Precept 7
Raytracer

Huiwen Chang • Mar 13 2016

Overview

Ray Intersection

- Triangle
- Sphere
- Cylinder
- Cone

Textures

- Checker Board
- Special

Shadow

Animation

Ray Intersection – *Triangle*

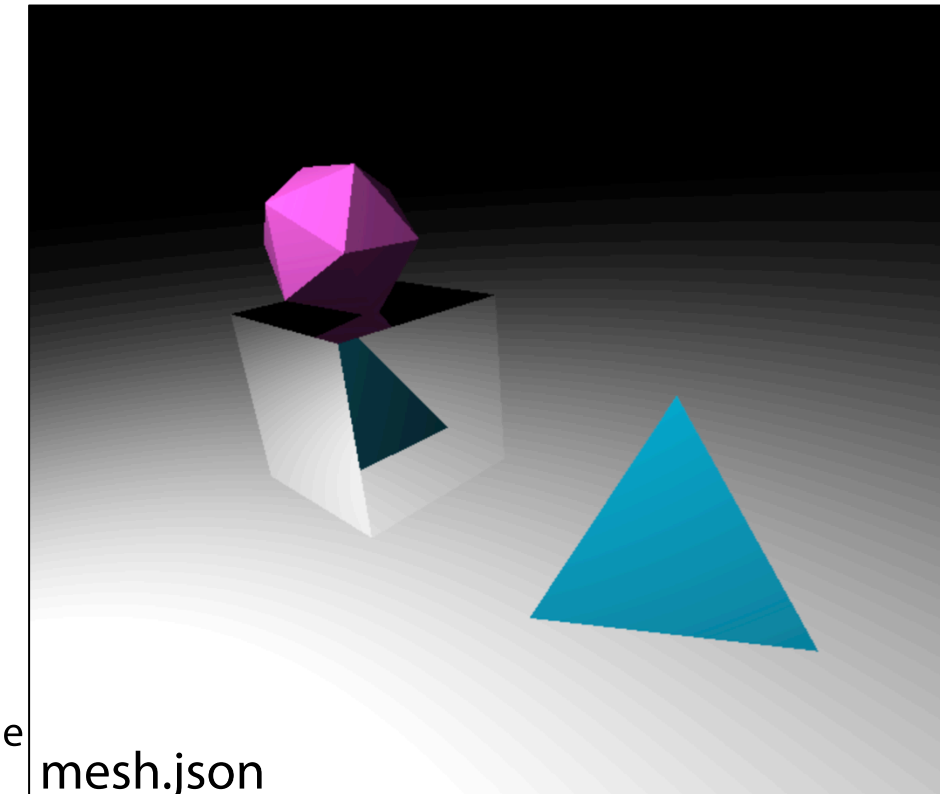
Plane:

$$N \cdot x = Dist$$

Triangle:

- Algebraic
- Geometric

For the normal on triangle at intersection:
no requirement for pointing inside or outside



Ray Intersection – *Triangle*

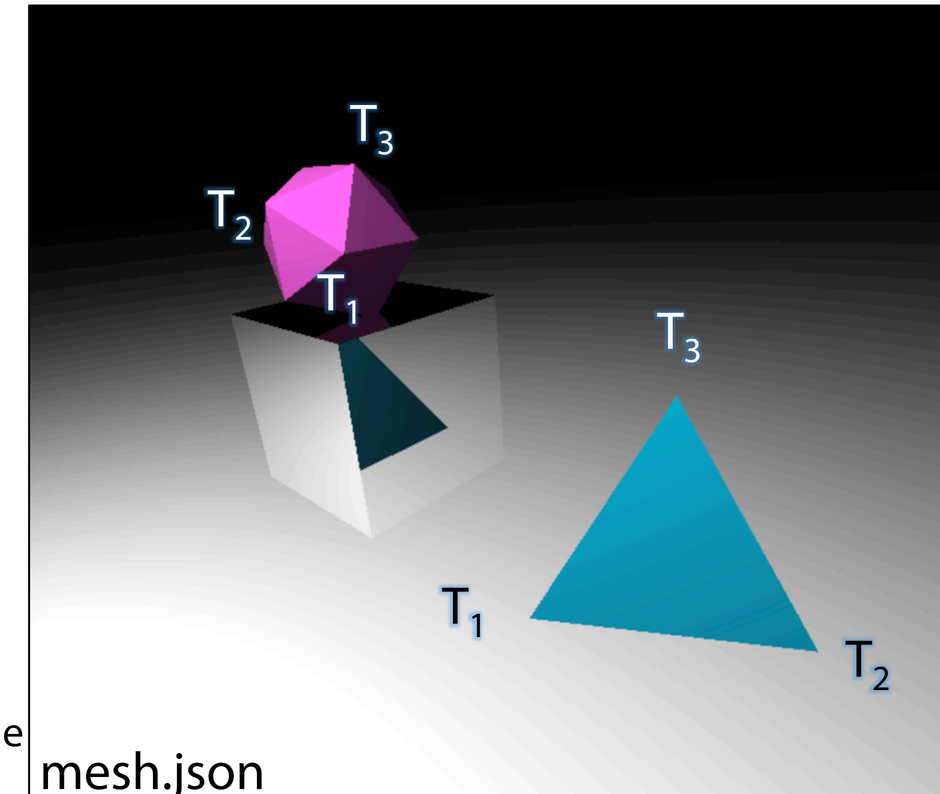
Plane:

$$N \cdot x = Dist$$

Triangle:

- Algebraic
- Geometric

For the normal on triangle at intersection:
no requirement for pointing inside or outside



Ray Intersection – *Triangle*

Geometric

$N = \text{normalize}(\ (T_2 - T_1) \times (T_3 - T_1)\ )$

$P \ldots$

$\alpha = \text{Area}(T_1T_2P) \ / \ \text{Area}(T_1T_2T_3)$

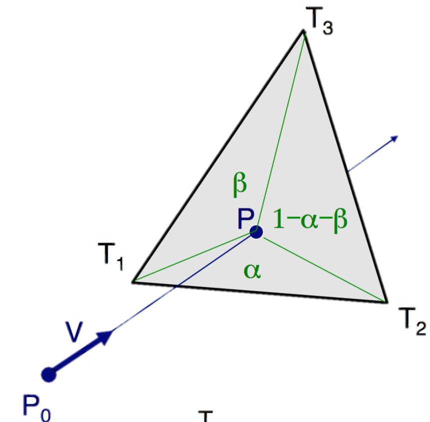
$\beta = \text{Area}(T_1PT_3) \ / \ \text{Area}(T_1T_2T_3)$

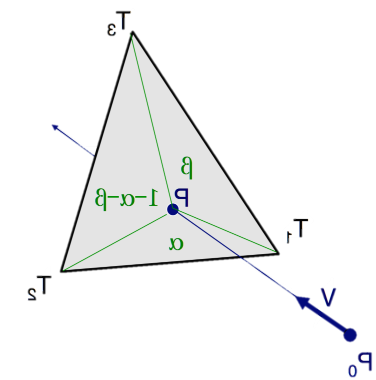
$\text{Area}(T_1T_2T_3) = \frac{1}{2} \ ||(T_2 - T_1) \times (T_3 - T_1)\ ||$

$\qquad\qquad\qquad\quad = \frac{1}{2} < (T_2 - T_1) \times (T_3 - T_1)\ , N>$

$\text{Area}(T_1T_2P) \ = \frac{1}{2} < (T_2 - T_1) \times (P - T_1)\ , N>$

$\text{Area}(T_1PT_3) \ = \frac{1}{2} < (P - T_1) \times (T_3 - T_1)\ , N>$





Ray Intersection – *Sphere*

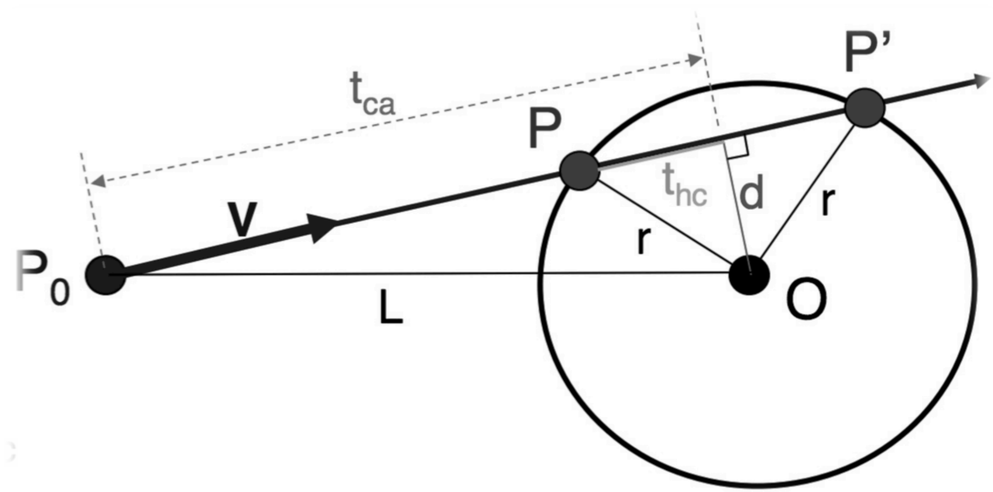
Look out reflecting/refracting ray!

$$P = P_0 + tV$$

$$t_1 = t_{ca} - t_{hc}$$ ✔

or

$$t_2 = t_{ca} + t_{hc}$$ ✗



Ray Intersection – *Sphere*

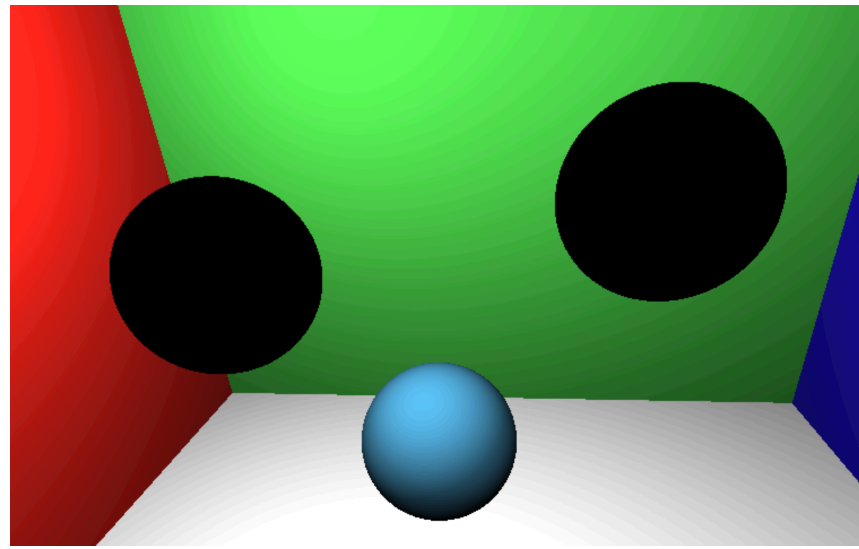
If we always choose the "nearest" one:

$$P = P_0 + tV$$

$$t_1 = t_{ca} - t_{hc}$$ ✔

or

$$t_2 = t_{ca} + t_{hc}$$ ✗



When a reflective/refractive ray bounds at an intersection, t1 = 0

Ray Intersection – *Sphere*

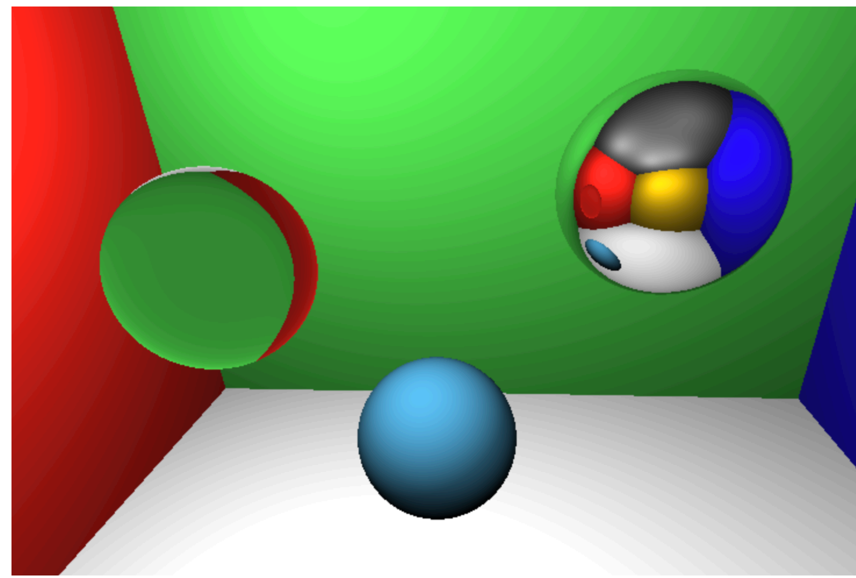
Check the “nearest valid(positive)” intersection

$$t_1 = t_{ca} - t_{hc}$$
$$t_2 = t_{ca} + t_{hc}$$

If (t1 > 0) return t1;

elseif (t2 > 0) return t2;

return INFINITY;



Ray Intersection – *Sphere*

Check the "nearest valid(positive)" intersection

$$t_1 = t_{ca} - t_{hc}$$
$$t_2 = t_{ca} + t_{hc}$$

Same for other primitives

```
If (t1 > 0) return t1;
elseif (t2 > 0) return t2;
return INFINITY;
```

Ray Intersection –*Cylinder*

1. Intersect with open cylinder

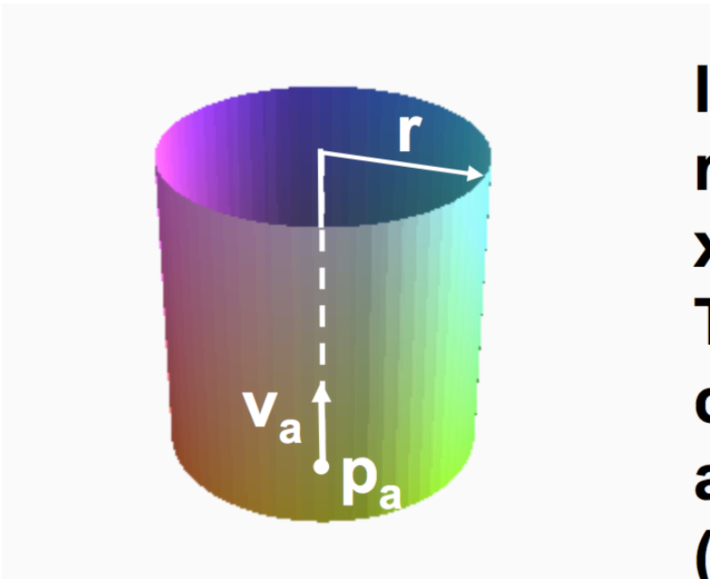
& Check if the intersection is between the planes

2. Intersect with two caps

3. Out of all intersections, choose the one with

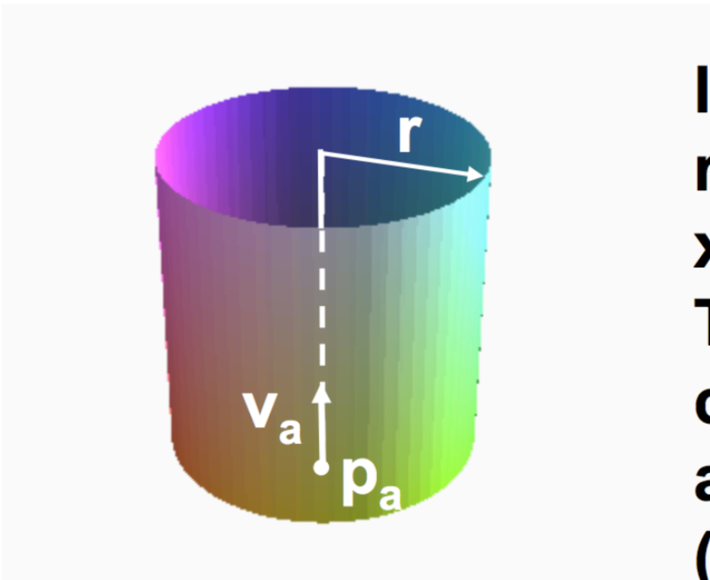
minimal dist

Ray Intersection – *Infinite Cylinder*



Infinite cylinder along y of radius r axis has equation $x^2 + z^2 - r^2 = 0$.
The equation for a more general cylinder of radius r oriented along a line $p_a + v_a t$:
$(q - p_a - (v_a, q - p_a)v_a)^2 - r^2 = 0$
where $q = (x,y,z)$ is a point on the cylinder.

Ray Intersection – *Infinite Cylinder*



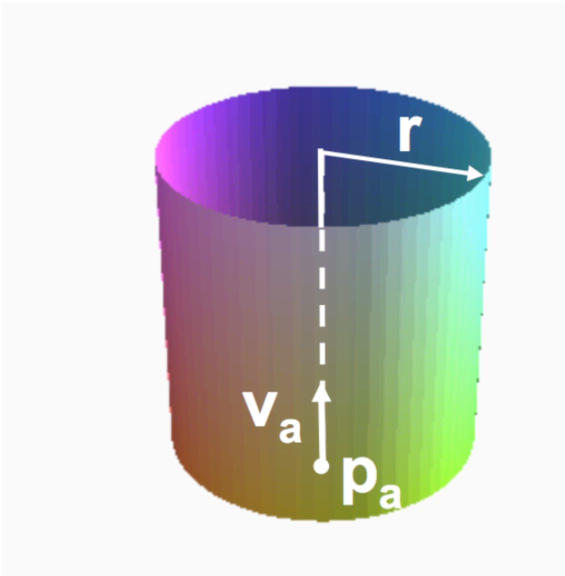
Infinite cylinder along y of radius r axis has equation $x^2 + z^2 - r^2 = 0$.

The equation for a more general cylinder of radius r oriented along a line $p_a + v_a t$:

$$(q - p_a - (v_a, q - p_a)v_a)^2 - r^2 = 0$$

where q = (x,y,z) is a point on the cylinder.

Ray Intersection – *Infinite Cylinder*



To find intersection points with a ray p + vt, substitute q = p + vt and solve:

$$(\mathbf{p} - \mathbf{p}_a + \mathbf{v}t - (\mathbf{v}_a, \mathbf{p} - \mathbf{p}_a + \mathbf{v}t)\mathbf{v}_a)^2 - \mathbf{r}^2 = 0$$

reduces to $\mathbf{A}t^2 + \mathbf{B}t + \mathbf{C} = 0$

with

$$\mathbf{A} = (\mathbf{v} - (\mathbf{v}, \mathbf{v}_a)\mathbf{v}_a)^2$$

$$\mathbf{B} = 2(\mathbf{v} - (\mathbf{v}, \mathbf{v}_a)\mathbf{v}_a, \Delta\mathbf{p} - (\Delta\mathbf{p}, \mathbf{v}_a)\mathbf{v}_a)$$

$$\mathbf{C} = (\Delta\mathbf{p} - (\Delta\mathbf{p}, \mathbf{v}_a)\mathbf{v}_a)^2 - \mathbf{r}^2$$

where $\Delta\mathbf{p} = \mathbf{p} - \mathbf{p}_a$

1999, Denis Zorin

Ray Intersection – *Cylinder*

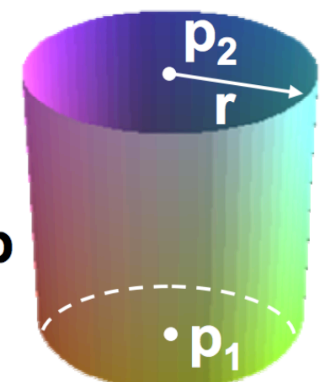
POV -ray like cylinder with caps : cap centers at p_1 and p_2, radius r.

Infinite cylinder equation: $p_a = p_1$, $v_a = (p_2 - p_1)/|p_2 - p_1|$

The finite cylinder (without caps) is described by equations:

$$(q - p_a - (v_a, q - p_a)v_a)^2 - r^2 = 0 \text{ and } (v_a, q - p_1) > 0 \text{ and}$$

$$(v_a, q - p_2) < 0$$



The equations for caps are:

$(v_a, q - p_1) = 0, (q - p_1)^2 < r^2$ **bottom cap**

$(v_a, q - p_2) = 0, (q - p_2)^2 < r^2$ **top cap**

Ray Intersection – *Cone*

Similar to cylinder:

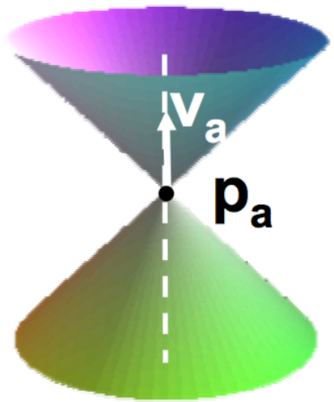
1. Intersect with open cone

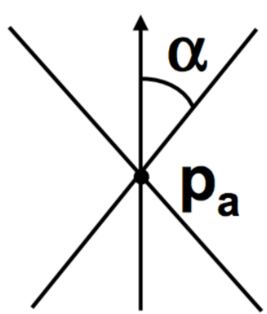
& Check if the intersection is between the planes

2. Intersect with the cap

3. Out of all intersections, choose the one with minimal dist

Ray Intersection – *Infinite Cone*



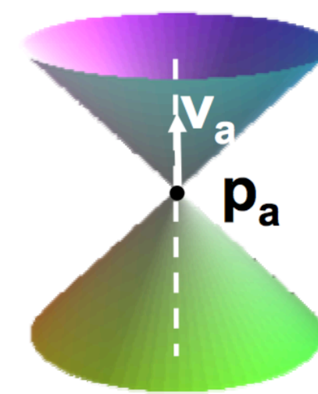


Infinite cone along y with apex half-angle α has equation $x^2 + z^2 - y^2 = 0$.
The equation for a more general cone oriented along a line $p_a + v_a t$, with apex at p_a:
$\cos^2\alpha\,(q - p_a - (v_a,q - p_a)v_a)^2 - \sin^2\alpha\,(v_a,q - p_a)^2 = 0$
where $q = (x,y,z)$ is a point on the cone, and v_a is assumed to be of unit length.

Ray Intersection – *Infinite Cone*

Similar to the case of the cylinder: substitute q = p+vt into the equation, find the coefficients A, B, C of the quadratic equation, solve for t. Denote $\Delta p = p - p_a$.

$$\cos^2\alpha\,(vt + \Delta p - (v_a, vt + \Delta p)v_a)^2 - \sin^2\alpha\,(v_a, vt + \Delta p)^2 = 0$$



$$A = \cos^2\alpha(v - (v, v_a)v_a)^2 - \sin^2\alpha(v, v_a)^2$$

$$B = 2\cos^2\alpha(v - (v, v_a)v_a, \Delta p - (\Delta p, v_a)v_a) - 2\sin^2\alpha(v, v_a)(\Delta p, v_a)$$

$$C = \cos^2\alpha(\Delta p - (\Delta p, v_a)v_a)^2 - \sin^2\alpha(\Delta p, v_a)^2$$

Ray Intersection – *Infinite Cone*

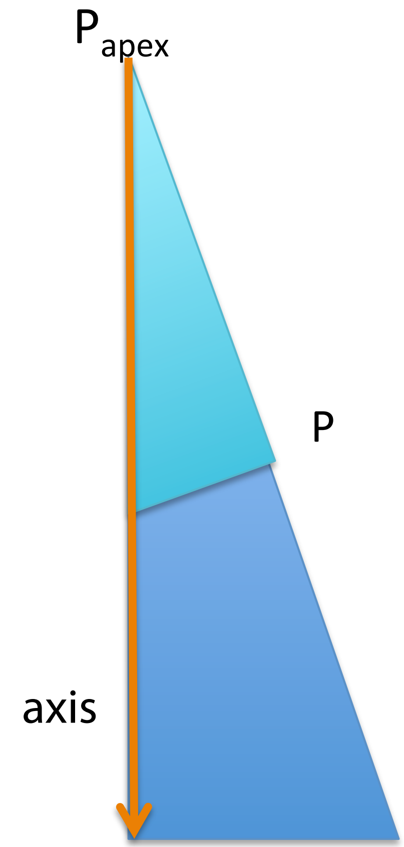
In the assignment,

N_{axis} = normalize(axis);

To get the normal in the infinite cone:

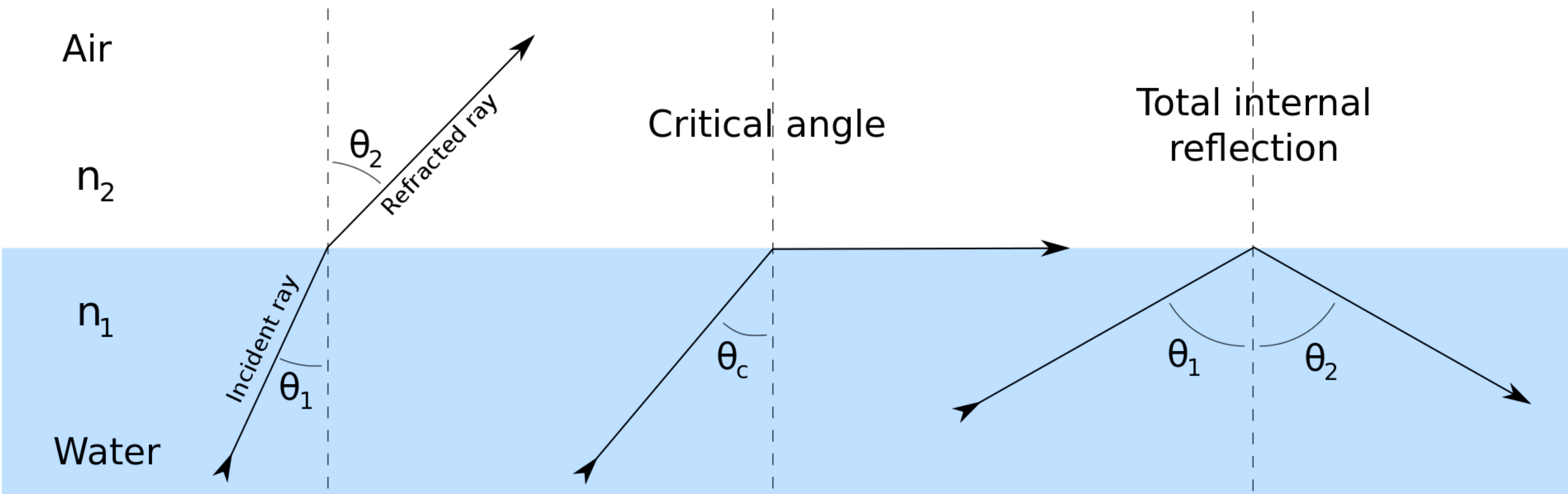
$E = P - P_{apex}$

Normal = normalize(E - ||E||/COS α * N_{axis});



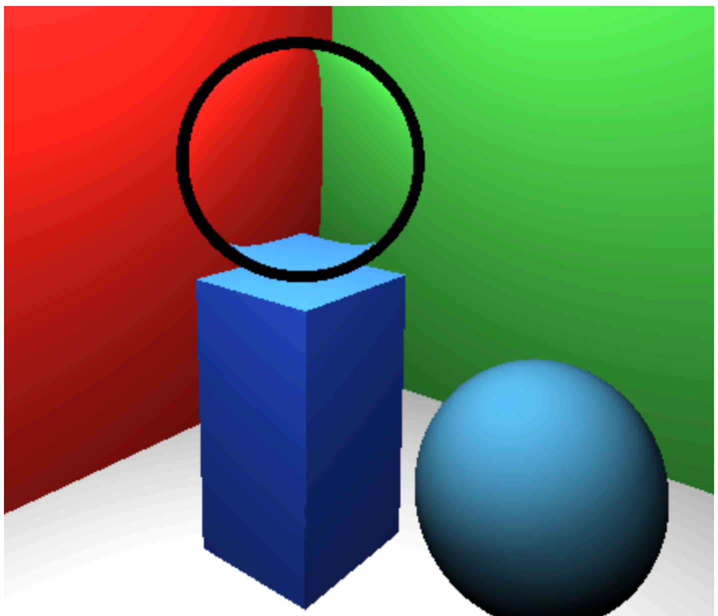
Refraction

From a medium with a higher refractive index to a lower one

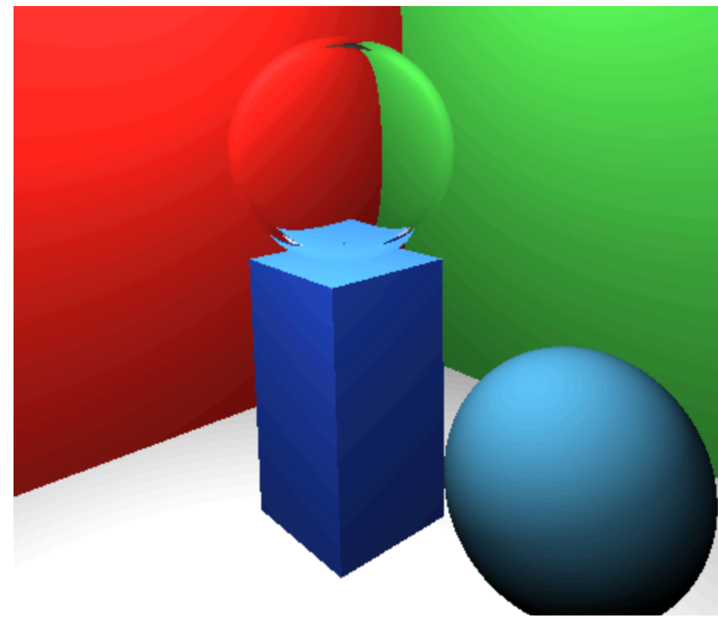


Refraction

When >critical angle,
you could let refraction return black ;



Or you could return internal reflection;



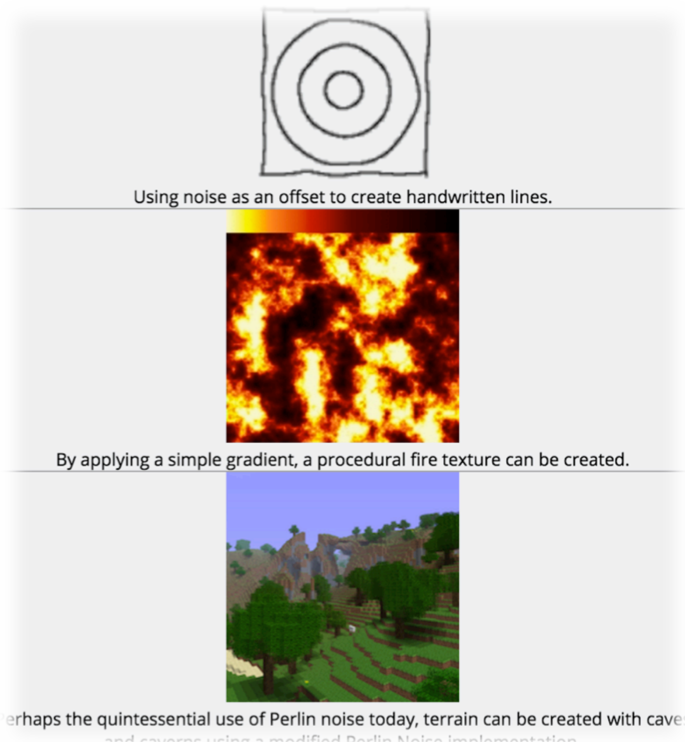
In this scene, the "refraction ratio" of the sphere = 1.1

Texture

- Checkerboard
  - 2D: floor(x)+floor(y) is odd/even
  - 3D: view normal as the z-axis for the new coordinate, then find x, y

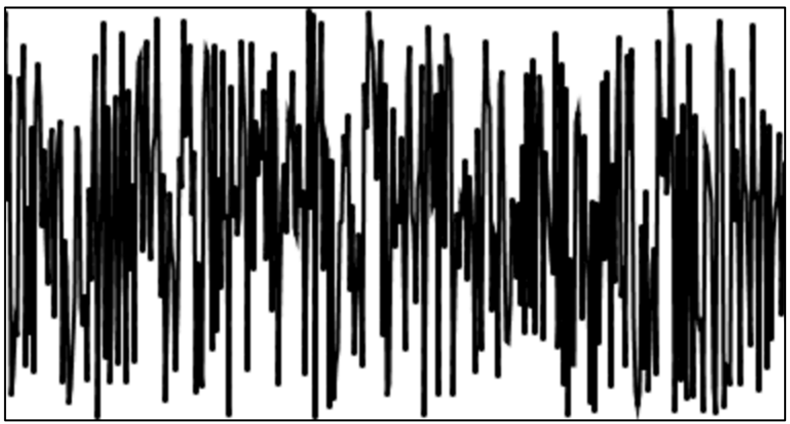
Texture

- Special: Perlin noise

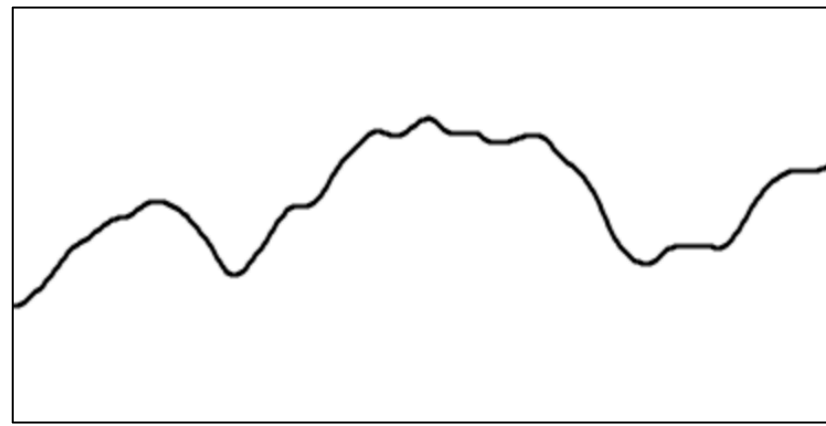


Perlin Noise

Random

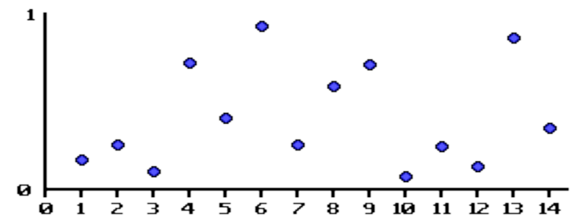


Perlin

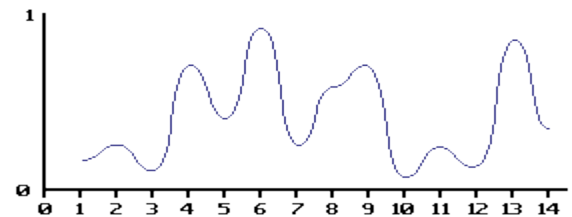


Idea

1. Generate random values at grid points.

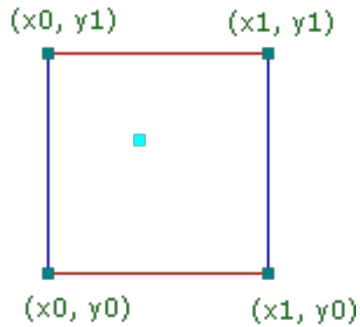


2. Interpolate smoothly between these values.



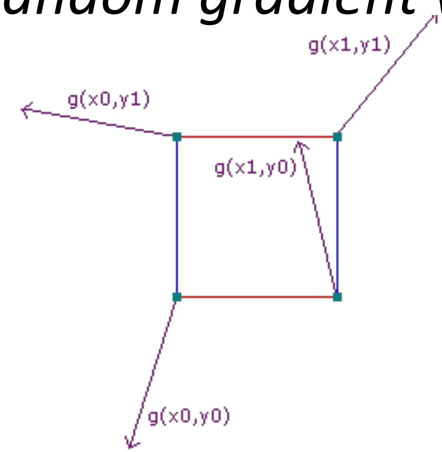
Perlin Noise

Step 1 Cut to grids



[Code for Random]()

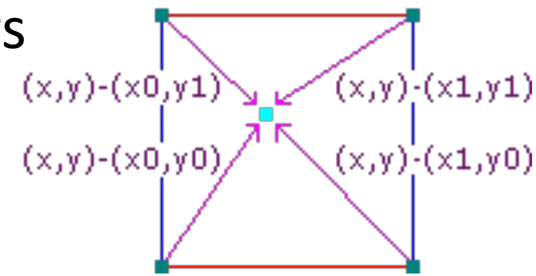
Step 2 *Pseudorandom gradient vector*.



[NEW] Randomly pick from
(1,1,0),(-1,1,0),(1,-1,0),(-1,-1,0),
(1,0,1),(-1,0,1),(1,0,-1),(-1,0,-1),
(0,1,1),(0,-1,1),(0,1,-1),(0,-1,-1)

Perlin Noise

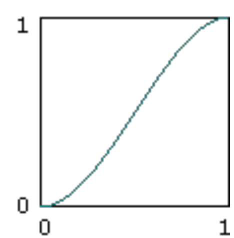
Step 3 Distance vectors

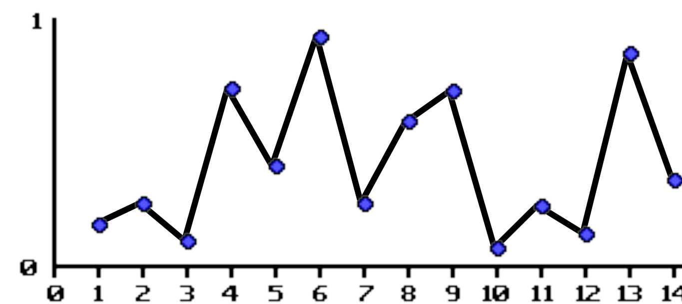


Step 4 Dot product on each grid point <Vgrad, Vdist>

Step 5 Interpolation

fade function: $6t^5-15t^4+10t^3$





Hard Shadow

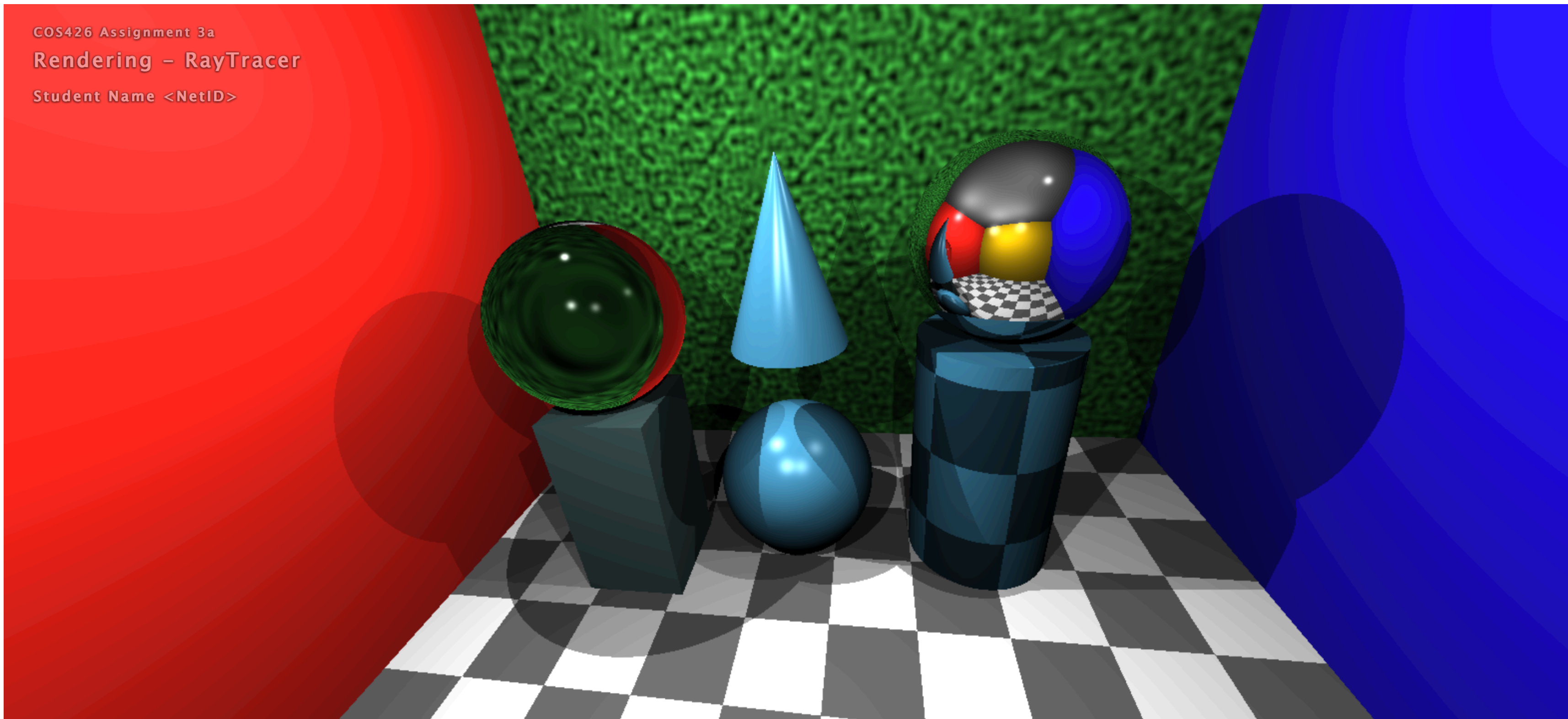
lightVec is the vector from that position to that light

Step 1 generate a ray from position to light

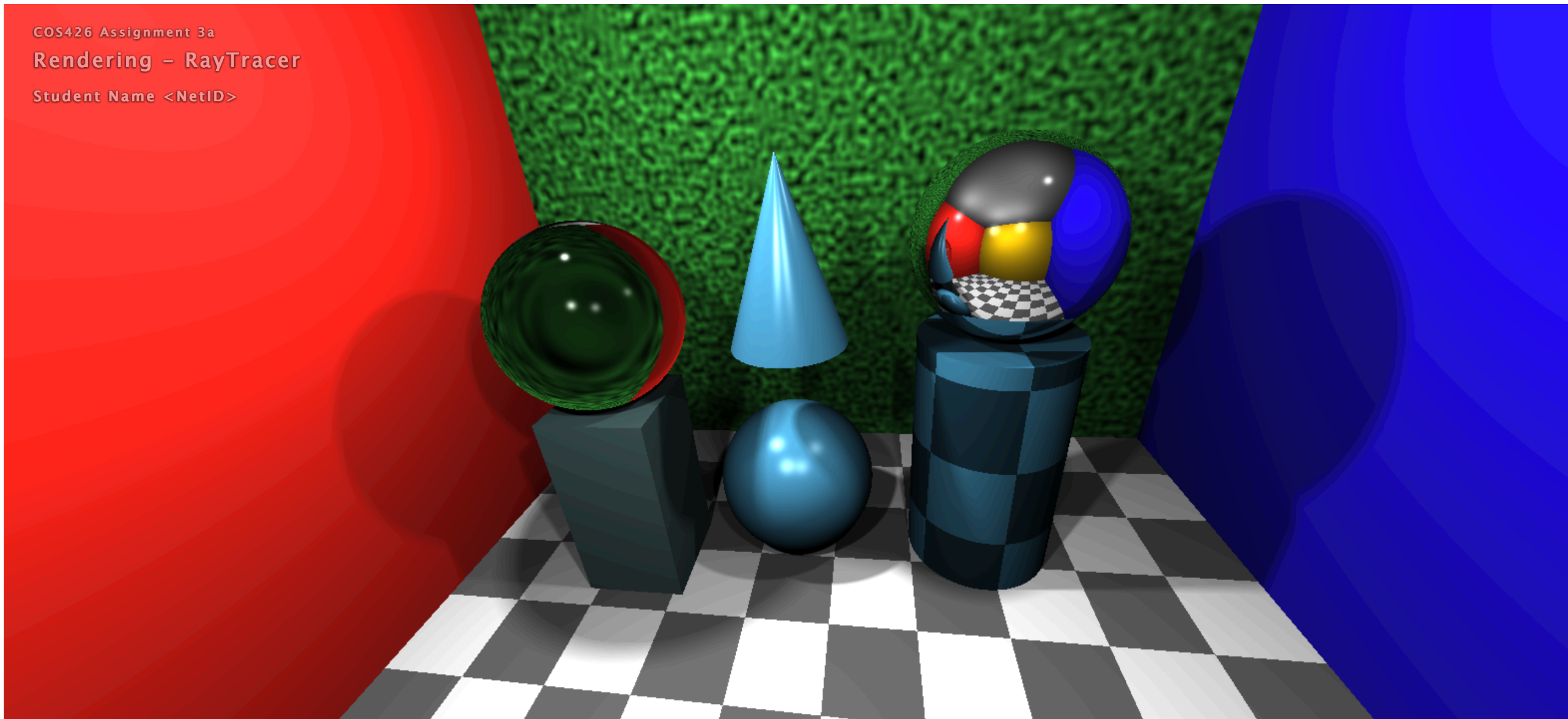
Step 2 find intersection length(distance)

Step 3 if distance is positive and before hitting the light (smaller than the length of lightVec) return true;

Hard Shadow


COS426 Assignment 3a
Rendering – RayTracer
Student Name <NetID>

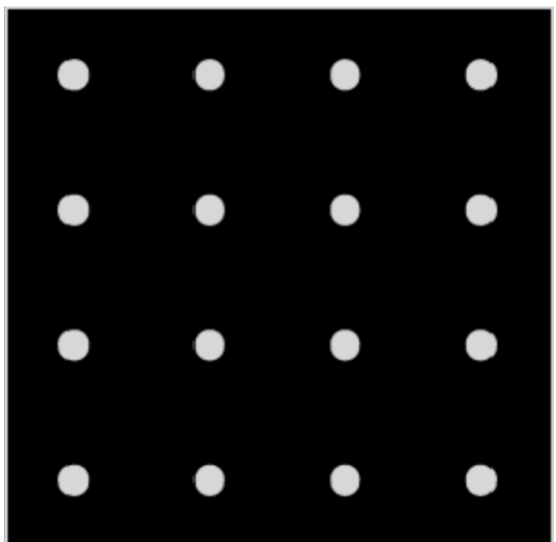
Soft Shadow


COS426 Assignment 3a
Rendering – RayTracer
Student Name <NetID>

Soft Shadow

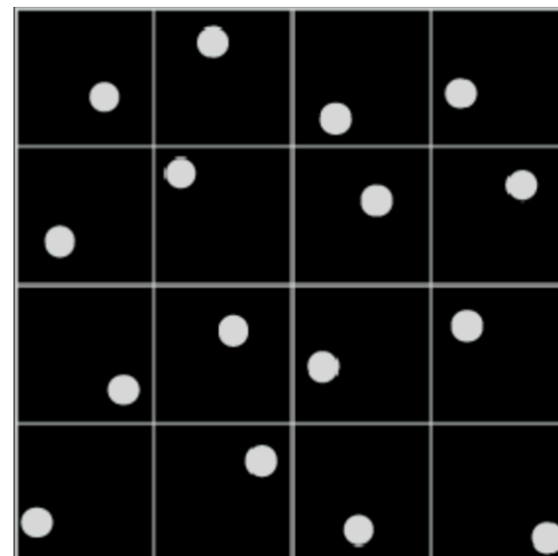
Shoot rays around the point light - grid

- Uniformly sample



✗

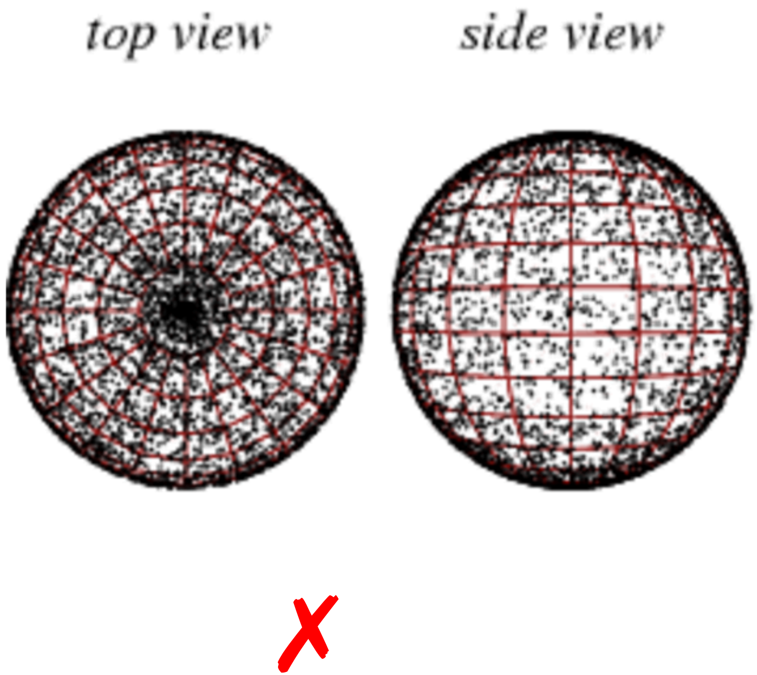
- Randomly sample



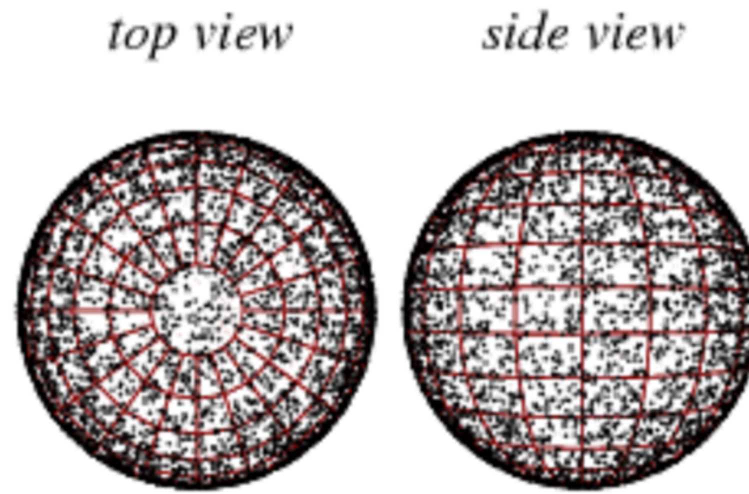
Soft Shadow

Shoot rays around the point light - sphere

- Sample by θ *[0, 2π)* and ϕ *[0, π)*



- Sample by *solid angle*



Soft Shadow

```
float pointShadowRatio ( vec3 pos, vec3 lightVec ) {
    float count = 0.0;
    for i = 1...k
    for j = 1...k
        Randomly Sample a new light ray around the original light
        if  not pointInShadow(pos, newLightVec)
          count += 1.0;
    return count/float(k*k);
}
```

In `function getLightContribution`

Comment `if pointInshadow return black;`

Modify `return` to `contribution * pointShadowRatio (or diffuseColor * pointShadowRatio)`

Soft Shadow

Randomly Sample :

(X, y, z) = (

$$2x_1\sqrt{1-x_1^2-x_2^2}$$
$$2x_2\sqrt{1-x_1^2-x_2^2}$$
$$1-2\left(x_1^2+x_2^2\right)$$

) random x1, x2 in [-1, 1]

or (

$$x=\sqrt{1-u^2}\ \cos\theta$$
$$y=\sqrt{1-u^2}\ \sin\theta$$
$$z=u,$$

) random u in [-1, 1], θ in [0, 2π)

Animation

In intersection with sphere,

Center.Y += K1*abs(round(frame/K2) – frame/K2))