An Algorithm for Generating Anti-Aliased Polygons for 3-D Applications

Ni Guangnan* Peter Tanner Marceli Wein Grant Bechthold

Division of Electrical Engineering
National Research Council
Ottawa, Ontario, Canada

ABSTRACT

An algorithm for generating anti-aliased polygons is described. The algorithm is simple, but powerful, combining color blending, Gouraud shading and Z-buffer hidden surface removal. A test bed has been constructed which consists of a Versabus M68000 computer (the graphics processor) and a custom interface between the M68000 and the frame buffer of a raster display. Written in "C", the package for generating anti-aliased polygons performs all calculations on a line segment basis rather than pixel basis. This requires less memory and can thus be moved to smaller systems. The prototype test results indicate that a higher graphics performance can be achieved on inexpensive systems by this algorithm and implementation.

This paper also suggests an alternative to the Red/Green/Blue color representation normally used in color displays by storing intensity and two normalized color components to simplify the calculation involved in anti-aliasing, color blending and shading.

KEYWORDS:
Computer graphics, raster displays, aliasing, hidden surface removal, scan conversion, color, color space.

RESUME:

On trouvera décrit dans cet article un algorithme de génération de polygones" anti-aliased". Cet algorithme est un algorithme simple mais puissant qui combine à la fois la technique du mélange des couleurs, la technique d'ombrage de Gouraud et la technique d'élimination des surfaces cachées par tampon Z. Un banc d'essai a été construit; il se compose d'un ordinateur VERSAbus M68000 (le processeur graphique) et d'une interface personnalisée reliant le M68000 et la mémoire-image d'une unité d'affichage à balayage récurrent. Rédigé en langage compilé, le progiciel de génération de polygones anti-confusion effectue tous les calculs sur une base de segments linéaires plutôt que de pixels. Cela réduit la mémoire utilisée et l'algorithme peut par conséquent être mis en oeuvre sur un petit ordinateur. Les résultats des essais effectués sur le prototype démontrent que cet algorithme permet d'obtenir, avec des systèmes peu coûteux, de meilleures performances graphiques.

On trouvera également décrite dans le présent article une nouvelle méthode de représentation des couleurs utilisée dans les affichages couleur (par opposition à la représentation trichromatique normale rouge/vert/bleu) qui consiste à stocker l'intensité ainsi que deux composantes chromatiques normalisées pour simplifier les calculs "d'anti-aliasing", de mélanges des couleurs et d'ombrage.

MOTS CLES: Informatique graphique, affichages à balayage récurrent, "anti-aliasing", élimination des surfaces cachées, conversion de balayage, couleur, espace couleurs.

*Mr. Ni Guangnan is from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. He is currently a guest worker with the National Research Council of Canada.

1.0 INTRODUCTION

Aliasing is a phenomenon associated with any quantization process. In raster displays it results in jagged edges and disappearing details. Increasing the resolution of displays can reduce this effect but at a high cost. On the other hand, subtle shadings can be used as an economical approach for anti-aliasing. Many such methods have been proposed ([1], [2], [3], [4]). Although the underlying principle of these methods is not new and they do not differ substantially from each other, it is worthwhile to make improvements both to simplify the implementation and to extend the capability to the generation of anti-aliased colored polygons with shading and hidden surface removal.

The algorithm described here is based on a simplification of Crow's algorithm using two pixels on each scan line, however the intensity of each pixel is determined directly by its subpixel position rather than a look-up table (Section 2.1). In addition, lines with slopes greater than one (absolute value) are treated similarly to lines with slopes less than one (Section 2.2), so that they can better fit the scan line conversion process. Since raster displays are well suited to the display of surfaces, the algorithm has been extended to generate anti-aliased 3-D polygons. The Z-buffer approach was chosen as a basis for hidden surface removal. Gouraud shading is also applied to produce more realistic images (Section 2.3). It is well known that anti-aliasing cannot be achieved without color blending performed at the boundary of different colored surfaces. A simple criterion is applied (Section 2.4) for performing color blending functions, which yields acceptable results.

All these functions must be combined into a single package. For this purpose, a test bed has been constructed, consisting of a M68000 computer and a custom interface (Section 3.1). The former acts as a graphics processor and communicates with the host computer. The latter enables the M68000 to access the frame buffer of a raster display. The M68000, running at 8MHz, with its firmware, memory and interface yields a cost effective configuration.

The package for implementing this algorithm was developed using the C language (Section 3.2). If the calculations for anti-aliasing, color blending, shading and hidden surface removal were performed on a pixel basis as in other algorithms, all the required information would have to be stored for each pixel. Thus, quite a large memory, greater than 1MB would be required. To reduce this memory requirement, the package performs all calculations on a scan-line segment basis. Consequently, it can run on smaller systems without difficulty.

The package currently runs both on the test bed and on a VAX 11-780. Testing this prototype has shown that it does accomplish its intended functions (Section 4). Images of high quality and realism can be produced at reasonable speed on the simple test bed system.

Anti-aliasing, color blending and shading involve manipulation of the pixel intensity. Performing these calculations in Red/Green/Blue color space is computationally inefficient. To improve this efficiency, the package employs a color representation which includes intensity and two normalized color components. Compared with the Red/Green/Blue color system (extensively used in color displays), the so-called i/r/g color notation features better color/intensity capability and lower computational expense. Other potential advantages exist and are described in the Appendix.

2.0 ALGORITHM

2.1 Anti-aliased Vector

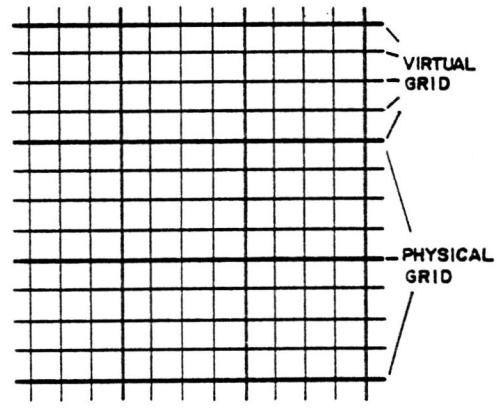

Fig. 2.1 A high resolution virtual grid superimposed on the physical grid.

The anti-aliased vector generation is based on the model of a frame buffer display with considerably higher resolution than the actual frame buffer. A virtual grid with this higher resolution is superimposed on the physical grid of the display (Fig. 2.1). A subpixel, then, is one square from the virtual grid. The resolution required depends on the available gray scale of the display device.

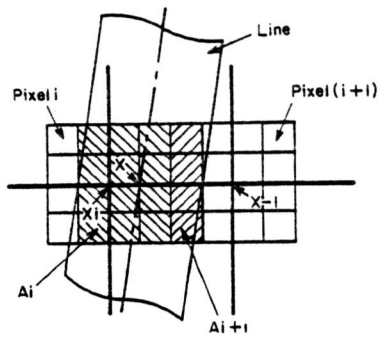

Fig. 2.2 The intersection of the line and the physical grid (X, as shown) determines the intensity of the two consecutive pixels -- pixel i and pixel (i+1).

Two consecutive pixels are shown in Fig. 2.2. Each pixel is defined as a square (the size of a physical grid unit). The line width is defined as the intersection of the vector and either an X grid line or a Y grid line depending on the slope of the line (Fig. 2.3). This makes 45 degree lines slightly thinner than horizontal or vertical lines. As long as the line is smooth, this is acceptable. In normal raster displays, 45 degree lines look slightly darker than horizontal or vertical lines; in our simple algorithm they are thinner.

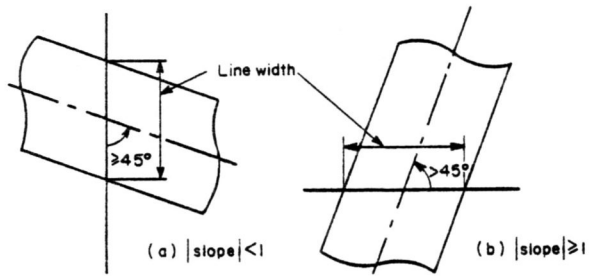

Fig. 2.3 The definition of the line width.

To determine the intensities of the two pixels in Fig. 2.2, the approximate overlap of the line on the pixels is determined by the intersection of the midpoint of the vector and the physical grid (to the nearest virtual grid unit). In Fig. 2.2 the intersection, measured in physical grid units, is X. We denote the fractional part of X in physical units as:

$$F(X) = X - \lfloor X \rfloor \qquad (2.1)$$

Where, $\lfloor X \rfloor$ is the greatest integer satisfying $\lfloor X \rfloor \leqslant X$. The overlap area (shaded area in Fig.2.2) is approximately expressed by:

$$Ai = 1 - F(X)$$
$$Ai+1 = F(X) \qquad (2.2)$$

The greater the inclination, the more precise this becomes. The absolute error distribution of Eq. (2.2) is plotted in Fig. 2.4. Note that the inclination is measured from the X grid for $|slope| \geqslant 1$ and from the Y grid for $|slope| < 1$, so it is always greater than or equal to 45 degrees. As can be seen from this plot, Eq. (2.2) yields quite a good approximation over most of the range. Thus in the anti-aliasing vector/polygon generation each pixel is properly intensified according to Eq. (2.2). No look-up table is required, making this approach very simple.

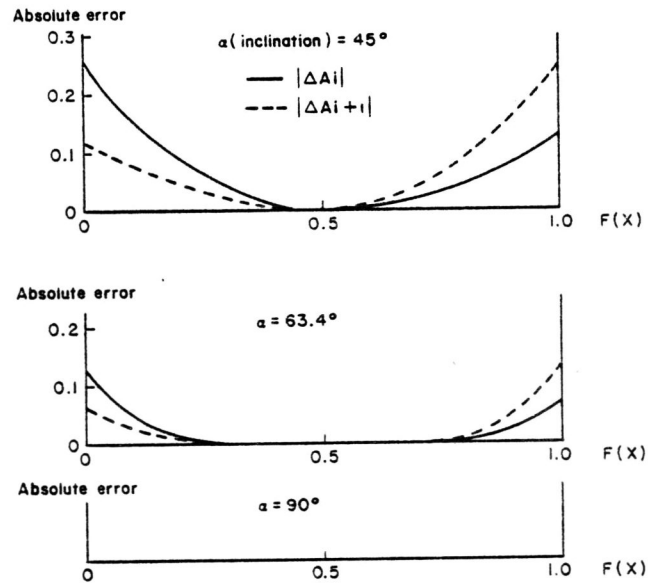

Fig. 2.4 Absolute error distribution of the Eq. (2.2).

Based on these principles the vector generator algorithm is explained below.

Given a vector drawn from $P_1(X_1, Y_1)$ to $P_2(X_2, Y_2)$ with $Y_1 \leqslant Y_2$, the following parameters are calculated:

$$\Delta X = X_2 - X_1$$
$$\Delta Y = Y_2 - Y_1$$
$$\text{If} \quad |\Delta Y| \geqslant |\Delta X|$$
$$\text{then } \delta X = -\Delta X / \Delta Y \qquad (2.3)$$
$$\delta Y = -1$$
$$\text{else } \delta X = \text{Sign}(\Delta X)$$
$$\delta Y = \Delta Y / \Delta X \quad (\text{Sign}(\Delta X))$$
$$\text{Here, Sign}(\Delta X) = 1, \text{ when } \Delta X \geqslant 0$$
$$\text{Sign}(\Delta X) = -1, \text{ when } \Delta X < 0 \qquad (2.4)$$

It should be noted that the X_1, Y_1, X_2, and

Y_2 are first rounded to integers (i.e. the physical grid unit). This is acceptable in most cases. By significantly increasing the complexity of the vector generator, the ends of the vector could be plotted more correctly.

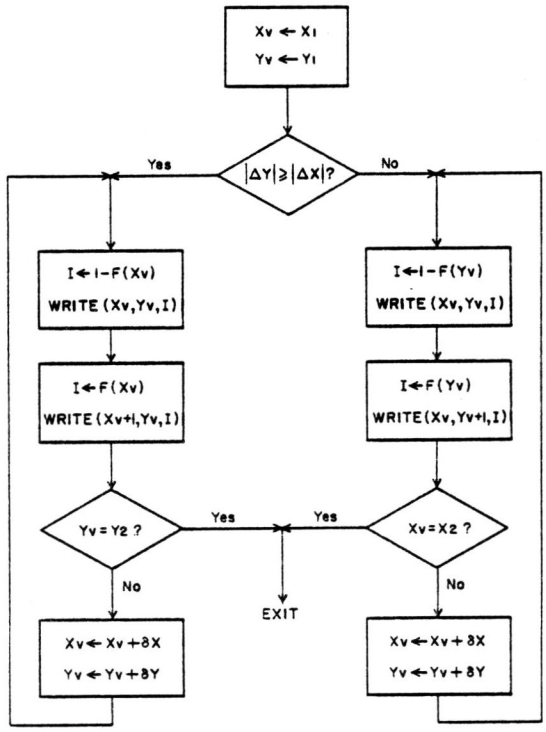

Fig. 2.5 Flow chart of the proposed anti-aliased vector generator.

In the flow chart in Fig. 2.5, the operation WRITE(Xv,Yv,I) represents a pixel with intensity value I to be written into the frame buffer at the address specified by Xv and Yv. Both Xv and Yv may have a fractional part, so the memory addressing is given by Xv and Yv -- the phsical grid unit. Although the fractional part has no influence on the WRITE operation, it does play a role in intensity calculation. This anti-aliasing vector generation is almost as simple as the normal vector generation, however, the simplicity of the vector generation does have a drawback. The lines drawn, although smooth, will have a minor repetitive intensity variation or pattern that changes (and is therefore more noticeable) when the line moves. This is known as the barber pole effect. These patterns are illustrated in Fig. 2.6. An intensity compensation technique can reduce this effect.

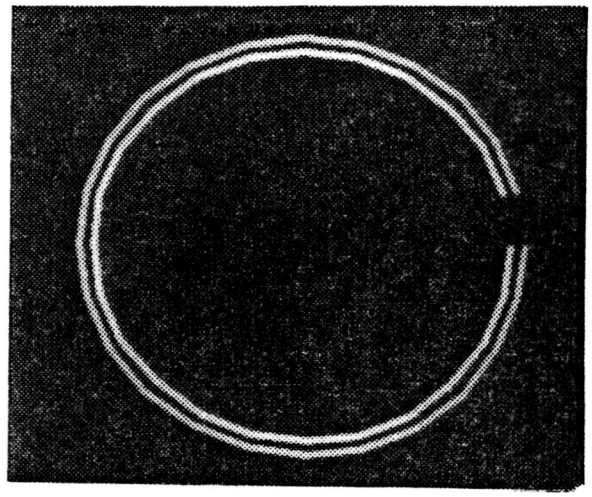

Fig. 2.6 Polylines generated by the normal vector generator (inner) and the anti-aliased vector generator(outer) without intensity compensation.

2.2 Anti-aliased Polygon Generation

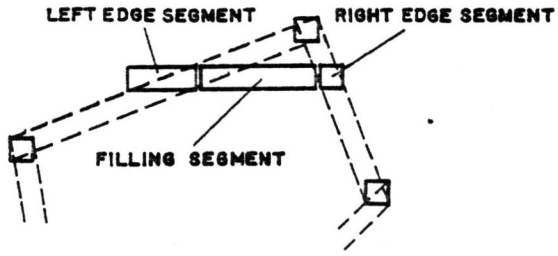

Fig. 2.7 Scan-line conversion of polygon combining anti-aliasing.

The algorithm described above can easily be extended to generate polygons in 2-D applications by first drawing anti-aliased polylines for the polygon boundaries, and then, filling the area enclosed by the polylines. It may be advantageous to modify the approach for certain applications. For example, the filling may be deferred in the case of displaying multi-polygons overlapping each other[9]. The algorithm may be tailored so that lines with $|dx|>|dy|$ are treated similar to the lines with $|dy|\geqslant|dx|$. Instead of writing a pixel at each x (refer to Fig. 2.5), a multi-pixel segment is written at each y. In other words, the pixel writing is deferred until a new y has been reached. This is particularly useful in

adapting the algorithm to scan-line conversion. As a result, the polygon can be converted to scan-lines no matter whether the slope of its edge is greater than one or not. Each scan-line consists of three segments: the left edge segment, the right edge segment and the filling segment as shown in Fig. 2.7. Note that the edge segment for a steep edge is a single pixel with the intensity determined by the subpixel position. However, a more horizontal edge may result in several pixels in the edge segment of the scan-line. The intensites of these edge segments as determined by subpixel positions, will vary linearly so that the two end pixels of the edge segments can express the whole edge segment. This forms the basis of the implementation described in Section 3.2.

2.3 Hidden Surface Removal And Shading

In 3-D applications it is useful to incorporate the Z-buffer technique for hidden surface removal. The Z-values on the edge can be linearly interpolated (similar to the X-value calculation in Fig. 2.5). The Z-values for non-edge pixels can be interpolated from the two edge pixels on the same scan-line.

Except for edge pixels, linear or Gouraud shading can be calculated in a similar manner. For edge pixels, the resultant intensity is calculated as the product of the intensity determined by the anti-aliasing algorithm and the intensity calculated by shading interpolation.

As described in Section 3.2, the picture is stored in memory in a line segment format. Each segment is expressed by its two end pixels. Consequently, only the Z-value and intensity of the two end pixels need to be calculated for storage while those pixels in between need be calculated only when necessary.

2.4 Color Blending

The color blending function is necessary for smoothing the boundary of different colored polygons. The criterion adopted here is straightforward. If two colors are blended, the resulting color is formed by additive primary mixing, with the resulting intensity equal to the maximun intensity of the two components. This has proved to be quite acceptable.

The Red/Green/Blue color representation normally used in color displays requires excessive computations when the anti-aliasing, shading and color blending are all combined. For this reason a color representation of intensity and two normalized color components is proposed. The main advantage lies in the separation of the intensity and color information. In this system the color components are normalized, so they are entirely independent of the intensity. (In color TV systems, the color components are not independent of intensity.) As a consequence, all the calculations relating to anti-aliasing and shading need only be performed on the intensity values. The color blending is also somewhat simplified with the use of the modified color system as described in the Appendix. It is only at the final stage of writing to the frame buffer that the intensity and normalized color components are converted to the Red/Green/Blue color representation for interfacing with regular color display.

3.0 IMPLEMENTATION

3.1 Hardware Configuration

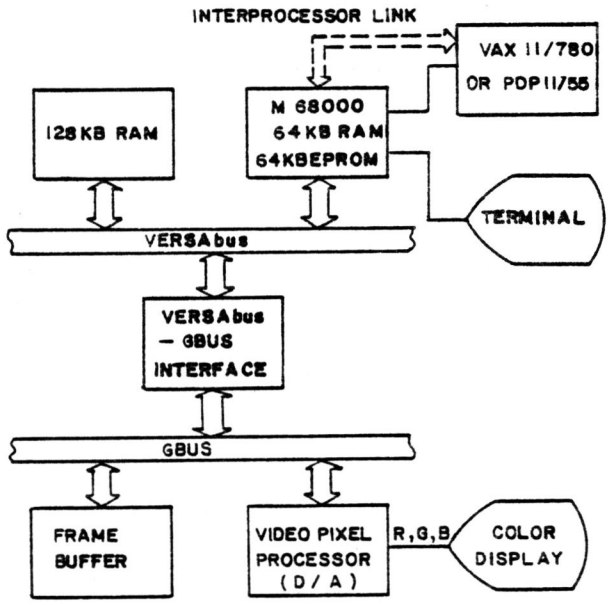

Fig. 3.1 A M68000 based prototype graphics system.

We have recently constructed a test bed based on a M68000 microprocessor (Fig. 3.1). The M68000 MPU board is a VERSAbus* module with 64KB RAM and 64KB EPROM on board and a 128KB RAM

---

* VERSAbus is a trademark of Motorola Inc.

module on the VERSAbus. The MPU Board has two serial ports, one connected to a terminal, the other to a host. The host port can be switched between a VAX 11-780 or a PDP-11/55. A custom interface enables the M68000 to access the frame buffer on the G-bus of the VDP* Graphics Display [10]. Initially, the VDP frame buffer was operated by a 2901 bit slice processor driven by the host computer (a PDP-11/55). In the current configuration, the M68000 can perform graphics functions independently of the 2901 and PDP-11/55.

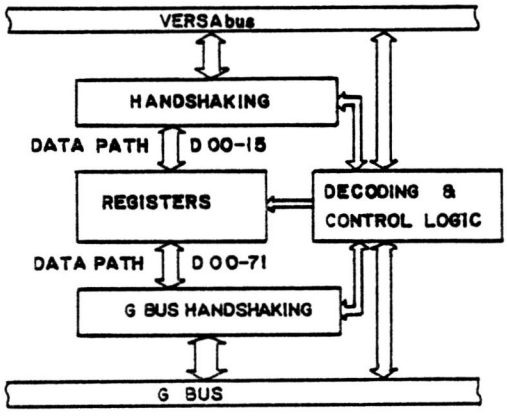

Fig. 3.2 Block diagram of the VERSAbus-GBUS Interface.

The interface is aimed at minimizing the overhead involved in converting the data from the asynchronous VERSAbus with its 16-bit data path (extendable to 32-bit), to the synchronous G-bus with its 72-bit data path. The interface circuitry must implement the necessary handshake for both buses, as well as convert the data format. The latter is solved by a scheme similar to that used by a two-port memory. As illustrated in Fig. 3.2, there are three 72-bit registers on the interface board. They can be directly accessed from both buses. To the VERSAbus side they appear as a 16-bit register file continuously mapped on to the M68000 address space. The pixel information can be directly manipulated in these registers before being sent to the frame buffer. Similarly, once the content of the frame buffer is read into the registers, it can then be made available to the M68000 immediately. Writing/reading these registers to/from the frame buffer on the G-bus is simply a memory write/read instruction for the M68000. In this way it is virtually transparent for the M68000 to access the frame buffer on the G-bus.

The Graphics support package resides in the EPROM of the MPU Board. In the stand-alone mode, the application programs are downloaded from the host to the M68000. The programs then run on the M68000 without host intervention. In the slave mode, the M68000 accepts the graphics routine calls from the application programs running on the host. However, the data rate of the serial port is a serious limiting factor. A fast interface will be installed to speed the interprocessor link between the M68000 and the host computer. This will also provide some freedom to distribute the graphics computation between the two computers in an optimal manner.

3.2 Software Description

In order to apply the described algorithm, four parameters are required for each pixel: the Z-value, intensity value, color, and a pixel attribute. The latter indicates whether or not the pixel is an edge pixel, and, if so, which type of edge. Obviously, more than 1MB of memory would be required to implement this algorithm if this information were stored for all pixels in a 512x512 grid. Since the algorithm is performed on a line segment basis, the memory requirement is reduced enabling the package to run on smaller systems. Table 3.1 lists some of the differences between pixel based storage and line segment storage.

It can be seen from Table 3.1 that the line segment method allows the package to run on smaller systems such as the test bed described here. The speed difference is not fully described by the Table. Generally speaking, the segment method will be slower for images consisting of a large number of polygons but quicker for pictures with fewer, larger polygons.

The whole package consists of three groups of routines.

(a) The routines for scan-line conversion of the polygon. Each scan-line is converted to three segments as shown in Fig. 2.7. The intensity and the Z-value are interpolated between vertices during the scan-line conversion. The segments are then fed to the later stages.

(b) The routines for resolving visibility of the segments. This is a sorting process. For each Y-value, the segments are sorted

---

* VDP is a product of NORPAK Ltd.

according to their X-value. The new segment may overlap the old one and many combinations may occur. There are routines for erasing a segment, inserting a segment, modifying a segment or performing combinations of these tasks. Certain color blending may also take place at this stage. A previously visible segment will return space to a free list if it is hidden. Conversely, a new visible segment generated will request space from the free list. If there is no space available it will ask for an allocation of new memory space. In this manner no memory is wasted.

(c) The routines for color blending to produce the segments for display.

Table 3.1 Element vs Performance

storage element	pixel based	line segment
memory requirement	>1MB at 512x512 resolution	proportional to image complexity, e.g. 128KB can accommodate 8192 line segments.
processing time	proportional to the polygon area.	proportional to the number of line segments.
program complexity	simpler	more complex

4.0 RESULTS AND DISCUSSIONS

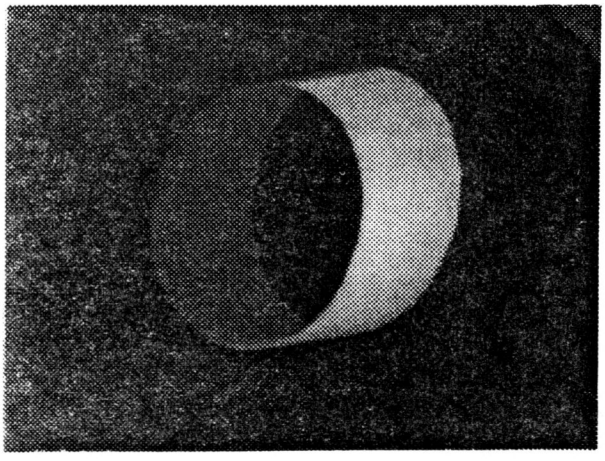

Fig. 4.1 (a) A cylinder generated by the anti-aliased polygon generator.

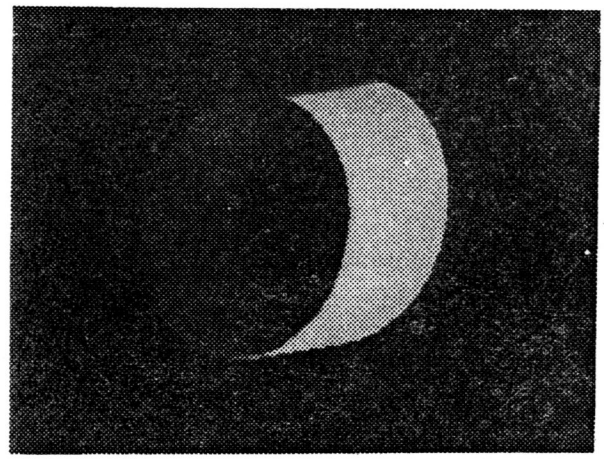

Fig. 4.1 (b) A cylinder generated by a normal polygon generator.

The package resides on the M68000 system as firmware. Fig. 4.1(a) is a cylinder generated by this package. Fig. 4.1(b) is the same object generated using a different program without anti-aliasing. The M68000 system described yields an acceptable performance. The time required for generating the image in Fig. 4.2 is about half a minute (the exact time depending on the rotation angle). Comparing this to the earlier implementation of the algorithm in FORTRAN on the PDP-11/55 the speed has been improved by roughly one order of magnitude. Shading was not included in this earlier implementation. Fig. 4.3 is an image demonstrating the color blending effect.

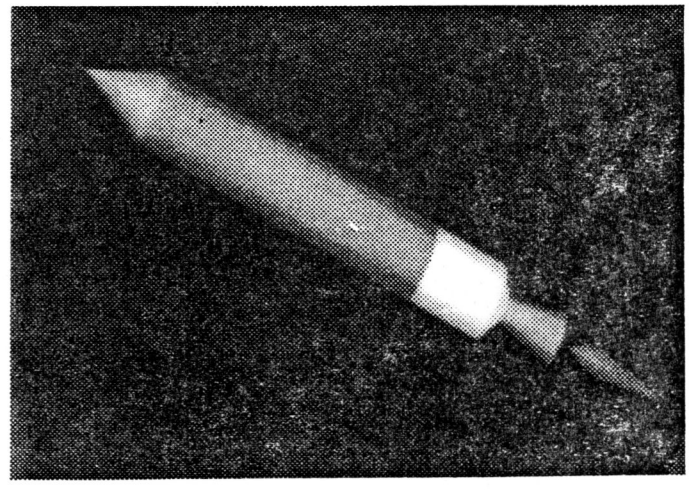

Fig. 4.2 An image generated on the M68000 prototype system.

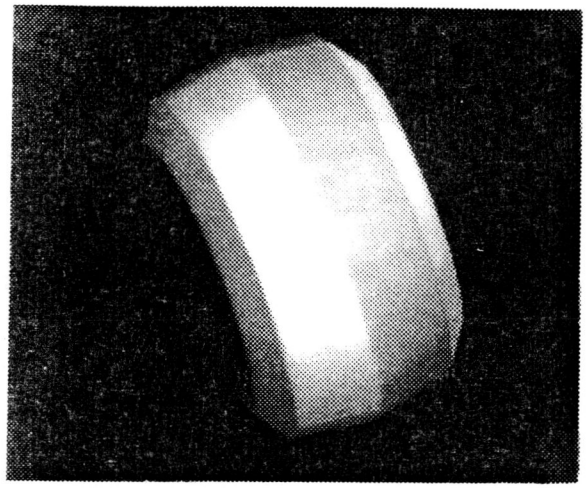

Fig. 4.3 An image demonstrating the color blending effect.

The same package running on the VAX 11/780 is about three times as fast (with no other intensive computation running concurrently). In view of the cost difference, the M68000 configuration yields a very good cost/performance ratio. It is expected that the speed can be improved either by employing custom hardware or multi-graphics processors, with each processor being a polygon generator.

In the current implementation, intersections produced by the Z-buffer algorithm (intersecting polygons) remain unsmoothed. Additional computation would be required to smooth these edges.

In summary, the algorithm and the particular implementation described provide an economical means of achieving a higher graphics performance on inexpensive systems. This could be valuable in many graphics applications.

Acknowledgement

The authors wish to thank several people in the Computer Graphics Section, Division of Electrical Engineering, particularly Mr. Art Binch, Mr. Jim McDougall, Mrs. Cathy Merritt, Mr. Jack Lee and Mr. Mike Duggan.

References

[1] F. Crow, "The Use of Gray Scale for Improved Raster Display of Vectors and Characters", Computer Graphics (Proc. Siggraph'78), Vol.12, No.3, Aug., 1978, pp.1-5.

[2] E. Piller and H. Widner, "Real-Time Raster Scan Unit with Improved Picture Quality", Computer Graphics, Vol.14, No. 1 & 2, July 1980, pp.15-38.

[3] J. Burros and H. Fuchs, "Generating Smooth 2-D Monocolor Line Drawings on Video Displays", Computer Graphics (Proc. Siggraph'79), pp.260-269.

[4] S. Gupta and R. Sproull, "Filtering Edges for Gray-Scale Displays", Computer Graphics (Proc. Siggraph'81), Vol.15, No.3, Aug., 1981, pp.1-5.

[5] G. Joblove and D. Greenberg, "Color Space for Computer Graphics", Computer Graphics (Proc. Siggraph'78), Vol.12, No.3, Aug., 1978, pp.20-25.

[6] A. Smith, "Color Gamut Transform Pairs", Computer Graphics (Proc. Siggraph'78), Vol.12, No.3, Aug., 1978, pp.12-19.

[7] H. Gouraud, "Continuous Shading of Curved Surfaces", IEEE Trans. Vol. C-20, No. 6, June 1971, pp.623-629.

[8] E. Catmull, "A Tutorial on Compensation Tables", Computer Graphics (Proc. Siggraph'79), Vol.13, No.2, Aug., 1979, pp.1-7.

[9] Guangnan Ni and Peter Tanner, "The Application of Anti-aliasing Technique for Displaying High Quality Chinese Characters", Proceeding of International Conference of the Chinese-Language Society, Sept. 1982, pp.37-45.

[10] M. Wein, N. Burtnyk, W. A. Davis and J. Norton, "A Raster Display System for Computer Graphics and Image Processing", 6th Man-Computer Communications Conference, May 1979, pp.115-125.

APPENDIX: Representing Color by Intensity and Two Normalized Color Components

The Red/Green/Blue color space normally adopted by color displays is not the optimal color space for calculating anti-aliasing, color blending, or shading. Other spaces such as HSV [5],[6] have certain advantages, but require conversion to Red/Green/Blue before they can be fed to the video stage, a non-trivial task. For this reason, a color representation is proposed which simplifies the pixel operations considerably but requires little extra hardware to implement the video stage.

This representation is a modification of the Red/Green/Blue notation. A separate item representing energy is introduced:

$$i = (R+G+B)\,K \qquad (A.1)$$

where, a constant (K) is used to adjust the i to be within the range 0-1. Although we call i "intensity", it is not the visual intensity perceived by a standard observer. We will call the latter "luminance" to distinguish it from "intensity". The system uses two normalized color components expressed by,

$$r = R/(R+G+B) \qquad (A.2)$$
$$g = G/(R+G+B) \qquad (A.3)$$

The third normalized color component is redundant and can be derived from Eq. (A.2) and (A.3):

$$b = 1-r-g \qquad (A.4)$$

Thus only i, r and g are included to form the color notation.

Converting i/r/g into Red/Green/Blue is simple:

$$R = i\cdot r\cdot K \qquad (A.5)$$
$$G = i\cdot g\cdot K \qquad (A.6)$$
$$B = i\cdot(1-r-g)\,K \qquad (A.7)$$

Again, a constant appears in the above formulas to scale the Red/Green/Blue components as required. Usually, this constant can be set to 1 for convenience.

It is well known that all physically reproducible colors in a color display are those within a triangle in the chromaticity diagram with the three vertices being the color primaries R, G and B. Assuming a color with three color components R+G+B≠ 1, we can always scale it so that R+G+B= 1. This is an intensity variation that does not change the color. Therefore, the lack of the constraint R+G+B= 1 does not influence the color gamut but provides intensity variability. In the Red/Green/Blue system, when the intensity decreases the color resolution decreases accordingly.

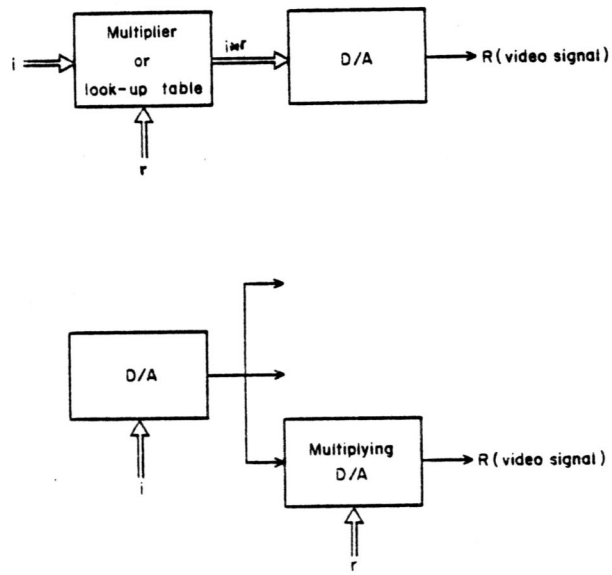

Fig. A.1 Possible hardware configuration of the video stage for the proposed i/r/g system.

Conversly, in the i/r/g system intensity and color information are mutually independent. The color resolution is unaffected by the intensity. Of course, provision must be made to effectively use the color information at lower intensities. The transformation from i/r/g to Red/Green/Blue specified by Eq. (A.5)-(A.7) may be carried out in hardware. Fig. A.1 depicts the possible configuration of a video stage tailored for the proposed i/r/g system. Either analog or digital high speed multipliers can perform the required transformation in real time, taking full advantage of the color-intensity independence.

Because the intensity calculations need to be applied to the intensity component alone rather than all the three components in Red/Green/Blue space, the i/r/g system is computationally efficient.

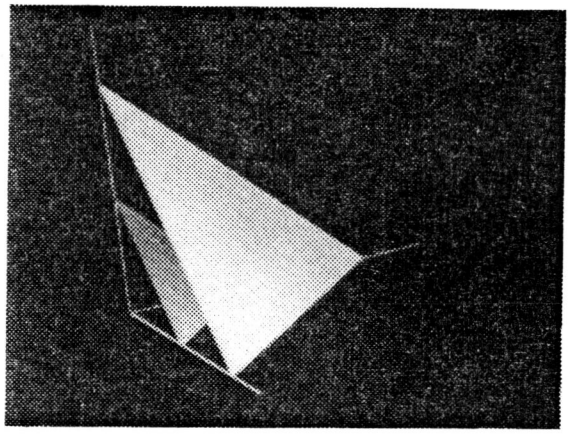

Fig. A.2 Constant intensity planes in Red/Green/Blue space, displayed i=1 (foreground) and i = 0.5 (background).