

# Teaching the Graphics Processing Pipeline: Cosmetic and Geometric Attribute Implications

**Jack Bresenham, Ph.D.**  
Winthrop University  
Rock Hill, SC 29733 USA  
bresenhamj@winthrop.edu

## Abstract

*The pipeline processing model for computer graphics typically includes stages for interpreting the display list <object entity descriptions, picture abstraction>, geometric manipulation <scaling, rotational, skewing transformations>, raster rendering <scanline conversion, dot image approximation>, and finally presentation <scanline refresh, image display>. At what point in the graphics processing pipeline do attributes actually become associated with picture entities? The answer depends upon the type of attribute and the specification intent for the attribute. I believe it is instructive for students to overtly and consciously become aware attributes such as line width and line style can have various interpretations. Cosmetic attributes are applied post-transformation; geometric attributes are applied pre-transformation. Two different types of clipping also can be considered as scissor clipping in post-rasterization discrete pixel space or as analytic clipping in pre-rasterization continuous space.*

**Keywords:** *computer graphics attributes: cosmetic & geometric*

## 1. Introduction

Is line width a purely cosmetic effect meant to mimic a finely sharpened 'thin' pencil or widely worn 'fat' pencil line thickness? In this case, line width is associated with geometric entities post-transformation. Engineering line weights are cosmetic attributes applied by a draftsman. Cosmetic attributes do not scale, skew, or otherwise distort. Asking for an architectural drawing to be redone on a paper twice as long and twice as wide treats line width as a cosmetic attribute since pens or pencils used will be the same as those used in the smaller original rendering. Pen nib selection or choice of pencil lead width determines line thickness in this cosmetic attribute instance.

Is line width a geometric property inherent in the original display list abstraction meant to mimic

photographic enlargement? In this case, line width will be associated with geometric entities pre-transformation. Having created areas for formerly zero-width lines, those areas must be processed through the full transformation stage with the effect that line width area can be scaled, skewed, or otherwise distorted. Pressing the enlargement by 2 button on a xerographic copier applies line width as a geometric attribute. Use of an overhead projector to produce large lines on a viewing screen some distance away from the projector is another example of a geometric interpretation of the attribute line width. Drawing or writing on an inflated balloon then letting much of the air escape is a 3-D example of a shrinking figure size, geometric width reduction.

Do dashes in a line style transform from rectangles to parallelograms with sharp points? Yes, they can, if one elects to apply line style as a geometric or pre-transformation attribute. Treated as a cosmetic or post-transformation attribute dashes will keep their rectangular shape. The basic question to resolve in one's **reference model** is whether a picture is first conceptually rendered, then has attributes applied, and is subsequently transformed and displayed or whether a picture is first transformed, then has attributes applied, and is subsequently displayed.

Both approaches are valid. Students should appreciate the subtleties and interactions involved in each reference model approach for attribute application. I prefer the more mnemonic nomenclature I first heard used by Dr. Brian Middleton: 'cosmetic' and 'geometric' attributes.

## 2. Motivation

Presenting simple line style implementation alternatives offers a good opportunity to reinforce transformation concepts. I use rounded, semicircular end caps that transform into elliptical caps for dashed lines as a simple example. One can readily derive the equation for the resulting ellipses ...but it can be messy to implement in

graphics hardware or software. If one approximates an original semicircle by chords along its perimeter, then, since lines always transform into lines, implementation is facilitated at reduced cost with concomitant reduced accuracy <fidelity>. The chords do introduce an early approximation; all approximation errors then subsequently propagate with the possibility that recognizable, visual degradation can be the result.

Clipping is another processing stage that can have alternative reference models. Should one first raster, at least conceptually raster, the full picture in infinite raster space then cut out the small viewing window to be actually displayed? Or, should one first clip the picture abstraction entities to an appropriately small viewing window and then raster that post-clip result. The latter is commonplace but it, too, introduces approximation error earlier than need be the case.

A fairly easy demonstration of the two clipping effects can be to first raster a full screen display of a picture. Blank out a small rectangle, perhaps for a pop up menu, then clip the abstract picture to retain just the portion for this small blank rectangle. Finally, render that newly clipped small part of the abstract picture to heal and restore the original full display picture. Discontinuities, likely, and undesirable mismatches off by a few pixels will be visible in analytic clipping.

Alternatively, consider blanking the same area for the small rectangular menu. This time, though, let's assume a fully rastered copy of the original picture has been retained in a second, nondisplayed buffer. Bit-Blit <copy> the pixels temporarily subsumed by the menu back into the displayed picture to restore the area overwritten by the small menu. Likely the result is different from the previously described restoration process; the picture will seamlessly be restored with no discontinuities in scissor clipping.

Here I like the term used by Dr. Bryan Roberts: **scissor clipping** to describe the model in which one first conceptually renders a pixel space picture then clips as is the case with the Bit-Blit approach. The alternative reference model in which one first clips the picture abstraction in continuous coordinates before rendering, then paints the clipped entity in discrete pixel space I refer to as **analytic clipping**.

The graphics pipeline can be used to demonstrate different visible results one can obtain even for the

simple attributes line width, line style, and clipping. Students can try a few examples to see for themselves how important a good understanding of one's underlying reference model is. The alternative models also offer opportunity to discuss introduction of early approximations with attendant error accumulation build-up.

### 3. Attribute Reference Model

Before implementing any graphics interface, it is essential to have a clear conceptual model in mind. Otherwise, well-meaning individuals may interpret a common specification differently owing to preconceived notions of where in the graphics processing pipeline an individual assumes attribute associations operations take place.

Clipping produces different visual results depending upon whether one performs a mathematical clip of theoretical, zero width loci of geometric entity boundaries or whether one first rasters then scissor clips the realized picture in raster space. A different picture will be drawn by those who consider a logic operation such as exclusive or to be applied after a polyline is drawn conceptually in an off-screen space then exclusive or'd to the displayed picture frame buffer. Those who consider logic operations to apply in an on-screen pixel by pixel manner dependent upon the algorithmic sequence by which the graphics system generates geometric entities during rendering will typically achieve a different visual effect.

A proper reference model should trivially let us answer questions such as: Do circular dots and rectangular dashes in a line style transform from circles to ellipses and from rectangles to parallelograms? Do wide lines join smoothly or will mitering be appropriate even though sharp spikes may result at the joining point? With a good reference model and agreed upon stages at which attribute actions are to be implemented in the graphics processing pipeline, if we don't like the answer, we should be able to change the reference model to achieve the result we really intended.

### 4. Example: Line Width & Line Style

Let's examine a few implications of two possible reference models that can be used in pipeline processing of Line Width and Line Style and interaction with general transformation and drawing mode. I'll call my alternatives:

1. Geometric <Str etched Rubber Sheet>
2. Cosmetic <Multiple Pen Nib Sizes>

Lines are specified by two end points such as  $\text{LINE}(x_0, y_0, x_1, y_1)$  for the line from  $(x_0, y_0)$  to  $(x_1, y_1)$ . Lines and any other geometric entity can be transformed by rotation, scaling, translation and shearing by use of a concatenated general 2-D transformation of the form:

$$\begin{bmatrix} u & v & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a & c & 0 \\ b & d & 0 \\ e & f & 1 \end{bmatrix}$$

The result for any affine transformation of a line is another line so, abstractly at least, we would have a line from  $(u_0, v_0)$  to  $(u_1, v_1)$  as the result of transformation. What we don't explicitly know is the effect expected from transformation of either line width or line style. The visual effect achieved by applying attributes before transformation can produce distinctly different pictures from that which would result from applying attributes only after all transformation is completed.

Line width typically is taken to be perpendicular to the defining line. Often specifications for line width fail to specify end point shape for area closure. What, then, shall we transform when dealing with a line width since, in reality, we must treat areas? What other effects interact with the process; what shapes are employed in line style?

Graphic primitives <lines, rectangles, elliptical arcs, ...> enter the pipeline with defining points in an abstract world coordinate space. The defining points imply a complete geometric locus which represents the infinitely thin, geometric abstraction of the graphic primitive form <contour or outline> in original, pre-transformation space. The essence of the pipeline processing is to collect associated attributes <such declarations as line width, line style, drawing mode, and clipping values> then to render conceptually the geometric form. The first conceptual rendering of the full locus in abstract model space is then subjected to an affine transformation <which can be the cumulative result of concatenation of multiple transformations> for composite scaling, rotation, translation, and shearing. The resulting post-transformation, conceptually abstract, picture obtained from the locus transformation potentially is clipped before then being mapped to device coordinate space.

When line width is implied in a geometric sense, graphic primitives such as a line, rectangle or ellipse inherently have an area definition. In the absence of

closure of the area owing to lack of line end shape specification, we could consider various possibilities for width processing. An implementation can define an end shape or end cap such as axially sheared, perpendicularly butted, semicircular, or triangular. For a perpendicular butt, we may want to use either a true end point reference or an extended end point reference in anticipation of joining line segments to form, say, a right angle corner.

With an anisotropic transformation, semicircular end caps can be seen ultimately on the display surface as partial ellipses capping ends of lines. As an example, try a reference line segment from  $(0,0)$  to  $(4,0)$  with a line width of two units on each side of the reference line. Let transformation matrix parameters be:  $a=2$ ,  $b=2$ ,  $c=3$ ,  $d=1$ ,  $e=0$ , and  $f=0$ . A pre-transformation semicircle centered at the origin could be the leftmost end cap joining the 'wide line' end points  $(0,2)$  and  $(0,-2)$ ; a comparable semicircle centered at  $(4,0)$  could join the rightmost end cap points from  $(4,-2)$  and  $(4,2)$ . Notice the two semicircles are drawn anticlockwise (CCW).

Applying the transformation yields boundary wide line segments from  $(4,2)$  to  $(12,14)$  and from  $(-4,-2)$  to  $(4,10)$ . Should you sketch the result of connecting the post-transformation end points from  $(4,2)$  to  $(-4,-2)$  with an origin centered semicircle having radius 2.5 and the other two endpoints from  $(4,10)$  to  $(12,14)$  with a comparable circle centered at  $(8,12)$  you can observe the effect is aesthetically lacking. With dot-dash line styles, it is important to keep the proper sense of directional movement consistent with the effect of a transformation. Notice here that the two semicircles would now properly be drawn clockwise (CW). Line style can cause a viewer to be aware of the direction in which a line is drawn. Attention must be paid to sense of movement after transformation.

A more consistent visual effect is to transform the appropriate semicircle from  $x^2 + y^2 - 4 = 0$  into the appropriate portion of the ellipse  $10u^2 - 16uv + 8v^2 - 64 = 0$  for the transformed 'leftmost' end cap to close an end of the wide line. With geometric transformation of graphic entities, a question implementers must answer is what shape is a dot or dash in a line invoking line style as an attribute.

If the *graphics system* is to provide inherent support for such attributes as width of a line or dot-dash style of a line as a geometric attribute, then reproduction of effects such as would be the case for stretching a rubber sheet or a 150% copier enlargement can guide our decision regarding

geometric attributes. The rubber sheet or magnifying glass model for functionality encourages a geometric interpretation of the attributes line width and line style.

By contrast, consider a different physical model to be the motivating reference for a computer graphics functional capability. Architectural and engineering drawings have a number of standard paper sizes. An 'A' size and an 'E' size drawing can be produced for the same picture content. Typically a draftsman will use the same mix of pen nib sizes in both drawings. Within any drawing, a draftsman likely will use several different pen nibs for line width to represent different meanings. For example, a map will use very thin lines for unimproved, secondary roads and will use much thicker lines for major superhighways or a dual carriageway.

A quill pen writing instrument can provide an analogy for cosmetic line width. Suppose a draftsman keeps a collection of ostrich, turkey, chicken, pigeon, sparrow, and hummingbird feathers. The choice of feather quill <pen nib> thus determines line width <weight>.

Patterns within an area may have symbolic meaning. Civil engineering applications need to represent concrete and other materials. Line patterns are one way of representing the material. If such a drawing is replicated on larger size paper, the lines representing the pattern for concrete should not have width enlarged. The effect to be reproduced by a computer graphics package is provision of a selection of line widths for cosmetic visual effect. In this situation line width and end cap shapes and dot-dashes should not be subject to transformation per se. The multiple pen nib size model for functionality ties a cosmetic interpretation to the attributes Line Width and Line Style.

From my development days as a graphics system implementer, I'll mention in passing that line style needs careful attention to consistency. Even in drawing a full circle, care must be taken to have the starting point on the circumference change so that rotated full circles in a pseudo-animation sequence have cosmetic line style properly rotated.

## 5. Clipping

Clipping is another task to which multiple interpretations can be applied. Many textbooks deal only with mathematical clipping, that is, traditional geometric intersection calculations. In traditional

geometry, parallel lines never meet or share a common point. Non-parallel lines do intersect and share one and only one common point. Such is not the case in raster graphics. Parallel lines can share common pixels and non-parallel lines can intersect in more than one common pixel in a raster environment. The line segment from (1,1) to (6,4) is parallel to the line segment from (0,0) to (5,3) yet their rastered representations share three pixels in common. The line segment from (0,0) to (5,4) and the line segment from (0,0) to (5,3) intersect geometrically only at the origin yet their rastered representations share three pixels in common.

With raster graphics, it makes a noticeable, visible difference whether one analytically clips a line segment locus then rasters the shortened line segment or whether one just straightforwardly rasters the full line segment onto a much larger canvas then cuts out the smaller 'clip window' of interest <scissor clipping>. I consider a pre-rastering clip to be an algebraic or mathematical clip or, that is, a traditional geometric clip. I consider a post-raster clip to be a scissor clip.

Raster pixels, conceptually, are small, zero-area dots centered at the integer grid intersection points <lattice points> of a rectangular, Cartesian coordinate system in which both the vertical and horizontal grid line spacing between pixels is unity. A clip border will be a line segment specified by its beginning and ending pixel coordinates. Pixels falling exactly on a border line segment are, by definition here, within the picture.

**Analytical clipping** examines the mathematical locus of each graphic entity without regard to its attributes. Any portion of an infinitely thin line outside the active clip window is discarded. Only that portion of the idealized line segment, now with new, likely fractional, end points is retained. If any portion of the line segment does lie within the clip window, then attributes such as width and style are applied. When a line segment's mathematical locus lies wholly outside the clip window, it is discarded in its entirety and no portion of its widened representation is drawn even if the widened portion would fall inside the clip window.

**Cosmetic clipping** can be thought of as first rendering an entire picture on an infinite area canvas then subsequently using a pair of scissors to cut out the clipped window of interest. I think of scissor clipping as 'raster first then clip'. I think of analytic clipping as 'mathematically clip first' then raster.

Suppose a rectangular clip area is specified to have sides parallel to the axes. Let the area be, border inclusive, defined by the area surrounded by the four lines  $y=0$ ,  $x=10$ ,  $y=6$ ,  $x=1$ . Seven-pixel wide lines having three pixels on each side of the respective reference line segments could be the line segment from  $(9,0)$  to  $(9,8)$  {the line  $x=9$ } and the line segment from  $(0,7)$  to  $(10,7)$  {the line  $y=7$ }. Analytic clipping would reject completely the line  $y=7$  while the line from  $(9,0)$  to  $(9,8)$  would be revised to a shorter line segment from  $(9,0)$  to  $(9,6)$ . Scissor clipping would retain both lines for conceptual rastering followed by a scissor clip.

Assume an identity matrix transformation. Analytic clipping would display for the line  $x=9$  the 35 pixels inside the clip area. The line  $y=7$  would contribute zero pixels.

Scissor clipping would display 50 pixels derived from each of the two original lines  $x=9$  and  $y=7$ . While the reference line  $y=7$  per se and its 'upper wide pixels' and 'leftmost pixels' would be discarded,  $y=7$  would contribute 30 pixels bounded by the rectangle formed from lines  $y=4$ ,  $x=10$ ,  $y=6$  and  $x=1$ . The line  $x=9$  would need to contribute its 35 pixels within the clip area. The resulting picture is that which would come from rastering the full lines then using a pair of scissors to cut out the clip area from the post-rastered picture. With scissor clipping, a large picture could be generated in successive swathes. For any form of raster monitor or dot-matrix printer using a bit map image and having limited memory, a picture drawn as successive swathes should be identical to the same picture drawn from a single rastering of the full picture into a very large memory.

For future reference, I'll note here that I've consciously double counted in individual Scissor clipping line segments the pixels in a rectangle bounded, edge inclusive, by  $y=4$ ,  $x=10$ ,  $y=6$  and  $x=6$ . In a replacement pixel drawing mode it won't affect the final picture seen on a display. In an 'Exclusive OR' drawing mode, what should happen to doubly drawn pixels?

Now let's consider some specific rastering situations which can arise. I'll contrive my example to offer integer intersections of lines and shall not cover additional considerations arising from non-integer intersections. Let the lower clip boundary be  $y=1$  and the rightmost boundary be  $x=9$ . The leftmost clip boundary can be  $x=0$  and the upper boundary can be  $y=3$ . For reference here, I'll claim a line segment from  $(0,0)$  to  $(27,3)$  will raster to:

5 pixels on scanline  $y=0$  from  $x=00$  to 04  
 9 pixels on scanline  $y=1$  from  $x=05$  to 13  
 9 pixels on scanline  $y=2$  from  $x=14$  to 22  
 5 pixels on scanline  $y=3$  from  $x=23$  to 27

The line segment from  $(0,0)$  to  $(27,3)$  intersects the boundary line  $y=1$  at  $(9,1)$ . It also intersects the boundary line  $x=9$  at  $(9,1)$ . Mathematical intersection leaves a single point. Even a single width, scissored line should draw the 5 pixels at  $(5,1)$   $(6,1)$   $(7,1)$   $(8,1)$  and  $(9,1)$ ; they clearly are within the boundary inclusive clip area. With wide lines, some sense of slope would be needed to determine how to do widening for this analytic clip instance of a single point  $(9,1)$ .

To continue the example, suppose the rightmost clip boundary were  $x=8$ . The line segment from  $(0,0)$  to  $(27,3)$  mathematically has its theoretical locus wholly outside the clip area.. Scissor clipping would present the four pixels  $(5,1)$   $(6,1)$   $(7,1)$   $(8,1)$ .

Next let the clip boundaries be given by  $y=0$ ,  $x=23$ ,  $y=3$  and  $x=4$ . A textbook clip would produce the line segment from  $(4,4/9)$  to  $(23,23/9)$ . Rounding to raster a perturbed line segment from  $(4,0)$  to  $(23,3)$  is not particularly accurate.

Actually rastering a line with a repetitive algorithm is not necessary to determine starting or terminating run lengths. The pixels too can be simply a direct mathematical evaluation or a run length slice evaluation. Cognizance also must be taken of the boundary line and its crossing direction. In the above example, in which an 'integer' boundary line  $y=K$  is examined for intersection with an  $x$ -major line  $y=y_0+((y_1-y_0)/(x_1-x_0))(x-x_0)$ , one can use  $y=K_-$  to calculate one extreme of the run along the scanline  $y=K$  and use  $y=K_+$  to calculate the other end of the run along the lower boundary. Clipping for raster devices also will need to process consistently equal error measure instances such as described by Boothroyd and Hamilton.

## 6. Drawing Mode

Drawing modes <such as Replace, Or, And, Exclusive Or, ...> are not uncommon as a computer graphics attribute. The question for a user can be: OK, I know what the dichotomous logic operation AND is; what are the operands for this binary operation on pixels? Does drawing mode affect a graphics entity such as a line as a whole or is it a combinatorial specification merely for pixels or is it meant to affect only bits within an implementation-specific, position dependent, ordered bit sequence internal

representation of a pixel before color table expansion? What's correct depends upon one's reference model.

For simplicity, I shall restrict my brief discussion of drawing modes to a monochrome or bi-level raster display environment in which a bit plane is used to refresh a display device. Pixels can be ON or OFF.

Exclusive Or (EXOR) is a drawing mode which usually can be relied upon to illustrate different picture presentation depending upon one's choice of reference model. It also is a good test to verify quirks in an algorithm; incautious use of 8-way symmetry in an incremental circle generation is exposed when drawing in EXOR mode if octant end points are doubly dotted.

In a Polyline from (0,0) to (3,0) to (7,0), how many times is the pixel at (3,0) drawn? If it is both the last pixel in the first line and the first pixel of the second line, it disappears owing to being drawn twice. A convention such as the first line segment in a polyline owns both its first and last pixel while all other line segments do not own their first pixel would cause a single drawing of the pixel at (3,0). What would it do for contiguous lines used to outline a closed simple area? Would pixels still be 'double dotted'?

Joined line segments with a small angle between them often share many pixels in common. Raster the polyline from (0,0) to (10,0) to (0,1). What should be expected using an EXOR drawing mode? The convention described in the preceding paragraph could leave the pixel at (10,0) visible but rather lonesome as its immediate neighbors would have disappeared owing to double dotting.

## 7. Conclusion

At the rendering level and for the display processing pipeline it is key to define clearly what objective or viewable action is intended. Parts of a graphics system can not be specified in a vacuum. If a reference model is established first, then decisions regarding implementation details can be made more consistently.

## 8. Acknowledgements

Thanks are owed to my colleagues on the IBM 3270 PC/GX graphics workstation at the development laboratory in Hursley, England, especially to Nancy Bull, Mike Davis, Adrian Gay,

Brian Middleton, Bryan Roberts, and Norm Sheen. I appreciate the support and encouragement from Steve Cunningham, Joaquim Madeira, and Mike McGrath for this GVE workshop. The specific numerical examples here were a part of an invited presentation at Ausgraph'90 International Conference on Computer Graphics; Michael Gigante did a great job with the conference.

## References

Arnold, D.B. and Bono, P.R. CGM and CGI: Metafile and interface standards for computer graphics. Springer-Verlag. 1988.

Boothroyd, J. and Hamilton, P.A. Exactly reversible plotter paths. Australian Computer Journal 2(No.1) p20-21. 1970.

Bresenham, J. E. Pixel processing fundamentals. IEEE Computer Graphics and Applications, 16(No15), p74-82. (January, 1996)

Bresenham, J. E. Computer graphics attributes and reference model alternatives. Proceedings: Ausgraph'90 International Conference on Computer Graphics. p413-424. (September, 1990)

Bresenham, J. E. Ambiguities in incremental line rastering. IEEE Computer Graphics and Applications, 7(No.5), p31-43. (May, 1987)

Gay, A. C. Experience in practical implementation of boundary-defined area fill. Fundamental algorithms for Computer Graphics, edited by R. A. Earnshaw. Springer-Verlag. p153-160. (1985)

Henderson, L.R. and Mumford, A.M. The computer graphics metafile. Butterworths. 1990.

Posch, K. C. and Fellner, W. D. The circle-brush algorithm. Theoretical Foundations of Computer Graphics and CAD, edited by R. A. Earnshaw. Springer-Verlag. p857-878. (1988)

# Appendix

