
Visibility algorithms

© 1999, Denis Zorin

Visibility algorithms

Visibility problem:

given a collection of objects in 3D and a camera,
for each point (x,y) in the image plane determine
the closest object point that projects to (x,y) .

Different types of algorithms:

- process objects one by one, update visibility for all pixels covered by object (Z buffer, painter's algorithm)
- process pixels one by one, for each pixel update visibility

© 1999, Denis Zorin

Z buffer

Assumptions:

- each pixel has storage for a z-value, in addition to RGB
- all objects are “scanconvertible” (typically are polygons, lines or points)

Algorithm:

initilize zbuf to maximal value

for each object

for each pixel (i,j) obtained by scan conversion

if $z_{new}(i,j) < z_{buf}(i,j)$

$z_{buf}(i,j) = z_{new}(i,j)$;

write pixel(i,j)

© 1999, Denis Zorin

Z buffer

What are z values?

Z values are obtained by applying the projection transform, that is, mapping the viewing frustum to the standard cube.

Z value increases with th distance to the camera.

Z values for each pixel are computed for each pixel covered by a polygon using linear interpolation of z values at vertices.

Typical Z buffer size: 24 bits (same as RGB combined).

© 1999, Denis Zorin

Painter's algorithm

Algorithm:

- Apply projection transform to all polygons.
- Sort all polygons by z, splitting intersecting polygons along z.
- Scan convert polygons in back to front order.
- Polygons that are closer overwrite those that are further away.

Drawbacks: requires sorting and splits; pixels are overwritten many times.

Front to back version: add a bit to pixel to indicate it was written.

© 1999, Denis Zorin

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
```

© 1999, Denis Zorin

Ray casting

Basic operation: intersect a ray with an object.

Object types are more varied than for Z-buffer:

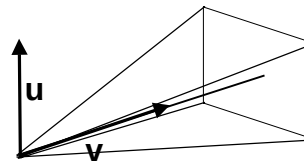
- polygon
- sphere
- cone
- cylinder
- general quadric
- height field
- ...

© 1999, Denis Zorin

Pixel rays

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

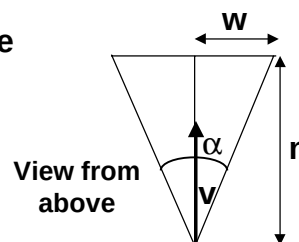
- c** position
- α** horizontal field of view
- v** viewing direction
- u** up direction
- s** aspect ration



Then the image half-width in the “virtual world” units is

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

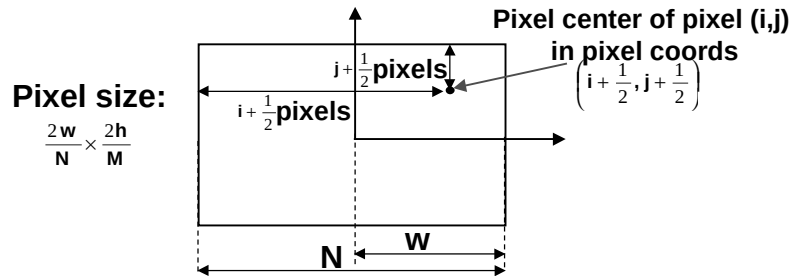
The half-height is $h = s n \operatorname{tg} \frac{\alpha}{2}$



© 1999, Denis Zorin

Pixel rays

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



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

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

© 1999, Denis Zorin

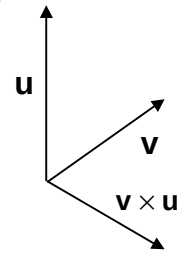
Pixel rays

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

Image center: $c + vn$

pixel (i, j) = $c + vn +$

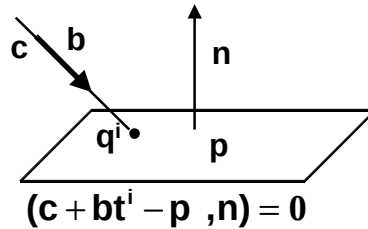
$$\left(h - \left(j + \frac{1}{2} \right) \frac{2h}{M} \right) u + \left(\left(i + \frac{1}{2} \right) \frac{2w}{N} - w \right) v \times u$$



© 1999, Denis Zorin

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, $b = p(i,j) - c$



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

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

© 1999, Denis Zorin

Intersection with a sphere

Two questions are important:

- is there an intersection?
- where are the intersection points?

Most rays do not hit a sphere if it is small enough, so a fast “no” to the first question will speed up our calculation.

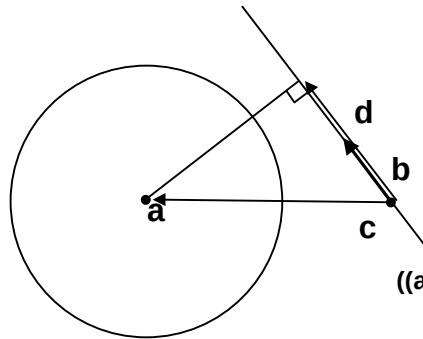
To answer the second question, we have to solve a quadratic equation. To answer the first, we do not have to.

© 1999, Denis Zorin

Intersection with a sphere

Question: is there intersection?

The distance from the center of the sphere to the ray should be less than radius.



Projection of $a-c$ on b :

$$d = \frac{(a-c, b)}{|b|^2} b$$

The square of length:

$$((a-c) - d)^2 = \left(a-c - \frac{(a-c, b)}{|b|^2} b \right)^2 = R^2$$

If $R^2 > r^2$, there is no intersection.

© 1999, Denis Zorin

Intersection with a sphere

Question: what are the intersection points?

Plug in the parametric ray equation into the sphere equation. Sphere equation can be written as $(a-q)^2=r^2$, where a is the center and q is a point on the sphere.

$$(a - c - bt)^2 = r^2$$

$$\underbrace{b^2 t^2}_A - 2 \underbrace{(a-c, b)}_B t + \underbrace{(a-c)^2 - r^2}_C = 0$$

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

Solutions of this equation, if any, are the values of parameter t for the intersection points.

© 1999, Denis Zorin