SUPERCOVER PLANE RASTERIZATION
A Rasterization Algorithm for Generating Supercover Plane Inside A Cube

T. Petković, S. Lončarić
University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, Zagreb, Croatia
tomislav.petkovic.jr@fer.hr, sven.loncaric@fer.hr

Keywords: computer graphics, supercover plane rasterization

Abstract: An analysis of a rasterization algorithm for generating supercover planes in 3D voxel space is presented. The derived algorithm is an extension to the classical 2D line rasterization algorithm. Additional voxels needed to form the supercover 3D plane are identified by rasterizing two additional 2D lines per volume slice. A discussion on how to modify the algorithm to rasterize finite supercover 3D plane segments with arbitrary parameters by using integer arithmetic only is given.

1 Introduction

One of the problems in various applications of computer graphics along with line drawing is rasterization of planary surfaces. Our ability to obtain huge volumetric data is ever increasing, especially in biomedical fields where 256$^3$ is currently the most common volume size. When analyzing such volumetric data-sets one is not always satisfied with the simple line and plane drawing and rasterization algorithms as some interesting and sometimes not desirable effects can be introduced, such as existence of tunnels in surfaces so for example 26-connected line can pass through the 18-connected plane (Cohen-Or and Kaufman, 1997).

Most popular algorithm for line drawing was presented by Bresenham (Bresenham, 1965), and first generic study on plane rasterization was done by Kim (Kim, 1984). A fast plane weaving algorithm for rasterizing 18-connected digital planes was described by Lincke at al. (Lincke and Wüthrich, 1999). In this article a simple 3D supercover plane generation algorithm is presented. The described algorithm is an extension of the weaving algorithm presented by Lincke et al.

The article is organized as follows: in section 2 we present the notation and definitions. In section 3 a review of exact weaving plane algorithm is given. Section 4 presents an idea for weaving the 3D supercover plane and explains details about the rasterization algorithm. We conclude the article in section 5.

2 Preliminaries

Let $\mathbb{R}$ be the set of real numbers and let $\mathbb{Z}$ be the set of integers. When digitizing three dimensional space a grid usually used is the cubic grid. As we are interested in finite volumes only a subset of $\mathbb{Z}^3$ is needed. So our area of interest is a subset of 3D Euclidean space $\mathbb{R}^3$ that consist of all points whose coordinates are integers and are within chosen intervals.

Square brackets denote both the rounding operator and a set of all residues module $q$ when the index is written, so $[x]$ is the rounding operator, but $[p]_q$ is the set of all residues of $p$ modulo $q$.

A voxel in $\mathbb{R}^3$ corresponding to a discrete point $(x,y,z) \in \mathbb{Z}^3$ is defined by the continuous unit cube with the center at $(x,y,z)$. Let us denote the cube associated with one voxel as $\mathbb{V}(x,y,z) = [x-\frac{1}{2},x+\frac{1}{2}] \times [y-\frac{1}{2},y+\frac{1}{2}] \times [z-\frac{1}{2},z+\frac{1}{2}]$. The voxel is sometimes called spel (spatial element), but note that a spel is not necessarily a voxel. A face of a cube associated with a voxel is just a face, but if two voxels are in relation then corresponding shared face is called a surfel (surface element). Let $X$ be any set and $p$ be a
binary relation on X. If \((p, q) \in \rho\) then \(p\) is \(\rho\)-adjacent of \(q\). If \(\rho\) is a symmetric relation then for any \(p, q \in X\) \((p, q) \in \rho\) if and only if \((q, p) \in \rho\). We say that \(p\) and \(q\) are \(\rho\)-adjacent (Herman, 1998).

For \(X = \mathbb{Z}^3\) we define three symmetric binary relations on \(\mathbb{Z}^3\) that correspond to 26, 18 and 6 voxel connectivity. For any two points \(p = (p_1, p_2, p_3)\) and \(q = (q_1, q_2, q_3)\) of \(\mathbb{Z}^3\) we say that they are 18-connected or \((p, q) \in \delta_3\) if and only if they share a face or an edge. The simplest plane rasterization in 3D is 18-connected, so adjacent voxels in a digital plane will share a face or an edge. Such plane is not tunnel-free and 26-connected line can pass through such plane. For some applications where this is not acceptable one must find a plane that is tunnel-free. A supercover plane is a likely candidate as it has some additional desirable properties and is also a tunnel-free structure. We will adopt the definitions by Andrés (Andrés, 2003):

**Definition 2.1 (Supercover)** A supercover \(S(X)\) of a continuous object \(X\) is the set of all the discrete points \(p \in \mathbb{Z}^n\) and associated voxels such that \(\forall(p) \cap X \neq \emptyset\). One of the drawbacks of the supercover objects is existence of bubbles.

**Definition 2.2 (Bubble)** A \(k\)-bubble is the supercover of an Euclidean point that has exactly \(k\) half-integer coordinates.

### 3 An Exact Weaving of Digital Plane

An attractive method to produce digital planes is by using weaving techniques (Lincke and Wüthrich, 1999). Basic idea is to decompose the rasterization of a surface into two orthogonal curve rasterizations. For the planes both curves are lines and by copying one along the other plane is obtained. The line being copied is usually called master and the line used for determining the positions of the master is called base. As all lines in the plane along the base are copies of the master and thus have the same chain code we only need to compute the rasterizations of the master and base lines. Copying the master line then completes the plane weaving.

We usually denote a line with the letter \(L\). As we are interested in lines with the same slope and different intercepts we only need to know the value of the intercept.

**Definition 3.1 (Straight line)** For any \(k \in \mathbb{Z}\) and a pair \(p, q\) of relatively prime numbers, \(q \neq 0\), the straight line \(L_k\) is a set of points \(L_k = \{(x, y) \in \mathbb{R}^2 : y = \frac{2}{q}x + \frac{k}{q}\}\).

However, when copying the master line one must notice that simple copying of the master along the base will not produce nearest neighbor rasterization as defined in (Wüthrich, 1998). To obtain a proper rasterization one must also consider shift in the line chain code that is introduced as the intercept changes. Lincke et al. (Lincke and Wüthrich, 1999) have presented an exact weaving rasterization algorithm for digital planes. Main result is the theorem stating how to compute a shift of any line at given position:

**Theorem 3.1 (Line shift)** Let \(L_e\) be a straight line given by \(y(x) = \frac{2}{q}x + e\) where \(p\) and \(q\) are relatively prime numbers, \(p, q \in \mathbb{Z}\), \(p \leq q, q \neq 0\) and \(e\) is an arbitrary real intercept. The shift \(s\) of \(L_e\) at position \(i \in \mathbb{Z}\) is given by \([s]_q = [r]_q^p\) with \(r = [pi + qe]_q\) if \(q\) is odd and \(r = \left[pi + qe + \frac{1}{2}\right]_q\) if \(q\) is even.

The weaving algorithm now copies the master line along the base, but the chain code is shifted by \(s\). Rasterization of the plane \(x + 2y - 5z = 0\) is shown in figure 1.

The plane weaving algorithm produces an exact rasterization of the plane \(Ax + By + Cz + D = 0\) with rational coefficients \(A, B, C\) and \(D\). Produced rasterization is 18-connected set that does not contain all the voxels a plane intersects. In figure 2 an 18-connected plane \(x + 2y - 5z = 0\) is superimposed over continuous plane. One can immediately notice that continuous plane is not contained within the 18-connected representation.
We will call two slices $S_i(k)$ and $S_j(l)$ adjacent iff $|k - l| = 1$.

Weaving algorithms compute 2D rasterization of a line in one slice which is then replicated for all other slices we are interested in. When we want to obtain a supercover rasterization of a plane segment we must trace two additional lines per slice along with the master line as is shown in figure 3. The master line (dot-dash line) corresponds to the line $L_m: z = -\frac{A}{B}x - \frac{C}{B}(By_0 + D)$, so the intercept is $\frac{1}{B}(By_0 + D)$ and $m = -\frac{A}{B}$. The lower line $L_l$ and the upper line $L_u$ are passing through planes with half-integer coordinates in $y$. Again, without loss of generality we can assume the slope of the plane along $y$ dimension is such that $z$ increases as we move from $y_0 - \frac{1}{2}$ to $y_0 + \frac{1}{2}$ (so $B > 0$). Upper and lower lines are now $L_u: z = -\frac{A}{B}x - \frac{1}{B}(By_0 + \frac{B}{2} + D)$ and $L_l: z = -\frac{A}{B}x - \frac{1}{B}(By_0 - \frac{B}{2} + D)$.

**Lemma 4.1 (Continuity)** For plane $P$ and two adjacent slices $S_i(k)$ and $S_j(l)$ either $L_l$ from $S_i(k)$ and $L_u$ from $S_j(l)$ or $L_u$ from $S_i(k)$ and $L_l$ from $S_j(k)$ are the same.

The result is obvious and follows immediately from $|n - m| = 1$. Together with the following theorem by Lincke et al. (Lincke and Wüthrich, 1999) it provides the basis for weaving supercover planes.

**Theorem 4.2 (Line Equivalence)** Let $L_k$ be the 2D line $y = \frac{px}{q} + \frac{e}{q}$ where $p$ and $q$ are relative primes. For all $k \in \mathbb{Z}$ the set of straight lines $\mathbb{L}_k$ having the same rasterization (up to shift) is an equivalence class and it contains all lines defined by $y = \frac{px}{q} + e$, $e \in \mathbb{R}$ with $\frac{k}{q} - \frac{1}{2q} < e \leq \frac{k}{q} + \frac{1}{2q}$ if $q$ is odd and $\frac{k-1}{q} < e \leq \frac{k}{q}$ if $q$ is even.
In our example the intercept \( e \) for upper and lower line is \( e = -\frac{1}{2}(By_0 + \frac{d}{2} + kB) \). We must show that for upper and lower lines in two adjacent slices we obtain the same line cover when shift is introduced—when we copy the master slice lower (or upper) line from slice \( S_0 \) we have the same.

**Lemma 4.3 (Equal shifts)** When copying the three lines in master slice \( S_i(0) \) to slice \( S_i(j) \) shifts \( s_m, s_u, s_l \) and \( s_i \) are at constant shift distance.

We must compute shifts for three lines \( L_m, L_u, L_l \) and \( L_i \). All three lines have the same slope, but the intercepts are different. By theorem 3.1 for odd \( C \) we have \( [s_{m}]_l = [[pj + qm]_{p^*}]_l = [[Aj + Bj + D][A^*]_c, \]
\( [s_{u}]_l = [[pj + qm]_p]_l = [[Aj + Bj + \pm \frac{d}{2} + D][A^*]_c, \) and \( [s_{l}]_l = [[pj + qm]_{p^*}]_l = [[Aj + Bj + \pm \frac{d}{2} + D][A^*]_c. \) As \( Aj + Bj + D \) is a whole number we have \( [s_{u}]_l \) and \( [s_{l}]_l \) are at constant shift distance. So two slices are shifted by \( s_i = \frac{1}{2}(Biy_0 - B + 2D - 1) \leq k_i \leq k_i + 1 \). By examining obtained inequalities for odd and even \( B \), and then for even \( C \) we can compute the difference between the intercepts. We obtain \( k_i + 1 = B + u - 1 = B. \)

**Lemma 4.4 (Switching)** When copying the upper, master and lower lines for two adjacent slices \( S_i(k) \) and \( S_i(l) \) covered voxels faces selected by upper line from one slice and lower line from another will be the same.

Let us first compute the shift distance between two master lines in two adjacent slices \( S_i(k) \) and \( S_i(l) \), \( l = k + 1 \). We have \( [s_{m}]_c = [[Ai + Bl + D][A^*]_c, \)
\( [s_{m}]_l = [[Ai + Bl + (k + 1) + D][A^*]_c = [s_{m}]_c + [BA^*]_c \), so two slices are shifted by \( [BA^*]_c \). Note that as the shift between two adjacent slices is the same we can simply shift-and-copy the chain codes from the previous slice. Now as \( [s_{l,k+1}]_c \) and \( [s_{u}]_c \) are shifted by \( \pm [[\frac{d}{2}]A^*]_c \), shifts for upper and lower line are also the same.

By combining those results we can state that intersection of voxel faces selected by the master and lower (or upper) line from slice \( S_i(k) \) and voxel faces selected by the master and upper (or lower) line from slice \( S_i(l) \) will form the supercover of the line shared by two slices. So when weaving supercover planes we could find two 2D supercover rasterizations of upper and lower lines (they must contain the master line), for example by using modified Bresenham algorithm presented in (Dedu, 2002). However, we can trace original line and only check whether upper or lower lines have non-empty intersection with upper or lower adjacent voxel as shown in figure 3.

### 4.1 Computing the cover for one slice

How can we compute the cover for one slice only? Our plane \( P \) is given by \( Ax + By + Cz + D = 0 \) with \( 0 < a \leq b \leq c \). If we start the line at coordinates \((x_0, y_0, z)\) we can compute the shifts \( [s_{m}]_c, [s_{u}]_c \), \( s_i \), and \( s_l \), and then we can copy the chain codes as done in (Lincke and Wüthrich, 1999) for naive planes. Now we have several possibilities when weaving a plane: a) we can compute the shift for each slice as done by Lincke et al. (Lincke and Wüthrich, 1999), or b) we can compute the shift difference between two adjacent slices and simply correct the shift from previous slice \( p^* \) required for the shift computation can be computed when rasterizing the master slice. Alternatively we can compute the starting values for error variables and rasterize each slice separately.

Let us compute the starting error for single slice. The real value of \( z \) coordinate is \( -\frac{4}{3}A_0 - \frac{1}{2}(By_0 + D) \), and \( [z] \) is the closest integer value. Now the error variable \( e_z \) is difference \( z - [z] \) scaled to \( 2C \), so
\[
e_z = 2C(z - [z]) = 2(-A_0 - By_0 - C[Z] + D).
\]

When computing the cover for one slice we also need the to know the error variables for upper and lower lines. When \( 0 < a \leq b < c \) the error from the slice defined by \( [z] \) for the lower line is \( e_l = e_u - a - b \), and for the upper line is \( e_u = e_z + a + b \). If either of \( e_l \) or \( e_u \) falls outside of the voxel the lower and upper lines will start at \([z] + 1 \) or \([z] - 1 \) respectively. Now we can trace those three lines simultaneously to obtain the cover for one slice.

The upper line and the lower line must be supercover lines. In the previous section we have shown that the chain codes for upper line and lower line are shifted \( \pm [[\frac{d}{2}]A^*]_c \), when compared to the master, however we must note that the computed shift is for the simple chain code. As we want to compute the supercover of the both upper and lower lines unfortunately the rounding operator must have different definitions for those lines. The rounding operator \( \lfloor x \rfloor \) is defined by \( k - \frac{1}{2} < x \leq k + \frac{1}{2}. \) The problem occurs when either of upper and lower lines passes exactly through the point with half-integer coordinates. For the upper line
as is shown in figure 3 we must select upper voxel when the line passes through back upper right voxel vertex (shown as a circle), and for the lower line we must select the lower voxel when the line passes through front lower left vertex (also shown as a circle). So the rounding operator is different for lower and upper lines.

By using similar reasoning as done in the previous section we can show that the shift for the supercover case will differ at most by one when compared to the shift $\pm \lfloor \frac{|A|}{2} \lfloor A \lfloor \rfloor$. The additional shift by one is consistent for all the slices and will not affect the cover shared between two slices. By copying and shifting the slice we can obtain the plane.

One rendering of a plane is shown in figure 4. Note that one should expect something similar here. Let us assume that $\hat{A} = [\alpha A], \hat{B} = [\alpha B]$ etc., where $\alpha$ is positive real constant. We can rewrite the (1) and (2) so $-\frac{1}{\gamma C}(\lfloor \alpha A \rfloor i + [\alpha B]j + [\alpha D]) = [-\frac{1}{\gamma}(A_i + B_j + D)]$ and consequently $\lfloor \gamma C \lfloor (\alpha A) j + [\alpha B]j + [\alpha D] \rfloor + \frac{1}{\gamma}(A_i + B_j + D) \rfloor < \frac{1}{2}$. Now we have

$$[\alpha A] < 2 \left( [\alpha A] i + [\alpha B] j - \frac{A_i + B_j + D}{C} \lfloor \alpha C \rfloor + [\alpha D] \right) + [\alpha C], \quad (3)$$

and as $0 < a, b, c$. As $\hat{C} = [\alpha C]$ is integer we can put $\alpha = \frac{1}{2}, n \in \mathbb{N}$, so $\hat{C} = n$. Now (3) transforms to

$$2 \left( \lfloor n \left\lfloor A \right\rfloor \right) - n \left\lfloor A \right\rfloor i + \left\lfloor n \left\lfloor B \right\rfloor \right) - n \left\lfloor D \right\rfloor j < |n|. \quad (4)$$

As $\lfloor n \left\lfloor \frac{a}{2} \right\rfloor \rfloor$ and other similar constructs are always between $\frac{1}{2}$ and $\frac{1}{2}$ we can find the worst case $|\hat{C}| = |n| > \max |i| + \max |j| + 1$. In fact when rasterizing arbitrary plane within finite volume we only need to check the size of the finite volume. Consequently, we can scale the coefficients so the largest one has the absolute value greater then $\max(|i_1|, |i_2|) + \max(|j_1|, |j_2|) + 1$.  

### 4.2 Restriction to a Finite Volume

Usually we are interested in computing a plane within a finite volume—usually a cube or a parallelepiped, so the plane-generating algorithm should be able to draw planes with arbitrary parameters within the subvolume.

Chain code of the digital 2D line $y = \frac{q}{p}n + e$ with $p, q \in \mathbb{N}$ and $0 < p \leq q$ is periodic. If $p$ and $q$ are relatively prime the period is $q$ (Pham, 1987). If we restrict the continuous plane $P : Ax + By + Cz + D = 0$ with arbitrary parameters (so $A, B, C, D \in \mathbb{R}$) to a cube we want to find another plane having the same rasterization, but with the coefficients being whole numbers. Without the loss of generality we can only consider the planes where $0 < a, b \leq c$ such that intersection with the parallelepiped is not an empty set. As the $C$ is the largest coefficient by absolute value when rasterizing a plane we must compute only the $z$ coordinates, $[z] = \lfloor -\frac{1}{C}(A_i + B_j + D) \rfloor$, so a digitized plane would be Dig($P$) = $\{(i, j, k) : i, j \in [i_1, i_2] \times [j_1, j_2], k = [\frac{1}{C}(A_i + B_j + C)]\}$. For digitized coordinates we require the following inequalities to have same solutions in $k$:

$$k - \frac{1}{2} \leq -\frac{1}{C}(A_i + B_j + D) \leq k + \frac{1}{2} \quad (1)$$

$$k - \frac{1}{2} \leq -\frac{1}{C}(A_i + B_j + D) \leq k + \frac{1}{2} \quad (2)$$

Here the $\hat{A}, \hat{B}, \hat{C}$ and $\hat{D}$ are whole numbers representing a plane that has same rasterization as the plane $P$. For the midpoint line drawing algorithm usually we must double all the values, so we can expect something similar here. Let us assume that $\hat{A} = [\alpha A], \hat{B} = [\alpha B]$ etc., where $\alpha$ is positive real constant. We can rewrite the (1) and (2) so $-\frac{1}{\gamma C}(\lfloor \alpha A \rfloor i + [\alpha B]j + [\alpha D]) = [-\frac{1}{\gamma}(A_i + B_j + D)]$ and consequently $\lfloor \gamma C \lfloor (\alpha A) j + [\alpha B]j + [\alpha D] \rfloor + \frac{1}{\gamma}(A_i + B_j + D) \rfloor < \frac{1}{2}$. Now we have

$|\alpha A| < 2 \left( [\alpha A] i + [\alpha B] j - \frac{A_i + B_j + D}{C} \lfloor \alpha C \rfloor + [\alpha D] \right) + [\alpha C], \quad (3)$

and as $0 < a, b, c$. As $\hat{C} = [\alpha C]$ is integer we can put $\alpha = \frac{1}{2}, n \in \mathbb{N}$, so $\hat{C} = n$. Now (3) transforms to

$$2 \left( \left\lfloor n \left\lfloor A \right\rfloor \right\rfloor - \left\lfloor A \right\rfloor i + \left\lfloor n \left\lfloor B \right\rfloor \right\rfloor - \left\lfloor D \right\rfloor j < |n|. \quad (4)$$

As $\left\lfloor n \left( \frac{a}{2} \right) \right\rfloor$ and other similar constructs are always between $\frac{1}{2}$ and $\frac{1}{2}$ we can find the worst case $|\hat{C}| = |n| > \max |i| + \max |j| + 1$. In fact when rasterizing arbitrary plane within finite volume we only need to check the size of the finite volume. Consequently, we can scale the coefficients so the largest one has the absolute value greater then $\max(|i_1|, |i_2|) + \max(|j_1|, |j_2|) + 1$.  

Figure 4: A 18-connected digital representation of the plane defined by $x + 2y - 5z = 0$ and it's supercover.
5 Conclusion

A supercover plane algorithm that uses only integer arithmetic was presented. Two variants are possible, one that simply traces a line for each slice and a weaving algorithm. Additionally it was shown that if we want to draw a finite segment of a plane we only need to scale and round the plane coefficients.

If the square plane segment of side lengths $n$ and $m$, $n < m$ has to be generated the complexity of first approach is $O(nm)$. The weaving approach needs to generate one line segment of the length $q$ and then it is copied $n$ times, so we can expect the complexity of $O(nq)$. Due to the large variety of available hardware performance analysis and code profiling was not done as it would probably be application and hardware specific, however we are currently working on this problem.

REFERENCES


