Ray tracing

Ray casting/ray tracing

Iterate over pixels, not objects.

Effects that are difficult with Z-buffer, are easy with ray tracing: shadows, reflections, transparency, procedural textures and objects.

Assume image plane is placed in the virtual space (e.g. front plane of the viewing frustum).

Algorithm:

```
for each pixel
    shoot a ray r from the camera to the pixel
    intersect with every object
    find closest intersection
```

Ray tracing

For each pixel shoot a ray R **from camera;**

```
    pixel = TraceRay(R)
```

The recursive ray tracing procedure:

`RGBvalue TraceRay(Ray R)`

- **shoot rays to all light sources;**
- **for all visible sources, compute RGB values** r_i
- **shoot reflected ray** R_{refl}; r_{refl}=TraceRay(R_{refl})
- **shoot refracted ray** R_{trans};r_{trans}= TraceRay(R_{trans})
- **compute resulting RGB value from** r_i, r_{refl}, r_{trans} **using the lighting model**

Some primitives

Finite primitives:

- **polygons**
- **spheres, cylinders, cones**
- **parts of general quadrics**

Infinite primitives:

- **planes**
- **infinite cylinders and cones**
- **general quadrics**

A finite primitive is often an intersection of an infinite with an area of space

Intersecting rays with objects

General approach:

Use whenever possible the implicit equation $F(q) = 0$ of the object or object parts. Use parametric equation of the line of the ray, $q = p+vt$.

Solve the equation $F(p+vt) = 0$ to find possible values of t. Find the minimal nonnegative value of t to get the intersection point (checking that t is nonegative is important: we want intersections with the ray starting from p, not with the whole line!

Scene Language

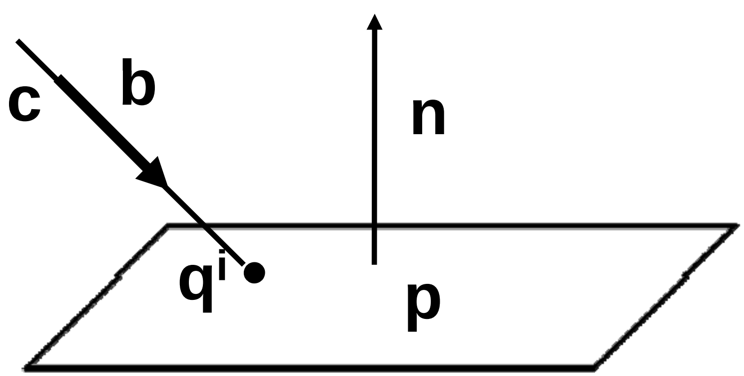
POV ray input language example example

```
camera {
  location  <0, 0, -8>
 look_at   <0, 0, 0>
}
sphere { <0.0, 0.0, 0.0>, 2
  finish {
    ambient 0.2
    diffuse 0.8
    phong 1
  }
  pigment { color red 1 green 0 blue 0 }
}
box { <-2.0, -0.2, -2.0>, <2.0, 0.2, 2.0>
   finish {
     ambient 0.2
     diffuse 0.8
   }
   pigment { color red 1 green 0 blue 1 }
   rotate <-20, 30, 0>
}

light_source { <-10, 3, -20> color red 1 green 1 blue 1 }
```

Intersecting a line and a plane

Same old trick: use the parametric equation for the line, implicit for the plane. In the case of a pixel ray, b = p(i,j)-c



$$((c + bt^i - p) \cdot n) = 0$$

$$t^i = -\frac{((c-p) \cdot n)}{(b \cdot n)}$$

Check for zero in the denominator; t^i should be positive for the intersection to be in front of the camera.

Intersecting a ray with a sphere

Sphere equation: $(q-c)^2 - r^2 = 0$

For a ray $q = p + vt$, we get $((p-c) + vt)^2 - r^2 = 0$

$(p-c)^2 + 2(p-c)\cdot v\, t + v^2 + t^2 - r^2 = 0$

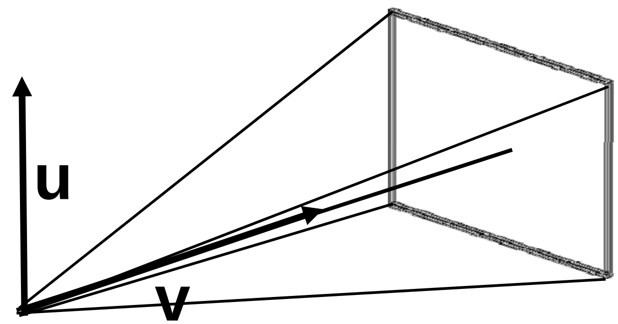
This quadratic equation in t may have no solutions (no intersection) or two (possibly coinciding) solutions (entry and exit points).

The correct point to return is the one that is closest to ray origin.

Pixel rays

Goal: Find direction of the ray to the center of the pixel (i,j). Let camera parameters be

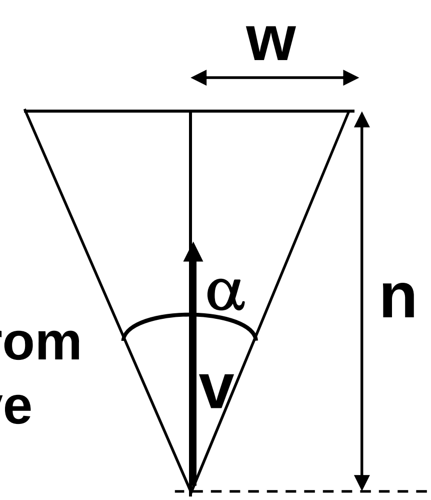
c	**position**
α	**horizontal field of view**
v	**viewing direction**
u	**up direction**
s	**aspect ration**



Then the image half-width in the "virtual world" units is

$$\mathbf{w} = \mathbf{n}\ \mathbf{tg}\frac{\alpha}{2}$$

The half-height is $\mathbf{h} = \mathbf{sn}\ \mathbf{tg}\frac{\alpha}{2}$



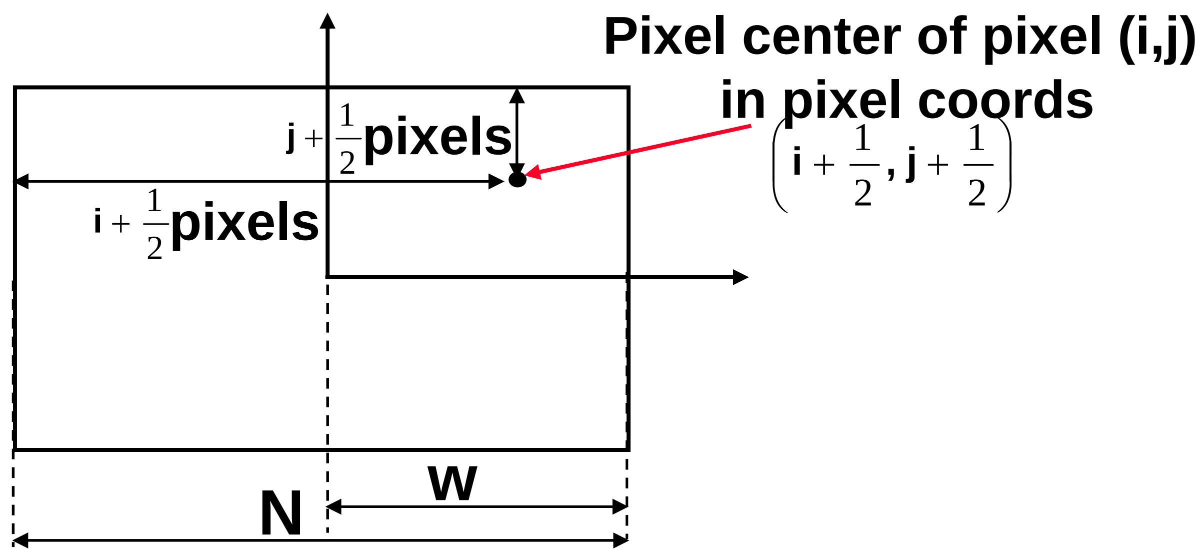
View from above

Pixel rays

From coordinates in pixel units to virtual world coordinates in image plane:

Pixel size:

$$\frac{2\mathbf{w}}{\mathbf{N}} \times \frac{2\mathbf{h}}{\mathbf{M}}$$



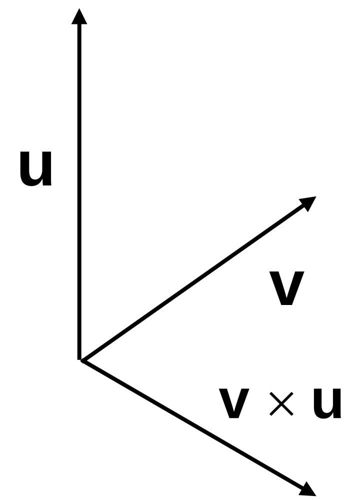
Displacements of the pixel from the image center in virtual space units:

$$\mathbf{h} - \left(\mathbf{j}+\frac{1}{2}\right)\frac{2\mathbf{h}}{\mathbf{M}}, \quad \left(\mathbf{i}+\frac{1}{2}\right)\frac{2\mathbf{w}}{\mathbf{N}} - \mathbf{w}$$

Pixel rays

Virtual world coordinates of pixel (i,j): image center + displacements.

Image center: c + vn



$$\mathbf{pixel}\ (\mathbf{i},\mathbf{j}) = \mathbf{c} + \mathbf{vn} + \left(\mathbf{h} - \left(\mathbf{j} + \frac{1}{2}\right)\frac{2\mathbf{h}}{\mathbf{M}}\right)\mathbf{u} + \left(\left(\mathbf{i} + \frac{1}{2}\right)\frac{2\mathbf{w}}{\mathbf{N}} - \mathbf{w}\right)\mathbf{v} \times \mathbf{u}$$