Precept 9

Huiwen Chang

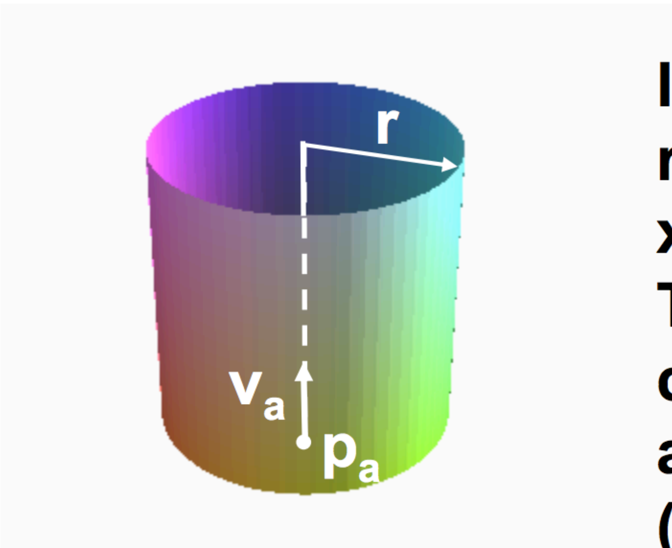
Raytracing

- Scene.js
- Raytracer.js
  - How to add new scene
- Frag/vertex: Shader.txt

Raytracing

- Function for Each Object
  - Find the intersection(ray, object)
  - Find the normal

Infinite cylinder-ray intersections



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.

Infinite cylinder-ray intersections

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

$$(p - p_a + vt - (v_a, p - p_a + vt)v_a)^2 - r^2 = 0$$

reduces to $At^2 + Bt + C = 0$

with

$$A = (v - (v, v_a)v_a)^2$$

$$B = 2(v - (v, v_a)v_a, \Delta p - (\Delta p, v_a)v_a)$$

$$C = (\Delta p - (\Delta p, v_a)v_a)^2 - r^2$$

where $\Delta p = p - p_a$

Finite cylinder-ray intersections

A finite cylinder with caps can be constructed as the intersection of an infinite cylinder with a slab between two parallel planes, which are perpendicular to the axis.

To intersect a ray with a cylinder with caps:

- intersect with the infinite cylidner;
- check if the intersection is between the planes;
- intersect with each plane;
- determine if the intersections are inside caps;
- out of all intersections choose the on with minimal t

Finite cylinder-ray intersections

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:

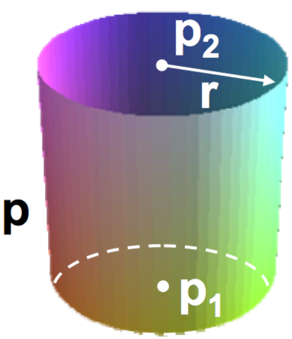
$$(q - p_a - (v_a, q - p_a)v_a)^2 - r^2 = 0 \text{ and } (v_a, q - p_1) > 0 \text{ and}$$

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

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



cylinder-ray intersections

Algorithm with equations:

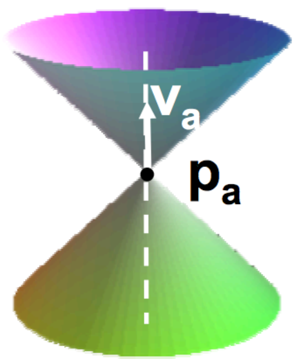
Step 1: Find solutions t_1 and t_2 of $At^2 + Bt + C = 0$

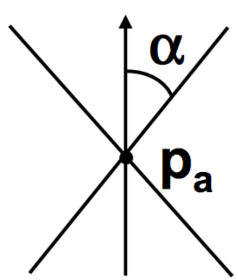
if they exist. Mark as intersection candidates the one(s) that are nonnegative and for which $(v_a, q_i - p_1) > 0$ and $(v_a, q_i - p_2) < 0$, where $q_i = p + v\, t_i$

Step 2: Compute t_3 and t_4, the parameter values for which the ray intersects the upper and lower planes of the caps.
If these intersections exists, mark as intersection candidates those that are nonegative and $(q_3 - p_1)^2 < r^2$ (respectively $(q_4 - p_2)^2 < r^2$).

In the set of candidates, pick the one with min. t.

Infinite cone-ray intersections





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.

Infinite cone-ray intersections

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 $\Delta p = p - p_a$.

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

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

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

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

Finite cone-ray intersections

A finite cone with caps can also be constructed as intersection of an infinite cone with a slab.

Intersections are computed exactly in the same way as for the cylinder, but instead of the quadratic equation for the infinite cylinder the equation for the infinite cone is used, and the caps may have different radii.

Both for cones and cylinders intersections can be computed somewhat more efficiently if we first transform the ray to a coordinate system aligned with the cone (cylinder). This requires extra programming to find such transformation.

Finite cone-ray intersections

POV-ray cone: cap centers (base point and cap point) at p_1 and p_2, cap radii r_1 and r_2.

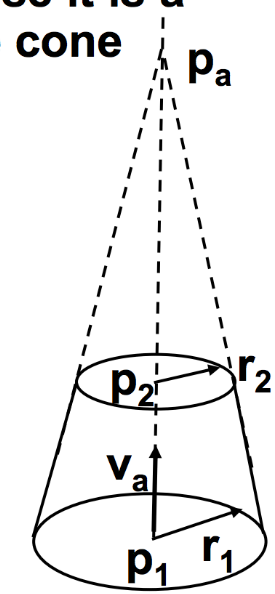
Then, assuming r_1 not equal to r_2 (otherwise it is a cylinder) in the equation of the infinite cone

apex: $p_a = p_1 + r_1(p_2 - p_1)/(r_1 - r_2)$;

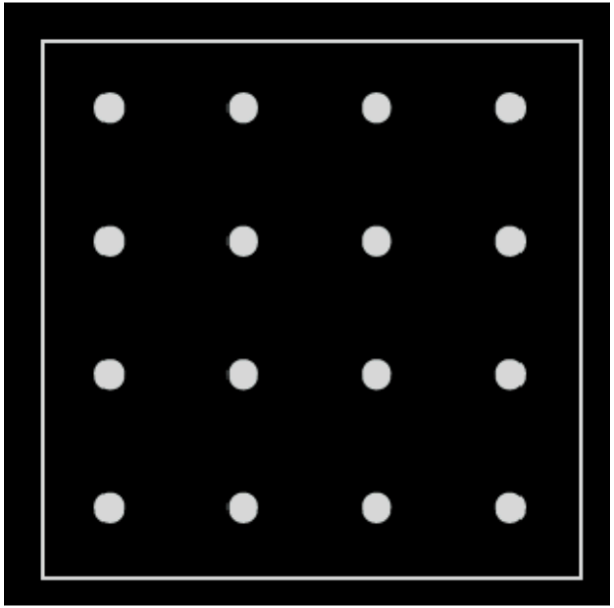
axis direction: $v_a = (p_2 - p_1)/|p_2 - p_1|$;

apex half-angle:

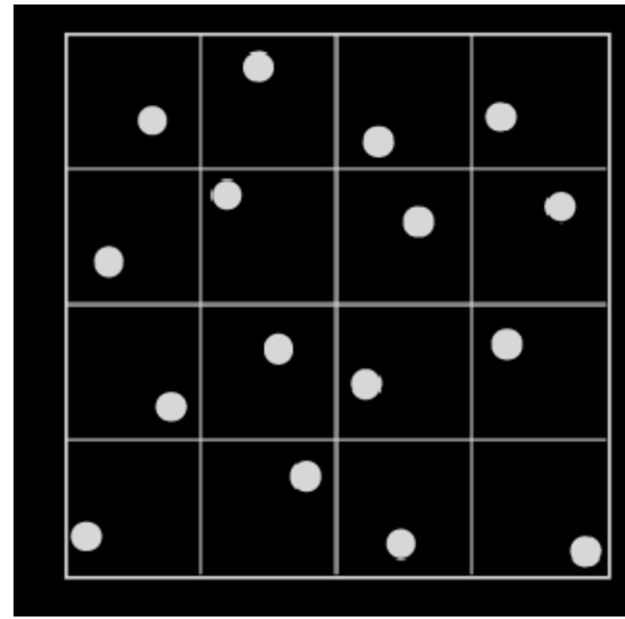
$tg\,\alpha = (r_1 - r_2)/|p_2 - p_1|$



Soft shadow



Soft shadow



Random?

Check out this site proposal, it might interest you: – Dan the Man Nov 18 '13 at 15:35

add a comment

8 Answers

active oldest votes

130

+100

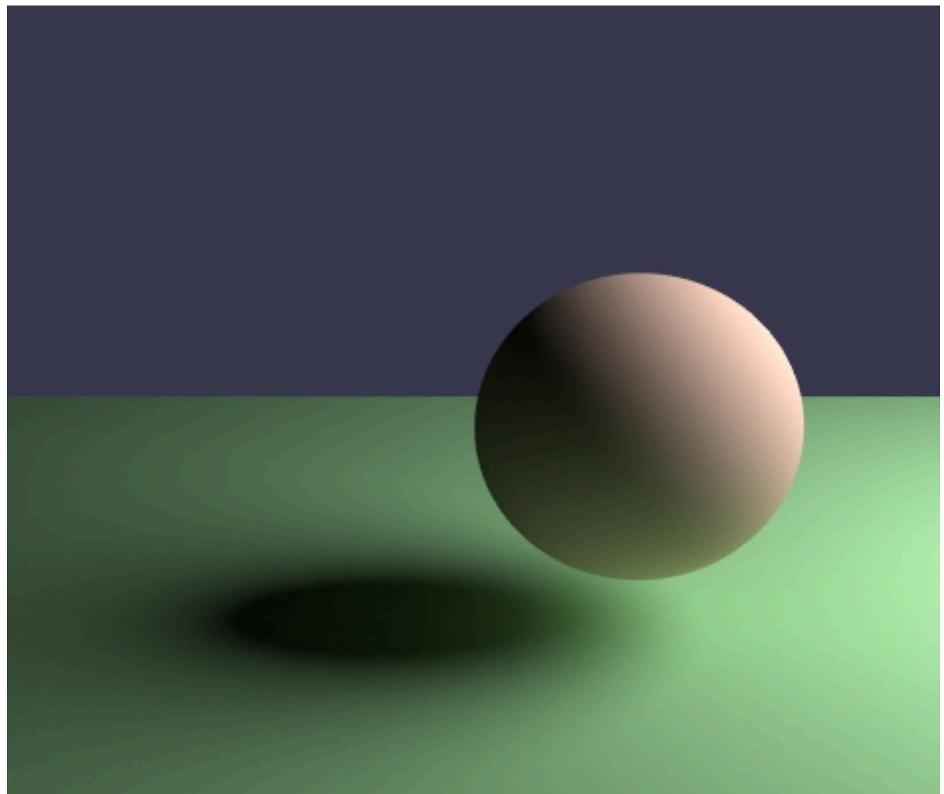
For very simple pseudorandom-looking stuff, I use this oneliner that I found on the internet somewhere:

```
float rand(vec2 co){
    return fract(sin(dot(co.xy ,vec2(12.9898,78.233))) * 43758.5453);
}
```

You can also generate a noise texture using whatever PRNG you like, then upload this in the normal fashion and sample the values in your shader; I can dig up a code sample later if you'd like.

Also, check out this file for GLSL implementations of Perlin and Simplex noise, by Stefan Gustavson

Soft shadow



Rasterizer

- shader
  - Need varying to send normal from vertex-shader to fragment-shader
  - Need varying for pixel position
  - Calculate eye position by cameraPosition