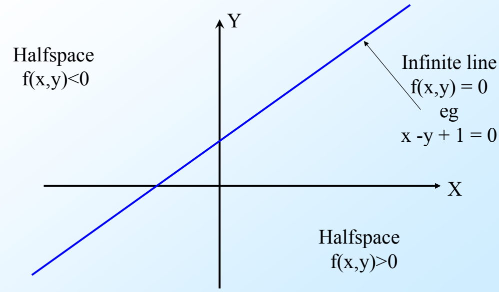
Interactive Computer Graphics

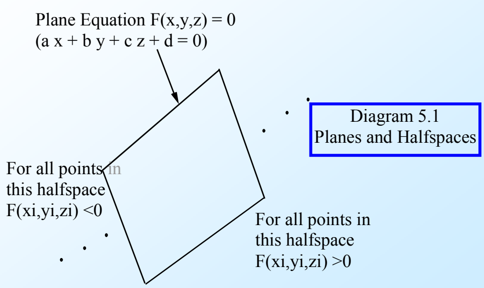
Lecture 4

Manipulation of Three Dimensional Objects

The Concept of a Halfspace



The same idea extends to three dimensions



Convex Objects

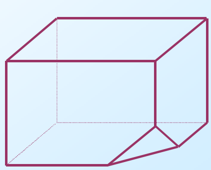
We can use the halfspace property for a number of algorithms for manipulating graphics scenes.

We will consider first convex objects, and the first algorithm is to determine whether an object is convex or not.

Two Definitions of Convex

1. A line joining any two points on the boundary lies inside the object.

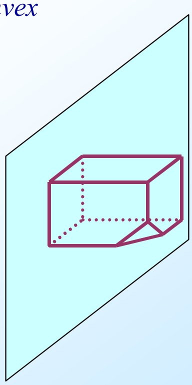
2. The object is the intersection of planar halfspaces.



Algorithm for determining if an object is convex

```
convex = true
for each face of the object
{   find the plane equation of the face f (x,y,z) = 0
    choose one object point (xi,yi,zi) not on the face
              and find sign( f (xi,yi,zi) )
    for all other points of the object
    {   if  (sign( f(xj,yj,zj) ) not = sign(f (xi,yi,zi) ) )
        then convex = false
    }
}
```

Testing for Convex



Testing for Containment

A frequently encountered problem is to determine whether a point is inside an object or not.

We need this for clipping algorithms

Algorithm for Containment

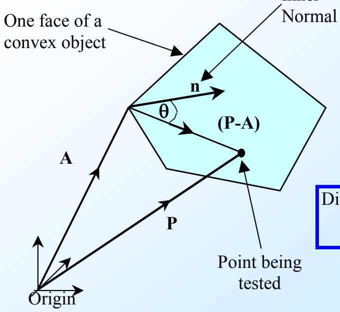
let the test point be [xt,yt,zt]
contained = *true*
for each face of the object
{ find the plane equation of the face f (x,y,z) = 0
choose one object point (xi,yi,zi) not on the face
and find *sign*(f (xi,yi,zi))
if (*sign*(f(xt,yt,zt)) not = *sign*(f (xi,yi,zi)))
then contained = *false*
}

Vector formulation

The same test can be expressed in vector form.

This avoids the need to calculate the Cartesian equation of the plane, if, in our data base we store the normal **n** vector to each face of our object.

Vector test for containment



Contained if θ is acute
ie Cos(θ) is positive
or **n**.(**P-A**) is positive
n.(**P-A**) = |**n**||**P-A**|Cos(θ)

Diagram 5.2 Containment within a convex object

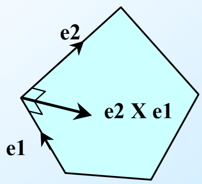
Normal vector to a face

The vector formulation does not require us to find the plane equation of a face, but it does require us to find a normal vector to the plane.

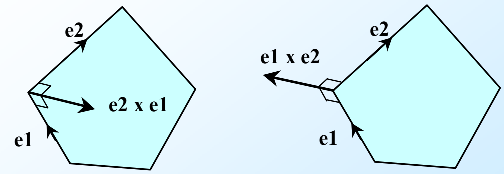
(same thing really since for plane ax + by +cz + d=0 a normal vector is [a,b,c])

Finding a normal vector

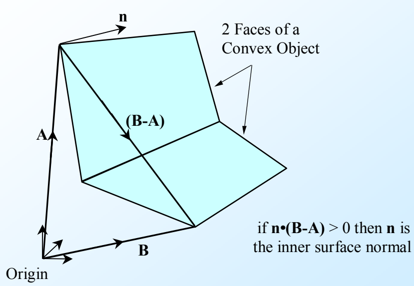
The normal vector can be found from the cross product of two vectors on the plane, say two edge vectors



But which normal vector points inwards?



Checking the normal direction



Problem Break

A face of a convex object lies in the plane

3x+5y+7z +1 = 0 and a vertex is {-1,-1,1}

The normal vector is therefore **n** = {3,5,7}

1. If another vertex of the object is {1,1,1} determine whether **n** is an inner or outer surface normal.
(see fig 6.4)

2. Determine whether the point {1,0,-1}is on the inside or the outside of the face.

Solution 1

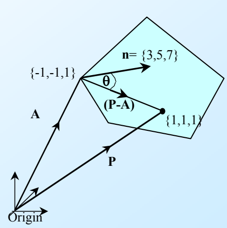
P-A = {1,1,1} - {-1,-1,1}
= {2,2,0}

n.(P-A) = 6 + 10 = 16

n.(P-A) is positive,

θ is acute

n is an inner normal



Solution 2

Method 1:

The plane has equation 3x+5y+7z+1=0

ie f(x,y,z) = 3x+5y+7z+1

For the internal point {1,1,1} f(1,1,1) = 16

For the test point {1,0,-1} f(1,0,-1) = -3

The signs are different, so the test point is on the outside

Solution 3

Method 2:

The inner surface normal is **n** = {3,5,7}

for the test point **P** = {1,0,-1} and vertex **A**={-1,-1,1}

P-A = {2,1,-2}

n.**(P-A)** = -3

Thus the angle to the normal is > 90

The point is on the outside

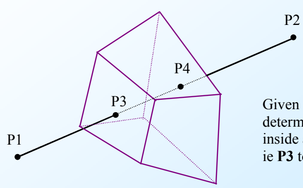
Clipping

Containment is an important property used in clipping algorithms.

Clipping is used to remove unwanted parts of a graphics scene before drawing.

It can be applied in computer aided design, and graphics scene design.

Clipping a line to a convex polyhedron



Given a line segment **P1** to **P2** determine the part of the line inside a convex object, ie **P3** to **P4**

Clipping algorithm

The algorithm checks the line against every face of the convex polyhedron.

It determines whether the end points of the line are on the inside or the outside of the face

This can be done by the halfspace or the dot product with the inner normal as before.

Case 1: Both P_1 and P_2 are on the outside

The line is completely clipped (no part of it is inside the polyhedron)

The algorithm terminates

Case 2: Both P_1 and P_2 are on the inside

There is no new information.

If there are more faces to test the algorithm continues to the next face.

Otherwise the line is completely inside the volume.

Case 3: P_1 is outside and P_2 is inside

Compute the intersection between the line and plane.

for any vector **p** lying on the plane **n.p** = 0

let the intersection point be $\mu_i \mathbf{P_2} + (1- \mu_i)\mathbf{P_1}$

if A ia a vertex of the object a vector on the plane is

$\mu_i \mathbf{P_2} + (1- \mu_i)\mathbf{P_1} - \mathbf{A}$

thus $\mathbf{n}.(\mu_i \mathbf{P_2} + (1- \mu_i)\mathbf{P_1} - \mathbf{A}) = 0$

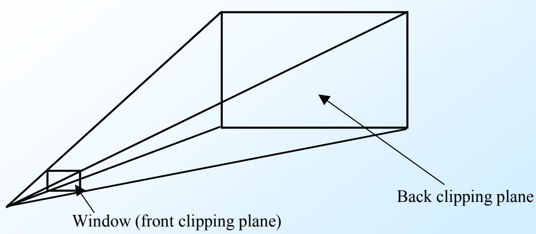
we can solve this for μ_i and hence find the point of intersection

Replace $\mathbf{P_1}$ with the intersection

Case 3: P_2 is outside and P_1 is inside

This is equivalent to case 3 with P1 and P2 exchanged

Clipping to a viewing volume



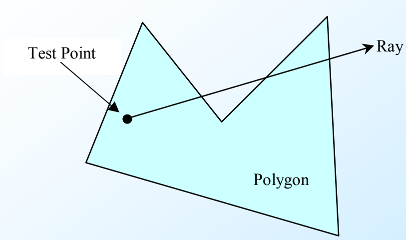
The visible parts of a graphics scene are bounded by six planes.
For efficient computation all invisible parts should be clipped

Concave Objects

Containment and clipping can also be carried out with concave objects.

Most algorithms are based on the ray containment test.

The Ray test in two dimensions



Find all intersections between the ray and the polygon edges.
If the number of intersections is odd the point is contained

Calculating intersections with rays

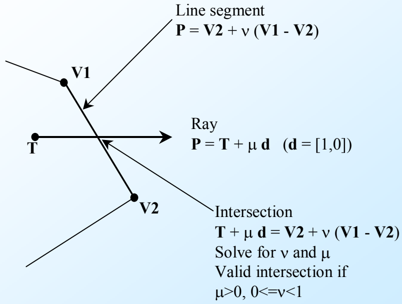
Rays have equivalent equations to lines, but go in only one direction. For test point T a ray is defined as

$$\mathbf{R} = \mathbf{T} + \mu\, \mathbf{d} \quad \mu>0$$

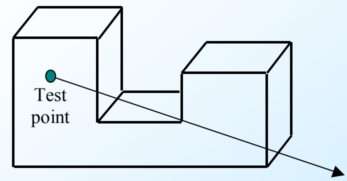
We choose a simple to compute direction eg

$$\mathbf{d} = [1,0,0]$$

Valid Intersections



Extending the ray test to 3D



A ray is projected in any direction.

If the number of intersections with the object is odd, then the test point is inside

3D Ray test

There are two stages:

1. Compute the intersection of the ray with the plane of each face.

2. If the intersection is in the positive part of the ray ($\mu>0$) check whether the intersection point is contained in the face.

The plane of a face

Unfortunately the plane of a face does not in general line up with the Cartesian axes, so the second part is not a two dimensional problem.

However, containment is invariant under orthographic projection, so it can be simply reduced to two dimensions.

Clipping to concave volumes

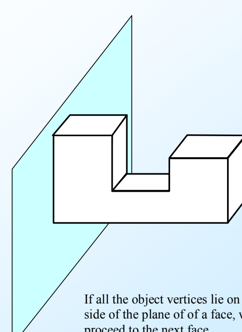
Find every intersection of the line to be clipped with the volume.

This divides the line into one or more segments.

Test a point on the first segment for containment

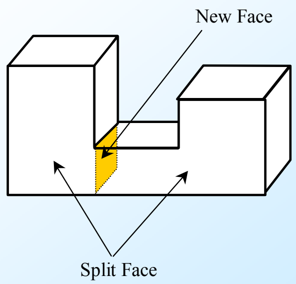
Adjacent segments will be alternately inside and out.

Splitting a volume into convex parts

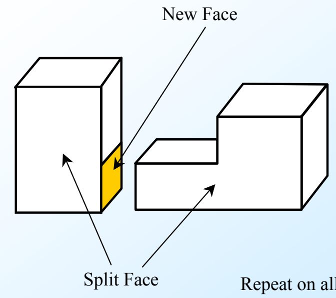


If all the object vertices lie on one side of the plane of of a face, we proceed to the next face

If the plane of a face cuts the object:



Split the Object



Repeat on all concave sub parts