On-Line Computer Graphics Notes

CLIPPING

Kenneth I. Joy Visualization and Graphics Research Group Department of Computer Science University of California, Davis

1 Overview

The primary use of clipping in computer graphics is to remove objects, lines or line segments that are outside the viewing volume. The viewing transformation is insensitive to the position of points relative to the viewing volume – especially those points behind the viewer – and it is necessary to remove these points before generating the view. Clipping can be done either in three-dimensional space, or in image space. The algorithms are nearly identical.

In this paper, we concentrate on clipping methods for polygons, clipping them against planes. We first develop an "inside/outside" test for points against a plane. This test is used to construct a clipping algorithm for line segments, and the line-segment clipping is used to develop the polygon-clipping algorithm. These methods are then applied to generate an algorithm that clips polygons in the viewing operation.

2 Inside/Outside Tests for Points Against a Plane

Given a plane defined by the normal vector \vec{n} and the point **P**.



This plane separates three-dimensional space into two half-spaces: one in the direction of the normal vector, which we will call the "in" side of the plane; and another in the opposite direction of the normal vector, which we will call the "out" side. If we have a point \mathbf{Q} in the plane then the vector $\langle \mathbf{Q} - \mathbf{P} \rangle$ is perpendicular to the normal vector \vec{n} , that is

$$\langle \mathbf{Q} - \mathbf{P} \rangle \cdot \vec{n} = 0$$

For an arbitrary point \mathbf{Q} in three-dimensional space, this dot product satisfies

$$<\mathbf{Q}-\mathbf{P}>\cdot\vec{n}~=~|<\mathbf{Q}-\mathbf{P}>||\vec{n}|\cos{ heta}$$

where θ is the angle between the vectors \vec{n} and $\langle \mathbf{Q} - \mathbf{P} \rangle$. So if the point \mathbf{Q} is on the "in" side of the plane, the angle θ must satisfy $0 \leq \theta < \frac{\pi}{2}$, and $\cos \theta$ must be positive. Since the dot product is positive if and only if $\cos \theta$ is positive, this says that the point is on the "in" side of the plane if and only if the dot product is positive. This case is illustrated in the figure below, where the figure has been drawn with the eyepoint on the surface of the plane.



Alternatively if the point \mathbf{Q} is on the "out" side of the plane, then the angle between the vectors \vec{n} and $\langle \mathbf{Q} - \mathbf{P} \rangle$ satisfies $\frac{\pi}{2} \langle \theta \leq \pi$, and $\cos \theta$ is negative. It follows that the point is on the "out" side of the plane if and only if the dot product is negative.

These facts can be combined to form an inside/outside test for points against a plane:

• A point \mathbf{Q} is defined to be on the "in" side of the plane (the side of the plane to which the normal

vector points) if

$$<\mathbf{Q}-\mathbf{P}>\cdot\vec{n}>0$$

• A point Q is defined to be on the "out" side of the plane if

$$<\mathbf{Q}-\mathbf{P}>\cdot\vec{n}<0$$

• A point **Q** is "on" the plane if

$$\langle \mathbf{Q} - \mathbf{P} \rangle \cdot \vec{n} = 0$$

We note that the third case $\langle \mathbf{Q} - \mathbf{P} \rangle \cdot \vec{n} = 0$ happens infrequently when using floating point arithmetic on computer systems. Normally this must be approximated by

$$|\langle \mathbf{Q} - \mathbf{P} \rangle \cdot \vec{n}| < \epsilon$$

where ϵ is a predetermined (small) constant.

3 Clipping Line Segments Against a Plane

The inside/outside test forms the basis for clipping line segments against a plane. Let \mathcal{P} be a plane defined by the normal vector \vec{n} and the point **P** and let $\overline{\mathbf{Q}_1 \mathbf{Q}_2}$ be the line segment defined by the two points \mathbf{Q}_1 and \mathbf{Q}_2 , as is shown in the following figure



Let $d_1 = \langle \mathbf{Q}_1 - \mathbf{P} \rangle \cdot \vec{n}$ and $d_2 = \langle \mathbf{Q}_2 - \mathbf{P} \rangle \cdot \vec{n}$. We can determine how the line segment is clipped by examining four cases:

- 1. Both points Q_1 and Q_2 are on the "in" side on the plane that is, the line segment is completely "in".
- 2. Both points Q_1 and Q_2 are on the "out" side on the plane that is, the line segment is completely "out".
- 3. Q_1 is on the "in" side of the plane and Q_2 is on the "out" side in which case the line segment crosses the plane. (This is the case for the figure above.)
- 4. \mathbf{Q}_2 is on the "in" side of the plane and \mathbf{Q}_1 is on the "out" side so again, the line segment crosses the plane.

We analyze these cases one at a time. Clearly the side of the plane on which Q_1 lies is determined by the sign of d_1 , and similarly, the sign of d_2 determines the side of the plane on which Q_2 lies.

• Case $I - d_1 \ge 0$ and $d_2 \ge 0$, or $d_1 \ge 0$ and $d_2 \ge 0$.

In this case the line segment $\overline{\mathbf{Q}_1 \mathbf{Q}_2}$ is on the "in" side of the plane.

• Case II $-d_1 \le 0$ and $d_2 \le 0$, or $d_1 \le 0$ and $d_2 \le 0$.

In this case the line segment is on the "out" side of the plane.

• **Case III** $-d_1 > 0$ and $d_2 < 0$

In this case, \mathbf{Q}_1 lies on the "in" side of the plane, and \mathbf{Q}_2 lies on the "out" side of the plane (the case shown in the illustration). Calculate the intersection I of the line and the plane by first noting that I is a point on the line $\overline{\mathbf{Q}_1\mathbf{Q}_2}$, and can be represented in the form

$$\mathbf{I} = \mathbf{Q}_1 + t < \mathbf{Q}_2 - \mathbf{Q}_1 >$$

To determine t, we first subtract the identity

$$\mathbf{P} = \mathbf{P} + t < \mathbf{P} - \mathbf{P} >$$

from both sides of the equation for I, giving

$$< \mathbf{I} - \mathbf{P} > = < \mathbf{Q}_1 - \mathbf{P} > + t (< \mathbf{Q}_2 - \mathbf{P} > - < \mathbf{Q}_1 - \mathbf{P} >)$$

Since I an P are both in the plane, $\langle I - P \rangle$ is a vector perpendicular to \vec{n} . So dotting both sides with the normal vector \vec{n} gives

$$0 = \langle \mathbf{I} - \mathbf{P} \rangle \cdot \vec{n}$$

= $\langle \mathbf{Q}_1 - \mathbf{P} \rangle \cdot \vec{n} + t (\langle \mathbf{Q}_2 - \mathbf{P} \rangle \cdot \vec{n} - \langle \mathbf{Q}_1 - \mathbf{P} \rangle \cdot \vec{n})$
= $d_1 + t (d_2 - d_1)$

and solving for t, we see that

$$t = \frac{d_1}{d_1 - d_2}$$

Once we have calculated I, we can say that the line segment $\overline{\mathbf{Q}_1 \mathbf{I}}$ lies on the "in" side of the plane and the line segment $\overline{\mathbf{IQ}_2}$ lies on the "out" side of the plane.

• **Case IV** $-d_1 < 0$ and $d_2 > 0$

In this case, Q_1 lies on the "out" side of the plane, and Q_2 lies on the "in" side of the plane. Calculate the intersection I of the line and the plane by

$$\mathbf{I} = \mathbf{Q}_1 + t(\mathbf{Q}_2 - \mathbf{Q}_1)$$

and with calculations similar to those above, we obtain

$$t = \frac{d_1}{d_1 - d_2}$$

Once we have calculated I, we can state that the line segment $\overline{\mathbf{Q}_1 \mathbf{I}}$ lies on the "out" side of the plane and the line segment $\overline{\mathbf{I}}\mathbf{Q}_2$ lies on the "in" side of the plane.

There is one additional case that should be considered. This happens when $d_1 = 0$ and $d_2 = 0$. In this case, the line segment $\overline{\mathbf{Q}_1 \mathbf{Q}_2}$ lies "in" the plane.

If we examine this closely, we see that the quantities

$$t = \frac{d_1}{d_1 - d_2}$$

make sense. When we write the equation $\mathbf{I} = \mathbf{Q}_1 + t < \mathbf{Q}_2 - \mathbf{Q}_1 >$, **I** lies on the line between \mathbf{Q}_1 and \mathbf{Q}_2 when $0 \le t \le 1$, and indeed the values $\frac{d_1}{d_1-d_2}$ are numbers between zero and one. For example, in case III we have that $d_1 > 0$ and $d_2 < 0$, so the denominator of the fraction is greater than the numerator. In case IV, $d_1 < 0$ and $d_2 > 0$, so the denominator of the fraction is less than the numerator in value, but the signs on both are negative and the denominator is greater in absolute value, again implying that the result is between zero and one.

4 Clipping a Convex Polygon

We can utilize the inside/outside test and the line-clipping algorithm to develop an algorithm for clipping a convex polygon. If we consider the vertices of the polygon one-at-a-time and keep track of when the edges of the polygon cross the plane, the algorithm is actually quite simple.

To implement this polygon clipping algorithm, we usually keep a list of points, commonly called the "in" list, which holds the resulting clipped polygon. The algorithm then iterates through the vertices of the polygon, and has the following steps:

• A vertex is on the "in" side of the plane is retained in the "in" list.

- A vertex is on the "out" side of the plane is discarded.
- If a vertex **P** is on the "in" side of the plane, and the previous vertex in our iteration was on the "out" side, the point-of-intersection **I** of the edge and the plane is calculated and placed on the "in" list. The vertex **P** is then placed on the "in" list.
- If a vertex is on the "out" side of the plane, and the previous vertex in our iteration was on the "in" side, the point-of-intersection I of the edge and the plane is calculated and placed on the "in" list. The vertex is discarded.

This algorithm can be implemented with a single loop through the points of the polygon. The only problem is that the edges are defined by two points, and so we must retain two dot products to make our comparisons. The algorithm is illustrated in the following pseudo-code algorithm.

Clipping Algorithm

Given a plane defined by \vec{n} and P Given vertices $\mathbf{Q}_0, \mathbf{Q}_1, ..., \mathbf{Q}_{n-1}$ Let pdot = 0Let $idot = \vec{n} \cdot \langle \mathbf{Q}_0 - \mathbf{P} \rangle$ for each vertex *i* Let $dot = \vec{n} \cdot \langle \mathbf{Q}_i - \mathbf{P} \rangle$ if dot * pdot < 0 then calculate $\mathbf{I} = \mathbf{Q}_{i-1} + t(\mathbf{Q}_i - \mathbf{Q}_{i-1})$ with $t = \frac{pdot}{pdot-dot}$ insert I into the "in" list end if if dot > 0 then **insert** \mathbf{Q}_i into the "in" list end if Let pdot = dotend for If pdot * idot < 0 then calculate $\mathbf{I} = \mathbf{Q}_{n-1} + t(\mathbf{Q}_0 - \mathbf{Q}_{n-1})$ with $t = \frac{pdot}{pdot-idot}$ insert I into the "in" list end if

If $\mathbf{Q}_0, \mathbf{Q}_1, ..., \mathbf{Q}_{n-1}$ are the *n* vertices of a polygon, we must insure that the last edge of the polygon, $\overline{\mathbf{Q}_{n-1}\mathbf{Q}_0}$, is checked. This is the reason for the last four statements of the algorithm. We must test to see if the last line segment (the one between Q_{n-1} and Q_0 crosses the plane, and if so, insert the intersection point into the in list.

This algorithm is guaranteed to work with convex polygons only – non-convex polygons can cause the algorithm to produce some false edges.

5 Clipping to a Convex Polyhedra

The three-dimensional analog of a polygon is a polyhedron (*e.g.*, a cube, a truncated pyramid, a dodecahedron). A convex polyhedron can be defined by a finite set of bounding planes $\mathcal{P}_0, \mathcal{P}_1, ..., \mathcal{P}_m$ and we can clip against the polyhedron by clipping against each plane in turn and using the output polygon of one step as the input polygon to the next. If, at any step, the output polygon is empty, then the process terminates.

We must define the planes so that the normal vectors point toward the inside of the polyhedron.

6 Clipping to the Viewing Pyramid

The viewing pyramid is a convex polyhedron – as is the image-space cube. The algorithm for clipping a single convex polygon against a plane can be utilized to clip a polygon against multiple planes of these regions. We simply clip against the planes one at a time, taking the output polygon of one clipping step as the input polygon to the next step.



The planes that bound the truncated viewing pyramid are defined by the following:

1. the top plane, defined by

normal vector
$$- < 0, -\cos\left(\frac{\alpha}{2}\right), -\sin\left(\frac{\alpha}{2}\right) >$$

point $-(0, 0, 0)$

2. the left plane, defined by

normal vector
$$- < \cos\left(\frac{\alpha}{2}\right), 0, -\sin\left(\frac{\alpha}{2}\right) >$$

point $-(0, 0, 0)$

3. the bottom plane, defined by

normal vector
$$- < 0, \cos\left(\frac{\alpha}{2}\right), -\sin\left(\frac{\alpha}{2}\right) >$$

point $-(0, 0, 0)$

4. the right plane, defined by

normal vector
$$- < -\cos\left(\frac{\alpha}{2}\right), 0, -\sin\left(\frac{\alpha}{2}\right) >$$

point $-(0, 0, 0)$

5. the near plane defined by

normal vector
$$- < 0, 0, -1 >$$

point $-(0, 0, -n)$

6. the far plane defined by

normal vector
$$- < 0, 0, 1 >$$

point $-(0, 0, -f)$

Given a polygon, the polygon is clipped against each plane in turn utilizing the result of one clipping operation as the input to the next. Clipping is terminated if all points are clipped out at any one stage.

7 Clipping against the Image Space Cube

It is useful to see how the clipping operation simplifies when we clip against the image space cube. Consider the figure below, where we represent the top face of the image space cube.



The top plane is defined by the point $\mathbf{P} = (0, 1, 0)$ and the normal vector $\vec{n} = < 0, -1, 0 >$. If we consider the point $\mathbf{Q} = (x, y, z)$, the in/out test tells us that \mathbf{Q} is in if

$$<\mathbf{Q}-\mathbf{P}>\cdot\vec{n}>0$$

That is,

$$0 << \mathbf{Q} - \mathbf{P} > \cdot \vec{n}$$

=< (x, y, z) - (0, 1, 0) > \cdot < 0, -1, 0 >
=< (x, y - 1, z) > \cdot < 0, -1, 0 >
= 1 - y

or equivalently that \mathbf{Q} is "in" when y < 1 – which in retrospect, is obvious.

But the dot product that corresponds to the "in" test for the top plane is just y - 1 for any point (x, y, z). Similarly, the dot product for the other planes are as follows:

- x 1 for the plane defined by point (1, 0, 0) and the normal vector $\langle -1, 0, 0 \rangle$,
- z 1 for the plane defined by point (0, 0, 1) and the normal vector < 0, 0, -1 >,
- x + 1 for the plane defined by point (-1, 0, 0) and the normal vector < 1, 0, 0 >,
- y + 1 for the plane defined by point (0, -1, 0) and the normal vector < 0, 1, 0 >,
- z + 1 for the plane defined by point (0, 0, -1) and the normal vector < 0, 0, -1 >,

and so the "inside/outside" test for clipping in these cases contain dot products that do not require multiplies – which makes the algorithm very efficient.

8 Clipping in Projective Space

If we are careful, we can also clip in projective space. Here line segments are represented in the same form as in three-dimensional coordinates – that is,

$$\mathbf{Q} = \mathbf{Q}_1 + t < \mathbf{Q}_2 - \mathbf{Q}_1 >$$

represents a line in projective space, where $\mathbf{Q}_1 = (x_1, y_1, z_1, w_1)$ and $\mathbf{Q}_2 = (x_2, y_2, z_2, w_2)$ are points in four-dimensional projective space. To find the three-dimensional line that corresponds to this four-space line, we project the line onto the w = 1 plane, dividing by the w coordinate. This is illustrated in the figure below, where the two w coordinates are assumed positive.



In this case, a line in projective space simply projects to a line in three-dimensional space. However if one of the w coordinates, say w_2 , is negative then the line projected back into three dimensional space "passes through infinity" as is shown in the next illustration.



The viewing transform produces this second case frequently, for when a polygon is behind the viewer, the viewing transform produces points with a negative w coordinate. So these lines are produced whenever a polygon has vertices both in front of, and behind the viewer.

But we can still clip in projective space. Consider the following illustration, where the line lies on both sides of the w = 0 space.



We can find the intersection of this line with the w = 0 space by calculating the intersection point I, where

$$\mathbf{I} = \mathbf{Q}_1 + t < \mathbf{Q}_2 - \mathbf{Q}_1 >$$

and since I is in the w = 0 space, we must have that the w coordinate of I is zero, that is

$$0 = w_1 + t(w_2 - w_1)$$

and this implies that

$$t = \frac{w_1}{w_1 - w_2} \tag{1}$$

So, we can calculate the intersection of a line in projective space with the w = 0 space, and clip.

A second example is given by the following figure. Here we have the space w = y and a line crossing this space.



Here we can again calculate the intersection \mathbf{I} of the line with the space by

$$\mathbf{I} = \mathbf{Q}_1 + t < \mathbf{Q}_2 - \mathbf{Q}_1 >$$

and since I is in the space w = y, we have

$$w_1 + t(w_2 - w_1) = y_1 + t(y_2 - y_1)$$

which can be solved for t, giving

$$t = \frac{w_1 - y_1}{(w_1 - y_1) - (w_2 - y_2)} \tag{2}$$

Since we can calculate the intersection, we can clip polygons against this space.

9 Clipping in the Viewing Operation

In the viewing operation, the camera transformation transforms points from world space to image space. To implement this transformation we implemented a viewing transformation that produces points in projective space such that, when we project the point back to three-dimensional space, we obtain points in the image-space cube.

The problem, of course, is this projection. The viewing transform can produce points with a negative w coordinate – for when a polygon is behind the viewer, the viewing transform produces points with a negative w coordinate. These points, when projected produce the line segments that "pass through infinity" in three-dimensional space – highly undesirable in computer graphics renderings. So the problem when clipping in the viewing operation is that the points must remain in projective space, but we must still clip on the image-space cube in three-dimensional space.

But we have all the machinery now: We have shown how to clip line segments when our planes and points are in three-dimensional space and have shown how to clip against the image-space cube; and we have also seen that we can (at least in a few cases) clip in projective space. So how do we combine these operations and clip line segments against the image-space cube when our points are in projective space? It's not too difficult.

Let's reexamine the case where we clipped on the top plane of the image space cube, but lets consider a point that has been projected back from projective space.



Here, our three-dimensional "inside/outside" test tells us that the points $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$ is "in" if

$$0<\vec{n}\cdot<(\frac{x}{w},\frac{y}{w},\frac{z}{w})-(0,1,0)>$$

or

$$0 << 0, -1, 0 > \cdot < \frac{x}{w}, \frac{y}{w} - 1, \frac{z}{w} >$$

= $1 - \frac{y}{w}$

which states, in the case that \boldsymbol{w} is positive, that the point is "in" only if

$$w > y$$
 or $w - y > 0$

Now consider a line that crosses the plane y = 1 and assume that this line has the two endpoints

$$\mathbf{Q}_1 = \left(\frac{x_1}{w_1}, \frac{y_1}{w_1}, \frac{z_1}{w_1}\right) \text{ and } \mathbf{Q}_2 = \left(\frac{x_2}{w_2}, \frac{y_2}{w_2}, \frac{z_2}{w_2}\right)$$

both of which have been projected back from projective space.



Any point

$$\mathbf{Q} = \left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$$

that projects to the plane at the top of the image-space cube has the property that y = w, and so clipping the line segment is equivalent to clipping the projective-space line segment defined by the two points

$$\mathbf{Q}_1^P = (x_1, y_1, z_1, w_1)$$
 and $\mathbf{Q}_2^P = (x_2, y_2, z_2, w_2)$

against the space w = y in projective space.

But we have done this in the sections above! We can calculate the intersection \mathbf{I}^P by

$$\mathbf{I}^P = \mathbf{Q}_1^P + t(\mathbf{Q}_2^P - \mathbf{Q}_1^P)$$

where t is the value

$$t = \frac{w_1 - y_1}{(w_1 - y_1) - (w_2 - y_2)}$$

and then the point \mathbf{I}^{P} projected back to image space is the intersection of the line $\overline{\mathbf{Q}_{1}\mathbf{Q}_{2}}$.



One should notice that the t used to calculate the intersection in homogeneous space is not the same as if the t were calculated in three-dimensional space. In the above figure, we can see that the t in projective space is close to one-half, but the intersection in three-dimensional space is not close to the midpoint.

Now we can do the whole job. We can utilize the clipping method for simple planes in projective space to clip against the image space cube.

1. In order to first remove the points with a negative w coordinate we clip our projective coordinates against the w = 0 space. If we have a point (x, y, z, w), we use the distance measure w, and if given two points (x_1, y_1, z_1, w_1) and (x_2, y_2, z_2, w_2) , we use the ratio

$$\frac{w_1}{w_1 - w_2}$$

to calculate the intersection with the space.

2. To remove those points beyond the top plane of the image-space cube, we clip our projective coordinates against the w = y space. If we have a point (x, y, z, w), we use the distance measure y - w, and if given two points (x_1, y_1, z_1, w_1) and (x_2, y_2, z_2, w_2) , we use the ratio

$$\frac{w_1 - y_1}{(w_1 - y_1) - (w_2 - y_2)}$$

to calculate the intersection with the space.

3. To remove those points beyond the bottom plane of the image-space cube, we clip our projective coordinates against the w = -y space. If we have a point (x, y, z, w), we use the distance measure y + w, and if given two points (x_1, y_1, z_1, w_1) and (x_2, y_2, z_2, w_2) , we use the ratio

$$\frac{w_1 + y_1}{(w_1 + y_1) - (w_2 + y_2)}$$

to calculate the intersection with the space.

4. To remove those points beyond the left plane of the image-space cube, we clip our projective coordinates against the w = -x space. If we have a point (x, y, z, w), we use the distance measure x + w, and if given two points (x_1, y_1, z_1, w_1) and (x_2, y_2, z_2, w_2) , we use the ratio

$$\frac{w_1 + x_1}{(w_1 + x_1) - (w_2 + x_2)}$$

to calculate the intersection with the space.

5. To remove those points beyond the right plane in our image-space cube, we clip our projective coordinates against the w = x space. If we have a point (x, y, z, w), we use the distance measure x - w, and if given two points (x_1, y_1, z_1, w_1) and (x_2, y_2, z_2, w_2) , we use the ratio

$$\frac{w_1 - x_1}{(w_1 - x_1) - (w_2 - x_2)}$$

to calculate the intersection with the space.

6. To remove those points beyond the front plane of the image-space cube (which corresponds to clipping on the near plane), we clip our projective coordinates against the w = z space. If we have a point (x, y, z, w), we use the distance measure z - w, and if given two points (x₁, y₁, z₁, w₁) and (x₂, y₂, z₂, w₂), we use the ratio

$$\frac{w_1 - z_1}{(w_1 - z_1) - (w_2 - z_2)}$$

to calculate the intersection with the space.

7. To remove those points beyond the back plane of our image-space cube (which corresponds to clipping on the far plane), we clip our projective coordinates against the w = -z space. If we have a point (x, y, z, w), we use the distance measure z + w, and if given two points (x_1, y_1, z_1, w_1) and (x_2, y_2, z_2, w_2) , we use the ratio

$$\frac{w_1 + z_1}{(w_1 + z_1) - (w_2 + z_2)}$$

to calculate the intersection with the space.

A polygon is clipped against each plane in turn, utilizing the output of one clipping operation as the input to the next. Clipping is terminated if all points are clipped out at any one stage.

It should be noted that we do not necessarily clip against the front and back planes of the image-space cube, as there are frequently applications in which we will wish to see the points outside of the area enclosed by these planes.

One final important fact.

If we examine the viewing transformation, we note that the points behind the viewer get transformed to points with a negative w coordinate. We have already seen that these points cause line segments to "pass through infinity" in homogeneous space. This could cause these line segments to appear to wrap around the screen, which is undesirable in our pictures. The reason for our first clip (of the seven total clips) is to eliminate these problem points by first clipping so that no points will have a negative w coordinate.

All contents copyright (c) 1996, 1997, 1998, 1999 Computer Science Department, University of California, Davis All rights reserved.