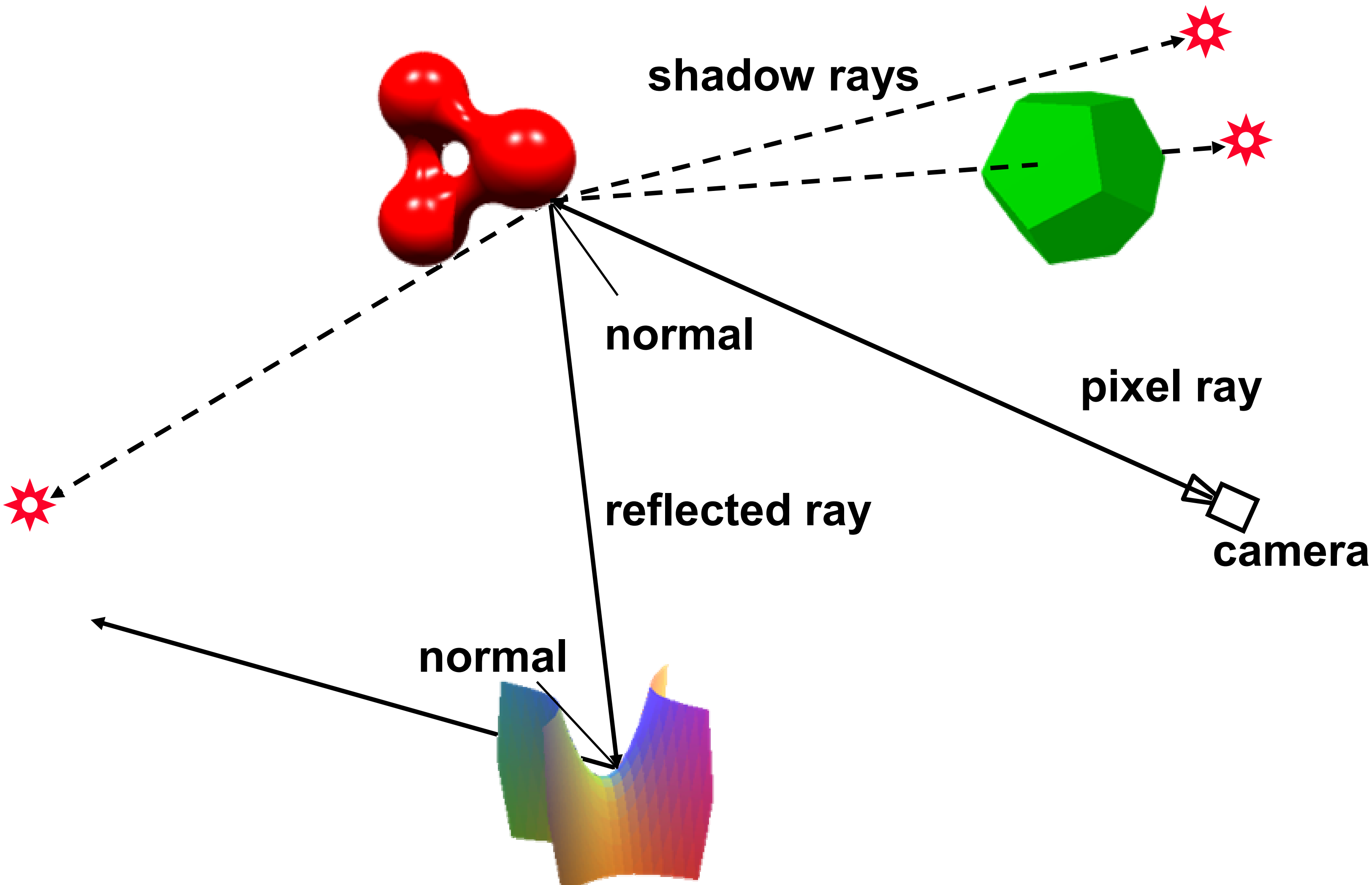
Ray tracing

© 1999, Denis Zorin

Ray tracing



© 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

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:

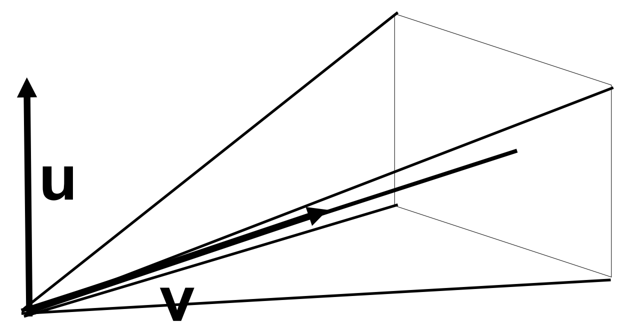
- **polygon**
- **sphere**
- **cone**
- **cylinder**
- **general quadric**
- **height field**
- **...**

© 1999, Denis Zorin

Pixel rays

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

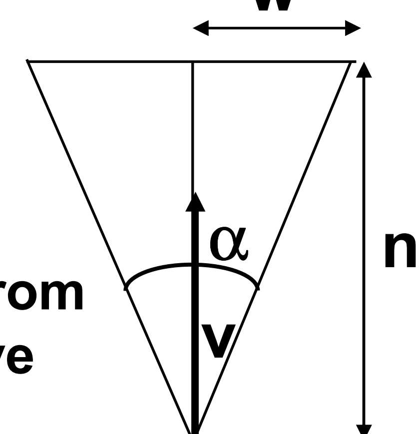
c	position
α	horizontal field of view
v	viewing direction
u	up direction
s	aspect ratio



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

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

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



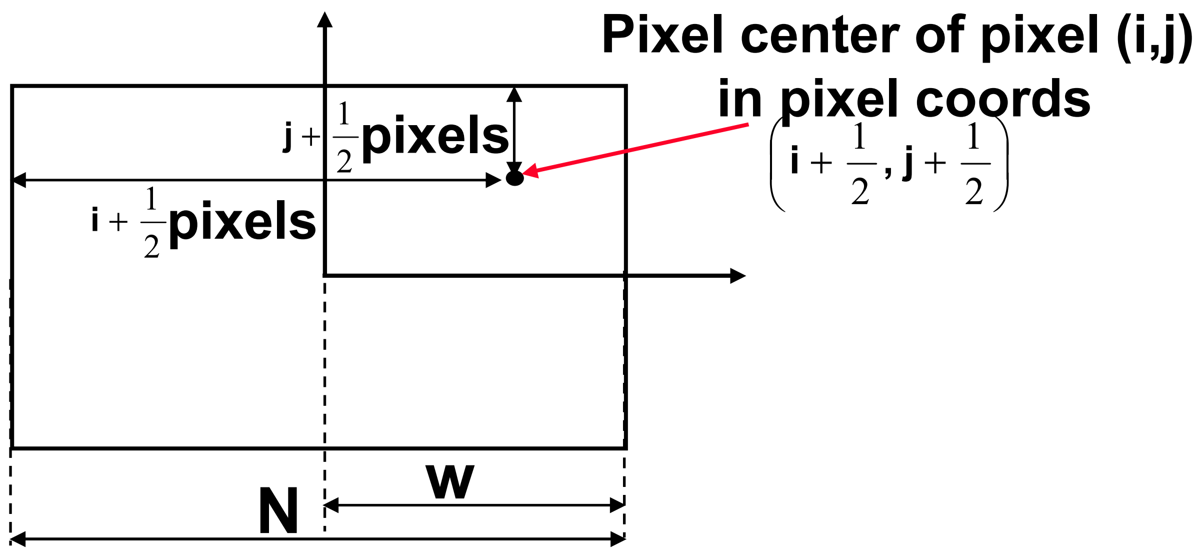
© 1999, Denis Zorin

Pixel rays

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

Pixel size:

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



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

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

© 1999, Denis Zorin

Pixel rays

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

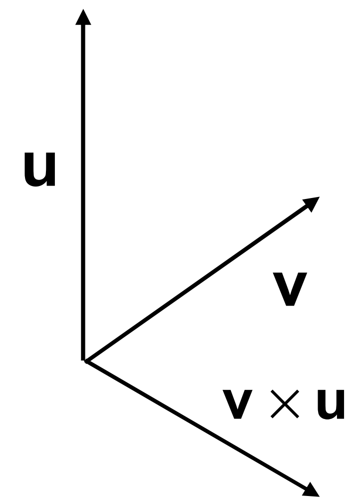


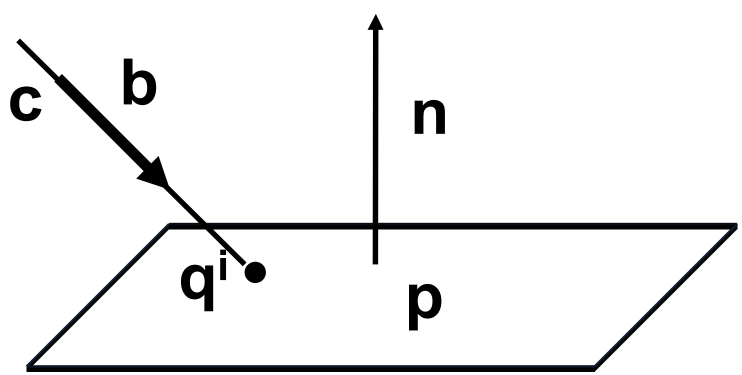
Image center: c + vn

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

© 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



$$(\mathbf{c} + \mathbf{b}t^i - \mathbf{p}, \mathbf{n}) = 0$$

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

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.

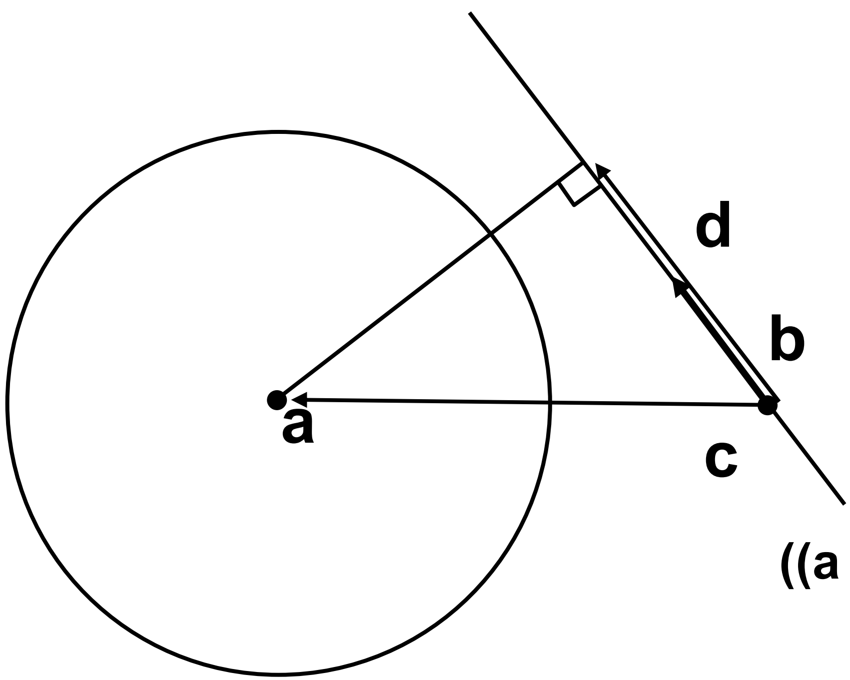
To answer the second question, we have to solve a quadratic equation. To answer the first, we do not have to.

© 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:

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

The square of length:

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

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 parameteric 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.

$$(\mathbf{a}-\mathbf{c}-\mathbf{b}t)^2 = r^2$$

$$\underbrace{\mathbf{b}^2}_{A} t^2 \underbrace{-2(\mathbf{a}-\mathbf{c},\mathbf{b})}_{B} t + \underbrace{(\mathbf{a}-\mathbf{c})^2 - r^2}_{C} = 0$$

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

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

© 1999, Denis Zorin

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

© 1999, Denis Zorin

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!

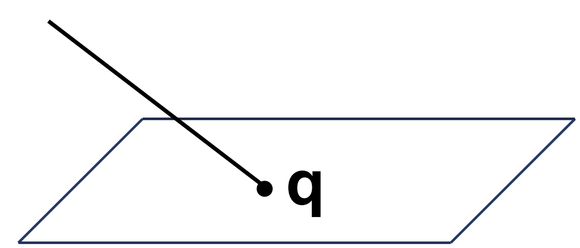
© 1999, Denis Zorin

Polygon-ray intersections

Two steps:

- intersect with the plane of the polygon
- check if the intersection point is inside the polygon

We know how to compute intersections with the plane (see prev. lecture). Let $q=[q_x,q_y,q_z]$ be the intersection point.



© 1999, Denis Zorin

Polygon-ray intersections

Possible to do the whole calculation using 3d points, but it is more efficient to use 2d points.

Converting to the coordinates in the plane is computationally expensive. Idea: project to a coordinate plane (XY, YZ, or XZ) by discarding one of the vector coordinates.

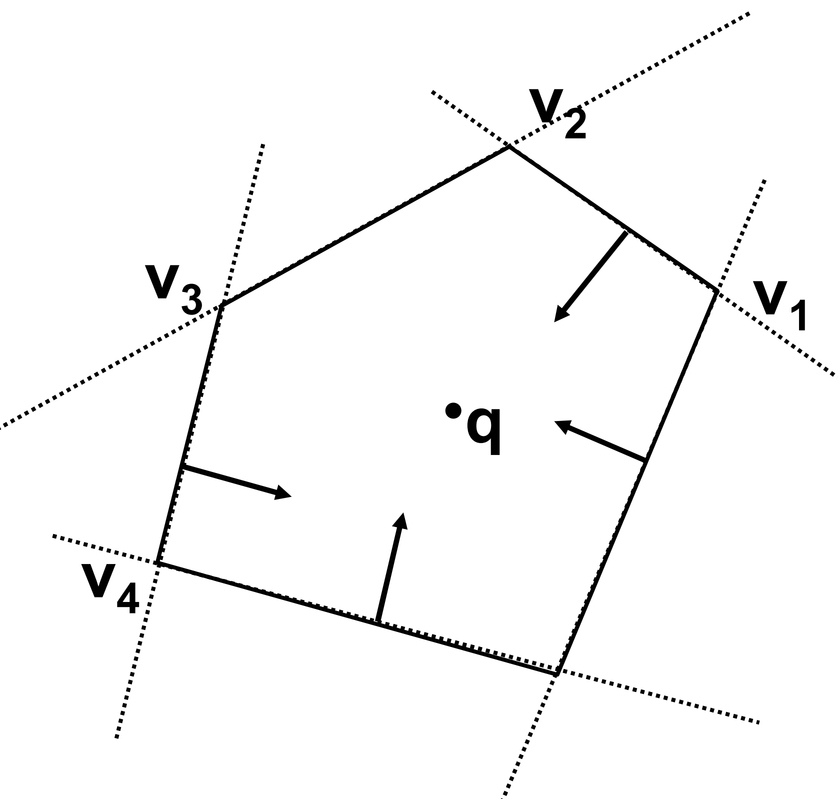
We cannot always discard, say, Z, because the polygon may project to an interval, if it is in a plane parallel to Z.

Choose the coordinate to discard so that the corresponding component of the normal to the polygon is maximal. E.g. if $n_x > n_y$ and $n_x > n_z$, discard X.

© 1999, Denis Zorin

2D Polygon-ray intersections

Now we can assume that all vertices v_i and the intersection point q are 2D points.



Assume that polygons are convex. A convex polygon is the intersection of a set of half-planes, bounded by the lines along the polygon edges. To be inside the polygon the point has to be in each half-plane. Recall that the implicit line equation can be used to check on which side of the line a point is.

© 1999, Denis Zorin

2D Polygon-ray intersections

Equation of the line through the edge connecting vertices v_i and v_{i+1}:

$$\left(v_i^y - v_{i+1}^y\right)\left(x - v_i^x\right) + \left(v_{i+1}^x - v_i^x\right)\left(y - v_i^y\right) = 0$$

If the quantity on the right-hand side is positive, then the point (x,y) is to the right of the edge, assuming we are looking from v_i to v_{i+1}.

Algorithm: if for each edge the quantity above is nonnegative for $x = q^x$, $y = q^y$ then the point q is in the polygon. Otherwise, it is not.

In the formulas x and y should be replaced by x and z if y coord. was dropped, or by y and z if x was dropped.

© 1999, Denis Zorin