Clipping Lines and Polygons

Must easily calculate where a segment crosses a plane:

Given a plane EQ $Q(P) = N \cdot P + d$

Q(P) is 0 if P is on Q
Q(P) is positive P if same side as N
Q(P) is negative if P is on opposite side
Q(P) is a "scaled", "signed" distance of P from plane.

Given two points, P0, P1, and plane eq Q:

Calculate distance of points from plane equation

$$d0 = Q(P0)$$
$$d1 = Q(P1)$$

Then the fraction of distance along segment where it crosses the plane:

$$t = \frac{0 - d0}{d1 - d0} = \frac{d0}{d0 - d1}$$

and the point of intersection is

$$I = P0 + t(P1 - P0)$$

Clipping plane equations:

Boundaries of the clipping coordinates (after homogeneous division)

$$-1 \le x/w \le 1$$
$$-1 \le y/w \le 1$$
$$0 \le z/w \le 1$$

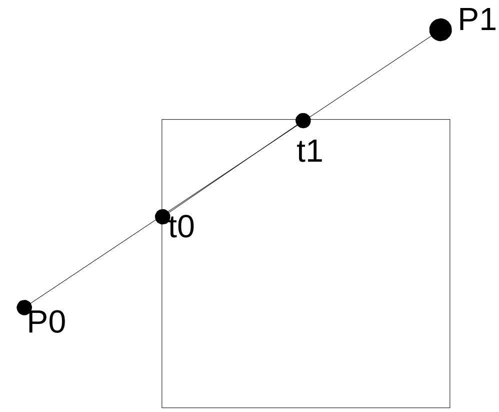
Converting these to pre-homogeneous-division plane equations yields

$$\begin{aligned}
-1 \le x/w &\quad\rightarrow\quad w + x \ge 0 \quad\rightarrow\quad [\ 1 \quad 0 \quad 0 \quad 1\ ] \\
x/w \le 1 &\quad\rightarrow\quad w - x \ge 0 \quad\rightarrow\quad [\ -1 \quad 0 \quad 0 \quad 1\ ] \\
-1 \le y/w &\quad\rightarrow\quad w + y \ge 0 \quad\rightarrow\quad [\ 0 \quad 1 \quad 0 \quad 1\ ] \\
y/w \le 1 &\quad\rightarrow\quad w - y \ge 0 \quad\rightarrow\quad [\ 0 \quad -1 \quad 0 \quad 1\ ] \\
0 \le z/w &\quad\rightarrow\quad z \ge 0 \quad\rightarrow\quad [\ 0 \quad 0 \quad 1 \quad 0\ ] \\
z/w \le 1 &\quad\rightarrow\quad w - z \ge 0 \quad\rightarrow\quad [\ 0 \quad 0 \quad -1 \quad 1\ ]
\end{aligned}$$

For testing purposes when writing a clipper,
provide better visual clues of the clipping boundaries by
modifying these to clip at 90% of the way to the boundary:

$$\begin{aligned}
-0.9 \le x/w &\quad\rightarrow\quad 0.9\mathrm{w} + x \ge 0 \quad\rightarrow\quad [\ 1 \quad 0 \quad 0 \quad 0.9\ ] \\
x/w \le 0.9 &\quad\rightarrow\quad 0.9\mathrm{w} - x \ge 0 \quad\rightarrow\quad [\ -1 \quad 0 \quad 0 \quad 0.9\ ] \\
-0.9 \le y/w &\quad\rightarrow\quad 0.9\mathrm{w} + y \ge 0 \quad\rightarrow\quad [\ 0 \quad 1 \quad 0 \quad 0.9\ ] \\
y/w \le 0.9 &\quad\rightarrow\quad 0.9\mathrm{w} - y \ge 0 \quad\rightarrow\quad [\ 0 \quad -1 \quad 0 \quad 0.9\ ] \\
0 \le z/w &\quad\rightarrow\quad z \ge 0 \quad\rightarrow\quad [\ 0 \quad 0 \quad 1 \quad 0\ ] \\
z/w \le 1 &\quad\rightarrow\quad w - z \ge 0 \quad\rightarrow\quad [\ 0 \quad 0 \quad -1 \quad 1\ ]
\end{aligned}$$

Line clipping



Representation of a line segment

Points on a line segment are represented with a parameter t representing fraction of distance along the segment

The two endpoints of the visible (non-clipped) portion of a segment are represented with t0 and t1.

Initial values (used to represent the full segment) are:

t0 = 0 and t1=1

As we clip, t0 will increase, and t1 will decrease.

First we work out how to clip to a single boundary:

Input: d0, d1
In-Out: t0, t1
Return: boolean indicating if any portion of line is visible

```
ClipB(d0, d1, t0, t1):
    if d0 <0 and d1 < 0: // Both endpoints outside boundary
        return False
    else if d0 >= 0 and d1 >= 0: // Both endpoints inside or on boundary
        return True
    else: // Endpoints on both sides of boundary – must clip
        t = d0/(d0-d1) // Cannot be a zero divide
        if d0 < 0:
            t0 = max(t0,t)
        else:
            t1 = min(t1,t)
        return True
```

Full algorithm calls single boundary alg 6 times:

In: vertices P0, P1
Out: vertices R0, R1
Returns: Boolean indicating if an portion is visible

```
Clip(P0,P1,   R0,R1):
    t0 = 0
    t1 = 1
    for Q in list of six plane equations:
        r = ClipB(Q(P0), Q(P1), t0, t1)
        if not r:
            return false // P0:P1 is outside this boundary so outside full window
    if t0 > t1:
        return false // corner case

    R0 = P0+t0(P1-P0)
    R1 = P0+t1(P1-P0)
    return true
```

Clipping Polygons
Sutherland-Hodgeman Algorithm

Algorithm

Pass a polygon through the following **PolyClip** procedure once for each clip boundary

PolyClip (single clipping plane pass):

Input: P -- array of vertex, Q a Plane EQ
Output: array of vertex
PolyClip(P):

```
R = empty list
S = P[size(P)-1]
for i = 0 to size(P)-1:
    T = P[i]
    if both S,T are on + side of Q:
        append T to R
    else if S is on + side of Q:
        I = intersection of segment S,T and plane Q
        append I to R // See note below
    else if T is on + side of Q:
        I = intersection of segment S,T and plane Q
        append I to R // See note below
        append T to R
    S = T
return R
```

Intersection of S,T with plane Q:

$$t = \frac{0 - Q(S)}{Q(T) - Q(S)} = \frac{Q(S)}{Q(S) - Q(T)}$$

$$I = S + t(T - S)$$

Must handle some zero and round-off casses:

To test for P being "on + side" of plane Q:
    include 0 as "on + side"
    include even within ϵ as "on + side"
So the test for "on + side" becomes
    if Q(P) > $-\epsilon$

Consider what it takes to do this in a streaming fashion:

Sequence of clippers and one last writer
    input-list ==> clipper1 ==> cliper2 ==> ... ==> clipper6 ==> output-list
Each clipper
    knows its plane EQ
    has a NextVertex method for input
    has a nextClipper reference
    calls nextClipper.NextVertex(...) instead of "append to R"
    has copies of first point and last point seen
    has a Done method which considers edge from last to first point.
Startup
    calls clipper1.NextVertex(P[i]) in a loop.