Ray-Tracing with Affine Transforms

(Preprint)

Otto J. A. Smith

Jan 11, 1995

Abstract

We derive affine transformations that change the basis of a three dimensional world space. The new basis for this space is chosen in such a manner that many ray-tracing calculations are simplified, particularly when doing z-buffering. We begin with the simple problem of determining whether a point is contained in a triangle and extend our derivation to three space. We present some algorithm optimizations that can make real-world computer programs faster.

1 Triangle Containment

Let $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3)$ be a triangle in 2-space. Each $\bar{\mathbf{v}}$ and $\bar{\mathbf{x}}$ are 2-dimensional row vectors. p_1 and p_2 are scalars. (FIGURE 1)

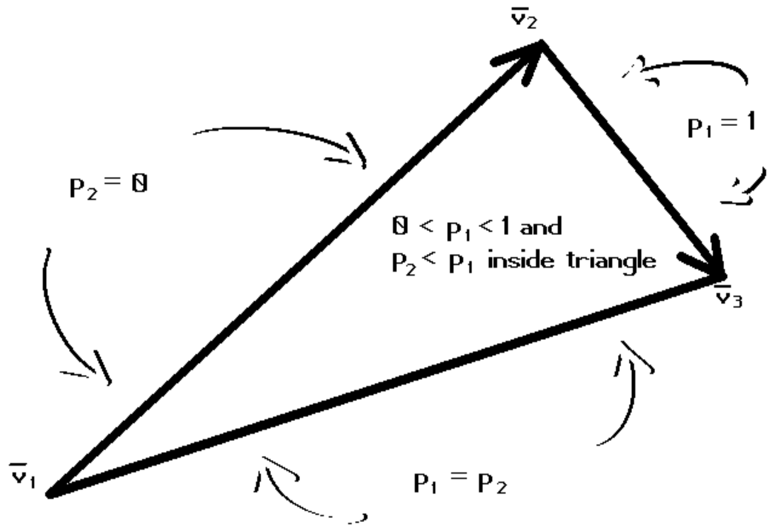

FIGURE 1

Given a point $\bar{\mathbf{x}}$, determine if $\bar{\mathbf{x}}$ is interior to $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3)$. In order to make this determination use vectors from the triangle $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3)$ to form a basis for the space, then any point $\bar{\mathbf{x}}$ can be represented by a parametric equation:

$$\bar{\mathbf{x}} = \bar{\mathbf{v}}_1 + p_1(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1) + p_2(\bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_2) \tag{1}$$

Now observe the following:

1. When $p_1 = 1$ and $0 < p_2 < 1$, then $\bar{\mathbf{x}}$ is on the line segment $(\bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3)$.

2. When $p_2 = 0$ and $0 < p_1 < 1$, then $\bar{\mathbf{x}}$ is on the line segment $(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2)$.

3. When $0 < p_1 < 1$ and $0 < p_2 < 1$ and $p_1 = p_2$, then $\bar{\mathbf{x}}$ is on the line segment $(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_3)$.

We constructed equation 1 in the manner given here in order to insure that the three observations made above were true. Now if $p_2 < p_1$ then $\bar{\mathbf{x}}$ is on the same side of the line determined by $(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_3)$ as $\bar{\mathbf{v}}_2$. If $p_2 > p_1$ then $\bar{\mathbf{x}}$ is on the opposite side of the line. In other words, the point $\bar{\mathbf{x}}$ is inside the triangle $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3)$ when the following are true:

1. $p_2 < p_1$
2. $0 < p_1 < 1$
3. $0 < p_2 < 1$

Given the triangle $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3)$ and $\bar{\mathbf{x}}$ one can calculate p_1 and p_2. Given the triangle $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3)$ alone, one can calculate a two by two working matrix $\mathbf{W}$ such that $(\bar{\mathbf{x}} - \bar{\mathbf{v}}_1)\mathbf{W} = [p_1, p_2]$ where $[p_1, p_2]$ is a row vector. We construct a matrix $\mathbf{M}$ and define $\mathbf{W} = \mathbf{M}^{-1}$. That is $\mathbf{W}\mathbf{M} = \mathbf{M}^{-1}\mathbf{M} = I$ where I is the identity matrix.

$$
\begin{aligned}
\bar{\mathbf{x}} &= \bar{\mathbf{v}}_1 + p_1(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1) + p_2(\bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_2) && (2)\\
\bar{\mathbf{x}} &= \bar{\mathbf{v}}_1 + [p_1, p_2]\begin{pmatrix} \bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1 \\ \bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_2 \end{pmatrix} && (3)\\
\text{Let } \mathbf{M} &= \begin{pmatrix} \bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1 \\ \bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_2 \end{pmatrix} && (4)\\
\text{Then } [p_1, p_2]\mathbf{M} &= \bar{\mathbf{x}} - \bar{\mathbf{v}}_1 && (5)\\
[p_1, p_2] &= (\bar{\mathbf{x}} - \bar{\mathbf{v}}_1)\mathbf{M}^{-1} && (6)\\
[p_1, p_2] &= (\bar{\mathbf{x}} - \bar{\mathbf{v}}_1)\mathbf{W} && (7)
\end{aligned}
$$

In applications where many points will be checked against a single triangle this is a fast algorithm. It is fastest if p_1 is calculated before p_2 and verified against the limits 0 and 1 so the algorithm can terminate prematurely if the point $\bar{\mathbf{x}}$ is not contained in the triangle. If there are many points, the cost of calculating $\mathbf{W} = \mathbf{M}^{-1}$ is amortized over many points. $\mathbf{W}$ is derived from the triangle and the focal point alone and not the point being checked. The point being checked is multiplied with the matrix $\mathbf{W}$. When $\mathbf{M}$ is singular, $\mathbf{W}$ does not exist since $\bar{\mathbf{v}}_1$,$\bar{\mathbf{v}}_2$ and $\bar{\mathbf{v}}_3$, all lie on a straight line. If this is the case the point is contained in the "triangle" (line segment) only if it is on the line defined by any two verticies from $\bar{\mathbf{v}}_1$,$\bar{\mathbf{v}}_2$ and $\bar{\mathbf{v}}_3$, and in addition has verticies from this set in both a positive and a negative direction.

2 Discussion of Basic Algorithm

I have found no references to this derivation of this technique in the literature although the technique presented here could be derived starting from barycentric coordinates. A discussion of this relationship is included in the appendix for this paper.

Before extending the algorithm to three (and higher dimensional spaces) and showing how it can simplify Ray-Tracing, particularly when doing z-buffering, we would like to enumerate some of its advantages and disadvantages.

One traditional method of determining containment in a triangle, or convex polygon, is to generate the equations for the bounding lines and then to determine which side of each bounding line the point falls on.

A second technique is based on the Jorden curve theorem[1].

I believe there are several reasons why the advantages of the triangle and hypertriangle containment algorithms presented here are not commonly used. One reason is that in most computer systems three dimensonal graphics geometry has been based on homogeneous[2] systems of coordinates and transforms taken from the theory of projective geometry, rather than affine transformations based in the theory of linear algebra. For most common transformations used in computer modeling systems, affine transforms and homogeneous transforms are equivalent. In a homogeneous transform matrix 12 of the cells represent a three vector affine transformation. The remaining four cells are used to generate 'projections'. Only one of these projections is actually used. In the approach taken in this paper, the equivalent projection is generated by skewing the affine transform. Through this and other manipulations we obtain more information from a single matrix multiply.

Another reason is that normally a basis for the space would be constructed

[1]Several of these techniques are compared by Eric Haines in Volume 5, Number 3, September 2, 1992, of the Ray Tracing News, an internet electronic journal. For triangles, his version of the barycentric coordinate system algorithm worked best for triangles, but the advantages were rapidly lost for polygons with more than three sides. We have some optimizations of the algorythm that drastically speed up these calculations.

[2]"homogenous", as used here does NOT mean a homogeneous system of equations, but derives from a projective geometry construct. Homogenous coordinates are used in computer graphics as a redundant method of storing vectors. That is in 3-space a vector of length 4 is used to store a coordinate. The equivalent of a matrix multiply, combined with a vector addition and a "projection" is then stored in a $4X4$ matrix.

from the vectors $\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1$ and $\bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_1$ (instead of $\bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_2$) so that hidden advantages of choosing other sets of vectors for the basis of the space have not been discovered. The point $\bar{\mathbf{v}}_1$ we call a common origin. The vector construction for detecting a point contained in a triangle given above in "Part 1, Triangle Containment" of this paper does not use a common origin. We will call techniques in which the vectors do not use a common origin Free Choice techniques. We will call techniques in which a common origin is used, Common Origin techniques.

I posted a common-origin algorithm on the Internet without comment about its derivation and recieved a reply from Prof. Dr. Heinrich Giesen pointing out that it is equivalent to the Barycentric coordinate technique, which is a well known technique. The demonstration of equivalency is included in the Appendix.

We will not use the common origin technique in this paper because by using free choice techniques, we can customize the results of our basis change to give us additional information that is useful and increases the efficiency of our algorithms.

3 Multi Dimensional Extensions

The techniques presented above are easily extended to multi-dimensional spaces. Let $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \ldots, \bar{\mathbf{v}}_n)$ be a hypertriangle in $(n-1)$-space. Given a point $\bar{\mathbf{x}}$, determine if $\bar{\mathbf{x}}$ is interior to $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \ldots, \bar{\mathbf{v}}_n)$. In order to make this determination use vectors from the hypertriangle $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \ldots, \bar{\mathbf{v}}_n)$ to form a basis for the space.

3.1 The Free Choice Extension

Any point $\bar{\mathbf{x}}$ can be represented by a parametric equation:

$$\bar{\mathbf{x}} = \bar{\mathbf{v}}_1 + p_1(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1) + p_2(\bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_2) + \cdots + p_{n-1}(\bar{\mathbf{v}}_n - \bar{\mathbf{v}}_{n-1}) \tag{8}$$

Now when all of the following are true, the point is contained in the hypertriangle.

1. When $0 < p_i < 1$, and
2. When given any i, j such that $i < j$ then $p_i \geq p_j$.

3.2 Common Origin Extension

In the common origin technique represent an arbitrary point $\bar{\mathbf{x}}$ as follows:

$$\bar{\mathbf{x}} = \bar{\mathbf{v}}_1 + p_1(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1) + p_2(\bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_1) + \cdots + p_{n-1}(\bar{\mathbf{v}}_n - \bar{\mathbf{v}}_1) \tag{9}$$

Now when all of the following are true, the point is contained in the hypertriangle.

1. When $0 < p_i$ and

2. When $\sum_{i=1}^{i=n} p_i < 1$.

Given the triangle $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \ldots, \bar{\mathbf{v}}_n)$ and $\bar{\mathbf{x}}$ we can calculate $p_1 \ \ldots \ p_n$. Given the triangle $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \ldots, \bar{\mathbf{v}}_n)$ alone we can calculate a working matrix $\mathbf{W} = \mathbf{M}^{-1}$ such that $(\bar{\mathbf{v}}_x - \bar{\mathbf{v}}_1)\mathbf{W} = [p_1, p_2, \ldots, p_n]$. In this paper we will most frequently use the free choice technique to choose vectors with which to calculate the matrix.

4 Two Perspective Problems

There are two perspective problems that are easily solved using hypertriangle containment.

The two problems for which we disclose solutions here are,

1. The ray tracing perspective projection problem.

2. The standard perspective problem.

4.1 The Standard Perspective Problem

The standard perspective problem is as follows. (FIGURE 2)

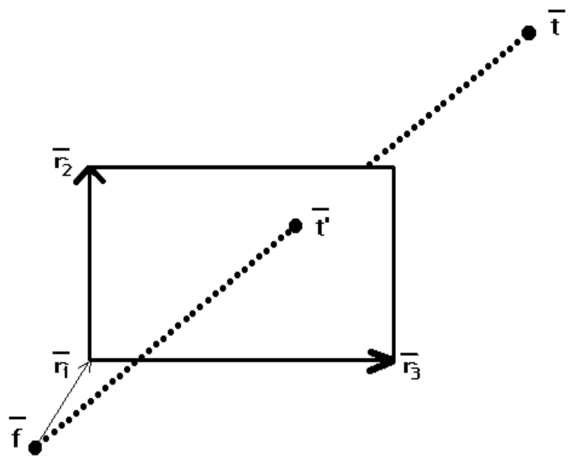

FIGURE 2

We are given the following information:

1. The location of a single point $\bar{\mathbf{t}}$, in world coordinates in three-space.
2. The location of the focal point, $\bar{\mathbf{f}}$ of an eye or a camera in world coordinates in three space.
3. The location of the projection plane P in world coordinates in three space. This location is determined by a set of three points, so P is defined by the set $\{\bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \bar{\mathbf{r}}_3\}$[3].

We are asked to determine the point at which a line from the focal point of the eye $\bar{\mathbf{f}}$ to the single point $\bar{\mathbf{t}}$ intersects the projection plane P. If this point of intersection exists, we will call it $\bar{\mathbf{t}}'$ and we are asked to determine its coordinates in the two dimensional rectangular coordinate system of the projection plane P. The projection plane represents the surface of a video display device, such as a CRT tube and the coordinate system in which we want the information returned is the pixel coordinates of the CRT tube.

[3]That is, the three points $\{\bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \bar{\mathbf{r}}_3\}$ determine a rectalinear area in three space that we are interested in, possibly the origin, the upper left hand corner and the lower right hand corner of a CRT tube.

4.2 The Ray Tracing Perspective Problem

The ray tracing perspective problem is as follows. (FIGURE 3)

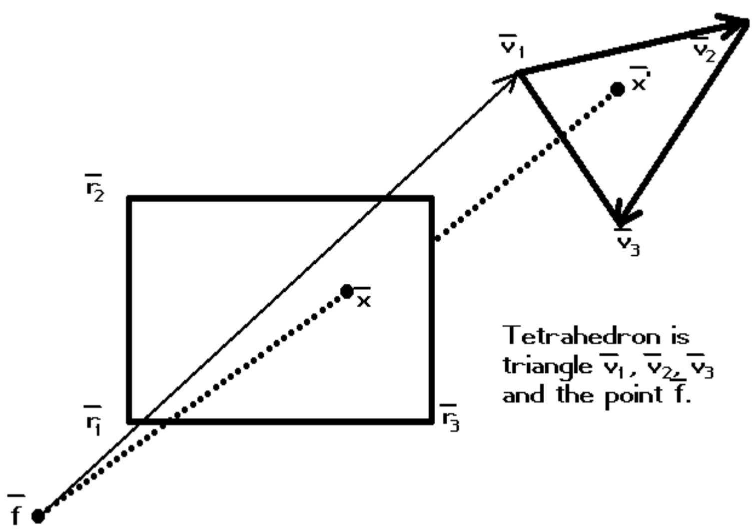

FIGURE 3

We are given the following information:

1. The location of the focal point, $\bar{\mathbf{f}}$ of an eye or a camera in world coordinates in three space.

2. The location of the projection plane P in world coordinates in three space, (this is three points, the origin of the screen, the upper left of the screen and the lower right).

3. A point $\bar{\mathbf{x}}$ on P in world coordinates[4].

[4]Translating pixel coordinates to world coordinates is easily accomplished. Let $\bar{\mathbf{x}}_0$ be the world space coordinate of the origin of the screen at pixel coordinates $(0, 0)$, let $\bar{\mathbf{x}}_1$ be the world space coordinates of the top left of the screen at pixel coordinates $(639, 0)$. The world space coordinates of $(X, 0)$ is then $\bar{\mathbf{x}}_0 + X(\bar{\mathbf{x}}_1/639)$. Similarly for the y axis. Generate 640 values for x and 400 values for y and access them by their indicies to avoid frequent recalculations when generating rays thruogh pixels.

4. A two dimensional triangle $T = \triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3)$ in world coordinates[5].

We are asked to determine two different things:

1. Does the line starting at the focal point of the eye $\bar{\mathbf{f}}$ and passing through the point $\bar{\mathbf{x}}$, intersect the triangle T? If the intersect exists call it $\bar{\mathbf{x}}'$.
2. If the intersection exists, what is a measure of the distance from the point $\bar{\mathbf{x}}$ to its intersection on the triangle T at $\bar{\mathbf{x}}'$.

5 Ray Tracing Problem Solved

Let us use the free choice tetrahedron containment algorithm to develop a ray tracing perspective technique. If the line starting at the focal point of the eye $\bar{\mathbf{f}}$ and passing through the point $\bar{\mathbf{x}}$ intersects the triangle T, the first item we are asked to determine, is equivalent to asking if the point $\bar{\mathbf{x}}$ is contained in the tetrahedron $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3, \bar{\mathbf{f}})$. That is: is $\bar{\mathbf{x}}$ contained in the tetrahedron defined by $T \cup \bar{\mathbf{f}}$? The second item we are asked to determine, is a measure of the distance from $\bar{\mathbf{x}}$ to its intersection on T. This measure is used for z-buffering. We get this measure as a side benifit of using the free choice technique of detecting whether $\bar{\mathbf{x}}$ is in $\triangle(\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{v}}_3, \bar{\mathbf{f}})$. Let the point $\bar{\mathbf{x}}$ be represented as[6]:

$$\bar{\mathbf{x}} = \bar{\mathbf{f}} + p_1(\bar{\mathbf{v}}_1 - f) + p_2(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1) + p_3(\bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_2) \tag{10}$$

Calculate the working matrix $\mathbf{W} = \mathbf{M}^{-1}$ that transforms a point into the new coordinate system represented by the above equation. That is:

$$\mathbf{M} = \begin{pmatrix} \bar{\mathbf{v}}_1 - f \\ \bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1 \\ \bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_2 \end{pmatrix} \tag{11}$$

[5] A triangle is the simplest object we may be asked to ray-trace. Polygons can be triangulated. Spheres, toruses and other mathematical objects may require more refined techniques.

[6] It is also possible to substitute a representation as in equation 13 below for equation 10 and change some of the checking conditions. This may be desirable in a real world program in order to make the code more consistant and shorter.

from the three by three matrix $\mathbf{M}$ calculate $\mathbf{W} = \mathbf{M}^{-1}$. Represent $\mathbf{W} = \mathbf{M}^{-1}$ as a matrix of column vectors.

$$\mathbf{W} = \begin{pmatrix} \bar{\mathbf{c}}_1 & \bar{\mathbf{c}}_2 & \bar{\mathbf{c}}_3 \end{pmatrix} \tag{12}$$

The algorithm to obtain the information we need is then as follows:

1. Calculate the dot product $p_1 = (\bar{\mathbf{x}} - \bar{\mathbf{f}}) \cdot \bar{\mathbf{c}}_1$. If this product is greater than one or less than zero the algorithm terminates. The point $\bar{\mathbf{x}}$ is not in the tetrahedron.

2. Calculate the dot product $p_2 = (\bar{\mathbf{x}} - \bar{\mathbf{f}}) \cdot \bar{\mathbf{c}}_2$. If this product is greater than p_1 or less than zero the algorithm terminates. The point $\bar{\mathbf{x}}$ is not in the tetrahedron.

3. Calculate the dot product $p_3 = (\bar{\mathbf{x}} - \bar{\mathbf{f}}) \cdot \bar{\mathbf{c}}_3$. If this product is greater than p_2 or less than zero the algorithm terminates. The point $\bar{\mathbf{x}}$ is not in the tetrahedron.

4. If the algorithm has not terminated, and we reach this step, the point $\bar{\mathbf{x}}$ is contained in the tetrahedron. We can use p_1 as a measure of the distance from $\bar{\mathbf{x}}$ to the triangle T since p_1 is the ratio of the length of the line segment $(\bar{\mathbf{f}}, \bar{\mathbf{x}})$ to the length of the line segment $(\bar{\mathbf{f}}, \bar{\mathbf{x}}')$ where $\bar{\mathbf{x}}'$ is the point at which the line intersects T. Since $(\bar{\mathbf{f}}, \bar{\mathbf{x}})$ has a fixed length, we can use the ratio as a measure.

In an actual computer system several things must happen in order to perform a ray tracing of a world object. First, areas that are not partially contained on the screen are clipped. One technique for doing this is disclosed as part of the algorithm for the standard perspective technique[7], then the real world coordinates for each pixel on the display screen is generated, then the ray from the focal point through the screen pixel is compared against triangles. The value of p_1, our measure, is used for z-buffering to determine which points are in front of or behind other points.

[7]There are other more effective ways of doing this clipping when using BSD trees and other data structures that partition the world space, rather than depending on the algorithms presented here alone.

6 Standard Perspective Problem Solved

In the standard perspective problem we are given the following information:

1. The location of a single point $\bar{\mathbf{t}}$, in world coordinates in three-space.
2. The location of the focal point, $\bar{\mathbf{f}}$ of an eye or a camera in world coordinates in three-space.
3. The location of the projection plane P in world coordinates in three space, (this is three points, the origin of the screen, the upper left of the screen and the lower right.).

Let the viewing area of P that we are interested in be represent by three coordinates in world-space. $P = \{\bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \bar{\mathbf{r}}_3\}$. That is, P is represented by the set of three vectors $\bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \bar{\mathbf{r}}_3$. In all "real world" situations, the vectors $\bar{\mathbf{r}}_2 - \bar{\mathbf{r}}_1$ and $\bar{\mathbf{r}}_3 - \bar{\mathbf{r}}_1$ are frequently chosen to be orthogonal, simply because the viewing area of the CRT is rectangluar and has right angle corners. We can think of $\bar{\mathbf{r}}_1$ as the lower left hand corner of the CRT, $\bar{\mathbf{r}}_2$ as the lower right hand corner and $\bar{\mathbf{r}}_3$ as the upper left hand corner. Let $\bar{\mathbf{t}}$ be represented by the equation,

$$\bar{\mathbf{t}} = f + p_1(\bar{\mathbf{v}}_1 - \bar{\mathbf{f}}) + p_2(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1) + p_3(\bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_1) \tag{13}$$

This equation has been generated differently than any of the above equations. It has been constructed this way in order to make two questions easy to answer.

1. Is $\bar{\mathbf{t}}$ a point that can be projected onto P such that $\bar{\mathbf{t}}'$, the projected point is in the rectangle determined by $\{\bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \bar{\mathbf{r}}_3, \bar{\mathbf{r}}_2 + (\bar{\mathbf{r}}_3 - \bar{\mathbf{r}}_1)\}$?
2. What are the coordinates of $\bar{\mathbf{t}}'$ in the two dimensional coordinate system determined by P where $\bar{\mathbf{r}}_1$ is chosen to have the coordinates $(0,0)$, $\bar{\mathbf{r}}_2$ is chosen to have the coordinates $(1,0)$ and $\bar{\mathbf{r}}_3$ is chosen to have the coordinates $(0,1)$[8]?

[8]This coordinate system is used to make reading this paper easy. In fact, in 'real world' computer applications, the coordinates would be chosen to represent pixel coordinates on a CRT tube. In otherwords, for a full screen 400 X 640 VGA system appropriate coordinates might be $\bar{\mathbf{r}}_1 = (0,0)$, $\bar{\mathbf{r}}_2 = (640,0)$ $\bar{\mathbf{r}}_3 = (0,400)$

Now construct $\mathbf{W} = \mathbf{M}^{-1}$ from $\mathbf{M}$ as determined by equation 13 above where we express $\mathbf{M}$ as:

$$\mathbf{M} = \begin{pmatrix} \bar{\mathbf{v}}_1 - f \\ \bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1 \\ \bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_1 \end{pmatrix} \tag{14}$$

We calculate $\mathbf{W} = \mathbf{M}^{-1}$ from $\mathbf{M}$ and let the columns of $\mathbf{W}$ be represented as:

$$\mathbf{W} = \begin{pmatrix} \bar{\mathbf{c}}_1 & \bar{\mathbf{c}}_2 & \bar{\mathbf{c}}_3 \end{pmatrix} \tag{15}$$

Now in order to solve the standard perspective problem we do the following:

1. Calculate $p_1 = (\bar{\mathbf{t}} - \bar{\mathbf{f}}) \cdot \bar{\mathbf{c}}_1$. If $p_1 < 1$ then the point $\bar{\mathbf{t}}$ does not project onto the rectangle of interest on P and the algorithm terminates.

2. Calculate $p_2 = (\bar{\mathbf{t}} - \bar{\mathbf{f}}) \cdot \bar{\mathbf{c}}_2$. If $p_2 < 0$ or $p_2 > 1$ then the point $\bar{\mathbf{t}}$ does not project onto the rectangle of interest on P and the algorithm terminates.

3. Calculate $p_3 = (\bar{\mathbf{t}} - \bar{\mathbf{f}}) \cdot \bar{\mathbf{c}}_3$. If $p_3 < 0$ or $p_3 > 1$ then the point $\bar{\mathbf{t}}$ does not project onto the rectangle of interest on P and the algorithm terminates.

4. If the algorithm has not terminated, then the point $\bar{\mathbf{t}}$ does project onto P in the area of interest and has the coordinates $\bar{\mathbf{t}} = (p_2/p_1, p_3/p_1)$

7 Optimizing the algorithms

There are several ways in which the algorithms can be optimized. Without going into extensive detail we will give several of them here. Some properties of the triangles are invarient from different viewpoints. In the ray tracing problem only one of the rows of the three by three matrix $\mathbf{M}$ depends upon the viewpoint when the matrix is generated using the free-choice technique. Consequently, two rows of the matrix $\mathbf{M}$ are invarient and need be calculated only once for all viewpoints. This has the added benefit, that if the inversion is done by calculating the determinant by expansion of minterms, some of the minterms need be calculated only once for all viewpoints. Optimizations similar to the above can be realized for triangles that share a common edge.

One can avoid dividing by the determinant when calculating the inverses and postpone this computation until the comparisons have been made. In this case comparisons are made against the values of the determinant instead of one, and the divisions are only made if the containment has been established and we need to complete the computation.

Any set of four non-coplanar (non-colinear) points in three space can act as a basis for the space. From these four points we can find an affine transformation that takes any point in world space into a new coordinate system defined by these points. We can also find an inverse transformation that takes points from our new coordinate system back into world space.

The affine transformation shares many of the advantages of the homogeneous transformation system. This fact is not new, and has been mentioned in the literature before. The main arguments in favor of using a homogeneous coordinate system have been that, first, it reduces the complexity of code. With modern object oriented languages the code is only minimally more complex for the basic operations with a great increase in simplicity for more complex operations. Secondly, homogeneous coordinate systems are used because of the ease of explaining there use and for historical reasons. The divisions needed in the last step of performing a projection, (see Standard Perspective Problem Solved) above are encoded as part of the homeogeneous matrix. There is a cost associated with this since normally the projection is only performed along one axis, so that the matrix contains elements that are always zero and regardless of which technique you use, these divisions eventually have to be performed.

This paper is an outgrowth of a larger research project in computer graphics that began with a 3-D modelling and display program that ran on an Apple II-E written around 1983. That system did not use homogeneous coordinates and transformations but used affine transforms instead to translate points and calculate perspective. The free choice algorithms are a recent invention derived from that work. Containment is calculated by generating an affine transformation that takes the point we are interested into a space in which simple comparisons can determine containment and the new coordinate system contains desirable information.

In that 1983 computer system, 'cameras', that is coordinate systems associated with a viewpoint were stored or created from four points in three space. For orthographic projections the three vectors generated from these four points were orthogonal. Each camera could be used to generate an affine transformation that would move a vector from world space to camera space. Perspective transforms were done as they are traditionally done rather than as presented herein above. Transforms could be convolved into new transforms as they are in systems using homogeneous coordinates. Inverses of transforms could also be calculated to take a vector from camera space to world space. Transforms could also be converted back to cameras.

The system provided a simple language in which sequences of transforms and objects could be generated mathematically.

Some major advantages of the system were.

1. It was significantly faster than equivalent systems on the same machine.

2. Planes and lines did not need to be expressed in their cartesian form since they were easily expressed simply as two points or three points respectively.

3. Projections onto polygons were simplified.

4. The 30% overhead associated with storage of homogeneous coordinate system matricies was avoided.

5. Transformations from any camera (coordinate system) to any other camera in the system was easily and simply achieved.

On a completely biased note, I prefer the kind of system presented here not only because I think it simplifies code, but because the Mathematics of what is happening appears to me to be simpler and more transparent to a user of the system.

A Appendix

Represent an arbitrary point $\bar{\mathbf{x}}$ using the equation:

$$\bar{\mathbf{x}} = \bar{\mathbf{v}}_1 + p_1(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1) + p_2(\bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_1) \tag{16}$$

Now the point $\bar{\mathbf{x}}$ is contained in the triangle only when all of the following conditions hold:

1. When $0 < p_2$ and
2. When $0 < p_1$ and
3. When $p_1 + p_2 < 1$.

We call this technique, the common origin technique. (FIGURE 4)

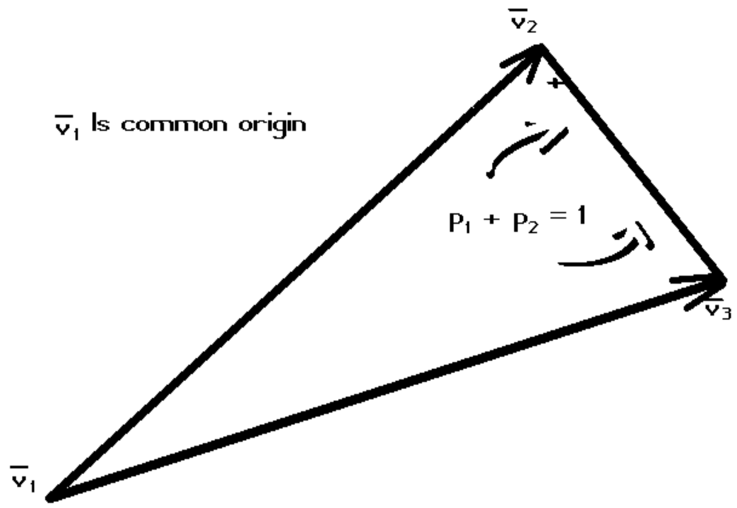

FIGURE 4

This is based upon a traditional convention that when calculating a new basis for a space, the origin of the vectors originate from a common point.

This is equivalent to the using barycentric coordinates. Barycentric coordinates are based upon the fact that any point can be represented as the sum of the verticies of a triangle times a weight, where the sum of the weights equals one.

$$\bar{\mathbf{x}} = p_0\bar{\mathbf{v}}_1 + p_1\bar{\mathbf{v}}_2 + p_2\bar{\mathbf{v}}_3 \tag{17}$$

$$p_0 + p_1 + p_2 = 1 \tag{18}$$

let $p_0 = 1 - p_1 - p_2$ then 17 above becomes

$$\bar{\mathbf{x}} = (1 - p_1 - p_2)\bar{\mathbf{v}}_1 + p_1\bar{\mathbf{v}}_2 + p_2\bar{\mathbf{v}}_3 \tag{19}$$

$$\bar{\mathbf{x}} = \bar{\mathbf{v}}_1 + p_1(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1) + p_2(\bar{\mathbf{v}}_3 - \bar{\mathbf{v}}_1) \tag{20}$$

This is identical to 16 above.