COSE436

Lecture 21: Ray Tracing

Nov 26, 2024

Won-Ki Jeong
(wkjeong@korea.ac.kr)

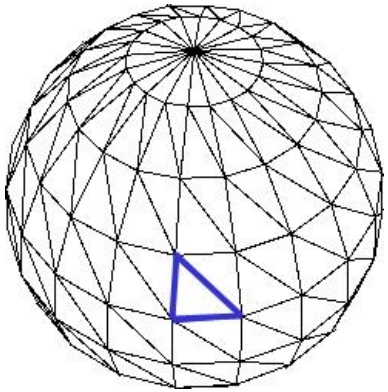
KOREA UNIVERSITY

Outlines

- Ray tracing basics
- Ray object intersection
- Recursive ray tracing

Object-Order Rendering

- For each object, find all the pixels influenced by the object and update their values
- Faster to compute
- No global effects
  - Shadows, multiple reflections...
- Raster graphics



(triangle rendered to screen)

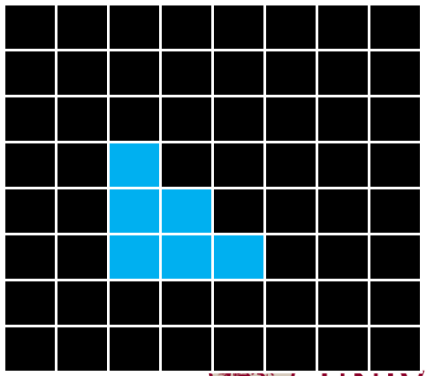
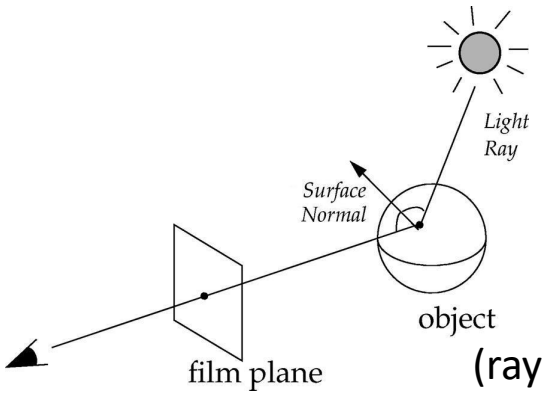


Image Courtesy of Van Dam

Image-Order Rendering

- For each pixel, find all the objects influence it and update its value
- Simpler to get working, global effects
- More expensive
- Ray tracing



(ray intersection rendered to screen)

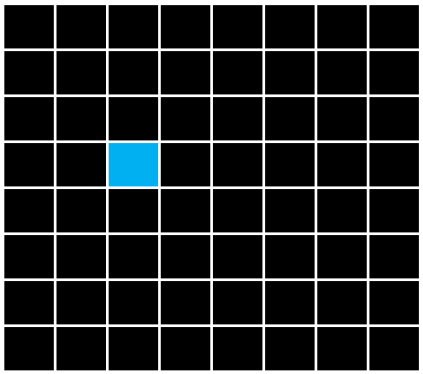


Image Courtesy of Van Dam

Ray Tracing

- Global effects?

Image Courtesy of Van Dam

Ray Tracing

- Transparency, reflection, shadow

Image Courtesy of Van Dam

Ray Tracing in Real-world

- Follow light rays from light source to the eye
  - Infinitely many light rays exist

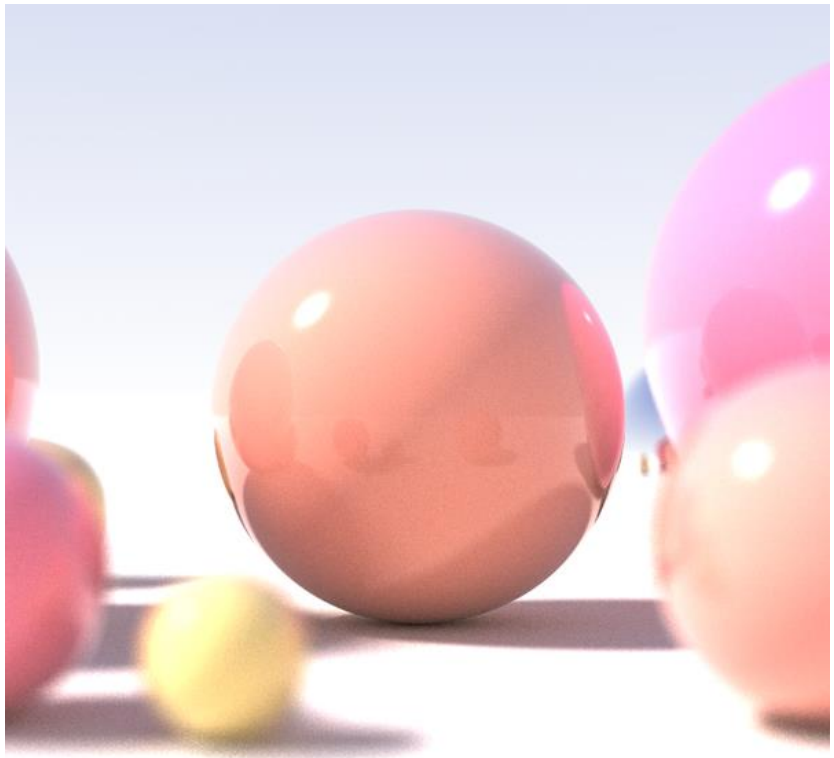
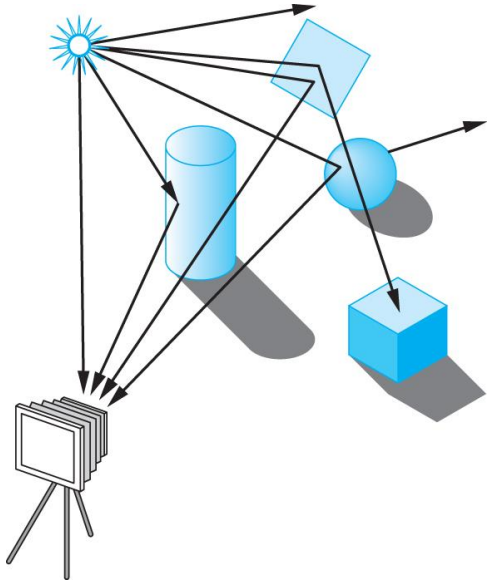
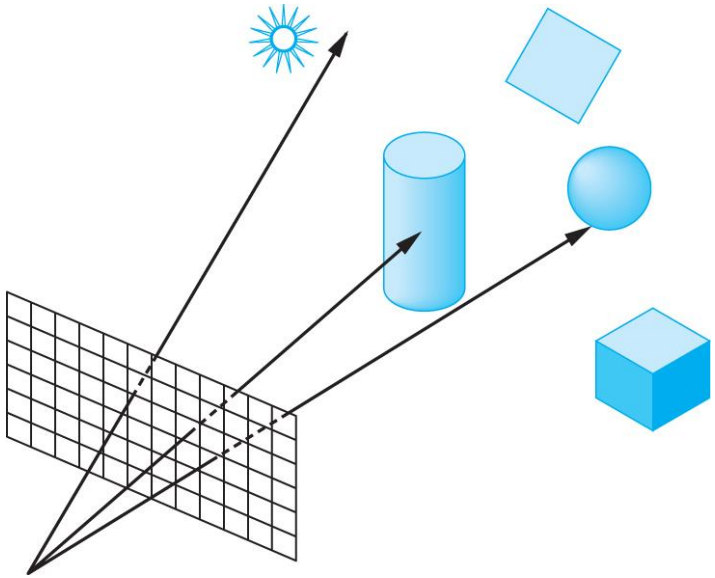


Image Courtesy of Wikipedia



Ray Tracing

- Only rays that reach the eye matter
- Trace light ray from the eye back through the image plane into the scene



Ray Tracing Components

- Ray generation
  - Origin and direction of viewing ray per pixel based on camera geometry
- Ray intersection
  - Finding closest object that intersecting the viewing ray
- Shading
  - Pixel color based on ray intersection

Basic Ray Tracing Algorithm

```
for each pixel do
    compute viewing ray
    find first object hit by ray and its surface normal n
    set pixel color to value computed from hit point, light, and n
```

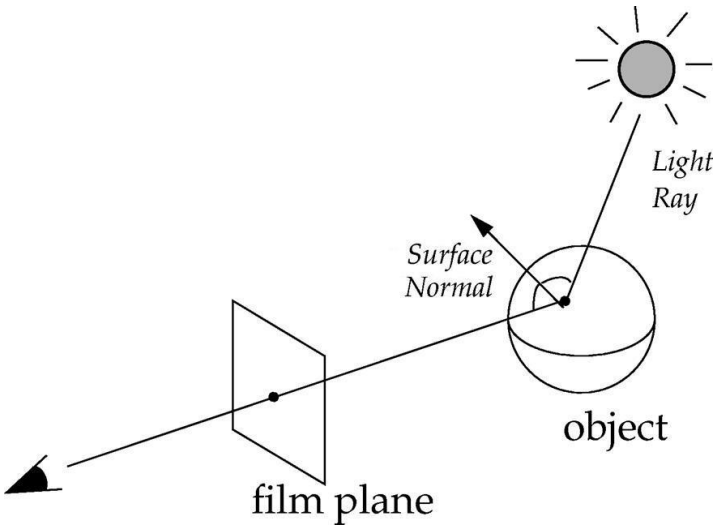
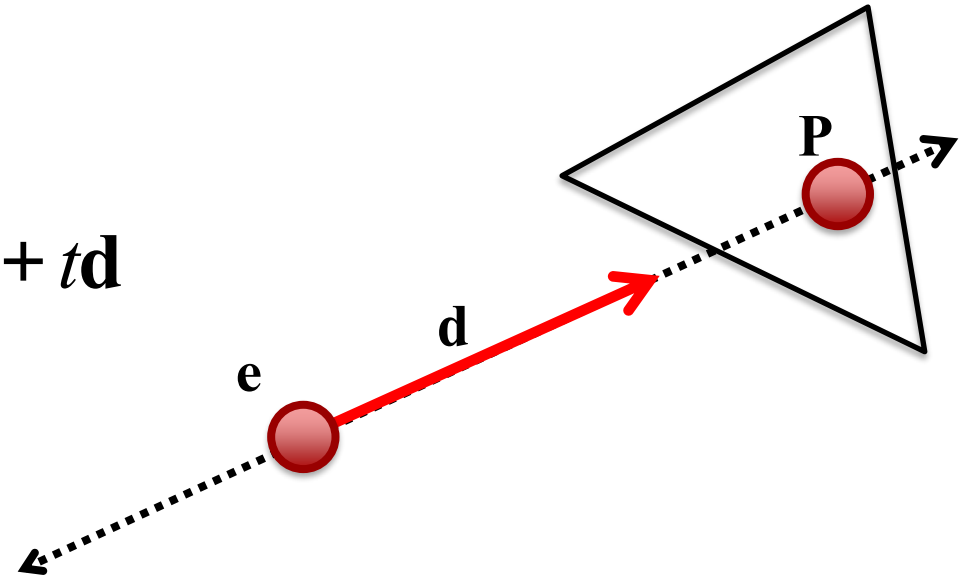


Image Courtesy of Van Dam

Definition of Ray

- Parametric line representation
  - Find t for intersection point P

$$\mathbf{P}(t) = \mathbf{e} + t\mathbf{d}$$



Camera (Eye) Frame

- Orthonormal coordinate frame
  - {u,v,w}
  - u: right, v: up, -w: view direction

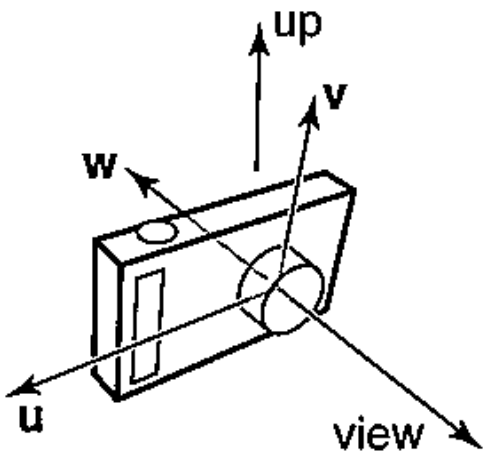
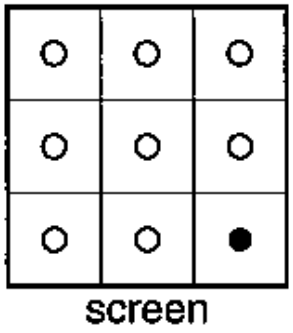


Image from Shirley's

Generate Ray for Sampling

- Each pixel on the screen maps to a single point in 3D space
  - A ray from eye position e pass through the point



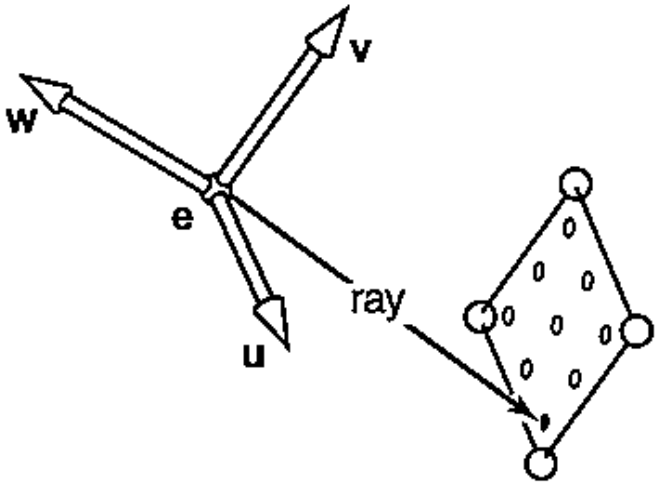
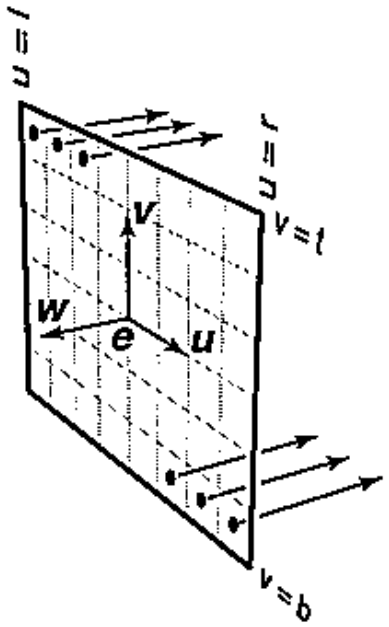


Image from Shirley's

Mapping Screen to View Plane

- Map n_x x n_y pixels into a rectangle (r-l) x (t-b)
  - Horizontal spacing: $(r-l)/n_x$
  - Vertical spacing: $(t-b)/n_y$
- Pixel (i,j) maps to (u,v)

$$u = l + (r - l)(i + 0.5) / n_x$$
$$v = b + (t - b)(j + 0.5) / n_y$$



Ray Generation: Orthogonal View

- Eye frame is on the view plane
- Eye position is at the center of the plane
- View directions are same for each pixel
- Rays have different origin

- Ray direction: $-\mathbf{w}$
- Ray origin: $\mathbf{e} + u\mathbf{u} + v\mathbf{v}$

scalar unit vector

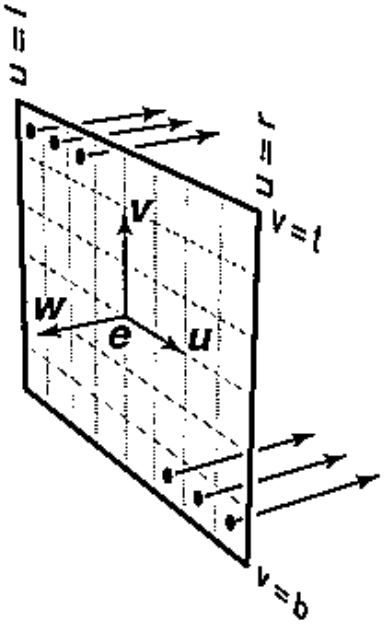


Image from Shirley's

Ray Generation: Perspective View

- All the rays have the same origin at the viewpoint
- View directions are different for each pixel
- Image plane is at distance *d*
- Ray direction: $-d\mathbf{w} + u\mathbf{u} + v\mathbf{v}$
- Ray origin: $\mathbf{e}$

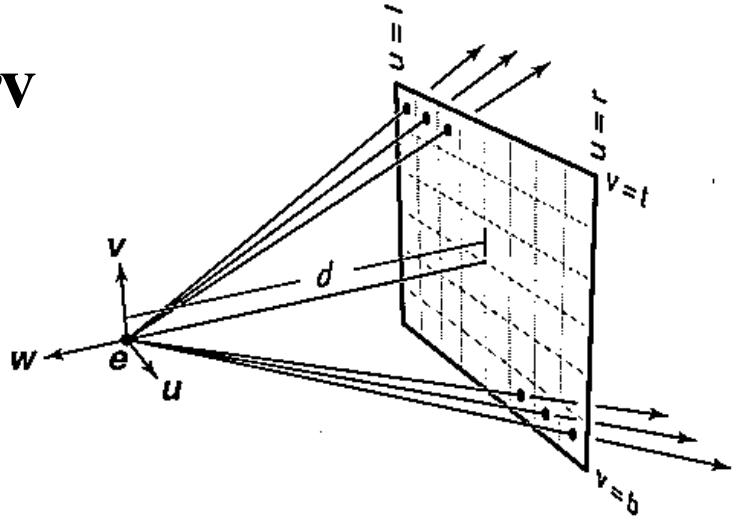


Image from Shirley's

Ray-Implicit Surface Intersection

- Ray

$$\mathbf{p}(t) = \mathbf{e} + t\mathbf{d}$$

- Implicit surface

$$f(\mathbf{p}) = 0$$

- Intersection between ray & surface

$$f(\mathbf{p}(t)) = f(\mathbf{e} + t\mathbf{d}) = 0$$

Ray-Sphere Intersection

- Sphere equation

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - R^2 = 0$$

$$(\mathbf{p} - \mathbf{c}) \times (\mathbf{p} - \mathbf{c}) - R^2 = 0$$

- Plug ray point into the sphere equation

$$(\mathbf{e} + t\mathbf{d} - \mathbf{c}) \times (\mathbf{e} + t\mathbf{d} - \mathbf{c}) - R^2 = 0$$

$$(\mathbf{d} \times \mathbf{d})t^2 + 2\mathbf{d} \times (\mathbf{e} - \mathbf{c})t + (\mathbf{e} - \mathbf{c}) \times (\mathbf{e} - \mathbf{c}) - R^2 = 0$$

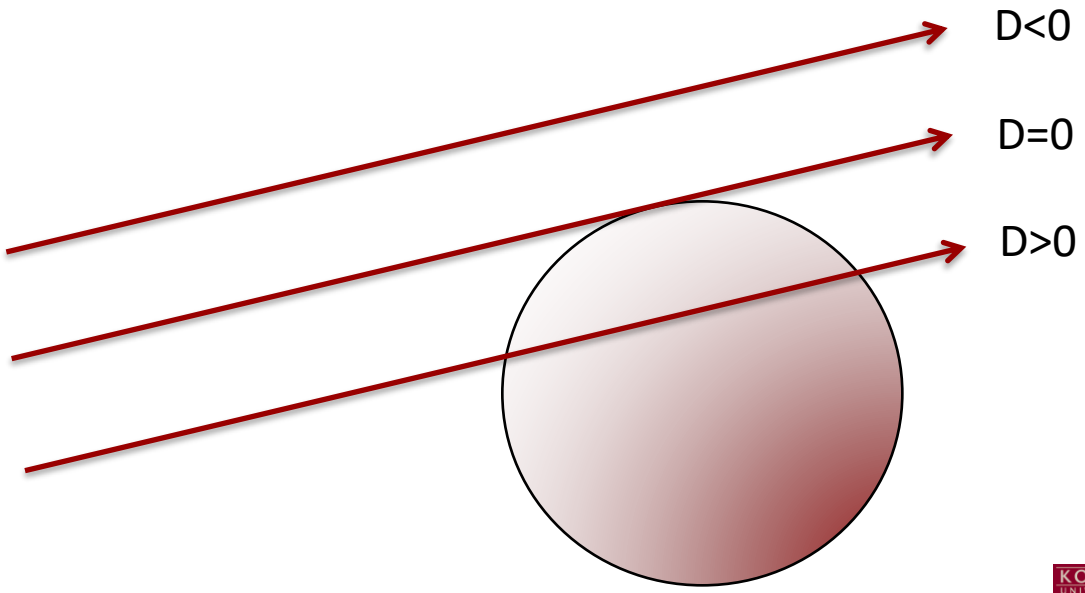
$At^2 + Bt + C = 0$: quadratic equation in t

Ray-Sphere Intersection

- Find t

$$t = \frac{-\mathbf{d} \times (\mathbf{e} - \mathbf{c}) \pm \sqrt{(\mathbf{d} \times (\mathbf{e} - \mathbf{c}))^2 - (\mathbf{d} \times \mathbf{d})((\mathbf{e} - \mathbf{c}) \times (\mathbf{e} - \mathbf{c}) - R^2)}}{(\mathbf{d} \times \mathbf{d})}$$

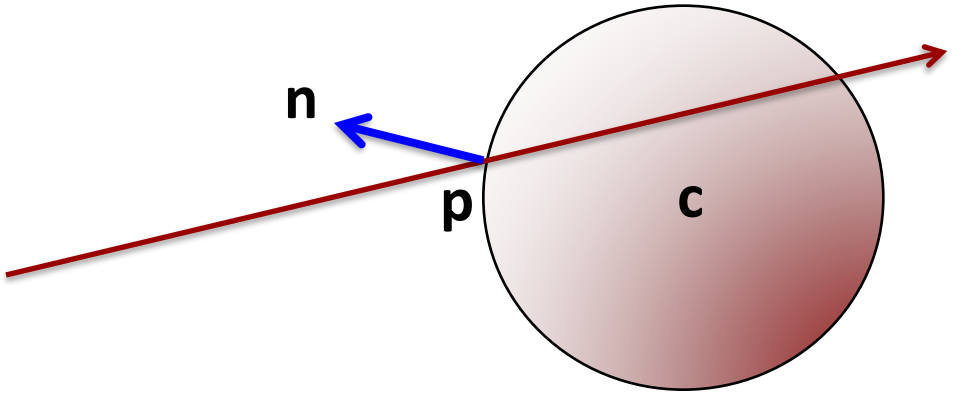
$$D = (\mathbf{d} \times (\mathbf{e} - \mathbf{c}))^2 - (\mathbf{d} \times \mathbf{d})((\mathbf{e} - \mathbf{c}) \times (\mathbf{e} - \mathbf{c}) - R^2)$$



Ray-Sphere Intersection

- (unit) Normal at intersection

$$\mathbf{n} = (\mathbf{p} - \mathbf{c}) / R$$



Ray-Parametric Surface Intersection

- Ray

$$\mathbf{p}(t) = \mathbf{e} + t\mathbf{d}$$

- Parameteric surface

$$\mathbf{f}(u,v) = (f(u,v), g(u,v), h(u,v))$$

- Intersection between ray & surface
  - Use numerical method (e.g., multivariate Newton's)

$$\left.\begin{aligned} x_e + tx_d &= f(u,v) \\ y_e + ty_d &= g(u,v) \\ z_e + tz_d &= h(u,v) \end{aligned}\right\} \text{ or, } \mathbf{e} + t\mathbf{d} = \mathbf{f}(u,v)$$

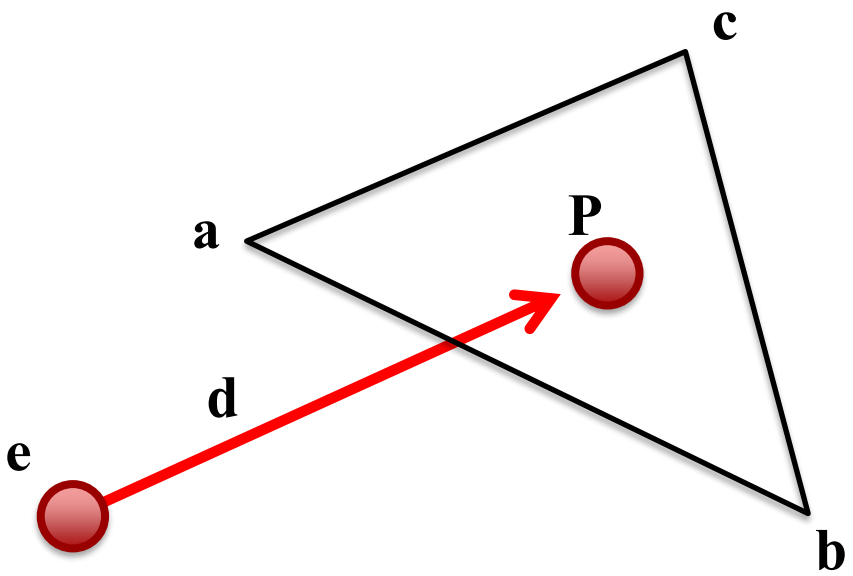
Ray-Triangle Intersection

- Using barycentric coordinate

$$\mathbf{e} + t\mathbf{d} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

- Inside test

$$\beta > 0, \gamma > 0, \text{and } \beta + \gamma < 1$$



Ray-Triangle Intersection

- Linear system of equations

$$x_e + tx_d = x_a + b(x_b - x_a) + g(x_c - x_a)$$
$$y_e + ty_d = y_a + b(y_b - y_a) + g(y_c - y_a)$$
$$z_e + tz_d = z_a + b(z_b - z_a) + g(z_c - z_a)$$

- Matrix form

$$\begin{bmatrix} x_a - x_b & x_a - x_c & x_d \\ y_a - y_b & y_a - y_c & y_d \\ z_a - z_b & z_a - z_c & z_d \end{bmatrix} \begin{bmatrix} b \\ g \\ t \end{bmatrix} = \begin{bmatrix} x_a - x_e \\ y_a - y_e \\ z_a - z_e \end{bmatrix}$$

Early Termination Algorithm

- Compute t, γ, β incrementally

```
bool raytri(ray r, vec3 a, vec3 b vec3 c, interval[t0,t1])
compute t
if(t<t0 or t>t1) then
    return false
compute γ
if(γ<0 or γ>1) then
    return false
compute β
if(β<0 or β>1) then
    return false
return true
```

Ray-Polygon Intersection

- Intersection test for arbitrary n-gon on a plane
- Plane intersection test

$$\mathbf{p} \cdot \mathbf{n} + c = 0$$

$$\mathbf{p}(t) = \mathbf{e} + t\,\mathbf{d}$$

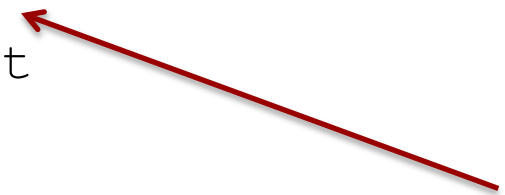
$$t = -(\mathbf{e} \cdot \mathbf{n} + c)/\,(\mathbf{d} \cdot \mathbf{n})$$

- Inside test
  - Cross product of two vectors determine which side the point is on

Intersecting a Group of Objects

- Find closest intersection to the eye

```
hit = false
for each object o in the group do
if(o is hit at t and t is in [t0,t1]) then
   hit = true
   hitobject = o
   t1 = t
return hit
```



update t1 with new t where t<t1

Shading

- Any illumination model can be used
- Phong model

$$\mathbf{I} = \mathbf{k}_a \mathbf{L}_a + \frac{1}{a + bd + cd^2}\left(\mathbf{k}_d (\mathbf{l} \times \mathbf{n}) \mathbf{L}_d + \mathbf{k}_s (\mathbf{r} \times \mathbf{v})^a \mathbf{L}_s\right)$$

Shadow

- Check whether intersection point is blocked by other object
- Create another ray to the light source and check intersection

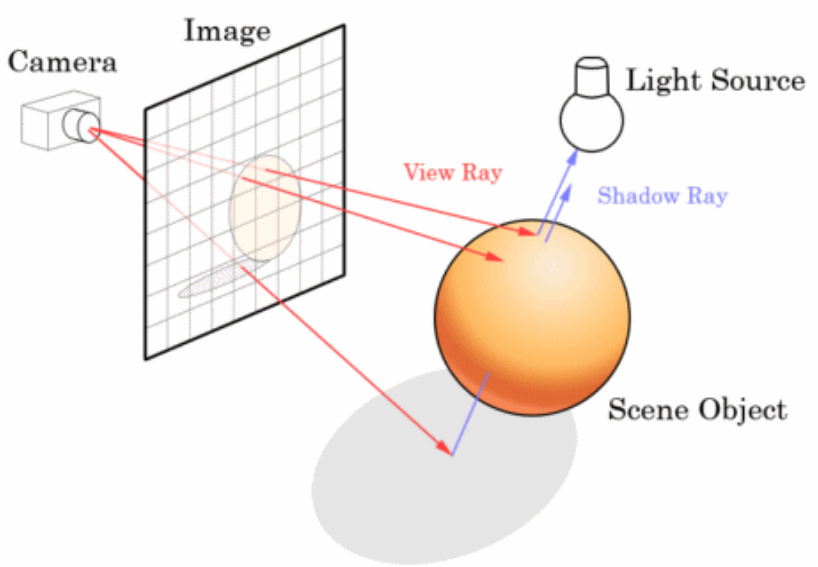


Image Courtesy of Van Dam

Note

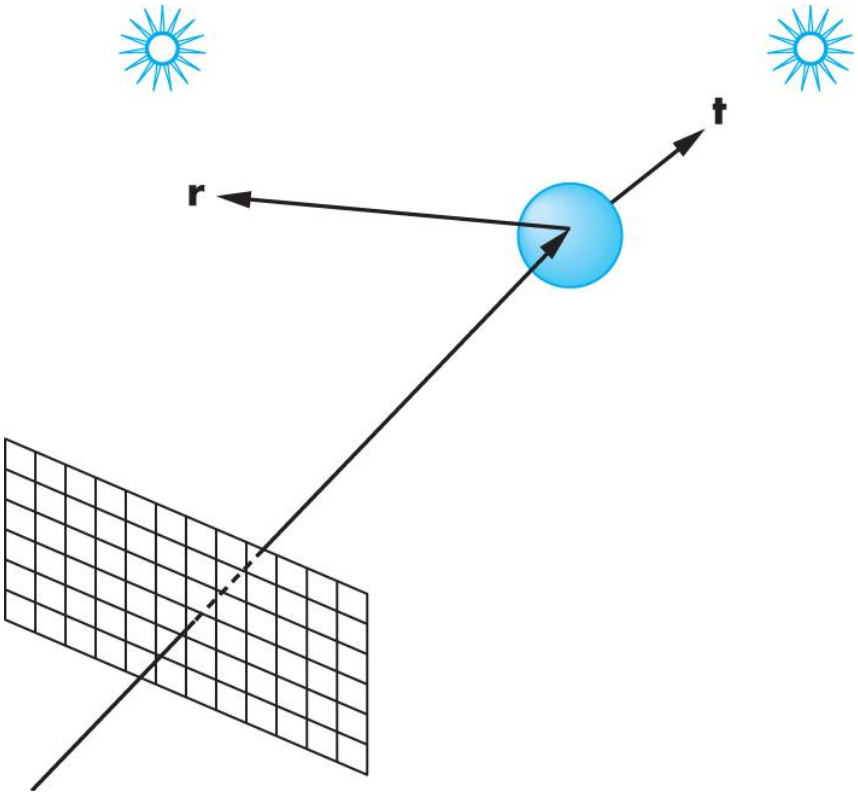
- For numerical issue, use [ε,∞] for shadow ray testing
  - If [0,∞] is used, then the current surface will be detected as intersection
- If under the shadow, ambient color is used to prevent absolute black
  - Ambient color from environment

Phong with Shadow

```
function raycolor(ray e+td, [t0,t1])
hit-record rec, srec
if(scene->hit(e+td,t0,t1,rec)) then
    p=e+(rec.t)d
    color c=rec.kaIa // ambient color
    if(not scene->hit(p+sl,ε,∞, srec)) then
        c = c + (diffuse + specular color)
    return c
else
    return background color
```

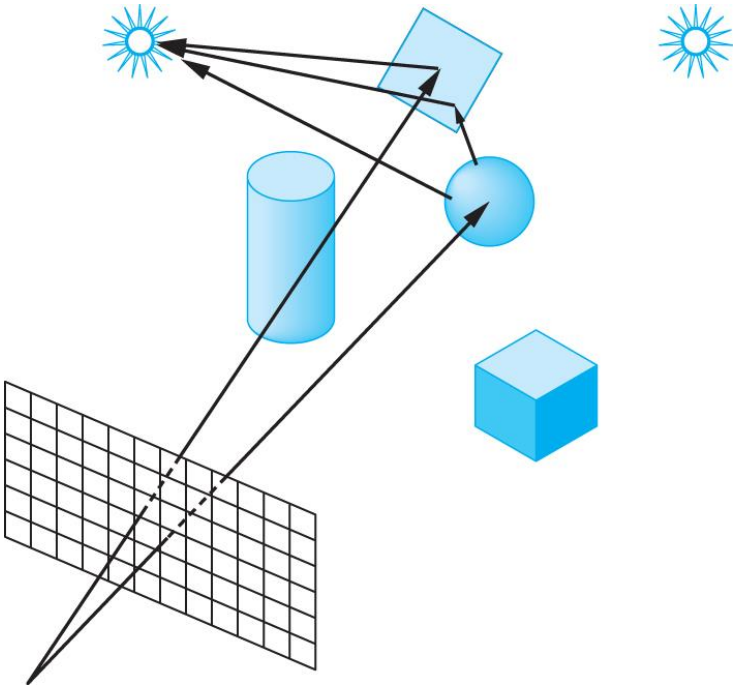
shadow test

Reflection and Refraction



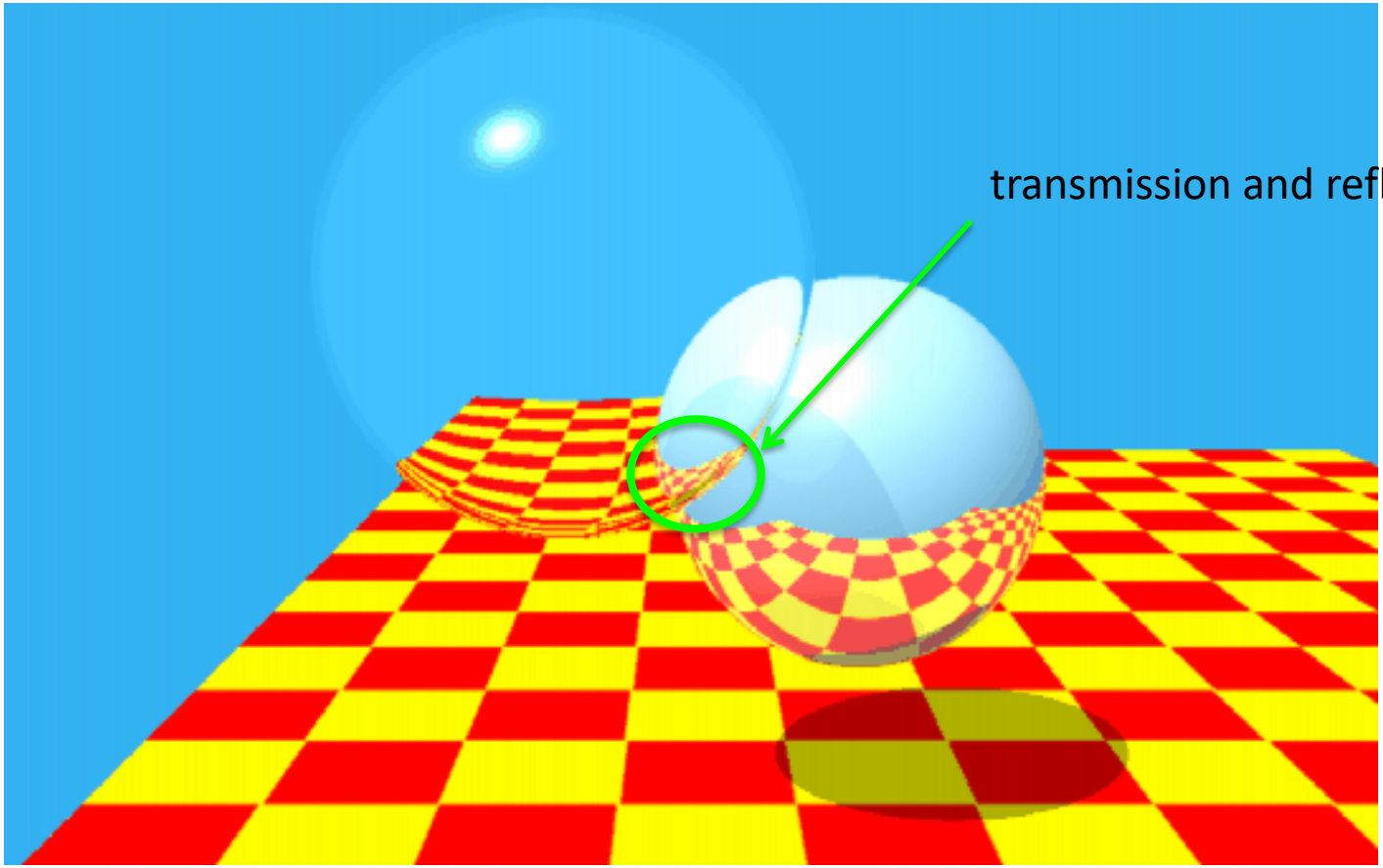
Recursive Ray Tracing

- Trace multiple reflection and refraction
- Process is recursive



Example of Recursive Ray Tracing

Whitted 1980



Recursive Ray Tracing

- Simulating global effect (whitted, 1979)
- At each point of intersection, cast rays to the directions likely to contribute most
  - Toward lights for shadows
  - Reflection direction for neighbor object
  - Through object for transparency
  - Ambient color

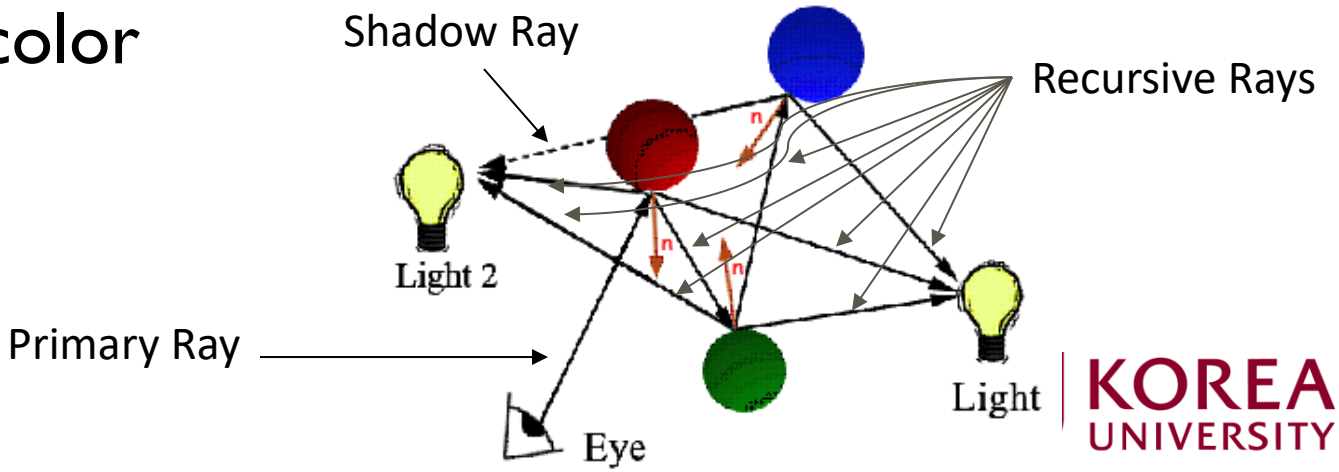
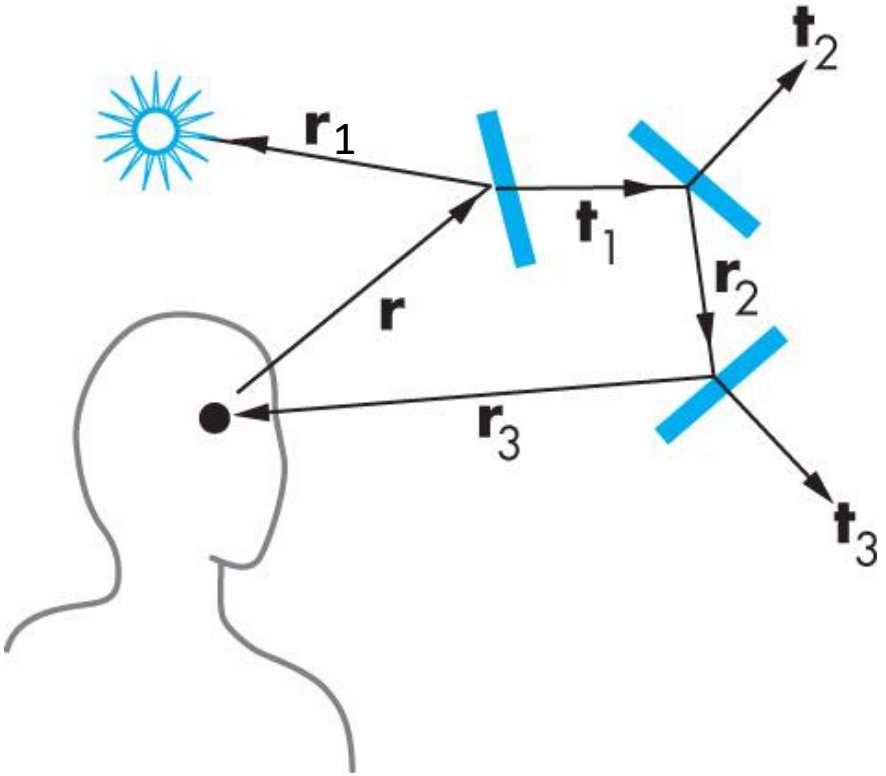


Image Courtesy of Van Dam

Recursive Ray Tracing

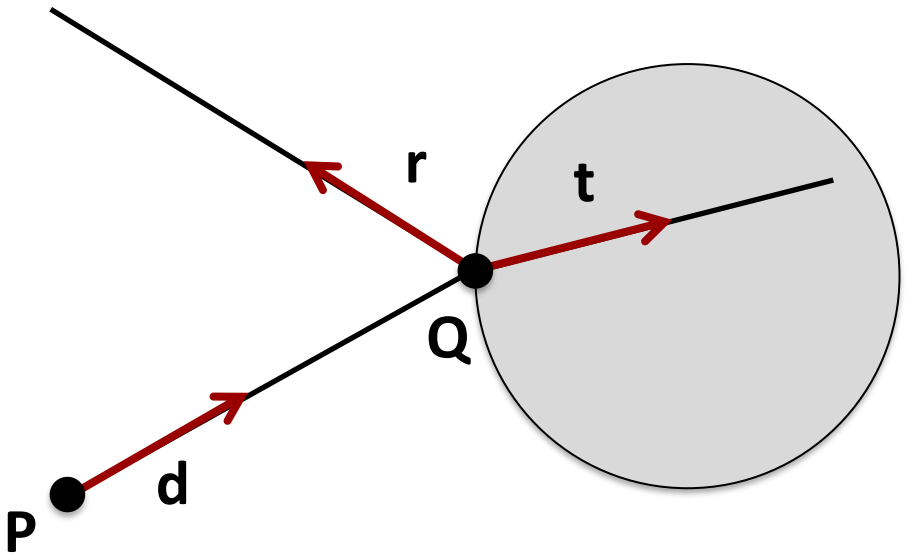
- Trace secondary rays at intersection
  - Shadow, reflection, refraction
- Recursively spawn new rays



Recursive Ray Tracing

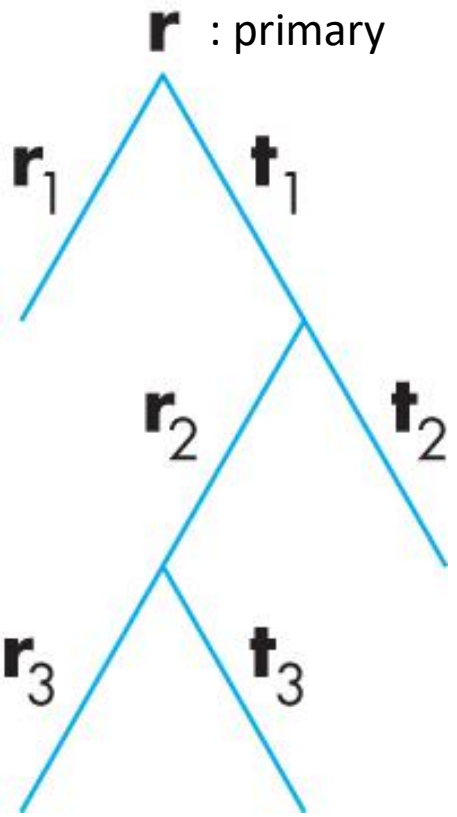
$$I = I_{direct} + I_{reflected} + I_{refracted}$$

$$I(P, \mathbf{d}) = I_{direct} + k_r I(Q, \mathbf{r}) + (1 - k_r) I(Q, \mathbf{t})$$



Ray Tree

- Per each ray
- Each node color contribution can be evaluated
- Stop when max depth reached



Aliasing

- Single sample per pixel introduce aliasing

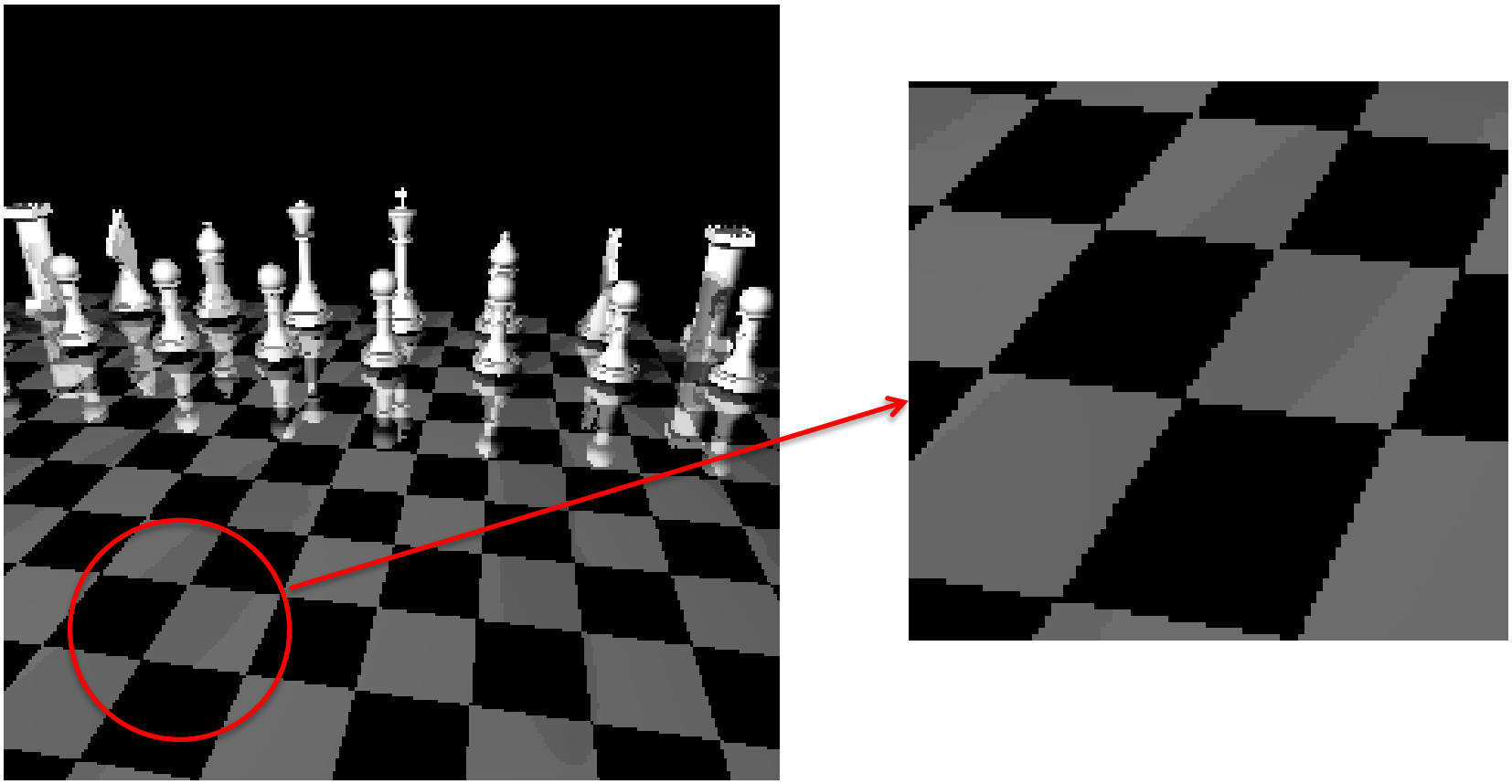


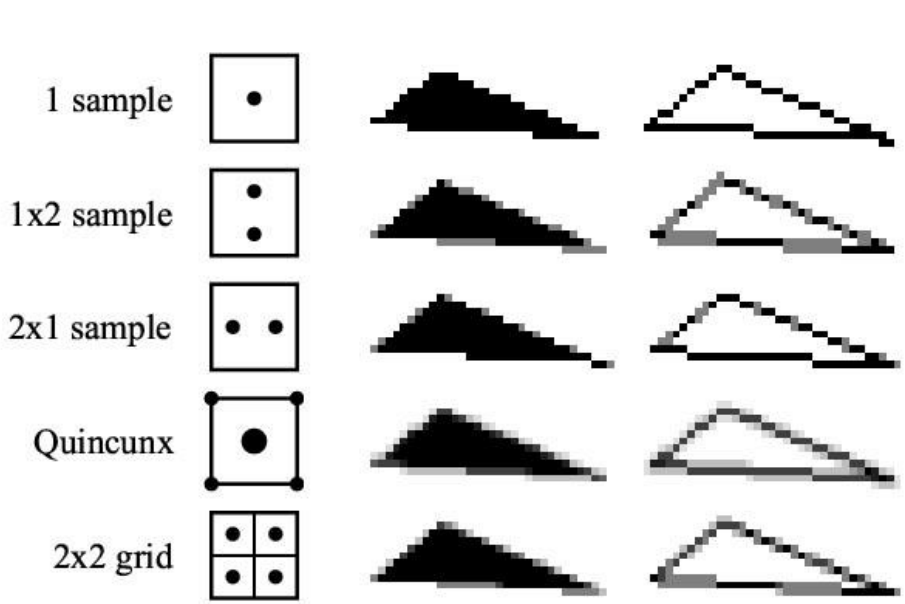
Image Courtesy of Van Dam

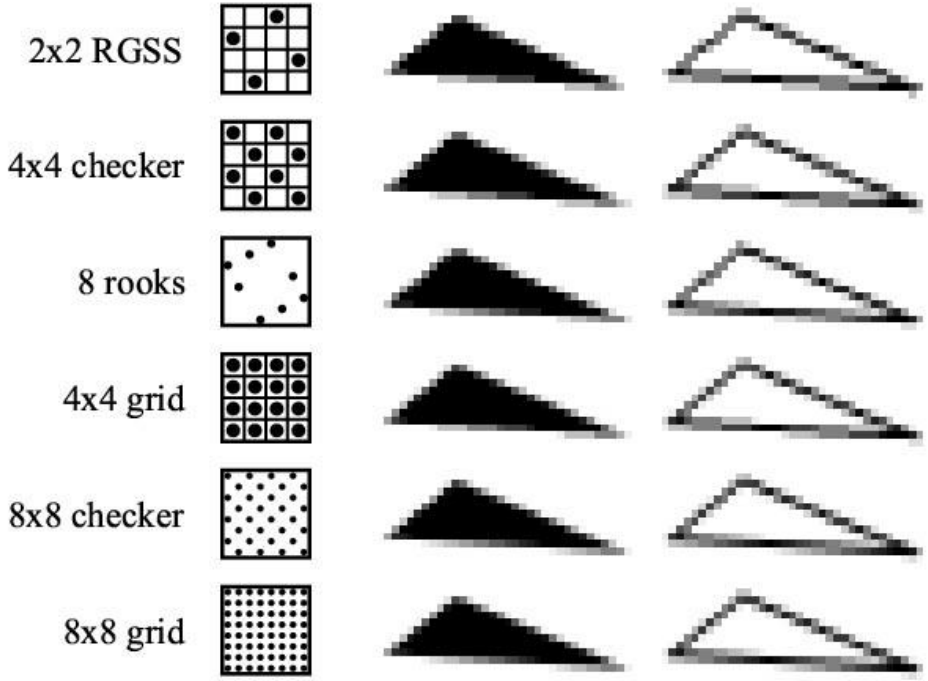
Anti-aliasing

- Supersampling
  - Corners and center
- Adaptive sampling
  - Increase sampling density in areas of rapid change in geometry and lighting
- Stochastic sampling
  - Random sampling

Image Courtesy of Van Dam

Supersampling Pattern





Supersampling

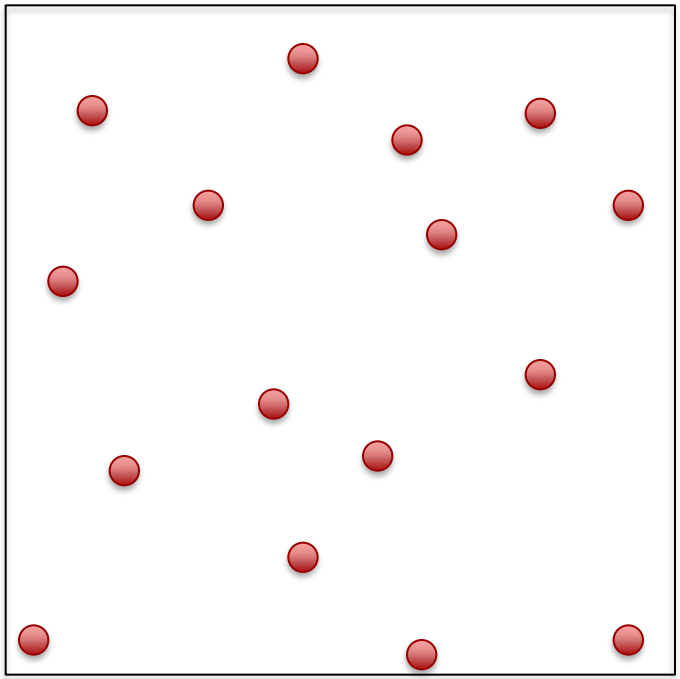
With SS

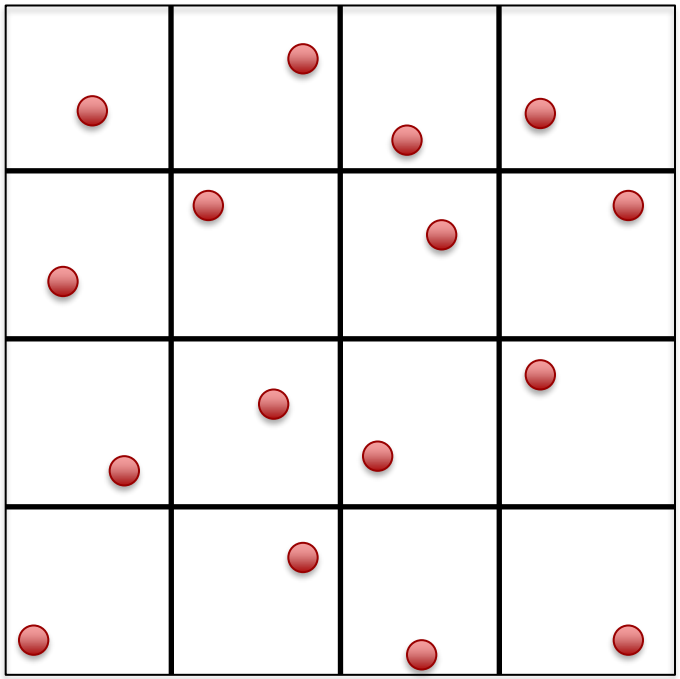
Without SS

Image Courtesy of Van Dam

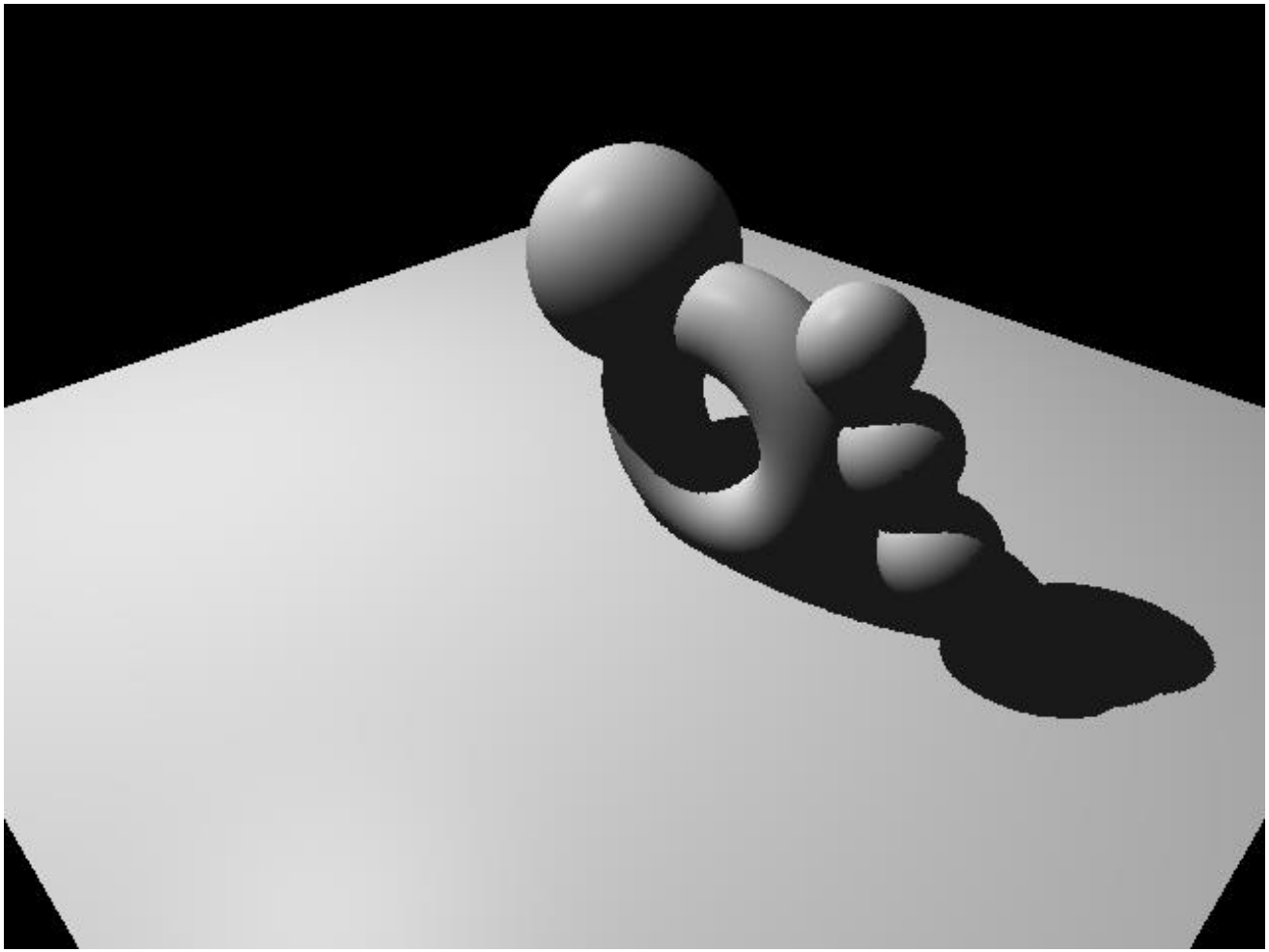
Jittering / Stratified Sampling

- One sample per pixel with irregular pattern





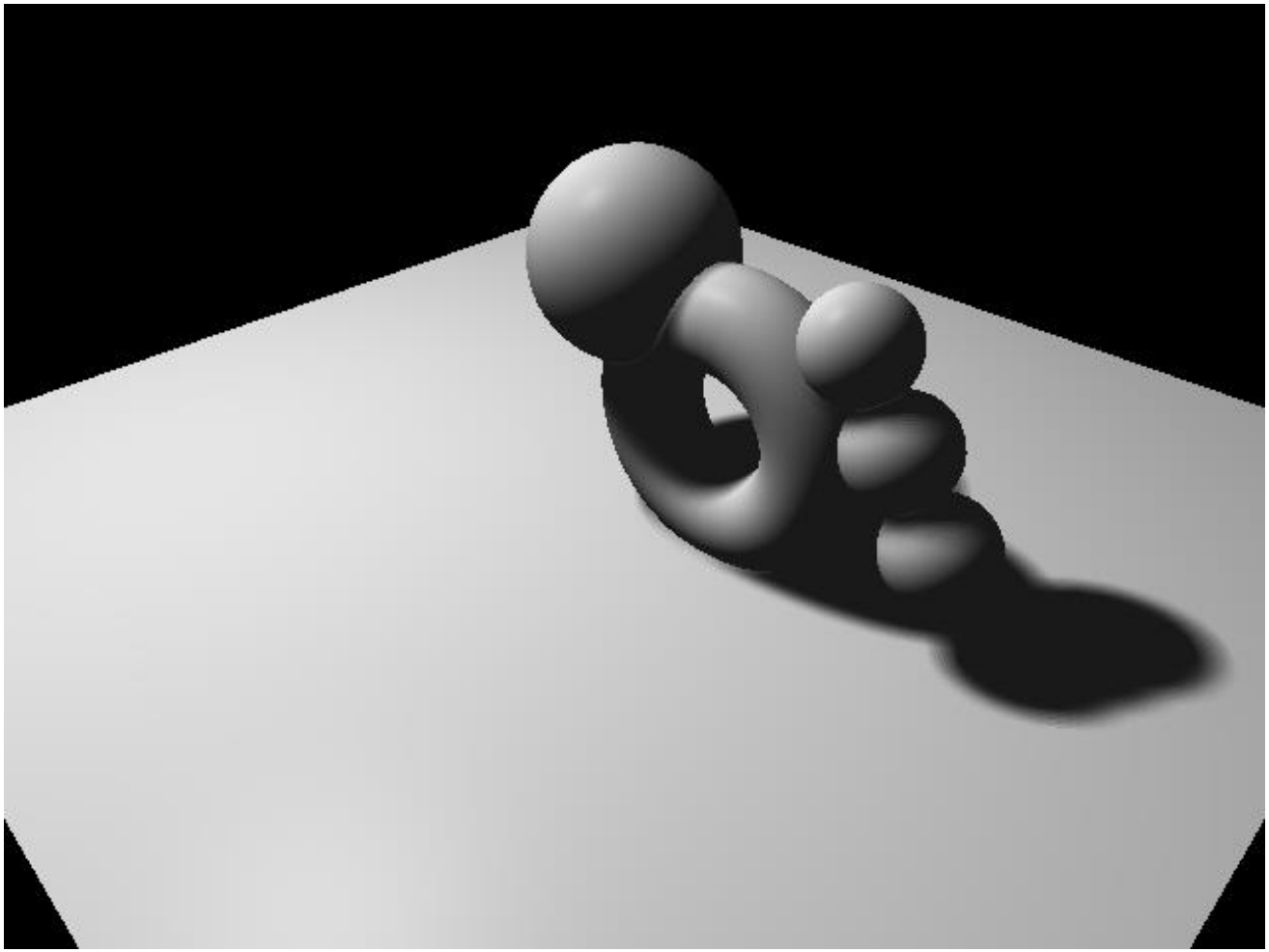
Shadow Revisited



Problem?

Image Courtesy of Gibson

Soft Shadow



More realistic

Image Courtesy of Gibson

Soft Shadow

- Area light source

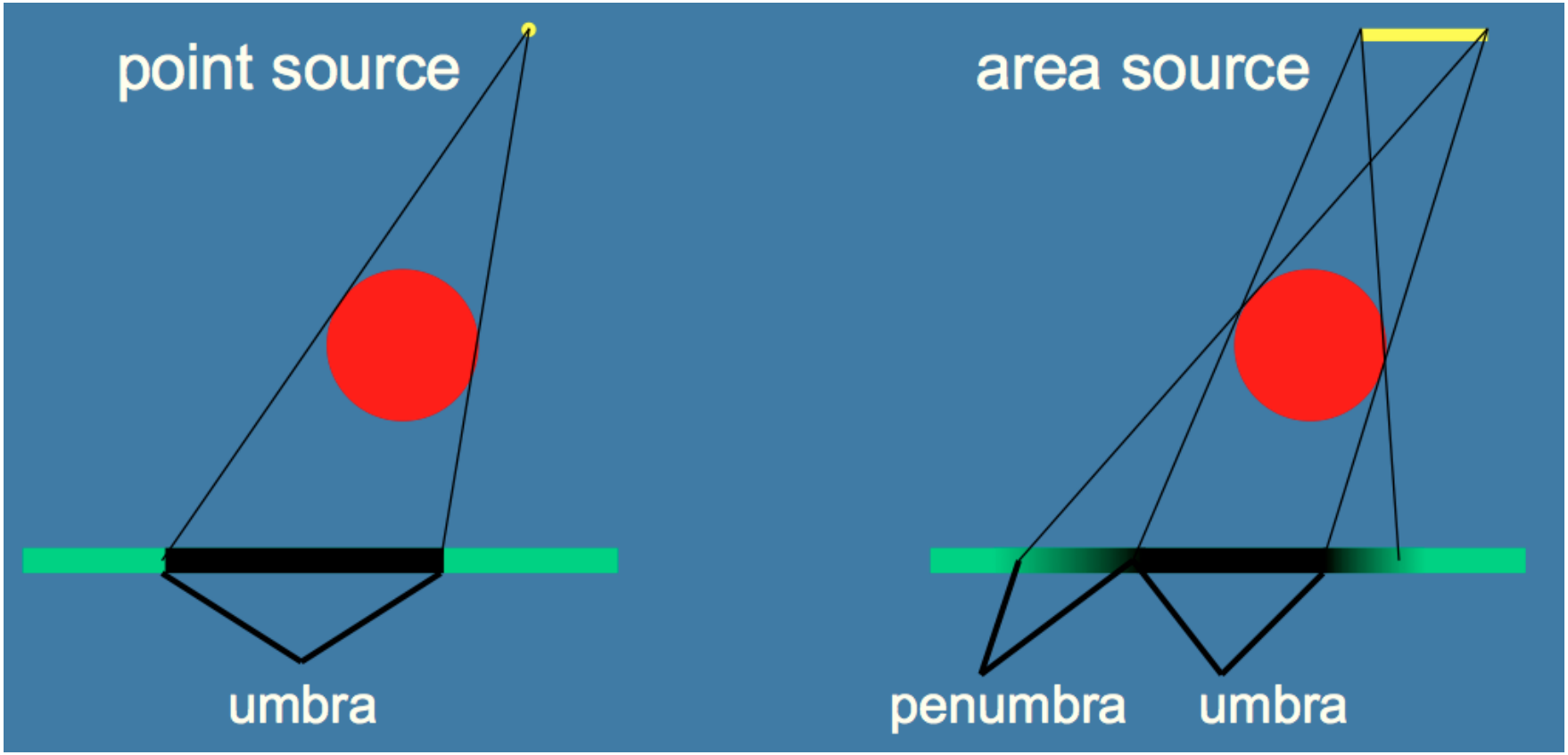


Image Courtesy of Moeller

Soft Shadow

- Approximate area light with multiple point light sources
- Jitter Sampling

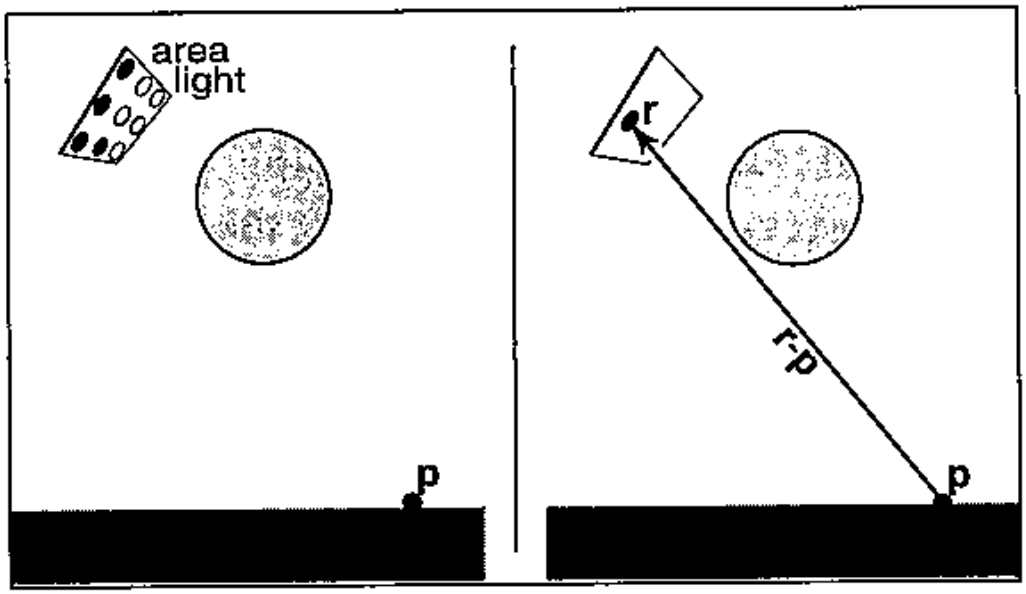
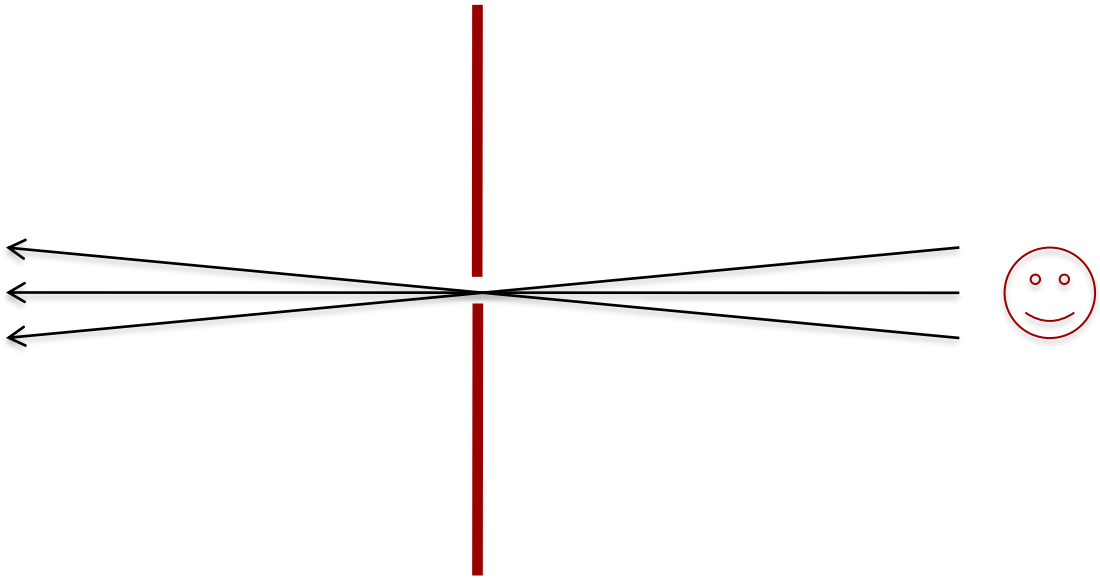


Image Courtesy of Shirley et al.

Image Courtesy of Dimson

Depth of Field

- Multi-sample with in-focus plane



focal plane

Acceleration of Ray Tracing

- Efficient spatial data structure
  - Bounding box hierarchy
  - Octree
  - BSP (binary space partitioning) tree
- Multi-threading
  - Each ray can be traced independently
- GPU
  - Large scale parallelism

Questions?

Image courtesy of NVIDIA