Factoring a Homogeneous Transformation for a more Efficient Graphics Pipeline

Salim S. Abi-Ezzi† and Michael J. Wozny‡

Abstract

We identify an intermediate coordinate system situated between world coordinates and display coordinates, which exhibits unique features for lighting calculations and for clipping in homogeneous coordinates. Our key contribution is an algorithm for extracting such a coordinate system from a homogeneous viewing transformation that relates WC to DC. The algorithm is based on factoring the transformation into a product of a Euclidean factor and a sparse (computationally cheap) but non-Euclidean factor.

A particularly strong application of the proposed technique is the graphical processing of curved surface primitives, such as what is needed in the PHIGS PLUS viewing pipeline. Furthermore, in PHIGS PLUS the graphical data is retained by the graphics system, therefore, it is possible to perform the factoring of the viewing transformation at creation time, and to take advantage of this factored form at traversal time.

1. Introduction

In recent graphics systems such as GKS-3D and PHIGS[1,2], the relationship between world coordinates (WC) and display coordinates (DC) is specified as a 4x4 homogeneous transformation. In the case of smooth shaded graphics based on Bezier or B-spline surface primitives, see PHIGS PLUS[3] and Figure 1, there is an important issue in regard to such a specification. On one hand, in order to avoid the cost of transforming the large volume of sample points generated during the tessellation of these surfaces, it is advantageous to transform control points to display coordinates before tessellation. On the other hand, conceptually, lighting takes place in world coordinates, and lighting is dependent on angles and distances which are distorted by non-Euclidean transformations.

Due to the distortions of angles and distances by perspective transformations, tessellation and lighting in WC versus DC results with different images. Even though rational Bezier and B-spline surfaces are closed under homogeneous transformations', the colours generated in DC will be different from those generated in WC, which will result with a perception of a different geometry. We ran experiments to study the magnitude of these differences, and we observed that they are usually dramatic even under straight forward perspective views; see Figure 2. Therefore, the problem is that we are interested in performing lighting computations in WC and tessellation in DC, and at the same time in order to perform the lighting computations we need the sample points generated by the tessellation process.

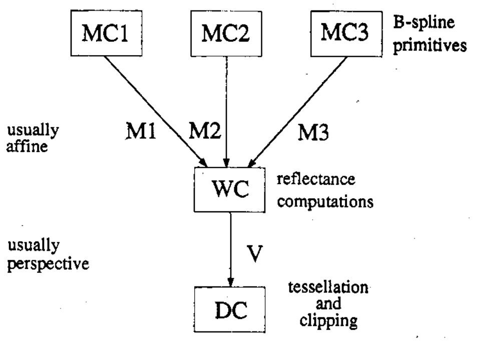


Figure 1. In a simplified look at the PHIGS+ pipeline, surface primitives are specified in different modeling coordinate (MC) systems and are mapped via modeling transformations to a unique world coordinate system. Conceptually, lighting takes place in WC while tessellation takes place in DC since the approximation criteria, which controls the fineness of tessellation, is specified in terms of pixel size.

Current solutions to this problem involve using cubic Hermitian polynomials to approximate the surface normal function in WC; see Rockwood et al[4] and Shantz et al[5]. Then the tessellation takes place in DC while the approxi-

This paper was presented at the EUROGRAPHICS Conference, Montreux, September 3-7, 1990 The editor's felt that members would appreciate the opportunity to see this interesting paper with the full colour illustrations and have therefore great pleasure in including it with this issue.

†Graphics Systems Products
Sun Microsystems, Inc
Mt View, CA 94043, USA

‡Design Research Center
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

#By closed we mean that the geometry is preserved by the transformation. We get the same geometry if we transform control points and tessellate post-transformation as we do by tessellating pre-transformation and then transforming the tessellants.

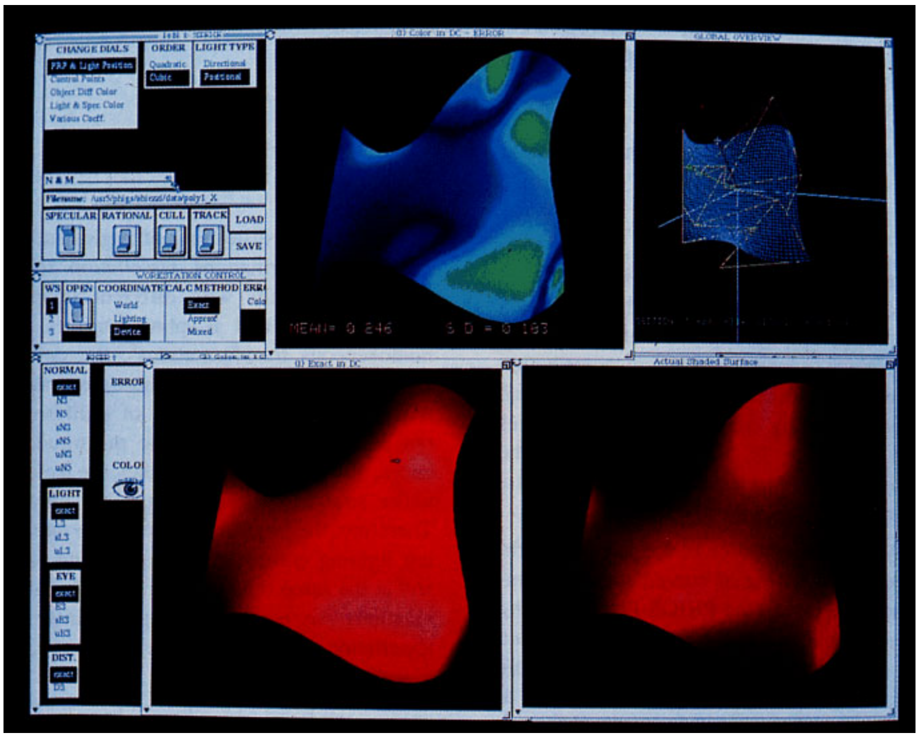
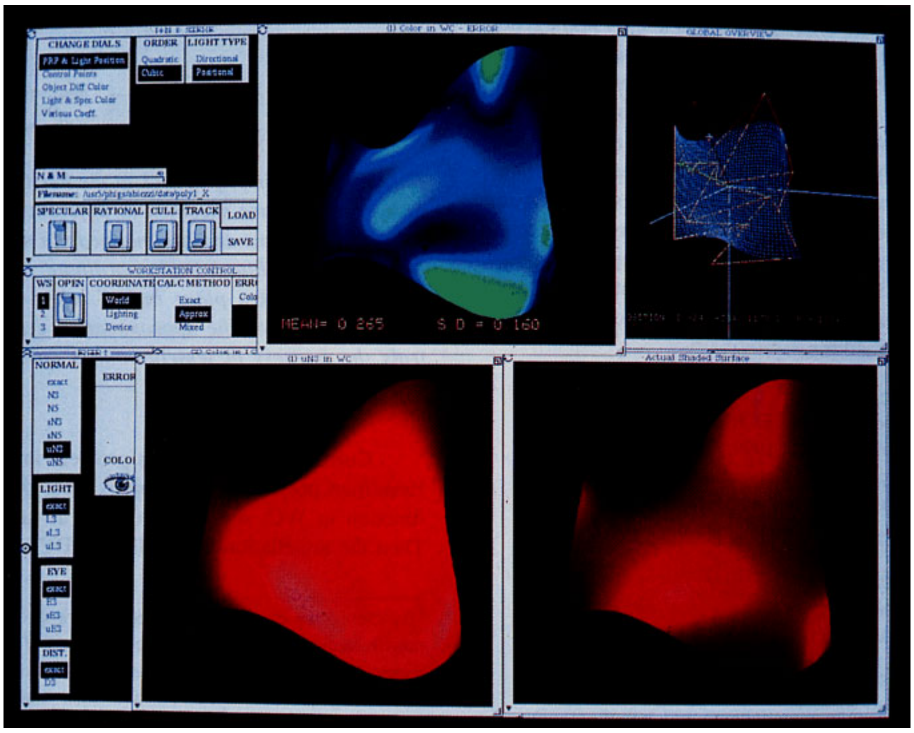


Figure 2. A general tool for visually comparing between two renderings of the same geometry. In this case we show the accurate rendering in the lower right window (by accurate rendering we mean that the lighting takes place in WC and the normal function is evaluated exactly), a distorted rendering due to a perspective transformation in the lower- left window, and the difference between the two in the upper window. A single positional light source is used.



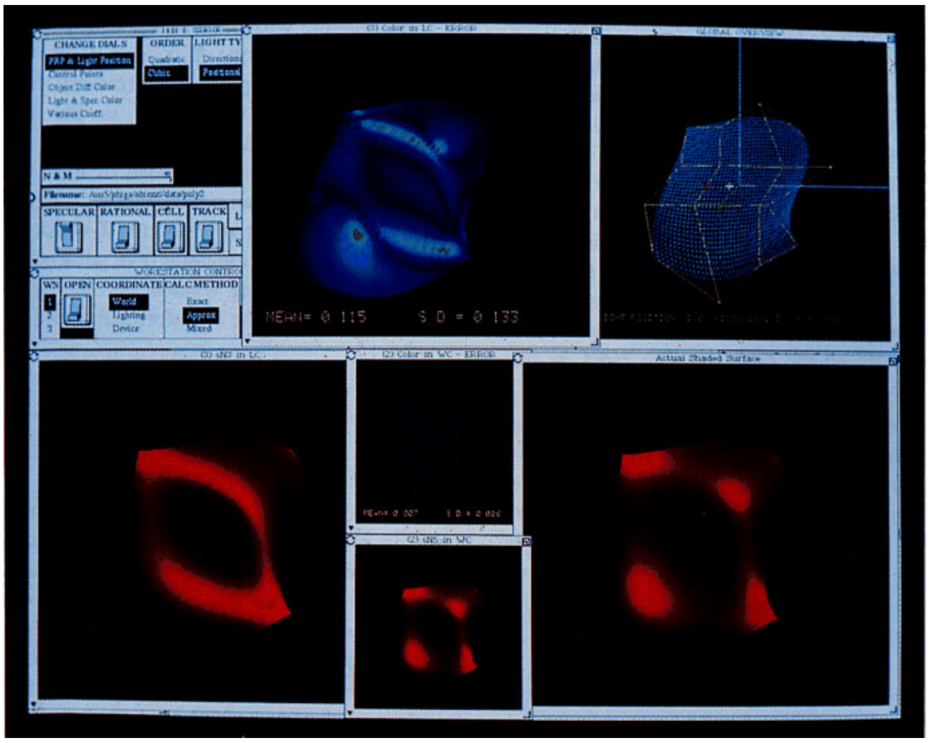


Figure 4. Here we show the accurate rendering in the lower right window, a distorted rendering due to the cubic Hermite approximation of the unnormalized normal function in the lower left window, and the difference between the two in the upper window. In the two small center windows we show that a quintic Hermitian approximation of the unnormalized normal function gives an exact accurate rendering, which is not surprising given that the function itself is quintic in that case.

mation to the normal function is evaluated in WC. This approach does not work for the following reasons:

1. The normal vector is not the only entity needed for lighting computations. There is also a need for the vectors from eye point to sample point in case of a perspective view, and from light, source to sample point in case of a finite light source. Notice that these vectors include distance information which is needed for attenuation computations. Therefore, it is not sufficient to approximate the normal function in WC, and at the same time it is not practical to approximate all the other needed parameters, since this would defy the efficiency goals of the process.
2. In spite of the above point it is still desirable to approximate the normal function for more efficient evaluation. Our experiments with using a cubic Hermitian to approximate the normal function, whether normalized or unnormalized, lead to unacceptable results; see Figures 3 and 4. A technique is acceptable only if it approximates the normal function to within a specified degree of tolerance. The approximation must have enough degrees of freedom to scale appropriately with the complexity and the degree of the underlying surface primitive, otherwise the approximation will likely to lead to meaningless results.

In this paper we present a method that produces accurate images, as if lighting took place in WC, without paying the price of transforming the large volume of sample points from WC to DC. The method is based on an algorithm for factoring a viewing transformation, which in effect identifies a coordinate system with unique properties for performing the tessellation and lighting processes, and hence it is called the lighting coordinate (LC) system.

The lighting coordinate system also has special characteristics for clipping in homogeneous coordinates. We exploit these characteristics and present a clipping solution that has distinct advantages over previously known ones.

Figure 3. Here we show the accurate rendering in the lower right window, a distorted rendering due to the cubic Hermite approximation of the normalized normal function in the lower left window, and the difference between the two in the upper window.

2. The Specification of a View

A view consists of both a viewing transformation, V, and a clipping volume called a viewport that is specified in DC. By convention, the viewport must be a rectangular parallelepiped with a standard orientation*; therefore, it has six faces and three sets of four parallel edges each. The faces can be appropriately labeled as the $X_{\min}$, $X_{\max}$, $Y_{\min}$, $Y_{\max}$, $Z_{\min}$, and $Z_{\max}$ faces; while the sets of lines that carry the edges are labeled the X, Y, and Z lines, depending on the axes to which they are parallel. Since the lines in each of the X, Y, and Z sets are parallel, they meet in ideal points† P_x, P_y, and P_z, respectively. By definition, the near and far faces of the viewport are the $Z_{\max}$ and $Z_{\min}$ faces, respectively‡.

Assuming the viewing transformation is non-singular, it's inverse V^{-1} maps the viewport into a *viewing window* in WC; see Figure 5. This viewing window could be either bounded or unbounded§, in either case it must have six faces and twelve edges that correspond to the faces and edges of the viewport. We use viewport labels for corresponding window entities, which include near and far window faces and the X, Y, and Z sets of window lines. Furthennore, from projective geometry the X, Y, and Z sets of window lines must meet in the following points: $Q_x = V^{-1}P_x$, $Q_y = V^{-1}P_y$, and $Q_z = V^{-1}P_z$, respectively; which may be either Euclidean or ideal. In Figure 5, Q_y and Q_z are Euclidean points, while Q_x is an ideal point. By definition, the point at which the Z-window lines meet, Q_z, is called the projection point (eye point) of the view. Notice that both the viewing window and the eye point of a view are implicit with the transformation and viewport of that view.

A view may be characterized by the nature of its viewing window, as discussed in the following sections.

2.1. A View in Canonical Form

In practice, a view is usually intended to model the process of an arbitrarily oriented camera capturing an image of WC. The *field of view* of such a camera is typically rectangular. Based on this observation we consider a view to be in *canonicalform* if-and-only-if the near and far faces of its window are rectangles; as opposed to arbitrary quadrilaterals. For this condition to hold, it is necessary and sufficient that 1) both Q_x and Q_y are ideal points, and 2) the directions corresponding to these points are orthogonal.*

*Standard orientation means the faces of the rectangular parallelepiped are parallel to the standard planes XY, YZ, and XZ.

†An ideal point in projective space, i.e. with a w = 0 coordinate, has an infinite point counterpart in Euclidean space, in which case it is represented as a direction.

‡In DC the positive Z-axis is assumed to be pointing towards the viewer.

§In a special case the viewing window is an unbounded truncated double pyramid, which in projective space intersects the ideal plane and in Euclidean space wraps around infinity.

*The PHIGS viewing utility routines, "compute view mapping" and "compute view orientation" always produce views in canonical form, since by convention the near and far faces of the viewing window are rectangular.

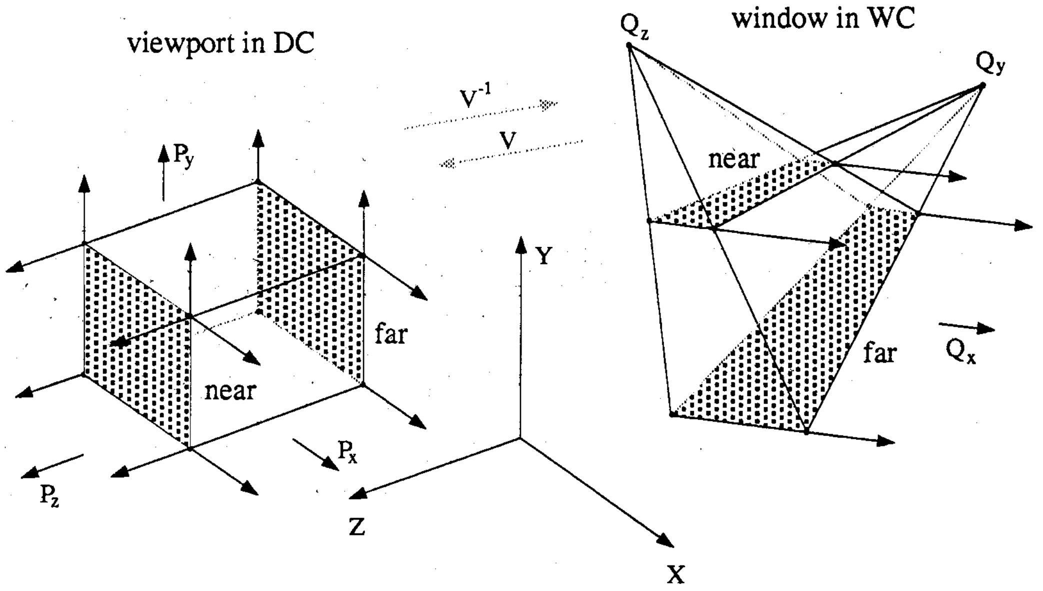


Figure 5. A view as a mapping ofa viewing volume with an arbitrary orientation in WC to a viewport with a standard orientation in DC. The view is not in canonicalform since Q_y is Euclidean.

The view depicted in Figure 5 is not in canonical form, since Q_y is not ideal. Notice that a view with an unbounded viewing window may still be canonical. In this paper we focus on views in canonical form, since they represent what's commonly used in practice*.

2.2. A Parallel View

A view is said to be a *parallel view* if-and-only-if its viewing window is a parallelepiped. This is an equivalent condition to having Q_x, Q_y, and Q_z as ideal points; or equivalently, having the viewing transformation as affine non-singular. Hence, a necessary but not sufficient condition for a view to be parallel is to have a projection point at infinity. Furthermore, not every parallel view is in canonical form, while every view that is in canonical form and whose projection point is ideal is a parallel view. In such a case, if the direction of projection is perpendicular to the near and far faces, then the view is orthonormal, otherwise it is oblique.

2.3. A Perspective View

A view is said to be a *perspective* view if-and-only-if its viewing window is not a parallelepiped. This is equivalent to having at least one of the points Q_x, Q_y, or Q_z as a Euclidean point; or equivaiently, having the viewing transformation as projective non-affine. A perspective view in canonical form must have a Euclidean projection point, since we know that both Q_x and Q_y must be ideal based on the requirement of the canonical form.

3. An Ideal Lighting Coordinate System

Conceptually, the lighting process takes place in world coordinates; however, due to an efficiency reason, we investigate a more appropriate coordinate system for performing this process. The reason is that a typical viewing transformation (which relates WC to DC), is usually a full 4 x 4 matrix at a cost of 28 FLOPs per point', which makes it expensive to transform the large volume of sample points after lighting to DC. Therefore, a coordinate system is considered to be ideal for lighting computations by meeting two requirements: 1) it leads to the same results as if the lighting computations were done in WC, and 2) it is computationally as close as possible to DC.

The approach we take for finding this Lighting Coordinate (LC) system is based on a factoring technique of the viewing transformation V. The technique attempts to express V as follows: $V = RE$, where E is Euclidean and R is as sparse as possible. After we achieve this form for V, then LC is the coordinate system that is related to WC via E.

Specifically, we give such factoring techniques for both perspective and parallel views in canonical form. In case a view is not in canonical form, which is atypical in common applications, then there is a need for a more general factoring algorithm with the same goal; see Abi-Ezzi[6]. In the worst case, we may have no choice but to perform the lighting process in WC.

The techniques we present work for both polynomial and rational surface primitives, since a Euclidean transformation is rigid regardless of the type of primitive.

3.1. The Case of a Perspective View

We can factor a perspective matrix V in canonical form into three factors V = GQE, having the following forms:

$$V = \begin{bmatrix} s_x & 0 & 0 & t_x' \\ 0 & s_y & 0 & t_y' \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & q_z & 1 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & t_x \\ a_{10} & a_{11} & a_{12} & t_y \\ a_{20} & a_{21} & a_{22} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (1)$$

where E is Euclidean and preserves both angles and distances. From these forms, it is clear that Q is a special shear in the W direction, and G is a scale and translate. It is convenient to think of the *perspective divide* (standard projection) as occurring immediately after Q. Figure 6 details the geometric steps associated with these factors; notice how LC is related to WC through E. (Figure 6 shows both the shear and perspective divide as one step, Figure 9 shows these steps separately.)

The beauty of this form is that now it is possible to avoid the costly application of V per sample point, without the sacrifice of distorted reflectance parameters. E, which is the majority (75%) of the computational cost of V, is combined with the composite modeling transformation.' which is only applied to the control points of primitives. Reflectance computations are performed in LC, then the rest of the operations associated with Q and G along with the division by $w^{\dagger}$ are applied to the numerous sample points Due to the nature of G, it may be combined with the workstation mapping step without added expense. In practice, the workstation mapping is usually applied separately.

*PHIGS separates the viewing transformation into three steps: 1) view orientation, 2) view mapping, and 3) workstation mapping. The view mapping, which maps Viewing Coordinates (VC) to Normalized Projection Coordinates (NPC) is usually a full 4 x 4. Therefore, in the case of the PHIGS pipeline performing lighting in VC is not more efficient than doing it in WC.

‡The composite modeling transformation may itself be projective, in which case the geometry of the patch will potentially get distorted while being mapped to WC. This situation is of no concern to us since conceptually lighting takes place in WC, i.e. after the distortions; in other words the distortions are considered to be intrinsic to the geometry to be lit.

'There is no escape from the division by w per sample point.

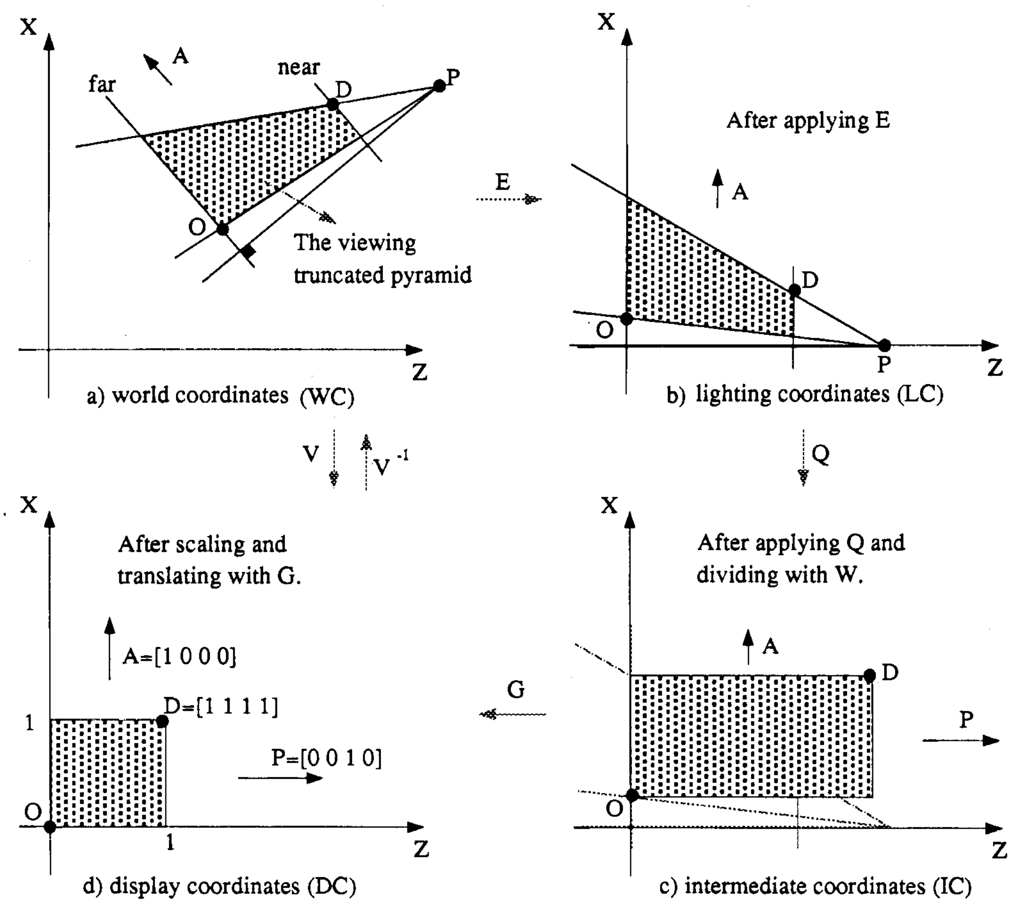

X
A
far
near
D
P
O
The viewing truncated pyramid
Z
a) world coordinates (WC)
E
After applying E
b) lighting coordinates (LC)
V
V -1
Q
After scaling and translating with G.
A=[1 0 0 0]
D=[1 1 1 1]
P=[0 0 1 0]
1
d) display coordinates (DC)
G
After applying Q and dividing with W.
c) intermediate coordinates (IC)

Figure 6. The factors of a perspective viewing transform, a) world coordinates, b) lighting coordinates where reflectance computations take place, c) intermediate coordinates, and d) display coordinates. The Y direction is not shown, but parameters in that direction behave exactly as do those in X.

The approach for factoring V is based on finding the sequence of steps that undo what V^{-1} does. The fundamental theorem of projective geometry states that a *3D* projective transformation, such as V^{-1}, is fully determined by knowing how it maps five points, no four of which are coplanar; see Penna et al7. For analyzing V^{-1}, it is convenient to inspect its affect on the following specific five points (the unit cube): $A^{dc} = [1,0,0,0]^T$, $B^{dc} = [0,1,0,0]^T$, $P^{dc} = [0,0,1,0]^T$, $O^{dc} = [0,0,0,1]^T$, and $D^{dc} = [1,1,1,1]^T$. These points are labeled as such to indicate the coordinate system to which we are referring, four different coordinate systems are depicted in Figure 6. Notice that some of these points are ideal, at infinity, and are represented by solid arrows in Figure 6. V^{-1} maps the shaded area in (d) to that in (a). Therefore, the images of points A^{dc}, B^{dc}, P^{dc}, and O^{dc} under V^{-1}, are the four column vectors of V^{-1}, and that of D^{dc} is the sum of these vectors. These points are A^{wc}, B^{wc}, P^{wc}, O^{wc}, and D^{wc} in Figure 7(a). It is clear that V is a perspective transformation in canonical form, if-and-only-if A^{wc}, B^{wc}, P^{wc}, O^{wc}, and D^{wc} determine a truncated pyramid with rectangular near and far faces. Next we show how to find the factors of V if it is in that form.

3.2. A Factoring Algorithm for Perspective Views

This algorithm accepts a 4 x 4 matrix, *V*, as input, then it determines whether or not V is valid (a perspective in canonical form), in which case it produces the factors of V as in Figure 6.

1. **Is V a perspective in canonical form?**
If V is singular then V is not valid, else if V_{3*} (the 4th row of V) = [0, 0, 0, 1] then it is a parallel view and should be handled by the algorithm given below, else find V^{-1} and from that determine:
    a. $A^{wc} = V^{-1}_{*0}$,
    b. $B^{wc} = V^{-1}_{*1}$,
    c. $P^{wc} = V^{-1}_{*2}$,

d. $O^{wc} = V^{-1}_{*3}$,

e. $D^{wc} = V^{-1}_{*0} + V^{-1}_{*1} + V^{-1}_{*2} + V^{-1}_{*3}$.

Notice, V being non-singular is a sufficient condition for P^{wc}, O^{wc}, and D^{wc} not to be collinear This is the case, since P^{dc}, O^{dc}, and D^{dc} are not collinear to start with. V is in canonical form if-and-only-if

a. $P^{wc}_w \neq 0$, $O^{wc}_w \neq 0$, and $D^{wc}_w \neq 0$ (i.e. P^{wc}, O^{wc}, and D^{wc} are Euclidian points).

b. $A^{wc}_w = B^{wc}_w = 0$ (A^{wc} and B^{wc} are ideal points).

c. A'' and B^{wc} are orthogonal (since they are ideal points then they correspond to directions).

In this case the viewing pyramid is uniquely defined and has an apex at P^{wc}, a near plane through D^{wc} spanned by A^{wc} and B^{wc}, and a far plane through O^{wc} also spanned by A^{wc} and B^{wc} .

2. **Finding the Euclidean factor:**
E is uniquely defined by rotating and translating the pyramid so that the far plane coincides with the XY plane, and P^{lc} lies on the Z-axis.

3. **Finding the shear in *W*:**
Q shears in the W direction so that P^{lc} lies on the $w = 0$ hyperplane. Hence, Q is such that $q_z = -1 / P^{lc}_z$.

4. **Finding the scale and translate:**
After the perspective division, P^{ic} goes to infinity along the positive Z-direction, and the truncated pyramid becomes a regularly-oriented, rectangular parallelepiped It then becomes straightforward to find G, which scales and translates the resulting box and coincides it with the unit cube.

3.3. The Case of a Parallel View

We can factor a general parallel viewing transformation V in canonical form, into three factors V = SHE having the following forms:

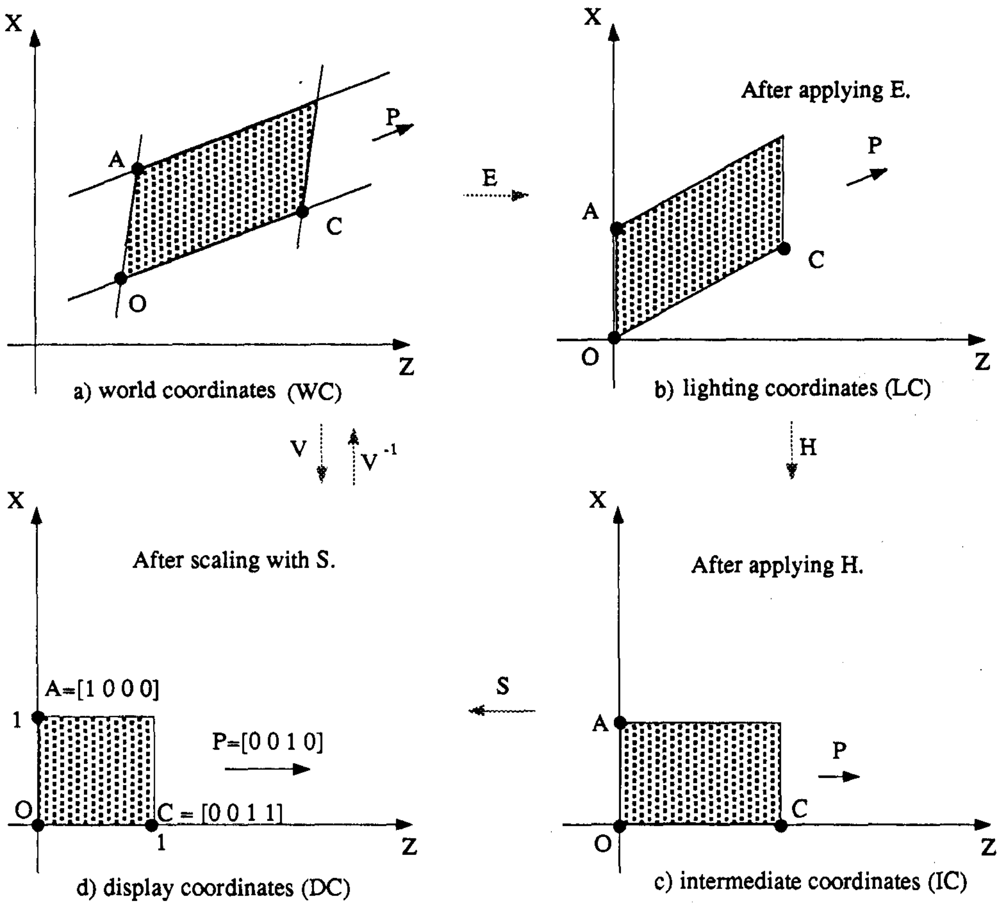


Figure 7. The factors of an oblique parallel viewing transformation, a) world coordinates, b) lighting coordinates where reflectance computations take place, c) intermediate coordinates, and d) display coordinates. The Y direction is not shown, but parameters in that direction behave exactly as do those in X.

$$V=\begin{bmatrix} s_x & 0 & 0 & 0\\ 0 & s_y & 0 & 0\\ 0 & 0 & s_z & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix} 1 & 0 & h_x & 0\\ 0 & 1 & h_y & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix} a_{00} & a_{01} & a_{02} & t_x\\ a_{10} & a_{11} & a_{12} & t_y\\ a_{20} & a_{21} & a_{22} & t_z\\ 0 & 0 & 0 & 1\end{bmatrix}, \quad (2)$$

where E is Euclidean and preserves both angles and distances, while H is a shear in both the X and Y directions, and S is a general scale. Figure 7 details the geometric steps associated with these factors. Again we see that the majority of the cost of V can be combined with the composite modeling transformation, which is only applied to the few control points. The non-rigid components of V, which are a shear and a non-uniform scale, are isolated and are applied to the sample points at a cost of, at most, 7 FLOPs per point. Notice that if V is an orthonormal parallel view, which is a common possibility, then there is no need for the shear.

Similar to the case of a perspective, the approach for factoring V is based on finding a sequence of steps that undo what V^{-1} does. The fundamental theorem of affine geometry states that a 3D affine transformation, such as V^{-1}, is fully determined by knowing how it maps four non-coplanar points; see DeRose8. For analyzing V^{-1}, it is convenient to inspect its impact on the following specific four points: $A^{dc} = [1, 0, 0, 1]^T$, $B^{dc} = [0, 1, 0, 1]^T$, $C^{dc} = [0, 0, 1, 1]^T$, and $O^{dc} = [0, 0, 0, 1]^T$; see Figure 7. V^{-1} maps the shaded area in 7(d) to that in 7(a). Therefore, the images of points A^{dc}, B^{dc}, C^{dc}, and O^{dc} can be extracted readily from the column vectors of V^{-1}. These points are A^{wc}, B^{wc}, C^{wc}, and O'' in Figure 7(a), which are sufficient to determine the nature of V, and to find the above factors of V if it is in canonical form.

3.4. A Factoring Algorithm for Parallel Views

This algorithm accepts a 4 x 4 matrix V as input, and determines if V is valid in which case the algorithm produces its factors as in Figure 7.

1. **Is *V* in canonical form?**
   If V is singular, then Vis not valid, else if $V_{3*} \neq [0, 0, 0, 1]$ then it is perspective and should be handled as above, else find V^{-1} and from that determine:
   a. $A^{wc} = V^{-1}_{*0} + V^{-1}_{*3}$;
   b. $B^{wc} = V^{-1}_{*1} + V^{-1}_{*3}$,
   c. $C^{wc} = V^{-1}_{*2} + V_{*3}$,
   d. $O^{wc} = V^{-1}_{*3}$.

   The fact that V is non-singular is a sufficient condition for these points not to be coplanar. V is in canonical form if-and-only-if $(\overline{O^{wc}A^{wc}})$ is orthogonal to $(\overline{O^{wc}B^{wc}})$.
2. **Is *V* orthonormal?**
   V is orthonormal if-and-only-if $(\overline{O^{wc}C^{wc}})$ is orthogonal to both $(\overline{O^{wc}A^{wc}})$ and $(\overline{O^{wc}B^{wc}})$.
3. **Find the Euclidean factor:**
   E is uniquely defined by translating O^{lc} to the origin, and then rotating and reflecting so that $A'' \in [OX)$, $B^{lc} \in [OY)$ and C^{lc} lies on the positive side of the XY plane. This is always possible due to the orthogonality of $(\overline{O^{wc}A^{wc}})$ to $(\overline{O^{wc}B^{wc}})$.
4. **Find the shear:**
   H is uniquely defined by shearing in both the X and Y directions so that $C^{lc} \in [OZ)$. Notice that if V is an orthonormal view, then there is no need for this shear. If V is oblique then the two shear parameters, h_x and h_y, can be determined as follows:
   a. $h_x = -C^{lc}_x / C^{lc}_z$.
   b. $h_y = -C^{lc}_y / C^{lc}_z$.

   $C^{lc}_z \neq 0$ since the four points A^{lc}, B^{lc}, C^{lc}, and O^{lc} are not coplanar.
5. **Find the scale:**
   The scaling parameters can be easily determined as follows:
   a. $s_x = 1 / A^{lc}_x$.
   b. $s_y = 1 / B^{lc}_y$.
   c. $s_z = 1 / C^{lc}_z$.

4. The Wrap around Infinity Phenomenon

The wrap around infinity phenomenon occurs when a primitive in projective 3D space intersects the ideal plane; i.e., it includes at least one point with $w = 0$. For example in Figure 8, the projective image through point O of line segment P_1P_2 from λ_1 unto λ_2 wraps around infinity. The image of point P_e is the ideal point with direction λ_2.

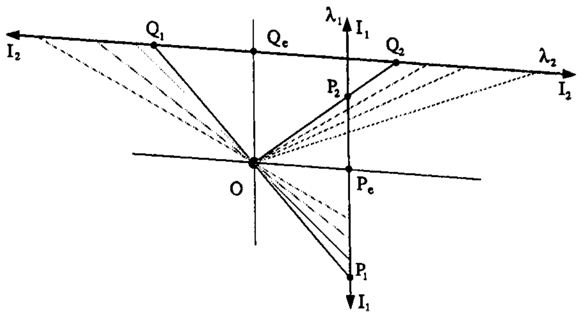


Figure 8. The wrap around infinity phenomenon as a consequence of a projection.

4.1. The Phenomenon in the Graphics Pipeline

Wrap around infinity occurs in the graphics pipeline due to the use of projective transformations. In the factored form of a perspective transformation, discussed in the previous section, this phenomenon has the chance of occurring during the Q step. This step consists of the two substeps of

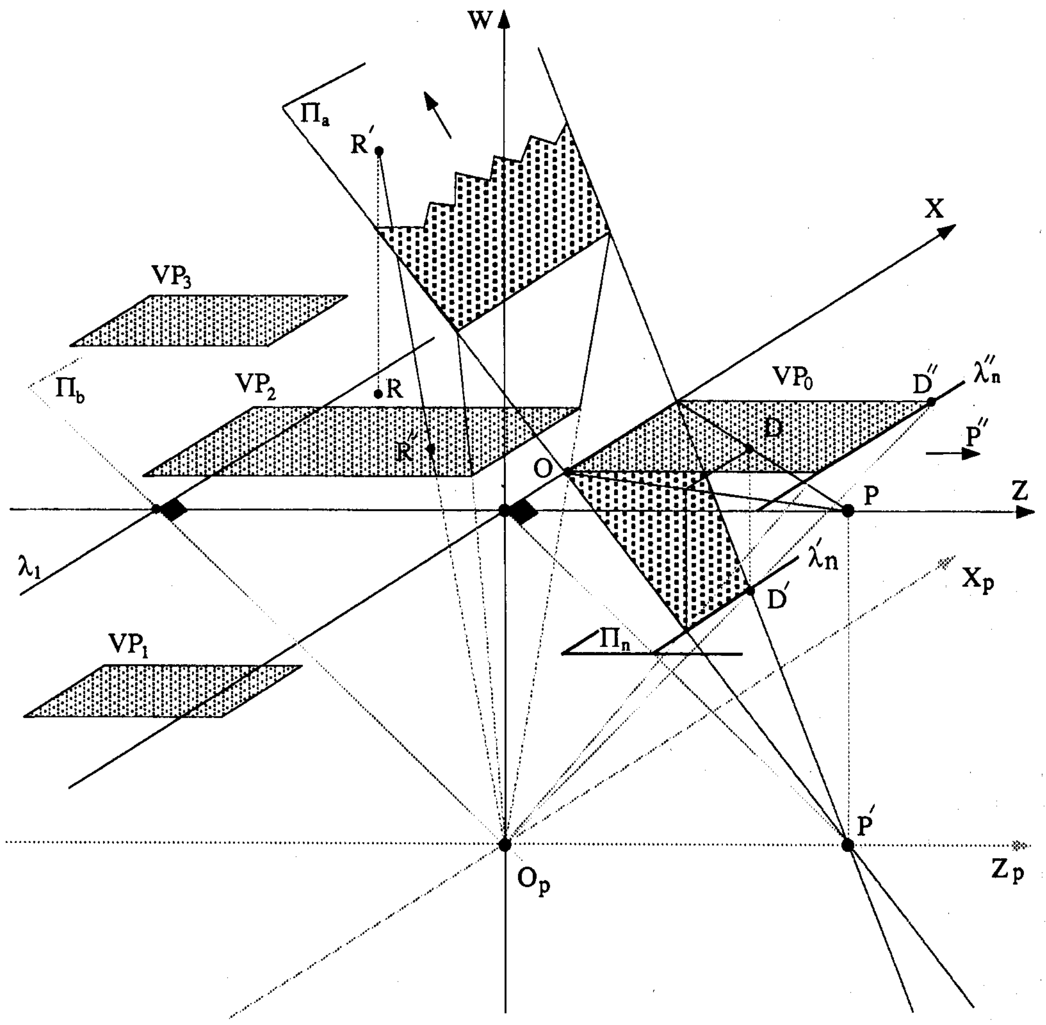


Figure 9. *The two substeps of Q: the primed points result after the shear in W of* Π_l *to* Π_a, *while the double primed points result after the projection of* Π_a *back to* Π_l *; for example points R, R' and R''.*

shearing, which maps LC to IC_{4D}*, and the projection which maps to IC. The detail of these substeps are shown in Figure 9. Define Π_0 and Π_1 to be the $w = 0$ and $w = 1$ hyperplanes, respectively. The shear substep maps Π_1 to Π_a, which is then projected back to Π_1 through O_p.

The wrap around infinity takes place when a primitive intersects Π_0 after the shear substep. This condition is equivalent to the case where in LC, before the shear, the primitive intersects a plane through the eye point (P^{lc}) and parallel to the near plane (both A^{lc} and B^{lc}). Assuming such a primitive is a bounded line segment [AB], then its image after the shear is [A'B'], and after the perspective projection is the two semi-infinite lines $(I, A''] \cup [B'', I)$ (where I is the ideal point of line $(\overline{A''B''})$; analogously to Figure 8. In order to accurately find the image of [AB], it is not sufficient to transform A and B into A'' and B'', respectively, and connect the transformed points since this produces a wrong result. The problem then is how to project a primitive that penetrates Π_0?

4.2. Existing Solutions for the Problem

This is a well-understood problem in computer graphics and traditionally the solution is referred to as "clipping in homogeneous coordinates" described in the well-known paper by Blinn and Newel[9]. Recently, a more efficient solution to the problem, referred to as the *W–clip*, was proposed by Herman and Reviczky[10].

We discovered that our factored form of a perspective transformation provides for a more efficient adaptation of

*IC_{4D} is the 4D coordinate system which becomes the 3D IC system after the projection.

the W-clip technique. Next we discuss the original W-clip approach, followed by the adapted form.

The original W-clip approach involves clipping a thin slice of 4D space surrounding Π_0 before the projection, thereby trimming the portions of primitives that would project to infinite points. The clip is relative to the two hyper-planes $\Pi_{\varepsilon+} \equiv w = +\varepsilon$ and $\Pi_{\varepsilon-} \equiv w = -\varepsilon$, where & is a very small positive number whose exact value depends on the floating-point precision that is used. These portions of primitives would be clipped by the bounded viewport in DC anyway. Therefore, the W-clip simply extracts the portions of primitives that would cause problems with the *w* division, without any loss in the final image on the screen. After the *w* division, a regular 3D clip takes place with respect to the boundaries of the viewport. Although the W-clip technique works very well, it introduces an extra clipping stage, namely the W-clip, in addition to the viewport clip. Our adaptation basically avoids this extra stage.

4.3. The Adapted W-clip Solution

In the case of the factored form of a perspective transformation, see Equation 1, we can implicitly obtain the effect of the W-clip operation by applying the far and near viewing clips before the projection; i.e. in IC_{4D}. This is the case since the images of the near and far clipping planes in IC_{4D} are parallel to Π_0. In order to clarify this we need the following definitions; see Figure 9:

1. $\Pi_a = Q\,\Pi_1$ (due to the nature of Q, $\Pi_a \cap \Pi_1$ is the *XY* plane)
2. Π_b: $(O_p \in \Pi_b) \wedge (\Pi_b \parallel \Pi_a)$
3. $\lambda_1 = \Pi_1 \cap \Pi_b$
4. λ_n''/λ_f'' the near/far plane of the viewport in IC
5. λ_n'/λ_f' the near/far plane of the viewport in IC_{4D}
6. Π_n: $(\lambda_n' \subset \Pi_n) \wedge (\Pi_n \parallel \Pi_0)$
   (below we show that $\lambda_n' \parallel \Pi_0$)
7. Π_f: $(\lambda_f' \subset \Pi_f) \wedge (\Pi_f \parallel \Pi_0)$
   (below we show that $\lambda_f' \parallel \Pi_0$)

Notice that the λs are 2D planes and the Πs are 3D hyper-planes, they're shown reduced one dimension in Figure 9. Also, λ_1 is a plane in IC that is parallel to the *XY* plane, and λ_1 is the projection of the ideal plane of Π_a. Conversely, the ideal plane of Π_a is the inverse-projection of λ_1. The inverse-projection maps points from Π_1 onto Π_a via projectors through O,; i.e., it maps the double-primed points in Figure 9 onto the single-primed points.

λ_n' and λ_f' are both parallel to Π_0 for two reasons:

1. λ_n'' and λ_f'' are both parallel to the *XY* plane since DC is related to IC via G^{-1} ;
2. due to the orientation of Π_a (special form of Q), the inverse-projection of planes that are parallel to the *XY* plane in IC, are planes in IC_{4D} that are parallel to Π_0.

Since we know that the viewport is bounded in IC then neither of λ_n' nor λ_f' can lie on Π_0, they are strictly parallel to it.

We have three cases to consider that are characterized by the position of the viewport relative to λ_1 in IC. Figure 9 shows representatives of these three cases; namely, VP_1, VP_2, and VP_3. VP_0 belongs to the same class as VP_1. The first case is characterized by having the viewport to the positive Z-side of λ_1, such as VP_0 and VP_1. In that case we clip relative to Π_n the 4D halfspace that contains Π_0, which also contains the negative *W* halfspace. This clearly subsumes both the near-viewing clip as well as the W-clip. The second case is characterized by having the viewport to the negative Z-side of λ_1, such as VP_3. In this case Π_f is between Π_0 and Π_n, so we clip relative to Π_f the 4D halfspace that contains Π_0. Therefore, the positive *W* halfspace is clipped. In both of these cases, we may perform the rest of the viewport clips in 3D after the projection, however, in practice it is more efficient to perform both the near (Π_n) and far (Π_f) clips in IC_{4D} since they're parallel to each other.

The third case is when the viewport intersects λ_1, such as VP_2, then we must clip the subspace between Π_n and Π_f in IC_{4D}, which contains Π_0. This clip clearly subsumes the near clip, far clip, and W-clip. Similarly to the previous two cases, we perform the rest of the viewport clips after the projection to 3D.

This case of having the viewport penetrating λ_1, corresponds to the situation where the viewing window in WC is an unbounded double pyramid. This is the case since the points on λ_1 are images of infinite points (or ideal points) in WC. Viewing with a double-pyramid window models the situation where a viewer has both front and back eyes, and where the back eyes invert and reverse the order (relative to the eye point) of the objects in the back and deposit them behind the objects in the front. This type of viewing clearly doesn't exist in the physical world we know; however, it does exist in the world of mathematics, and should be included for completeness. An advantage of computer graphics is that it is not limited to representations from the physical world as we know it.

Therefore the advantage of the adapted W-clip technique is to avoid an extra clipping stage by accomplishing the W-clip implicitly with the near and far clips. This adaptation is possible only due to the factored form of the viewing transformation, which results with λ_n' and λ_f' that are parallel to Π_0.

Herman and Reviczky suggested an optimization technique for the original W-clip, which determines if both $\Pi_{\varepsilon+}$ and $\Pi_{\varepsilon-}$ clips are needed, or if one of them is sufficient. The technique involves the use of inverse-projection to map the eight vertices of a viewport onto Π_a in IC_{4D}, and then to check if the convex hull of these points intersect Π_0. In case it does not, then only one of the two clips is sufficient; see Herman et al10. A more economical and equivalent alternative to the above test is to check the relation of the viewport and λ_1 in IC; unless the viewport intersects λ_1 only one clip is sufficient. This technique has three drawbacks: 1) it is expensive to carry out the required test; 2) even if the test is positive, an extra clip is still needed in addition to the original clips; and 3) the test needs to be repeated every time the composite modeling transformation changes, which tends to happen during traversal. In the case of the adapted W-clip technique the whole process is independent of the modeling transformation, and hence we can perform the pre-analyses for clipping at creation time just like we can factor the viewing transformation at that time. This pre-analysis includes finding λ_1 and its relation to the viewport.

5. Conclusion

We presented a method to perform accurate lighting of surface primitives without sacrificing efficiency. The method is based on factoring the viewing transformation in order to identify an ideal coordinate system for the lighting process. The lighting coordinate system is related via a Euclidean transformation to world coordinates, and it is computationally close to device coordinates.

The lighting coordinate system also has special features for clipping in homogeneous coordinates. We take advantage of these features and provide an efficient solution, in which we apply the near and far clipping before the projection to 3D and the rest of the clipping in 3D.

Both methods have a direct application in the PHIGS PLUS pipeline. Furthermore, due to the nature of the PHIGS model of retaining graphical data, it is possible to perform the setup processing needed by both methods at creation time. It is desirable to perform the factoring of the viewing transformation and the preliminary analysis for clipping at creation time, and to take advantage of the results during the potentially numerous subsequent traversals.

Acknowledgements

We would like to thank Steven Kader and Jan Bohn for implementing SurfaceTool, and Bob O'Bara for reviewing and commenting on this paper.

References

1. International Organisation for Standardisation (ISO), "Graphical kernel System for Three Dimensions (GKS-3D), Functional Description," ISO/DIS 8805, ISO Central Secretariat (1989).
2. International Organisation for Standardisation (ISO), "Programmer's Hierarchical Interactive Graphics System (PHIGS), Functional Description," ISO/IEC 9592-1, ISO Central Secretariat (1988).
3. International Organisation for Standardisation (ISO), "Programmer's Hierarchical Interactive Graphics System - Plus Lumiere Und Surfaces (PHIGS PLUS), Functional Description," ISO/IEC 9592-4: 199x, Draft Proposal, ISO Central Secretariat (March 1990).
4. Alyn Rockwood, Kurt Heaton, and Tom Davis, "Real-Time Rendering of Trimmed Surfaces," Computer Graphics 23(3) (August 1989).
5. Michael Shantz and Sheue-Ling Lien, "Shading Bicubic Patches," Computer Graphics 21(4), pp. 189-196 (July 1987).
6. Salim Abi-Ezzi, "The Graphical Processing of B-splines in a Highly Dynamic Environment," RPI Ph.D. dissertation, RDRC-TR-89001, Troy, New York (May 1989).
7. Michael Penna and Richard Patterson, Projective Geometry and its Applications to Computer Graphics, Prentice Hall (1986).
8. Tony DeRose, ''Geometric Programming: A Coordinate-Free Approach," SIGGRAPH '88 tutorial notes #25 (August 1988).
9. James Blinn and Martin Newell, "Clipping in Homogeneous Coordinates," Proc. SIGGRAPH '78, pp. 245-251 (1978).
10. Ivan Herman and Janos Reviczky, "A Mean to Improve the GKS-3D/PHIGS Output Pipeline Implementation," Proc. EUROGRAPHICS '87 (September 1987).