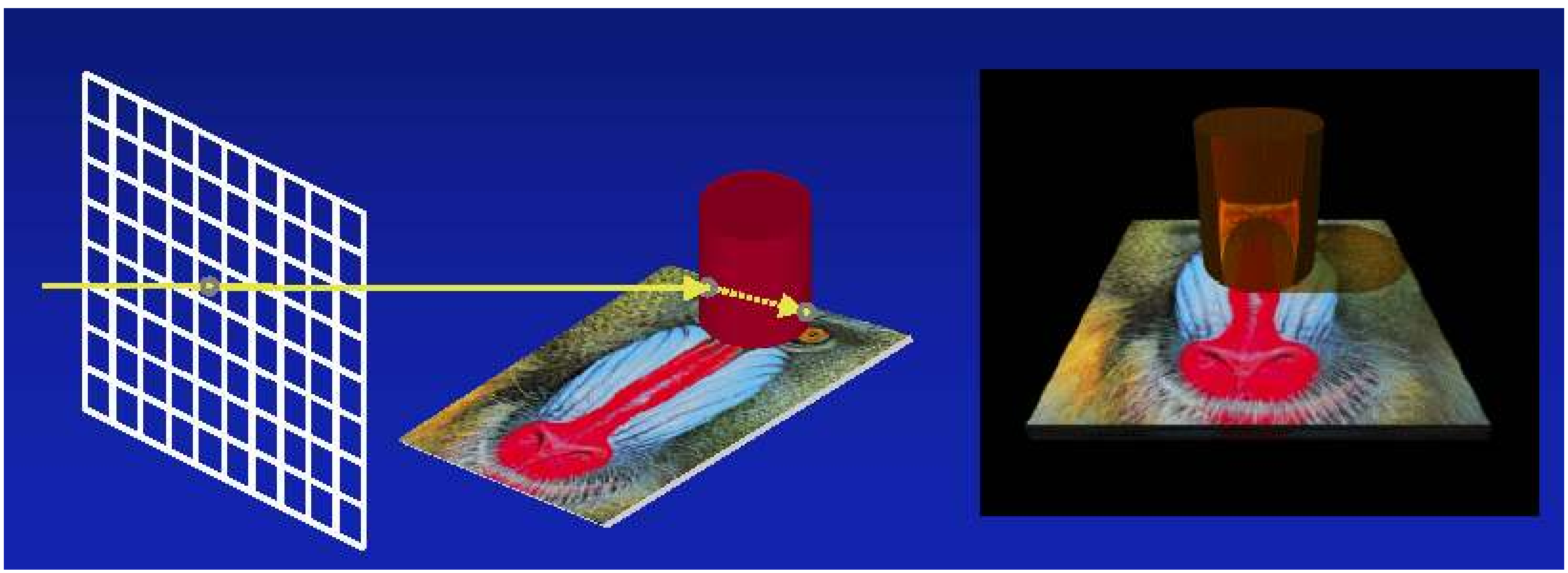
Recursive raytracing

- If the object is reflective, a reflected ray (about the surface normal) is cast into the scene



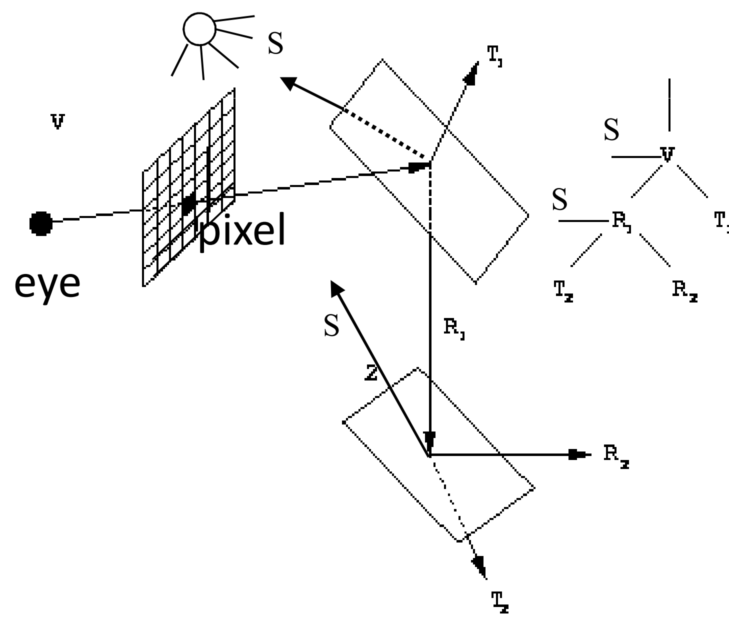
Recursive raytracing

- If the object is transparent, a refracted ray (about the surface normal) is cast into the scene



Recursive raytracing

- New rays are generated for reflection, transmission (refraction) and shadow feelers
- Rays are parameters of a recursive function
  - Detects all visible surfaces intersected by rays, shades them, and returns the result (= color)



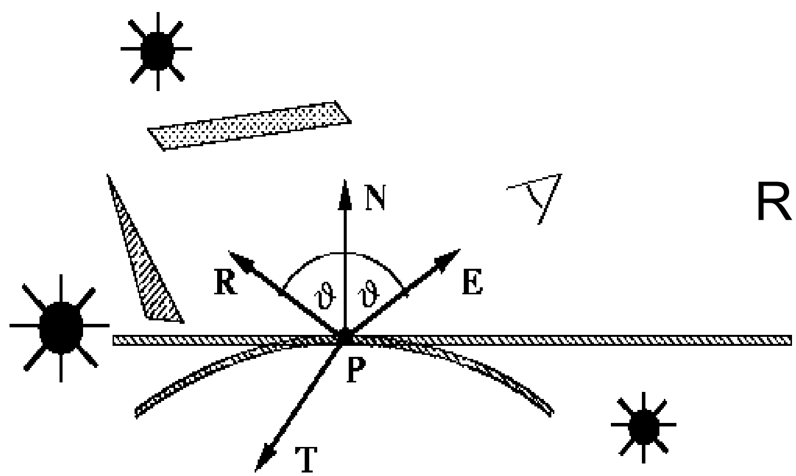
Lighting model

- Lighting on a surface is combined of
  - Ambient +
  - Diffuse +
  - Specular (highlights and reflection) +
  - Transmitted (refracted)

- Equivalent to the Phong-model **plus** contributions from **reflected and refracted rays**

Lighting model

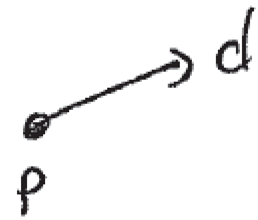
- $L_{sum} = L_{Phong} + r_r L_r + r_t L_t$
  - L_r is the luminance of the reflected ray
  - L_t is the luminance of the transmitted ray
  - r_r is the reflectance (in [0,1]) for ideal reflection
  - r_t is the reflectance (in [0,1]) for ideal transmission



$R = -E + 2(N \cdot E)N$

Rays

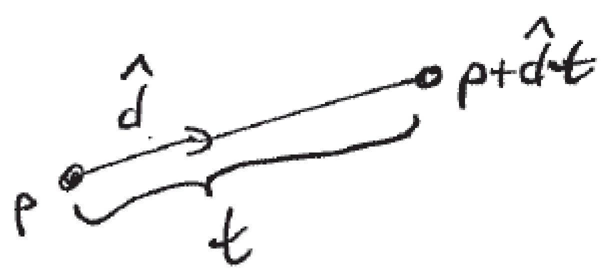
- Data structure
  - Point of origin **p** + direction **d**



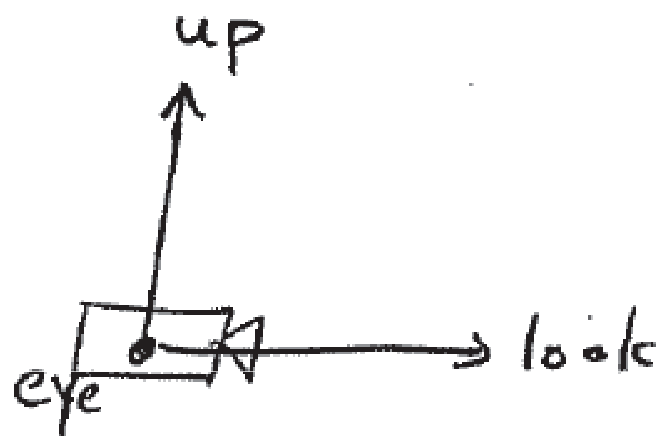
- Parametric equation

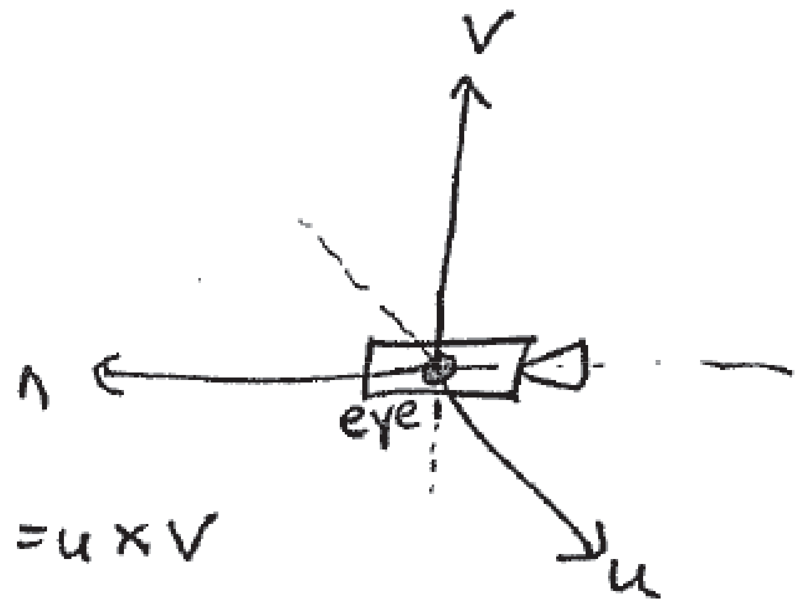
$$r(t) = p + d \cdot t$$

- If **d** is normalized, t is distance from **p** to **p**+t**d**



Camera setup

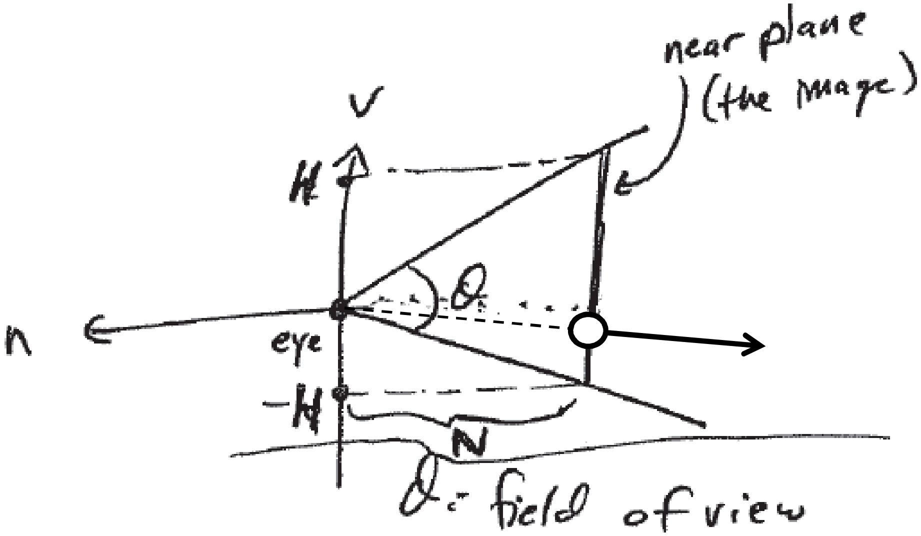


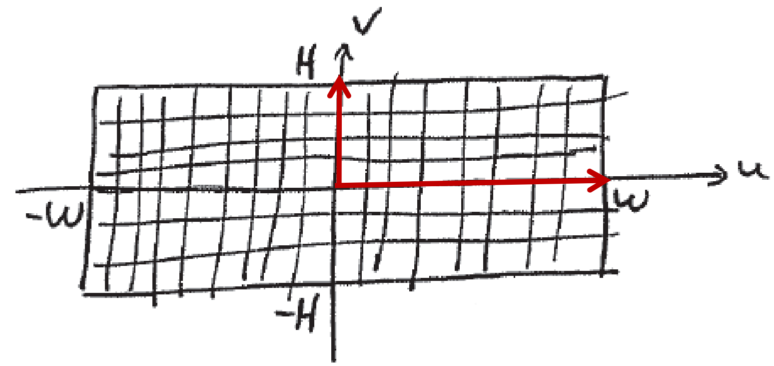


canonical openGL camera

u = x-axis
v = y-axis
n = z-axis (camera looks down -z)
eye = (0,0,0)

Camera setup





Aspect ratio r = W/H
Determine W and H from θ and r

pixel $x, y \in [-1, 1]^2$

$\vec{u}x + \vec{v}y - \boxed{\vec{n}(\text{near})} \rightarrow N\,\mathbf{n}$

Intersecting rays with objects

- Object representation?
  - Implicit equations make intersections easier
  - Surface is a set of points that satisfy

$$\mathbf{F(x,y,z) = 0}$$

  - Example: sphere at the origin with radius 1

$$\mathbf{F(x,y,z) = x^2 + y^2 + z^2 - 1}$$

  - As opposed to its explicit representation

$$s(u,v) = \begin{pmatrix} \cos u \cos v \\ \sin u \cos v \\ \sin v \end{pmatrix} \qquad \begin{matrix} u \in [0, 2\pi) \\ v \in [-\frac{\pi}{2}, \frac{\pi}{2}] \end{matrix}$$

Intersecting rays with objects

- Normal vector to an implicit surface

$$n(x_0, y_0, z_0) = (\nabla F)(x_0, y_0, z_0)$$

$$= \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right)(x_0, y_0, z_0)$$

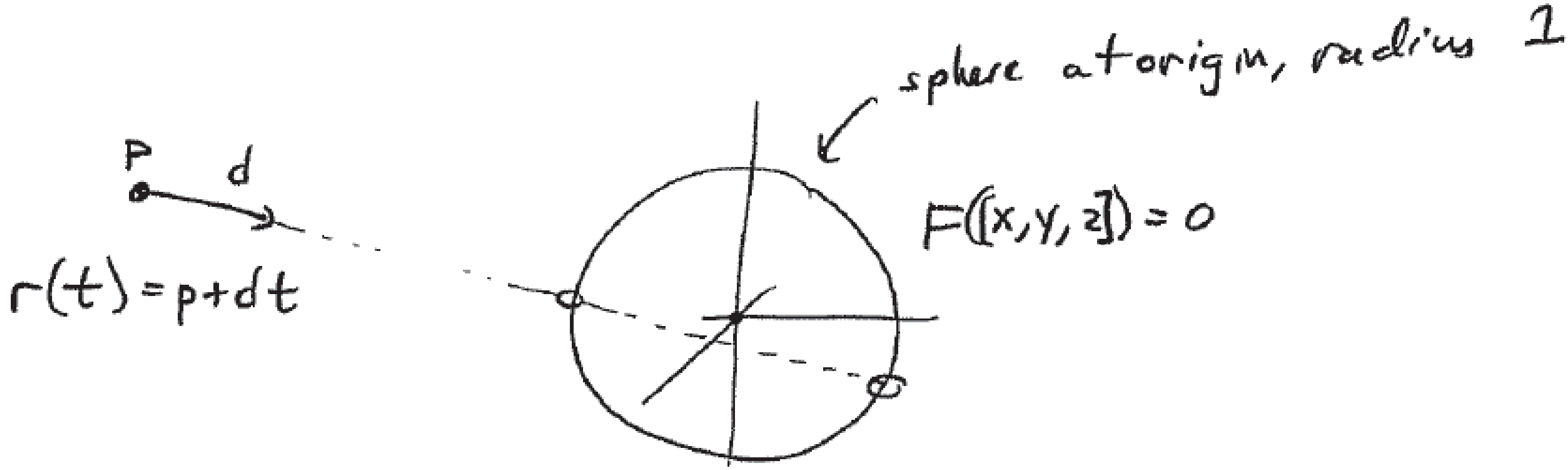
- For the previous example of a sphere

$$F(x,y,z) = x^2 + y^2 + z^2 - 1$$

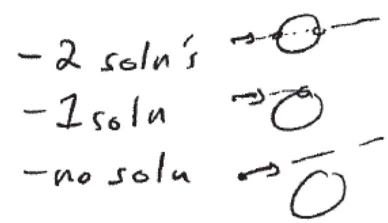
$$\mathbf{n(x,y,z) = [2x, 2y, 2z]^T}$$

Intersecting rays with objects

- Given a ray (**p**,**d**)



- Intersection points found by solving



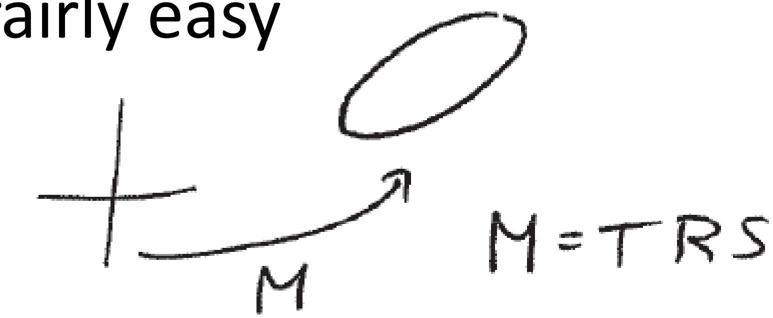
$$F(\mathbf{p}+t\mathbf{d}) = 0 \quad \text{for } t$$

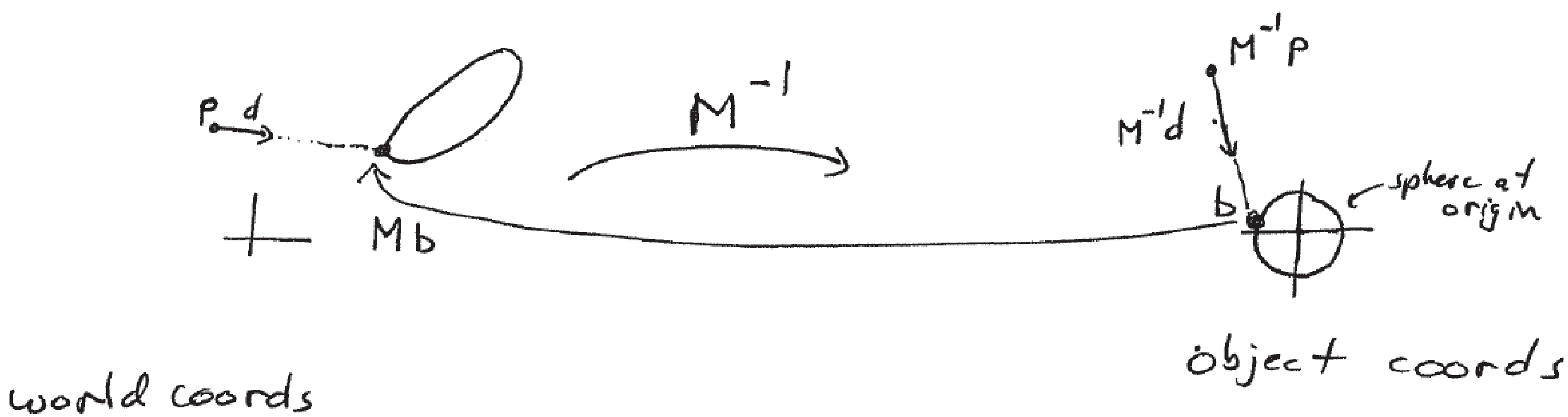
- Quadratic in t for $F(x,y,z) = x^2 + y^2 + z^2 - 1$

http://en.wikipedia.org/wiki/Quadratic_equation

What if object is not at origin?

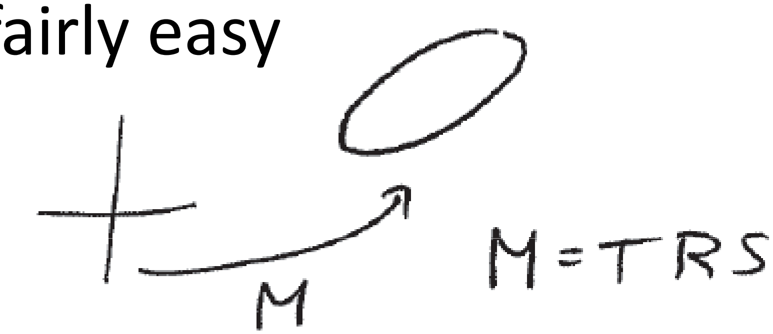
- Implicit equation becomes more complex
  - although sphere is still fairly easy
- Transform the ray with $\mathbf{M}^{-1}$





What if object is not at origin?

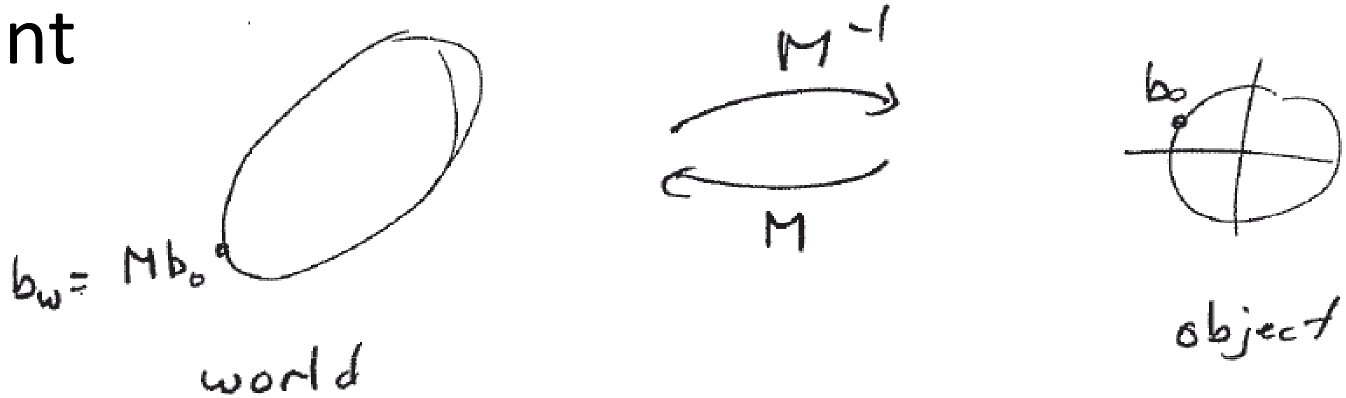
- Implicit equation becomes more complex
  - although sphere is still fairly easy
- Transform the ray with $\mathbf{M}^{-1}$
- Transform back the intersection point(s)
  - t is the same in both (don't normalize $\mathbf{M}^{-1}\mathbf{d}$)
- In other words, solve



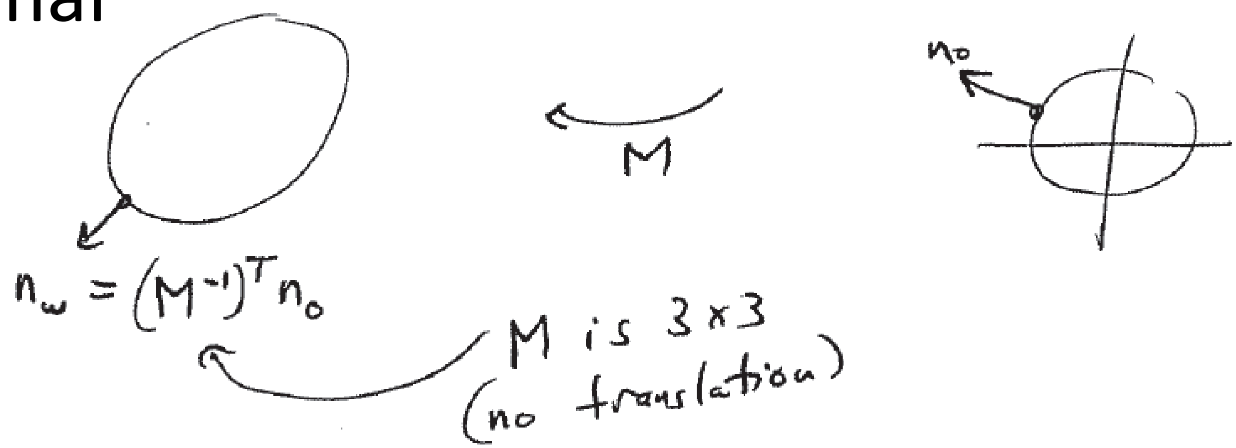
$$F(\mathbf{M}^{-1}\mathbf{p} + t(\mathbf{M}^{-1}\mathbf{d})) = 0$$

Transforming intersection

- Point

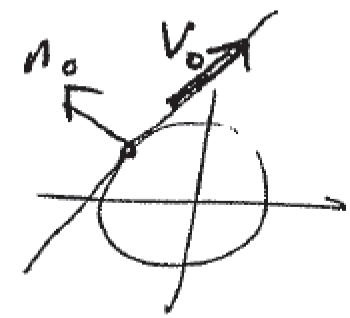


- Normal



Transforming intersection

- Why $(\mathbf{M}^{-1})^T$ for normals?
  - Given a vector $\mathbf{v}_o$ in the tangent plane then



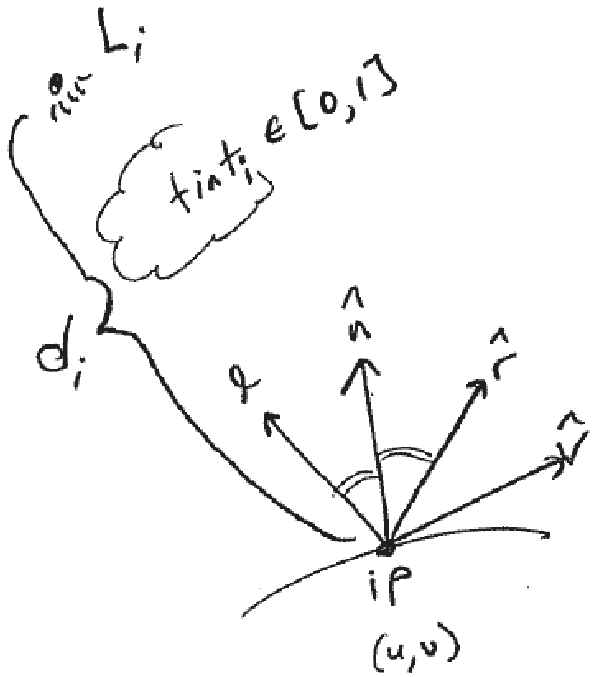
$$\mathbf{n}_o \cdot \mathbf{v}_o = 0 \quad \text{(orthogonal)}$$

$$\mathbf{n}_o^T \mathbf{v}_o = 0$$

$$\mathbf{n}_o^T (\mathbf{M}^{-1} \mathbf{M})\, \mathbf{v}_o = 0$$

$$\underbrace{((\mathbf{M}^{-1})^T \mathbf{n}_o)^T}_{\mathbf{n}_w^T} \underbrace{\mathbf{M}\, \mathbf{v}_o}_{\mathbf{v}_w} = 0$$

Lighting recap



$$I_{L_i} = L_i \cdot k_a \cdot T(u,v)$$ — ambient

$$+ L_i \cdot atten(d_i) \cdot tint_i \cdot T(u,v) \cdot k_d \cdot \max(0, \hat{n} \cdot \hat{\ell})$$ — diffuse

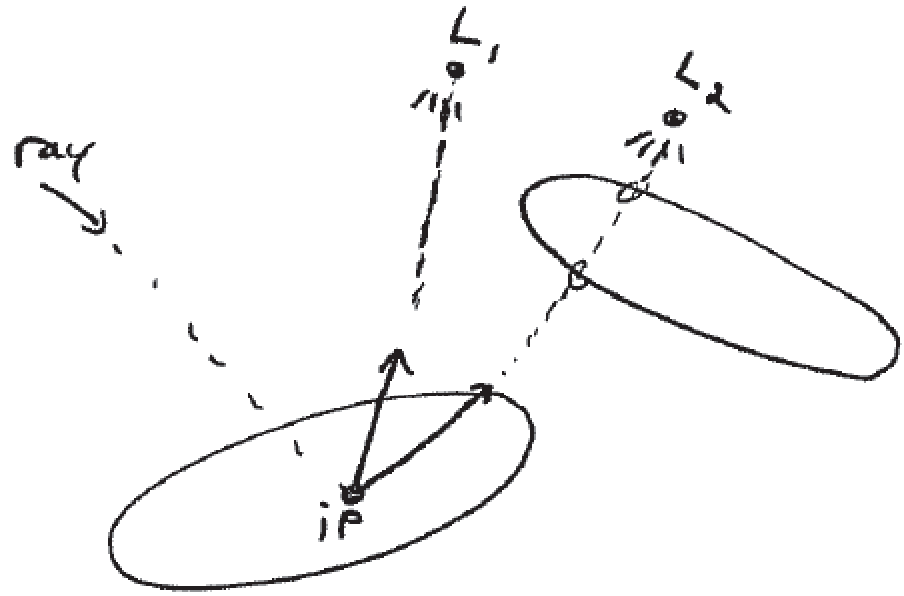
$$+ L_i \cdot atten(d_i) \cdot tint_i \cdot k_s \cdot \max(0, \hat{v} \cdot \hat{r})^{shiny}$$ — specular

use 0 when $\hat{n} \cdot \hat{\ell} < 0$

$$I_L = \Sigma I_{L_i}$$

Shadows

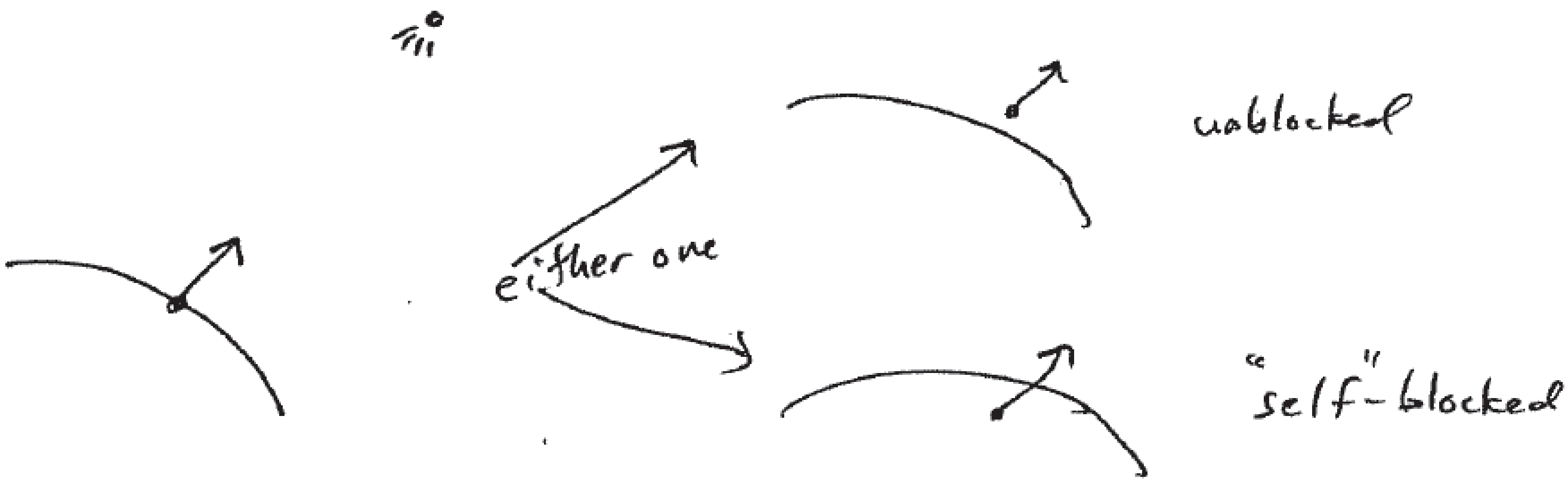
- Check if anything is between light and intersection point (ip) that would block light



- Send a ray from ip to each light to check if light illuminates ip
- Light blocked, tint = 0, otherwise tint = 1
  - Will modify later for transparency

Shadows

- Ray cannot start at ip due to rounding errors



- When finding intersections, use a minimum acceptable value of t to be some eps > 0

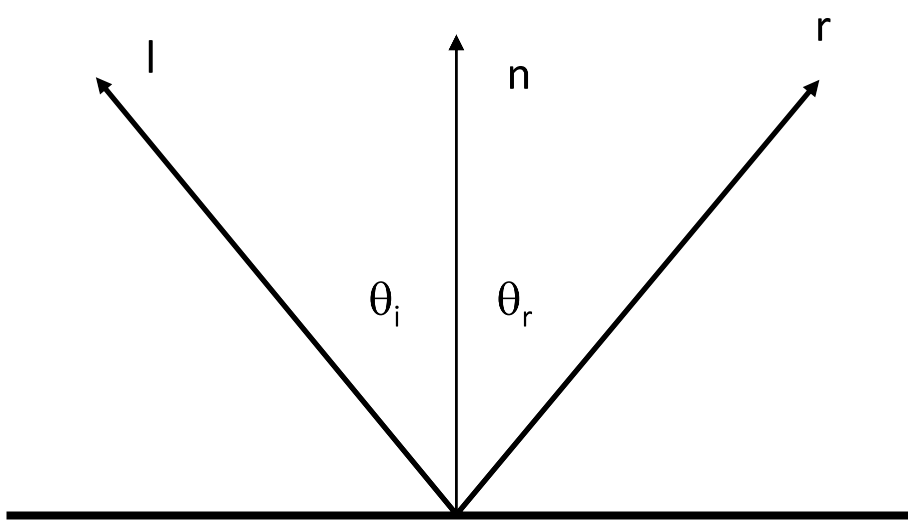
Ideal reflection

- Mirror reflection by law of reflection
  - The incident and reflected ray form the same angle with the surface normal
  - The incident and reflected ray and surface normal all lie in the same plane
  - In polar coordinates: $\theta_r = \theta_i$ and $\phi_r = \phi_i + \pi$
  - For view ray $\mathbf{l}$ and (normalized) normal $\mathbf{n}$

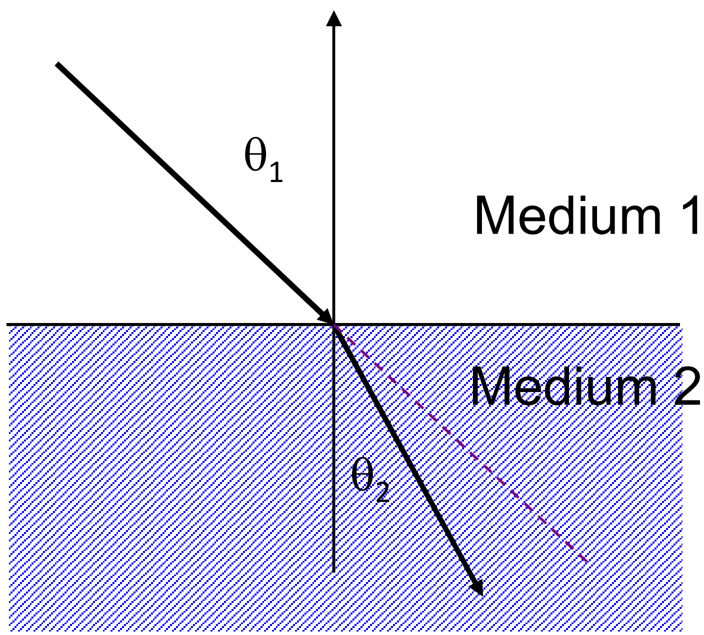
$$\mathbf{r} = -\mathbf{l} + 2\,(\mathbf{l} \cdot \mathbf{n})\,\mathbf{n}$$

Ideal reflection

Geometry of
Reflection law



Geometry of
refraction law



Ideal reflection

Total reflection

- Transition from optically dense to less dense material $n_2 < n_1$
  - Rays refracted away from the surface normal
  - There exists an incident angle θ_T with refraction angle of 90º

$$\sin\theta_T = \frac{n_2}{n_1}.$$

- Once θ_T is exceeded
  - All light reflected on the boundary between media
  - Total reflection