NOTES ON CSG IN A PIXEL-POWERS SYSTEM

By Jack Goldfeather

September 26, 1985

INTRODUCTION

In these notes we set out the mathematical foundation for drawing a CSG object in a Pixel-powers system. For the purposes of these notes the Pixel-powers system can be thought of as an intelligent frame buffer that has at each pixel:

a) a Quadratic Expression Evaluator (QEE) that receives the coefficients A,B,C,D,E,F as input and evaluates the expression $Ax^2 + Bxy + Cy^2 + Dx + Ey + F$. The speed of the Pixel-powers system is due in large part to the fact that this calculation is done simultaneously at each pixel when the coefficients are broadcast to the system.

b) a one-bit ALU

c) 128 bits of memory

Because of the capabilities mentioned above, the Pixel-powers system is naturally suited to performing CSG calculations and display. For CSG objects containing ??? primitives we believe that smooth-shaded display can be performed in real time!!

The first section lays out the theoretical groundwork for manipulating CSG trees. The second section analyzes the problem of visibility of CSG objects. The third section adapts the display algorithm developed in section 2 to the Pixel-powers system.

this basically is the definition of a canonical form for CSG objects and proof that any CSG tree can be put in this form

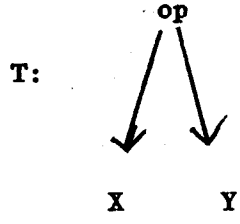
SECTION 1. CSG-TREES AND CSG EXPRESSIONS

In this section we show that CSG objects can be represented by CSG trees that have a "normal" form that is especially suitable for display purposes. Specifically, when a tree is in normal form, the object that the tree represents can be drawn by sending only primitives to the display system.

Def: A CSG-tree,T, is a complete binary tree whose non-leaf nodes are chosen from the set OP = {∪,∩ ,-} and whose leaf nodes are chosen from a set of solid primitives, PRIM = {X,Y,Z,...}.

Def: The expression for a CSG-tree T, denoted exp(T), is defined inductively as follows:

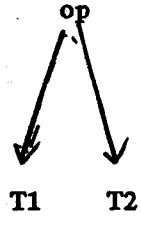
if T is of height 1 i.e



where op is in OP

then exp(T) = X op Y .

If T is of height n, n>1, then T can be decomposed as



where T1, T2 are CSG-trees of height n-1.

Then exp(T) = exp(T1) op exp(T2).

Def: The solid represented by T is the set in 3-space found by evaluating exp(T).

Convention: We will use the convention that if no parentheses appear in an expression, then association is from left to right. For example,

A op B op C op D means ((A op B) op C) op D.

Def: Using the convention mentioned above, we will say that a tree T is simple if exp(T) = X_1 op X_2 op ... op X_k where each "op" is either "−" or "∩", and each X_i belongs to PRIM.

Def: A CSG-tree T is normal if exp(T) is the union of simple trees. Similiarly we say that exp(T) is normal if it is the union of simple expressions. As we will see the normal form for a CSG-tree is especially suitable for display purposes. Briefly, the idea here is that a simple expression can be sent to the display system one primitive at a time without having the host do any computations to determine how each succeeding primitive will effect previous ones. In addition, each simple expression is easily compared with other simple expressions using the z-buffer algorithm.

Def: Two CSG trees are equivalent if they represent the same solid.

Theorem A: Every CSG tree is equivalent to a normal CSG tree.

The proof of this theorem will follow several lemmas.

Lemma 1. The following identities hold:

a) $X \cap (Y - Z) = X \cap Y - Z$

b) $X \cap (Y \cup Z) = (X \cap Y) \cup (X \cap Z)$

c) $X \cap (Y \cap Z) = X \cap Y \cap Z$

d) $X - (Y - Z) = (X - Y) \cup (X \cap Z)$

e) $X - (Y \cup Z) = X - Y - Z$

f) $X - (Y \cap Z) = (X - Y) \cup (X - Z)$

Proof: Straightforward set inclusion arguments.

Lemma 2. Suppose E_1 and E_2 are simple expressions. Then

a) $E_1 - E_2$ can be written in normal form.

b) $E_1 \cap E_2$ can be written in normal form.

Proof: a) Suppose E and F are simple expressions. We will show that E-F is normal using induction on the number of primitives in F. If F has only one term then E-F is simple and hence normal. Suppose the lemma holds for all simple expressions F which have n-1 terms. Suppose F has n terms. Then we can write F as F= G op X, where G is a simple expression with n-1 terms, X is a primitive and op is either "$\cap$" or "-". Using Lemma 1, we rewrite E-F:

E-F = E - (G op X) = (E - G)$\cup$(E op' X), where op'= "$\cap$" if op = "-" and op' = "-" if op = "$\cap$".

E-G is normal by the inductive hypothesis and E op' X is simple so E-F can be written in normal form.B

b) Use argument similiar to the one in a).

Lemma 3. a) The difference of normal expressions can be written in normal form.

b) The intersection of normal expressions can be written in normal form.

Proof: Use Lemma 2 and induction on the number of simple terms in the second expression.

Proof of Theorem A: Induction on the height of the tree T.

Every tree of height 1 is normal. Suppose the theorem holds for all trees of height n-1. Let T be a tree of height n. Then T can be decomposed as T= T_1 op T_2, where T_1 and T_2 are trees of height n-1. If op = "$\cup$" then T is

normal. If op = "$\cap$" or "$-$" then Lemma 3 and the inductive hypothesis imply T can be written in normal form.

An Example: Consider the non-simple expression

$$(X - Y) - ((U \cap V) - (S - T)).$$

Using Lemma 1, the expression can be rewritten as follows:

$$(X - Y) - ((U \cap V) - (S - T))$$
$$= ((X - Y) - (U \cap V)) \cup ((X - Y) \cap (S - T)) \quad \text{Lemma 1 d)}$$
$$= (X - Y - U) \cup (X - Y - V) \cup (X - Y \cap S - T) \quad \text{Lemma 1 f) and a)}$$

SECTION 2. VISIBILITY AND SIMPLE EXPRESSIONS

Let dX denote the boundary of a solid X. In order to display X it suffices to display dX. However, time will be wasted computing those parts of dX which are never visible. On the other hand, there is a natural algebraic interaction between the boundary operator d and the set operations so that the display of complicated CSG objects can be broken down into the successive displays of simpler ones. It might appear to be a significant saving to compute only the visible portion, $d_V X$, of dX. However, d_V does not interact well with set operations. For example, it is not possible to express $d_V(X-Y)$ in terms of $d_V X$ and $d_V Y$.
In this section we develop the notion of "local visibility" and show that there is still a natural algebraic interaction between the locally visible boundary operator d_F and the set operations. In addition, we will show in the next section that the local visibility calculation is much cheaper than the calculation of all of dX.

Def: A scene Q is a collection of solids in 3-space. A point x in 3-space is visible from a viewpoint P in the scene Q if no point of Q is on the line of sight between x and P (The line of sight could be orthographic or perspective).

Def: A point x in 3-space is locally visible from P in the scene Q if there is a ball B(x) centered at x, such that x is visible from P in the scene Q ∩ B(x).

Def: $d_F X$ is the subset of dX which is locally visible.

$d_B X = dX - d_F X$.

The interaction between the boundary operator, d, and the CSG operations is given by the following theorem.

Theorem:

a) $d(X \cup Y) = (dX - Y) \cup (dY - X)$

b) $d(X \cap Y) = (dX \cap Y) \cup (dY \cap X)$

c) $d(X - Y) = (dX - Y) \cup (dY \cap X)$

As mentioned above, we want to preserve something like this interaction when we replace d by d_F or d_B. To do this we first introduce the Display equivalence relation:

Def: Two sets of objects in 3-space are equivalent mod Display ($\equiv$) if they produce the same 2-D projection.

Example: If Y is totally obscured by X then $X \cup Y \equiv X$ mod Display.

The following theorem summarizes the boundary interaction for d, d_F and d_B mod Display.

Theorem: Suppose X and Y are CSG objects. Then

a) $d(X \cup Y) \equiv d_F X \cup d_F Y$ mod Display

b) $d(X \cap Y) \equiv (d_F X \cap Y) \cup (d_F Y \cap X)$ mod Display

c) $d(X - Y) \equiv (d_F X - Y) \cup (d_B Y \cap X)$ mod Display

We can use this theorem together with the fundamental theorem of the last section to describe an algorithm for CSG display. The idea is to compute only the boundaries (d_F and d_B) for the primitive objects and not any complicated objects formed from them. As we will see in the next theorem, this is possible when the tree representing the CSG object is in normal form.

THE DISPLAY THEOREM: Let $E = E_0 \ op_1 \ E_1 \ op_2 \cdots \ op_k \ E_k$ be a simple expression. Suppose

$$P_i = \begin{cases} F & \text{if } op_i = \text{"}\cap\text{" or } i=0 \\ B & \text{if } op_i = \text{"}-\text{"} \end{cases}$$

Then

$dE =$

$$\bigcup_{i=0}^{k} \left(d_{P_i} E_i \ op_1 \ E_1 \ \cdots op_{i-1} \ E_{i-1} \ op_{i+1} \ E_{i+1} \cdots op_k \ E_k \right)$$

SECTION 3 PIXEL-POWERS AND THE DISPLAY ALGORITHM

In this section we adapt the previous theorem to the Pixel-powers system. The QEE enables the system to deal with any primitive solid whose boundary can be divided into patches each of which is a quadratic or linear surface. Such primitives include, boxes, spheres, ellipsoids, cylinders, cones, hyperboloids, paraboloids, etc.

The general idea is to build up the image one primitive at a time. For each primitive X, dX is generated in the frame buffer and is "pared down" by passing by it the other primitives in the same simple expression as X.

In order to implement this we use the ENABLE register and allocate part of each pixel memory as follows:

a) a 30 bit ZMIN buffer

b) a 30 bit ZTEMP buffer

c) two 24 bit color buffers COL1 and COL2

The Algorithm

```
displaycsg(T)
{
   determine normal form for exp(T);
   for each simple expression E in exp(T) {
    for each primitive X_i in E {
     for each patch on d_p_i X {
/* Scan Conversion */    { ENABLE pixels inside 2-D projection of patch;
/* Depth Computation */    compute z values and store in ZTEMP;
                          for each additional primitive X_j in E

                            { case(op_j):
                                op_j = "∩" : disable pixels outside X_j;
/* CSG operation */             op_j = "-" : disable pixels inside X_j;

                            }

                          disable pixels for which ZMIN<ZTEMP;
                          replace ZMIN by ZTEMP for ENABLED pixels;
/* Shading */             compute color for ENABLED pixels;
                      }
                  }
```

}

}

We will now consider in some detail each of the four major phases in the algorithm.

Scan Conversion

The scan conversion of quadratic primitives is much like the scan conversion of polygons in the pixel-planes system. The one major difference is that we do not divide the surface of an object into patches before the geometric transformation phase. Rather, we divide the primitive into two patches after eye transformation: the front facing (i.e. locally visible) part and the back-facing part. The figure shows what the boundary of each patch would look like for various solid primitives. The important observation is that in each case, the boundary curves are quadratic so that we can use the QEE to determine on which side of a boundary curve a pixel lies. More precisely, in order to ENABLE the appropriate pixels inside a patch, the equation of each boundary curve is written in the form $f(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$, where $f>0$ for points on the same side of the curve as the patch. The six coefficients A,B,C,D,E, and F are broadcast and those pixels for which $f(x,y)<0$ are disabled.

Depth Computation

Each of the surface patches satifies an equation of the form

$$Ax^2+By^2+Cz^2+Dxy+Exz+Fyz+Gx+Hy+Iz+J=0$$

There are two cases to consider:

Case 1: No z^2 term. Then $z = Q(x,y)$ where $Q(x,y)$ is quadratic in x and y. Then the QEE can be used directly to compute the z value by broadcasting the 6 coefficients.

Case 2: non-zero z^2 term. Then $z = L + \sqrt{Q}$, where L is linear and Q is quadratic in x and y. Then the square root function can be approximated using a piecewise linear function in the following way:

Let M^2 be the maximum value of Q in the scan converted region. Subdivide the interval [0,M] into n pieces. Each subinterval produces a corresponding subregion of the scan converted object. These subregions are scan converted just as they would be for the entire object. There are several choices for this subdivision including (a) a uniform subdivision (b) a subdivision that produces uniform error. In the ith subinterval replace the square root function by a linear function, so $\sqrt{Q}$ is replaced by $(a_i Q + b_i)$. Hence z is approximated by a quadratic function. Preliminary results show that n does not have to be very large in order to get good approximations. A value of n between 3 and 8 seems fine in most cases.

CSG Operation

Deciding whether a point (x,y,z) is inside or outside of a primitive Y ia much like the shadow phase in pixel-planes. Each patch of Y is solved for z and using the approximation mentioned above, z=f(x,y) is a quadratic function that can be broadcast and compared with the stored ZTEMP value. Pixels are turned off if they are on the wrong side of the patch. Much time can be saved if for each dX, only those primitives Y that have a non-trivial interaction with dX are sent. A simple bounding box elimination will cut down on the length of each simple expression. For example, if we want to draw $dX - Y \cap Z$ and the bounding boxes for X and Y are disjoint, then it suffices to draw $dX \cap Z$. Similiarly, if the bounding boxes for X and Z are disjoint, then we can skip the rendering of this part altogether.

Shading

It can be shown that the exact shading formula for quadratic primitives is of the form:

Shade at (x,y) = $(L + \sqrt{Q})/D$

where L is linear, Q is quadratic and D = is the square root of a sum of squares of linear terms ~~l1,l2,l3~~ l_1, l_2, l_3 in x,y and z. We approximate $\sqrt{Q}$ with a single term aQ+b. We approximate D in 2 steps: (i) replace D by $l_1+l_2+l_3$ (ii) replace all occurrences of square roots in z in $l_1+l_2+l_3$ with a single linear approximation. While these approximations appear to be drastic, the effect is still "smooth-shaded" with appropriate highlights.