<http://www.sublogiccorp.com/contents/displayArticle.aspx?0_3>

|  |  |  |
| --- | --- | --- |
| **3D Computer Graphics ... you don’t need glasses! (1977 SubLogic Inaugural Article)** |  |  |

|  |  |  |
| --- | --- | --- |
| **by Bruce A. Artwick** |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_0.gif |  | |  |  |  |   From the October 1977 issue of Kilobaud Microcomputing magazine.  \* \* \*  Graphics is an area that I’m sure most of us find exciting. When you consider some of the applications (games, educational programs, etc.) that we’ll see sophisticated graphics, such as 3D, used for in the years to come it gets even more exciting!  Bruce’s article describes a package he and his associates have developed and are marketing through their company, Sublogic. Included in the article is a listing for a "starter program" which you can get running with minimum hardware and memory. — John Craig, Editor, Kilobaud Magazine.  \* \* \*  A highway engineer is casually driving down a road which doesn’t exist, enjoying the scenery. Meanwhile, a pilot is in a spin, airspeed climbing, watching the runway which is coming up fast! It’s three-dimensional, dynamic computer graphics which is making both simulations possible, and if similar types of computerized display systems were available to everyone, who knows how many interesting events would be occurring.  Until this time, however, only a limited number of real-time 3D systems were available. The hardware involved was usually large, bipolar, fast, and very expensive, ranging from tens to hundreds of thousands of dollars. This is hardly what a hobbyist could afford.  Dedicated 3D digital projection hardware is still prohibitively expensive, and such an investment is not desirable simply because it is dedicated to only one task.  Graphics display devices (raster scan bit maps), on the other hand, are a different matter. Prices are at an all time low, putting conventional, although somewhat coarse 2D graphics within reasonable economic reach. To enter the world of three dimensions, all that is needed is a low-priced graphics generator, and the microcomputer can quite successfully take on the role if applied properly.  Software is the key to making a microcomputer perform as an efficient 3D to 2D transformer. Running at real-time speeds with limited memory demands efficiency, and although a high level language such as BASIC could be used, speed-wise the results would be disappointing.  Optimized assembly language combined with good real-time programming techniques and double or triple precision arithmetic can increase speed on the order of ten to one hundred times.  The Sublogic Three-Dimensional Micrographics Package is a collection of such graphics programs which can be used in whole or on a subroutine basis for computerized 3D graphic simulations. Three-dimensional data bases may be observed in a dynamic mode from any position or angle in space.  Versions of the graphics package are currently available in 6800 assembly language and in BASIC for slower graphics. This article will present some of the methods used in the graphics package and will provide insight into the problems faced by the graphics programmer. The micro or minicomputer user can take it from there in putting 3D graphics to use in his special application.   |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  |  |  | | --- | | Views of 3-D transforms and projections decorating article. |   **Graphics Quality**  What quality of graphics can be expected from a micro processor? Broadcast quality, speed and density are definitely not possible. Even the large machines have trouble here. Processing bandwidth is the limiting factor. An eight-bit machine with a two MHz clock simply cannot generate a six MHz bandwidth picture.  It should be kept in mind that 3D to 2D conversion is not a trivial process either. The majority of the processor’s time is spent doing mathematical calculations which cut down projection speeds drastically.  Projection frame rates and image complexity are inversely proportional. If five minutes are allowed to generate one frame, an extremely dense image is attainable. A dynamic display of ten frames per second, on the other hand, will be relatively sparse. The software should be capable of making speed/complexity trade-offs at the user’s discretion.  **A 3D Graphics Algorithm**  There’s much more to 3D projection than meets the eye. There are the obvious tasks of data base transformation (which considers the viewer’s position and viewing angle), and 3D to 2D perspective projection. Added to this is the necessary task of off-screen line elimination and clipping for objects which fall off or partially off the screen. There are also the optional tasks of hidden line elimination, shading, shadowing and reflection.  For the 3D Micro graphics Package, in the interest of reasonable speed and memory requirements, the simple wire frame line transformation, clipping and perspective projection method was chosen. Fig. 1 shows the type of image which a microcomputer can reasonably produce on a low-cost display system.   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_2.gif |  | |  |  |  |  |  | | --- | | Fig. 1. A wide angle view of a few simple architectural forms. A 6800 microcomputer can produce this simple scene at a few frames per second if very efficient programming is used. The 200 x 200 bit map display can best be described as coarse. |   Fig. 2 shows display characteristics and performance.   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_3.gif |  | |  |  |  |  |  | | --- | | Fig. 2 Characteristics of the Sublogic 3D Micrographics Program number 3DG68V3. |   Modularity is important in graphics programming since many subtasks must be performed. Fig. 3 shows a good program partitioning.   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_4.gif |  | |  |  |  |  |  | | --- | | Fig. 3. The major operational modules of the display program. Modular partioning allows for future expansion and multiprocessing. |   **Data Base Structure**  The data base is essentially the world which is going to be projected, represented as a series of lines in a three coordinate space. Each line is represented as a start point (X,Y,Z) and end point in memory. By properly transforming the two points, the line between them is also transformed.  Since much time is needed to transform a point, a 2:1 time savings can be realized on large objects by using strings of lines or start and continue points. The lines (a to b) and (b to c) would be expressed as (a to b to c) resulting in three point transformations instead of four.  A number of methods for building data bases exist. One is to have a computer digitize a drawing or picture. Another is to have the user generate it with a joystick arrangement. For the data base sizes used by a microcomputer system, generation of a data base by hand with grid paper is probably the best choice. Fig. 4 shows a simple data base which shall be used in the examples henceforth. The sixteen points are numbered in the optimal order for stringing lines together with continue points.   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_5.gif |  | |  |  |  |  |  | | --- | | Fig. 4. A road and a pyramid. This simple data base will be used for illustrative purposes. Notice the coordinate numbering sequence and the X, Y and Z directions. |   **Space/Screen Coordinate System**  The 3D coordinate system is set up so the resulting 2D image displays X and Y on the screen in typical graph axis fashion with Z representing depth.  **Perspective Projection**  3D to 2D perspective projection is really an easy process. First, imagine a flat projection of a 3D scene on a 2D screen; that is, merely eliminate the Z component and plot the lines on the screen. Fig. 5a shows a cube being transformed.   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_6.gif |  | |  |  |  |  |  | | --- | | Fig. 5a. A flat projection of a cube onto a screen. The perspective projection pushes points in the distance toward the screen’s center by dividing by the depth (Z).  Fig. 5b. The perspective projection pushes points in the distance toward the screen’s center by dividing by the depth (Z). |   In Fig. 5a the cube has indeed been projected, but there is no depth consideration. Whether the cube was a foot or a mile away the result would look the same. Optical geometry shows that lines appear to get shorter as the inverse of the distance. By plotting X/Z and Y/Z instead of just X and Y, points progressively further away will be pushed closer to zero which is the center of the screen (see Fig. 5b).  Note that, in addition to perspective, the small cube is now visible. This is due to the viewer’s cone (as opposed to tunnel) of vision. In the case of the rectangular screen this becomes the viewing pyramid. This concept is used extensively in the program. Fig. 6 illustrates the viewing pyramid.   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_7.gif |  | |  |  |  |  |  | | --- | | Fig. 6. The viewing pyramid encloses all objects which can be seen and thus displayed. In theory, the viewing pyramid angles change as one gets closer to the screen. |   **Line Clipping**  Line clipping is the most difficult procedure in the display program. The following questions must be asked of each line in the data base: - 1. Does the line belong on the screen or is it outside the viewing pyramid? - 2. Does the line somehow intersect the viewing pyramid if one or both points are off the screen  Using the line’s end points, an equation for the line can be set up. Solved simultaneously with the equations for the planes of the pyramid, a line can essentially be pushed toward the pyramid’s boundaries as shown in Fig. 7.   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_8.gif |  | |  |  |  |  |  | | --- | | Fig. 7. Clipping. Only part of the line is visible (within the viewing pyramid). Before projection the lines must be clipped to meet the screen boundaries. |   It should be noted that 2D clipping (which some graphic terminals possess) will not work for 3D points due to the pyramid’s increase in dimensions along the Z axis. For exact equations and more information about the coding and clipping of lines, the Sublogic package or a text such as ’Principles of Interactive Computer Graphics’ by Newman and Sproull should be consulted.  **Data Base Transformation**  Data base transformation is what gives the viewer the amazing capability of being able to go anywhere and look at a scene from any direction or angle. Through a geometric transform, the world (3D data base) is first translated in the X, Y and Z directions by adding an offset to all data base points. The world is then rotated by multiplying by a heading, pitch and finally a bank equation.  The four equations are most easily handled by the computer in matrix form, creating the need for a four by four transformation matrix derived from the offsets and rotation angles. Data base points are multiplied by the matrix in a vector times matrix manner. Fig. 8 shows the data base being rotated. It is interesting to note that it is the world which moves, not the viewer.   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_9.gif |  | |  |  |  |  |  | | --- | | Fig. 8. Data base translation and rotation permits views from any direction or angle. Notice that it is the world that moves. |   The matrix shown in Fig. 9 is the proper transform and need only be created once for each image. A separate subroutine in the program is set aside to do this.   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_10.gif |  | |  |  |  |  |  | | --- | | Fig. 9. The translation matrix (above) is multiplied by the rotation matrix yielding the transformation matrix. Fortunately, this matrix need only be computed once per frame. |   **That’s About It — Or Is It?**  In principle that’s about all there is to getting a 3D wire frame projection on the screen; transform, clip and project. Using a microcomputer to accomplish this, however, poses a number of interesting problems. For any serious graphics, double precision accuracy (on an eight-bit machine) is a must. High speed double precision matrix multiplying, clipping and projection routines are needed.  If the display device being used does not have vector drawing capabilities, the microcomputer must also take on this responsibility.  Running off the end of the data base is another problem. If a point is near the end of the data base and the viewer moves a little too much in the wrong direction (remember, it’s the world which actually moves), the point can "fall off the edge of the world" or integer overflow. Scaling, saturation, or a smaller data base can help here.  The projection of the integer world with its discrete point values can cause severe distortion with points close to the viewer. At one integer unit from the screen, for example, a point within the viewing pyramid can only be in one of nine positions( X= - 1 ,0,1 and Y=-1 ,0,1 ) which makes for some pretty rough movements of near lines. The Sublogic package uses a spherical compression method to reduce this problem.  Programming consideration should also be given to "running into points." If a point falls at the base of the viewing pyramid (z=0), a divide by zero will result upon projection.  **Conclusion**  Getting 3D graphics and dynamic 3D graphics up on a microcomputer system requires at least 8K of memory for a 6800 or 6500 series processor, and possibly more for an 8080-based system.  Dynamic graphics, in addition, requires some sort of screen erase or update feature to remove the old display frame before the new one appears. Individually erasing vectors can cause image degradation due to line intersection erasures.  As for microcomputer preference, any one will do, with speed and memory size being the main difference in the final result. The 6800 systems work extremely well due to the large quantity of mathematics being performed in a real-time mode. Incidentally, the Sublogic package in assembly language form was first developed on a Southwest Technical Products 6800 System.  For those interested in multiprocessors, graphics is a great application as the algorithm partitions very nicely. A 6800 processor can do all the complex transforms, for example, and an 8080 system can generate vectors and control the screen display. The images shown in Figs. 1 and 10 were, in fact, generated using a SWTP 6800 for the 3D routines and a Graphics One 200 x 200 dot matrix intelligent graphics terminal for vector generation.  A 3D display program can, of course, be written completely by the user, or a whole display program can be purchased from Sublogic. A few books exist on the subject, but floating point algorithms are usually given. For speed’s sake a good text on integer programming should be consulted (this is also hard to find).  Three-dimensional micro computer graphics is now available to all microcomputer users. Real-time simulations, games, modeling and art are now possible. This is just a subset of the total list, however, which is indeed limitless.   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_11.gif |  | |  |  |  |  |  | | --- | | Fig. 10. Crashing through a runway can be quite safe with 3D graphics. The cube is in the aircraft’s reference frame and remains the same in all frames. |   **A 3D Graphics Program For Experimentation and Familiarization**  A somewhat condensed version of the BASIC Sublogic 3D Micrographics program is flowcharted in Fig. 11, and a listing is provided (Program A). With this program, anyone with a CPU, 4K BASIC and a console terminal can get started with 3D graphics.  Extensive rotation capability has been omitted to limit program size and memory requirements, but two viewing angles (front and rear) have been provided in lieu of rotation. Users with about 17K of memory can use a BASIC interpreter with trig functions to generate the complete matrix (see Fig. 9 and listing block 8400).   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_12.gif |  | |  |  |  |  |  | | --- | | Fig. 11. Flowchart for 3DGKB.V01. |   This program is very modular and permits easy interfacing to any user’s system. The input and output handlers simply ask the user for input parameters (3D start and end points of lines and the viewer’s 3D position) and writes the resulting 2D output display point values on the console terminal.  This simple I/O protocol allows the user to first get the program up and tested without a graphics display device. The user can then substitute his own I/O handlers (blocks 8100 and 8300) to load display files (arrays of many lines), format the output, and send it to his ’Polymorphics,’ ’Dazzler,’ ’SWTP’ or ’Matrox’ display (to name just a few). The USER, POKE and PEEK functions available with most BASICs will be quite useful here.  Fig. 12 shows a sample run. All input parameters are floating point. Although a viewer’s angle of 1.0 is said to be normal, .6 or .7 looks much more natural.   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_13.gif |  | |  |  |  |  |  | | --- | | Fig. 12. A sample run of 3DGKB.V01. |   A unique line generator is included in the 8900 block. Only twenty-one statements are required to generate all the points on a line. No trig functions, multiplications or divisions are needed. The Graphics One photographs of Figs. 1 and 10 were generated using an assembly language version of this routine.  Complete Three-Dimensional Micrographics Packages in BASIC for slow graphics and 6800 assembly language for high speed applications are available from Sublogic, PO Box 3442, Culver City CA 90230, for $22 and $28 respectively. Both packages include valuable interface and applications information. The 2900 Iine 6800 version includes a "Kansas City standard" tape for easy entry.  **Program A. 3DGKBV01 Listing.**  8000 REM 3D GRAPHICS VERSION 3DGKB.V01 5/27/77 8002 REM AUTHOR- B. ARTWICK FOR SUBLOGIC CO. 8004 REM PROGRAM USE-SLOW SPEED 3D GRAPHICS FOR 8006 REM EXPERIMENTATION AND FAMILIARIZATION. 8008 REM LANGUAGE- MINIMAL SET BASIC (SWTP 4K TYPE) 8010 REM MEMORY REQUIRED- 11K INCLUDING 4K FOR BASIC. 8012 REM FEATURES- PERSPECTIVE PROJECTION, CLIPPING, 8014 REM VARIABLE VIEWING WINDOW, LINE GENERATOR, 8016 REM COORDINATE TRANSLATION. 8018 REM OPTIONS AVAILABLE FROM SUBLOGIC- OPTIMIZED 8020 REM CLIPPING, COORDINATE ROTATION, ARRAY 8022 REM HANDLING, 6800 ASSEMBLY VERSIONS FOR 8024 REM SERIOUS DYNAMIC APPLICATIONS. 8026 REM EQUIPMENT NEEDED- A CPU AND CONSOLE TERMINAL 8028 REM ARE ENOUGH TO GET STARTED. A GRAPHICS 8030 REM DISPLAY DEVICE IS HIGHLY DESIRABLE. 8034 REM VARIABLE DEFINITION - - - - - - - - - - - - - - - 8036 REM X,Y,Z ARRAYS 1 THROUGH 5 ARE FOR: 8038 REM 1= 3D START PT. 2= 2D START PT. (NO Z) 8040 REM 3= VIEWER’S TRANSLATIONAL POSITION 8042 REM 4= 2D END PT. 5= 3D END PT. 8044 REM C= CLIPPING CODE ARRAY 8046 REM T= TRANSFORMATION MATRIX ARRAY 8048 REM P,B AND H= PITCH, BANK AND HEADING 8050 REM P2= PROJECTION FLAG 8052 REM W,V= WINDOW AND FIELD OF VIEW 8053 REM A,B,D,E,G,H,M,N,S= TEMPORARY STORAGE  8054 DIM X(5),Y(5),Z(5),C(8),T(16) 8100 REM INPUT HANDLER - - - - - - - - - - - - - - - 8105 REM ALL VARIABLES NEEDED TO TRANSFORM, CLIP 8110 REM AND PROJECT ONE LINE ARE ASKED FOR. 8120 PRINT "SCREEN WIDTH? (W)" 8125 INPUT W 8126 W=W/2 8130 PRINT "FIELD OF VIEW? (V)" 8135 INPUT V 8140 PRINT "VIEWER’S LOCATION? (X,Y,Z)" 8145 INPUT X(3),Y(3),Z(3) 8150 PRINT "VIEWER’S ANGLE? (P,B,H)" 8155 INPUT P,B,H 8160 REM CREATE THE TRANSFORMATION MATRIX 8165 GOSUB 8442 8170 PRINT "START PT. COORDINATE? (X,Y,Z)" 8175 INPUT X(1),Y(1),Z(1) 8180 PRINT "END PT. COORDINATE? (X,Y,Z)" 8185 INPUT X(5),Y(5),Z(5) 8200 REM EXECUTION CONTROL - - - - - - - - - - - - - - - 8203 REM A LINE IN 3D SPACE IS TRANSFORMED TO 8206 REM A 2D SCREEN LINE. THE INPUT HANDLER HAS 8209 REM ALREADY BUILT THE TRANSFORMATION MATRIX 8218 REM PERFORM TRANSLATION,ROTATION AND 8219 REM VIEWING WINDOW CALCULATIONS 8221 GOSUB 8530 8224 REM CLIP THE LINE 8227 GOSUB 8608 8230 REM PROJECT THE LINE 8233 GOSUB 8830 8236 REM DISPLAY THE RESULTS 8239 GOSUB 8330 8242 REM ASK FOR ANOTHER LOOP, NEW VIEW OR STOP 8245 PRINT "NEW LINE=1, NEW VIEW=2, STOP=3" 8248 INPUT A 8257 ON A GOTO 8170,8140,8260 8260 END  8300 REM OUTPUT HANDLER - - - - - - - - - - - - - - - 8305 REM THE SCREEN COORDINATES OR "LINE OFF 8310 REM THE SCREEN" MESSAGE IS PRINTED. 8315 REM THE USER CAN REPLACE STATEMENT 8345 WITH 8320 REM "RETURN" IF PIXELS ARE NOT NEEDED 8330 IF P2=0 THEN GOTO 8355 8335 PRINT "THE SCREEN LINE IS:" 8340 PRINT X(2);", ";Y(2);" TO ";X(4);", ";Y(4) 8345 GOSUB 8909 8350 RETURN 8355 PRINT "THE LINE IS OFF THE SCREEN" 8360 RETURN  8400 REM TRANSFORMATION MATRIX GENERATOR  8403 REM NORMALLY THE FULL TRANSFORMATION MATRIX 8406 REM WOULD BE CREATED. TO SAVE MEMORY, ONLY 8409 REM TWO VIEWS ARE POSSIBLE WITH THIS VERSION: 8412 REM FRONT AND REAR VIEWS. 8414 REM BASIC MUST INITIALIZE T MATRIX TO ZERO. 8442 PRINT "FRONT VIEW=1, REAR VIEW=2" 8445 INPUT A 8448 IF A=2 THEN GOTO 8480 8451 REM FRONT VIEW MATRIX 8454 T(1)=1 8457 T(6)=1 8460 T(11)=V 8463 T(13)=-X(3) 8466 T(14)=-Y(3) 8469 T(15)=-Z(3)\*V 8472 T(16)=1 8475 RETURN 8478 REM REAR VIEW MATRIX 8480 T(1)=-1 8482 T(6)=1 8484 T(11)=-V 8486 T(13)=X(3) 8488 T(14)=-Y(3) 8490 T(15)=Z(3)\*V 8492 T(16)=1 8494 RETURN  8500 REM MATRIX MULTIPLIER - - - - - - - - - - - - - - - 8505 REM THE TWO INPUT VECTORS (START AND END 8510 REM POINTS) ARE MULTIPLIED BY THE 8515 REM TRANSFORMATION MATRIX "T". 8520 REM [X,Y,Z,1] T1 T2 T3 T4 [X’,Y’,Z’,1] 8521 REM . T5 T6 T7 T8 8522 REM . T9 T10 T11 T12 8523 REM . T13 T14 T15 T16 8530 FOR A=1 TO 5 STEP 4 8535 G=X(A) 8540 H=Y(A) 8545 X(A)=G\*T(1)+H\*T(5)+Z(A)\*T(9)+T(13) 8550 Y(A)=G\*T(2)+H\*T(6)+Z(A)\*T(10)+T(14) 8555 Z(A)=G\*T(3)+H\*T(7)+Z(A)\*T(11)+T(15) 8560 NEXT A 8565 RETURN  8600 REM LINE CLIPPER - - - - - - - - - - - - - - - 8601 REM LINES ARE CODED WITH RELATION TO THE 8602 REM VIEWING PYRAMID AND CLIPPED IF 8603 REM NEEDED. THE PROJECTION CODE IS SET. 8606 REM CODE THE START AND END POINT. 8608 FOR A=1 TO 5 STEP 4 8610 C(A)=0 8612 C(A+1)=0 8614 C(A+2)=0 8616 C(A+3)=0 8618 IF X(A) < -Z(A) THEN C(A)=1 8620 IF X(A) > Z(A) THEN C(A+1)=1 8622 IF Y(A) < -Z(A) THEN C(A+2)=1 8624 IF Y(A) > Z(A) THEN C(A+3)=1 8626 NEXT A 8630 REM OFF SCREEN LINE CHECKING 8632 FOR A=1 TO 4 STEP 1 8634 IF C(A)=0 THEN GO TO 8638 8636 IF C(A)=C(A+4) THEN GOTO 8668 8638 NEXT A 8642 REM START POINT OFF SCREEN CHECKING 8644 FOR A=1 TO 4 STEP 1 8646 IF C(A)=1 THEN GOTO 8676 8648 NEXT A 8652 REM END POINT OFF SCREEN CHECKING 8654 FOR A = 5 TO 8 STEP 1 8656 IF C(A)=1 THEN GOTO 8686 8658 NEXT A 8660 REM THE LINE IS ON THE SCREEN. SET P2. 8662 P2=1 8664 RETURN 8666 REM THE LINE IS OFF THE SCREEN. RESET P2. 8668 P2=0 8670 RETURN 8674 REM CLIP THE START POINT 8676 A=1 8678 B=5 8680 GOTO 8694 8684 REM CLIP THE END POINT 8686 A=5 8688 B=1 8692 REM CLIP. PUSH POINT XYZ A TOWARD XYZ B. 8694 IF C(A)=1 THEN GOTO 8728 8696 IF C(A+1)=1 THEN GOTO 8714 8698 IF C(A+2)=1 THEN GOTO 8742 8700 IF C(A+3)=1 THEN GOTO 8756 8704 REM THE LINE IS ON THE SCREEN. 8706 GOTO 8662 8712 REM PUSH XYZ A LEFT 8714 K=(Z(A)-X(A))/(X(B)-X(A)-Z(B)+Z(A)) 8716 X(A)=K\*(Z(B)-Z(A))+Z(A) 8718 Y(A)=K\*(Y(B)-Y(A))+Y(A) 8720 Z(A)=X(A) 8722 GOTO 8608 8724 REM 8726 REM PUSH XYZ A RIGHT 8728 K=(Z(A)+X(A))/(X(A)-X(B)-Z(B)+Z(A)) 8730 X(A)=K\*(Z(A)-Z(B))-Z(A) 8732 Y(A)=K\*(Y(B)-Y(A))+Y(A) 8734 Z(A)=-X(A) 8736 GOTO 8608 8740 REM PUSH XYZ A UP 8742 K=(Z(A)+Y(A))/(Y(A)-Y(B)-Z(B)+Z(A)) 8744 X(A)=K\*(X(B)-X(A))+X(A) 8746 Y(A)=K\*(Z(A)-Z(B))-Z(A) 8748 Z(A)=-Y(A) 8750 GOTO 8608 8754 REM PUSH XYZ A DOWN 8756 K=(Z(A)-Y(A))/(Y(B)-Y(A)-Z(B)+Z(A)) 8758 X(A)=K\*(X(B)-X(A))+X(A) 8760 Y(A)=K\*(Z(B)-Z(A))+Z(A) 8762 Z(A)=Y(A) 8764 GOTO 8608  8800 REM PERSPECTIVE PROJECTION - - - - - - - - - - - - - - - 8805 REM THE LINE IS NOT PROJECTED IF P2=0 8810 REM IF P2=1 THE LINE IS CHECKED FOR A 8815 REM PYRAMID BASE CRASH (X=Y=Z=0) 8820 REM AND IS THEN PROJECTED 8830 IF P2=0 THEN RETURN 8840 IF Z(1)=0 THEN GOTO 8890 8845 IF Z(5)=0 THEN GOTO 8890 8855 X(2)=X(1)/Z(1)\*W 8860 Y(2)=Y(1)/Z(1)\*W 8865 X(4)=X(5)/Z(5)\*W 8870 Y(4)=Y(5)/Z(5)\*W 8875 RETURN 8890 PRINT "LINE CRASHED AT PYRAMID’S BASE" 8893 PRINT X(1);Y(1);Z(1);" ";X(5);Y(5);Z(5) 8895 P2=0 8896 RETURN  8900 REM LINE GENERATOR - - - - - - - - - - - - - - - 8903 REM A SUM TRACKING ALGORITHM IS USED TO 8906 REM GENERATE ALL PIXELS BETWEEN 2 LINES. 8909 X(2)=INT (X(2)+.5) 8912 Y(2)=INT (Y(2)+.5) 8915 X(4)=INT (X(4)+.5) 8918 Y(4)=INT (Y(4)+.5) 8921 S=0 8924 M=1 8927 N=1 8930 D=X(4)-X(2) 8933 IF D<0 THEN M=-1 8936 IF D<0 THEN D=-D 8939 IF D=0 THEN S=-1 8942 E=Y(4)-Y(2) 8945 IF E<0 THEN N=-1 8948 IF E<0 THEN E=-E 8951 PRINT "PIXEL= ";X(2);Y(2) 8954 IF X(2)=X(4) THEN GOTO 8978 8957 IF S<0 THEN GOTO 8969 8960 X(2)=X(2)+M 8963 S=S-E 8966 GOTO 8951 8969 Y(2)=Y(2)+N 8972 S=S+D 8975 GOTO 8951 8978 IF Y(2)=Y(4) THEN RETURN 8981 GOTO 8957 8990 REM PROGRAM END - - - - - - - - - - - - - - -   |  |  |  | | --- | --- | --- | |  |  |  | |  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_14.gif |  | |  |  |  |  |  | | --- | | Large view of 3D transform and projection decorating article. | |

This program is very modular and permits easy interfacing to any user’s system. The input and output handlers simply ask the user for input parameters (3D start and end points of lines and the viewer’s 3D position) and writes the resulting 2D output display point values on the console terminal.

This simple I/O protocol allows the user to first get the program up and tested without a graphics display device. The user can then substitute his own I/O handlers (blocks 8100 and 8300) to load display files (arrays of many lines), format the output, and send it to his ’Polymorphics,’ ’Dazzler,’ ’SWTP’ or ’Matrox’ display (to name just a few). The USER, POKE and PEEK functions available with most BASICs will be quite useful here.

Fig. 12 shows a sample run. All input parameters are floating point. Although a viewer’s angle of 1.0 is said to be normal, .6 or .7 looks much more natural.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | http://www.sublogiccorp.com/imgArticle/page500/Page500_3_13.gif |  |
|  |  |  |

|  |
| --- |
| Fig. 12. A sample run of 3DGKB.V01 |

A unique line generator is included in the 8900 block. Only twenty-one statements are required to generate all the points on a line. No trig functions, multiplications or divisions are needed. The Graphics One photographs of Figs. 1 and 10 were generated using an assembly language version of this routine.

Complete Three-Dimensional Micrographics Packages in BASIC for slow graphics and 6800 assembly language for high speed applications are available from Sublogic, PO Box 3442, Culver City CA 90230, for $22 and $28 respectively. Both packages include valuable interface and applications information. The 2900 Iine 6800 version includes a "Kansas City standard" tape for easy entry.

Program A. 3DGKBV01 Listing.

8000 REM 3D GRAPHICS VERSION 3DGKB.V01 5/27/77  
8002 REM AUTHOR- B. ARTWICK FOR SUBLOGIC CO.  
8004 REM PROGRAM USE-SLOW SPEED 3D GRAPHICS FOR  
8006 REM EXPERIMENTATION AND FAMILIARIZATION.  
8008 REM LANGUAGE- MINIMAL SET BASIC (SWTP 4K TYPE)  
8010 REM MEMORY REQUIRED- 11K INCLUDING 4K FOR BASIC.  
8012 REM FEATURES- PERSPECTIVE PROJECTION, CLIPPING,  
8014 REM VARIABLE VIEWING WINDOW, LINE GENERATOR,  
8016 REM COORDINATE TRANSLATION.  
8018 REM OPTIONS AVAILABLE FROM SUBLOGIC- OPTIMIZED  
8020 REM CLIPPING, COORDINATE ROTATION, ARRAY  
8022 REM HANDLING, 6800 ASSEMBLY VERSIONS FOR  
8024 REM SERIOUS DYNAMIC APPLICATIONS.  
8026 REM EQUIPMENT NEEDED- A CPU AND CONSOLE TERMINAL  
8028 REM ARE ENOUGH TO GET STARTED. A GRAPHICS  
8030 REM DISPLAY DEVICE IS HIGHLY DESIRABLE.  
8034 REM VARIABLE DEFINITION - - - - - - - - - - - - - - -  
8036 REM X,Y,Z ARRAYS 1 THROUGH 5 ARE FOR:  
8038 REM 1= 3D START PT. 2= 2D START PT. (NO Z)  
8040 REM 3= VIEWER’S TRANSLATIONAL POSITION  
8042 REM 4= 2D END PT. 5= 3D END PT.  
8044 REM C= CLIPPING CODE ARRAY  
8046 REM T= TRANSFORMATION MATRIX ARRAY 8048  
REM P,B AND H= PITCH, BANK AND HEADING  
8050 REM P2= PROJECTION FLAG  
8052 REM W,V= WINDOW AND FIELD OF VIEW  
8053 REM A,B,D,E,G,H,M,N,S= TEMPORARY STORAGE

8054 DIM X(5),Y(5),Z(5),C(8),T(16)  
8100 REM INPUT HANDLER - - - - - - - - - - - - - - -  
8105 REM ALL VARIABLES NEEDED TO TRANSFORM, CLIP  
8110 REM AND PROJECT ONE LINE ARE ASKED FOR.  
8120 PRINT "SCREEN WIDTH? (W)"  
8125 INPUT W  
8126 W=W/2  
8130 PRINT "FIELD OF VIEW? (V)"  
8135 INPUT V  
8140 PRINT "VIEWER’S LOCATION? (X,Y,Z)"  
8145 INPUT X(3),Y(3),Z(3)  
8150 PRINT "VIEWER’S ANGLE? (P,B,H)"  
8155 INPUT P,B,H  
8160 REM CREATE THE TRANSFORMATION MATRIX  
8165 GOSUB 8442  
8170 PRINT "START PT. COORDINATE? (X,Y,Z)"  
8175 INPUT X(1),Y(1),Z(1)  
8180 PRINT "END PT. COORDINATE? (X,Y,Z)"  
8185 INPUT X(5),Y(5),Z(5)  
8200 REM EXECUTION CONTROL - - - - - - - - - - - - - - -  
8203 REM A LINE IN 3D SPACE IS TRANSFORMED TO  
8206 REM A 2D SCREEN LINE. THE INPUT HANDLER HAS  
8209 REM ALREADY BUILT THE TRANSFORMATION MATRIX  
8218 REM PERFORM TRANSLATION,ROTATION AND  
8219 REM VIEWING WINDOW CALCULATIONS  
8221 GOSUB 8530  
8224 REM CLIP THE LINE  
8227 GOSUB 8608  
8230 REM PROJECT THE LINE  
8233 GOSUB 8830  
8236 REM DISPLAY THE RESULTS  
8239 GOSUB 8330  
8242 REM ASK FOR ANOTHER LOOP, NEW VIEW OR STOP  
8245 PRINT "NEW LINE=1, NEW VIEW=2, STOP=3"  
8248 INPUT A  
8257 ON A GOTO 8170,8140,8260  
8260 END

8300 REM OUTPUT HANDLER - - - - - - - - - - - - - - -  
8305 REM THE SCREEN COORDINATES OR "LINE OFF  
8310 REM THE SCREEN" MESSAGE IS PRINTED.  
8315 REM THE USER CAN REPLACE STATEMENT 8345 WITH  
8320 REM "RETURN" IF PIXELS ARE NOT NEEDED  
8330 IF P2=0 THEN GOTO 8355  
8335 PRINT "THE SCREEN LINE IS:"  
8340 PRINT X(2);", ";Y(2);" TO ";X(4);", ";Y(4)  
8345 GOSUB 8909  
8350 RETURN  
8355 PRINT "THE LINE IS OFF THE SCREEN"  
8360 RETURN

8400 REM TRANSFORMATION MATRIX GENERATOR   
8403 REM NORMALLY THE FULL TRANSFORMATION MATRIX  
8406 REM WOULD BE CREATED. TO SAVE MEMORY, ONLY  
8409 REM TWO VIEWS ARE POSSIBLE WITH THIS VERSION:  
8412 REM FRONT AND REAR VIEWS.  
8414 REM BASIC MUST INITIALIZE T MATRIX TO ZERO.  
8442 PRINT "FRONT VIEW=1, REAR VIEW=2"  
8445 INPUT A  
8448 IF A=2 THEN GOTO 8480  
8451 REM FRONT VIEW MATRIX  
8454 T(1)=1  
8457 T(6)=1  
8460 T(11)=V  
8463 T(13)=-X(3)  
8466 T(14)=-Y(3)  
8469 T(15)=-Z(3)\*V  
8472 T(16)=1  
8475 RETURN  
8478 REM REAR VIEW MATRIX  
8480 T(1)=-1  
8482 T(6)=1  
8484 T(11)=-V  
8486 T(13)=X(3)  
8488 T(14)=-Y(3)  
8490 T(15)=Z(3)\*V  
8492 T(16)=1  
8494 RETURN

8500 REM MATRIX MULTIPLIER - - - - - - - - - - - - - - -  
8505 REM THE TWO INPUT VECTORS (START AND END  
8510 REM POINTS) ARE MULTIPLIED BY THE  
8515 REM TRANSFORMATION MATRIX "T".  
8520 REM [X,Y,Z,1] T1 T2 T3 T4 [X’,Y’,Z’,1]  
8521 REM . T5 T6 T7 T8  
8522 REM . T9 T10 T11 T12  
8523 REM . T13 T14 T15 T16  
8530 FOR A=1 TO 5 STEP 4  
8535 G=X(A)  
8540 H=Y(A)  
8545 X(A)=G\*T(1)+H\*T(5)+Z(A)\*T(9)+T(13)  
8550 Y(A)=G\*T(2)+H\*T(6)+Z(A)\*T(10)+T(14)  
8555 Z(A)=G\*T(3)+H\*T(7)+Z(A)\*T(11)+T(15)  
8560 NEXT A  
8565 RETURN

8600 REM LINE CLIPPER - - - - - - - - - - - - - - -  
8601 REM LINES ARE CODED WITH RELATION TO THE  
8602 REM VIEWING PYRAMID AND CLIPPED IF  
8603 REM NEEDED. THE PROJECTION CODE IS SET.  
8606 REM CODE THE START AND END POINT.  
8608 FOR A=1 TO 5 STEP 4  
8610 C(A)=0  
8612 C(A+1)=0  
8614 C(A+2)=0  
8616 C(A+3)=0  
8618 IF X(A) < -Z(A) THEN C(A)=1  
8620 IF X(A) > Z(A) THEN C(A+1)=1  
8622 IF Y(A) < -Z(A) THEN C(A+2)=1  
8624 IF Y(A) > Z(A) THEN C(A+3)=1  
8626 NEXT A  
8630 REM OFF SCREEN LINE CHECKING  
8632 FOR A=1 TO 4 STEP 1  
8634 IF C(A)=0 THEN GO TO 8638  
8636 IF C(A)=C(A+4) THEN GOTO 8668  
8638 NEXT A  
8642 REM START POINT OFF SCREEN CHECKING  
8644 FOR A=1 TO 4 STEP 1  
8646 IF C(A)=1 THEN GOTO 8676  
8648 NEXT A  
8652 REM END POINT OFF SCREEN CHECKING  
8654 FOR A = 5 TO 8 STEP 1  
8656 IF C(A)=1 THEN GOTO 8686  
8658 NEXT A  
8660 REM THE LINE IS ON THE SCREEN. SET P2.  
8662 P2=1  
8664 RETURN  
8666 REM THE LINE IS OFF THE SCREEN. RESET P2.  
8668 P2=0  
8670 RETURN  
8674 REM CLIP THE START POINT  
8676 A=1  
8678 B=5  
8680 GOTO 8694  
8684 REM CLIP THE END POINT  
8686 A=5  
8688 B=1  
8692 REM CLIP. PUSH POINT XYZ A TOWARD XYZ B.  
8694 IF C(A)=1 THEN GOTO 8728  
8696 IF C(A+1)=1 THEN GOTO 8714  
8698 IF C(A+2)=1 THEN GOTO 8742  
8700 IF C(A+3)=1 THEN GOTO 8756  
8704 REM THE LINE IS ON THE SCREEN.  
8706 GOTO 8662  
8712 REM PUSH XYZ A LEFT  
8714 K=(Z(A)-X(A))/(X(B)-X(A)-Z(B)+Z(A))  
8716 X(A)=K\*(Z(B)-Z(A))+Z(A)  
8718 Y(A)=K\*(Y(B)-Y(A))+Y(A)  
8720 Z(A)=X(A)  
8722 GOTO 8608  
8724 REM  
8726 REM PUSH XYZ A RIGHT  
8728 K=(Z(A)+X(A))/(X(A)-X(B)-Z(B)+Z(A))  
8730 X(A)=K\*(Z(A)-Z(B))-Z(A)  
8732 Y(A)=K\*(Y(B)-Y(A))+Y(A)  
8734 Z(A)=-X(A)  
8736 GOTO 8608  
8740 REM PUSH XYZ A UP  
8742 K=(Z(A)+Y(A))/(Y(A)-Y(B)-Z(B)+Z(A))  
8744 X(A)=K\*(X(B)-X(A))+X(A)  
8746 Y(A)=K\*(Z(A)-Z(B))-Z(A)  
8748 Z(A)=-Y(A)  
8750 GOTO 8608  
8754 REM PUSH XYZ A DOWN  
8756 K=(Z(A)-Y(A))/(Y(B)-Y(A)-Z(B)+Z(A))  
8758 X(A)=K\*(X(B)-X(A))+X(A)  
8760 Y(A)=K\*(Z(B)-Z(A))+Z(A)  
8762 Z(A)=Y(A)  
8764 GOTO 8608

8800 REM PERSPECTIVE PROJECTION - - - - - - - - - - - - - - -  
8805 REM THE LINE IS NOT PROJECTED IF P2=0  
8810 REM IF P2=1 THE LINE IS CHECKED FOR A  
8815 REM PYRAMID BASE CRASH (X=Y=Z=0)  
8820 REM AND IS THEN PROJECTED  
8830 IF P2=0 THEN RETURN  
8840 IF Z(1)=0 THEN GOTO 8890  
8845 IF Z(5)=0 THEN GOTO 8890  
8855 X(2)=X(1)/Z(1)\*W  
8860 Y(2)=Y(1)/Z(1)\*W  
8865 X(4)=X(5)/Z(5)\*W  
8870 Y(4)=Y(5)/Z(5)\*W  
8875 RETURN  
8890 PRINT "LINE CRASHED AT PYRAMID’S BASE"  
8893 PRINT X(1);Y(1);Z(1);" ";X(5);Y(5);Z(5)  
8895 P2=0  
8896 RETURN

8900 REM LINE GENERATOR - - - - - - - - - - - - - - -  
8903 REM A SUM TRACKING ALGORITHM IS USED TO  
8906 REM GENERATE ALL PIXELS BETWEEN 2 LINES.  
8909 X(2)=INT (X(2)+.5)  
8912 Y(2)=INT (Y(2)+.5)  
8915 X(4)=INT (X(4)+.5)  
8918 Y(4)=INT (Y(4)+.5)  
8921 S=0  
8924 M=1  
8927 N=1  
8930 D=X(4)-X(2)  
8933 IF D<0 THEN M=-1  
8936 IF D<0 THEN D=-D  
8939 IF D=0 THEN S=-1  
8942 E=Y(4)-Y(2)  
8945 IF E<0 THEN N=-1  
8948 IF E<0 THEN E=-E  
8951 PRINT "PIXEL= ";X(2);Y(2)  
8954 IF X(2)=X(4) THEN GOTO 8978  
8957 IF S<0 THEN GOTO 8969  
8960 X(2)=X(2)+M  
8963 S=S-E  
8966 GOTO 8951  
8969 Y(2)=Y(2)+N  
8972 S=S+D  
8975 GOTO 8951  
8978 IF Y(2)=Y(4) THEN RETURN  
8981 GOTO 8957  
8990 REM PROGRAM END - - - - - - - - - - - - - - -

|  |  |
| --- | --- |
| http://www.sublogiccorp.com/imgArticle/page500/Page500_3_14.gif |  |
|  |  |  |

|  |
| --- |
| Large view of 3D transform and projection decorating article |