Micro computer animation

Peter Sterling Franks

Follow this and additional works at: https://scholarworks.rit.edu/theses

Recommended Citation
MICRO COMPUTER ANIMATION

BY PETER STERLING FRANKS

MAY 27, 1986
Approvals

Chief Advisor: Prof. James Ver Hague

James Ver Hague Date: 10/30/86

Associate Advisor: Prof. Robert Keough

Robert Keough Date: 11-11-86

Associate Advisor: Prof. Guy Johnson

Guy Johnson Date: 11-21-86

Special Assistant to the
Dean for Graduate Affairs: Prof. Philip Bornarth

Philip Bornarth Date: 12/10/86

Dean, College of
Fine and Applied Arts: Dr. Robert Johnston

Dr. Robert H. Johnston Ph.D. Date: 12/15/86

I, Peter Franks, insist to be contacted each time a request for publication is made.

I can be reached at the following address:

Peter Franks

Date:———
Chief Advisor: James Ver Hague
Associate Advisor: Robert Keough
Associate Advisor: Guy Johnson
Acknowledgments

Special thanks should go to Danny LePage for his time and effort in providing critical information for the completion of my thesis. I would like to thank Jim Ver Hague, Robert Keough and Guy Johnson for their patience and support throughout this entire project.
# Table of Contents

Thesis Committee ................................................................. i
Acknowledgments ................................................................. ii
Table of Contents ............................................................... iii
List of Illustrations ............................................................... iv
List of Appendices ............................................................... v
Introduction ................................................................. vi
Technical Research .............................................................. 1
   A. Programming Structure and Sprites .......................... 1
   B. Programming Languages ............................................. 4
   C. Complications of Artronics ........................................... 5
Adapting Halo for Artronics ................................................. 7
Creating an Animation Program for the Artronics ................. 11
Documentation ..................................................................... 15
   A. Booting System and Software ..................................... 15
   B. BASIC Editor ............................................................ 16
   C. Development of Program ............................................. 17
   D. Animation .............................................................. 21
Conclusion ..................................................................... 22
Illustrations ..................................................................... 25
Bibliography ..................................................................... 28
Appendix A. Artronics Sync Parameters
Appendix B. LineTest Program
Appendix C. Line Animation Program
Appendix D. Test Animation Program
Appendix E. Super Animation Program (SAP)
Appendix F. Offset for Routines
List of Illustrations

1. Superline1 .................................................. 25
2. Superline2 .................................................. 25
3. Superline3 .................................................. 26
4. Omniline ................................................... 26
5. Omniline1 .................................................. 27
List of Appendices

Appendix A—Artronics Sync Parameters
Appendix B—Linetest Program
Appendix C—Line Animation Program
Appendix D—Test Animation Program
Appendix E—Super Animation Program (SAP)
Appendix F—Offset for Routines
Introduction

The purpose of my thesis is to investigate the possibilities of creating a two-dimensional and a three-dimensional computer animation program for the Artronics Computer Graphic System. This thesis will develop the ground work for future development in bit map computer graphics at RIT, enabling the designer to have much more flexibility in creating imagery through programming. The thesis may be described as a technical thesis, but the graphic designer is the most important benefactor.

The two basic types of computer animation are raster or bit map graphics and vector graphics. Vector graphics is the most widely used type of computer animation used in the commercial industry. The problem with vector graphics is hardware. Vector computer graphics hardware is very expensive, but requires no more extra database than raster graphics. The effects of vector graphics can be spectacular. Raster graphics was introduced in the mid-seventies. The advantage of raster graphics is the refresh buffers which enable an image to be handled with relatively inexpensive hardware, and the use of standard television technology. The Artronics Computer Graphics System is a raster computer graphics system and is a fairly inexpensive system.

The Artronics Computer Graphic system has been developed around the IBM personal computer. This particular system is based on the Leading Edge personal computer, which is comparable to the IBM PC. My thesis develops around the hypothesis that I should be able to combine the programming of the Leading Edge and the PCPAINT program of Artronics. This hypothesis turned into a very complicated situation which took over
five months of technical research. There were two critical aspects to the entire project. The first was the ability to access 256 colors in the PCPAINT program of the Artronics through programming. Most personal computers only have a maximum of 32 colors. The ability to provide 256 colors to a designer enables the designer to have a much wider variety of possibilities. The other crucial aspect is the possibility of realtime animation or frame-by-frame animation. Frame-by-frame animation is not particularly difficult to create until the exclusive (XOR) function is implemented. XOR is the fundamental concept in the development of bit map animation. The XOR or complement mode is useful when wishing to draw an image temporarily, and removing the image leaving the original image. This can be accomplished by writing the image twice in complement mode, since the complement of the complement of a color is the original color. The XOR function is extremely important for realtime animation and will be the determining factor for animation on the Artronics. The idea is to create an image with the Artronics PCPAINT program and then animate the object over a background created in the PCPAINT program.

There is a great deal of technical research and development which must be investigated. I will provide a fundamental foundation for the further development of bit map graphics. My thesis is meant to encourage designers to realize the possibilities of bit graphics, and to demonstrate that programming can be and should be a major consideration in the education of the computer graphic designer.
Technical Research

A. Programming Structures and Sprites

It is extremely difficult to develop an animation software package from scratch. Most companies or educational institutions have teams of programmers that spend long periods of time on the development of just one software package. The time constraints for the completion of this thesis made it extremely difficult for one person to create a software package. The logical alternative was to begin my investigation with existing software.

It was necessary to choose a starting point which would provide a solid foundation for the research. The Apple IIe supports an animation software package called TAKE 1. This software enables a designer to create two dimensional animation. The program is fairly well organized and easy to learn. The program seemed ideal for the Artronics.

The one main feature of the TAKE 1 animation package is that it uses a computer graphic function called exclusive OR. This function enables the designer to define an area around an object. Once the object has been defined, it turns the area around the object into a transparent overlay. The animation effect is then achieved by placing and moving the object over a background. It is a marvelous function which allows a designer to create many types of two-dimensional animation.

The TAKE 1 program has another strength besides exclusive OR. The program menus are straightforward and readable. This makes it easy for the designer to flip through the menus and run the program. The program enables the designer to create actors, backgrounds, screens and movies fairly quickly. I believe the TAKE 1 program
would be a very impressive software package if it could run on the Artronics.

There are a couple of drawbacks with the TAKE 1 program, which could be resolved on the Artronics. The first drawback is the number of colors. The Apple TAKE 1 program provides the designer with only sixteen colors. This is very limited compared to the 256 colors of the Artronics. The other problem with the Apple is low resolution. The Apple IIe has a resolution of 280 x 192 whereas the Artronics has a resolution of 512 x 480. The combination of greater color capabilities and resolution on the Artronics would provide the designer with a tool to create complicated images and animation.

Most computer animation software packages are written in either "C" or assembly language. C programming language is relatively new to computer graphics, but it is becoming the language of computer graphics. Assembly language is often used in writing computer programs. This enables the programmer to calculate large and complex mathematical routines in the matter of a split second. Assembly language enables a program to be quick and efficient. The problem with assembly language is the code can become extremely long. This code makes it difficult for an inexperienced computer programmer to debug. Time contraints dictated that the time be spent in a more productive manner. I decided to try and locate some essential parts in the program. There are subroutines in TAKE 1 which define an area for animation and the exclusive OR of that area. TAKE 1 refers to these functions as Sprites. The Sprite subroutine could be interpreted into an understandable syntax and rewritten for the Artronics. I was unable to locate any of the subroutines for Sprites. A programmer and software company try to make it as difficult as possible for someone else to pirate their software.
Sprites are technically the area which is being XORed (exclusive OR). The process of performing the sprite function becomes much more complicated. The process involved in running the actual sprite is technically difficult on the Artronics. The subroutine code in assembly language on the Apple activates the display buffer and color register in memory. The color register and display buffer are stored in two different locations in memory. Once the information has been stored in these two locations, a “black box” is needed to execute both memory locations and plot the points on the screen. The entire process will not work without this “black box”. The “black box” acts as a device that links in memory the color register and display buffer. Therefore, not only is the sprite subroutine needed to run the program, but also the black box to display the visual information.

Sprites are very common with the Commodore Computer System. I researched the possibilities of converting the Commodore sprites into a code which can be used on the Artronics. Commodore publishes many books on how to program sprites and create animation. The problem with these sprites is that the Commodore computer is specially designed to handle the commands for sprites. Commodore does not release the information on how to structure the hardware and software to accept the commands for programming sprites. Therefore, it is not feasible to use Commodore sprite functions in creating sprites.

The original concept of developing an Artronics animation software package based on the Apple Computer and TAKE 1 proved unworkable. I concluded that it would take a computer science software programmer to develop animation using this process of converting TAKE 1 to the Artronics. The next step was to investigate the possibilities
of programming directly on the Artronics and to find which language would be most suitable for animation on the Artronics.

B. Programming Languages

The programming language is critical to the development of an animation program. The speed of drawing and redrawing an image with XOR is the significant factor for realtime animation. Realtime animation becomes increasingly difficult when combining hardware and software capabilities. The next step was to find software that is compatible with the Leading Edge computer. The Artronics has a Number Nine Computer board built into the Leading Edge. A programming language that is most compatible with the Number Nine board had to be chosen. The options are assembly language, BASIC or C. They all have strengths and weaknesses when conforming to the Leading Edge.

The same difficulties exist with assembly language as encountered with debugging the TAKE 1 animation program. It would take some time to learn and write assembly programs. Assembly would also be difficult to compile and run with the Artronics. The most exciting possibility of assembly is the speed at which the program could run. This is possible due to the ability of assembly commands to pick the functions right off of the Number Nine board and would probably give us realtime animation.

C is the most popular computer graphics language being used today. C is a fairly easy programming language to learn and has possibilities of producing realtime animation. The problem with C from a programming aspect is that it allows for sloppy programming and difficult debugging. C is the language which must be used in future pro-
programming on the Artronics. Danney LePage has developed a C editor and compiler for the Artronics. It is fairly easy to write, compile and run programs. One drawback with the package is that the programmer must write out complete routines to perform functions such as drawing a line. These extra calculations waste valuable time for the purpose of animation especially when the functions become complex.

The final possibility is BASIC. The Leading Edge computer provides a basic package to program the Artronics. It is a standard BASIC editor and compiler. The major problem with this BASIC package is the same problem with C. The programmer must write out the entire routine to perform a function. Since we are dealing with BASIC, extremely slow graphics are produced. BASIC is a three step compiling process before graphics can be drawn on the screen. The BASIC program must first be compiled by the BASIC compiler, then transferred into assembly code, and finally the routines have to be accessed off of the Number Nine board. In addition, the program requires functions to be written out as a procedure. BASIC is not particularly well suited for animation, but BASIC is the ideal computer language for beginning programmers to learn the fundamentals of computer graphics programming.

C. Complications of Artronics

The final decision on the programming language depended on the internal structure of the Artronics. A former student was able to program on the Artronics. He was able to draw a line and change the color intensity of the palette. The only way this student was able to control the color palette and draw a line was with the help of Artronics,
Inc. Artronics gave this student the memory locations of the screen coordinates and the color palette in assembly code. This meant that his BASIC program was poking the locations on the board in creating imagery. This information implied that all the graphic routines must be on the board and all I needed to do was access these commands.

The Artronics runs off of an eight bit, gen lock, banked Number Nine board. This board has a capability of 256 displayable colors and runs on a 18.432 megahertz signal. This information became extremely important as the investigation developed. The next step was to contact the makers of the graphics board, the Number Nine Computer Company. I was hoping that Number Nine would provide valuable information as far as accessing routines off of the board. Number Nine would not release any information, but they suggested I speak with Media Cybernetics Corporation. Media Cybernetics creates software which is designed to interact with the Number Nine board.

Media Cybernetics makes a software package called Halo. Halo allows a programmer to program directly to a Number Nine board. The software package enables a programmer to use C or BASIC. The cost of the software is $300. The software package is supplied with one programming language and each additional programming language cost $150. Jim VerHague and I decided to purchase the BASIC package. For the time being, BASIC would have much more value to the Computer Graphics Design Programmer. The amount of speed lost from not using C would be offset in the educational value of BASIC to beginning programmers. This package is much quicker than the Leading Edge BASIC package and should provide exciting computer animation. The next step to investigate was the capabilities of the Number Nine board of the Artronics with Halo.
Adapting Halo for Artronics

The Halo software is rich with various types of features. For example, Halo allows a programmer to interface with a wide variety of input and output devices. Halo can provide both bitmapped and stroke lines, circles and text. This software allows the programmer to construct menus through window management. The commands are very simple to use in creating graphics. Halo provides a Learnhalo program which instructs the programmer on how to use the features of Halo. Unfortunately, the Learnhalo program is not presently compatible with the Artronics.

The Learnhalo program was loaded. The computer shut down when it attempted to run. This failure was definitely a serious problem. I decided to call Media Cybernetics, Artronics, and Number Nine for information. Each insisted the problem was with the other company. I finally deduced that the software was not compatible with the Number Nine board. This problem meant that the software had to be modified to conform with the board. This change led to the first of many major breakthroughs.

Meanwhile, another student happened to be working on direct-to-video capabilities for the Artronics. He was adapting a Lenco Sync Generator and a Color Encoder to the Artronics. This hardware was not compatible with the normal Number Nine board. The normal board does not have the video functions to enable the syncing of all the hardware. Artronics provided RIT with a new Number Nine board with the necessary video commands. The new board was installed and conformed to the Artronics hardware. While he was working on the direct-to-video, I was unsuccessfully trying to run the Learnhalo program on other hardware. As a last resort, I decided to try and run Learnhalo on
the new video Number Nine board. The program worked to a degree. There was an image on the screen, but it was completely distorted. The difference between the new and old board was the megahertz signal. The regular board runs on a 20 megahertz signal while the new board runs on a 18.432 megahertz signal.

Next, I concentrated on why the screen was being completely distorted. The image drawn by Learnhalo was flickering and being stretched across the screen on a diagonal. This seemed to be another serious problem. I consulted Artronics, Media Cybernetics, and the Number Nine Company. They were of no help and passed the blame on the other company. I then decided to get some outside consultation from a software programming specialist. Mr. Danny LePage is a Masters student at RIT in Computer Science. Danny has the knowledge of computer hardware and software to figure out almost any problem. Danny realized the problem immediately. The scan rate of the monitor was not in sync with the device driver and the Number Nine board. This problem could be solved if Artronics and Media Cybernetics cooperate with one another. First, I had to call Artronics and find out the sync parameters for the Artronics. After a struggle, Artronics finally released the parameters (See Appendix A). Next, Media Cybernetics had to reveal the location of the parameters in the device driver. The company refused to release this information. Obviously, Media Cybernetics did not want Danny or I to reprogram their device driver files. This meant that Danny had to take an extremely long shot to try to locate the parameters on the device driver. The Debug program of Leading Edge was used to break into the device driver program. Again, we seemed to get a lucky break. Danny was able to find the location of the parameters within an hour. Danny replaced the old
sync parameters with the new set of sync parameters given to us from Artronics. We were anxiously anticipating the outcome of the new sync parameters in the reconstructed device driver. We loaded and ran the Learnhalo program. Learnhalo ran without a hitch, drawing lines, circles and boxes in full color! This breakthrough was a major accomplishment and gave renewed hopes for the success of this project.

The next step was to create a BASIC program and then run the program. The Halo software comes with five disks. Each disk is critical in developing a working software package. The five disks are BASIC Programming disk, Learnhalo disk, a Locator disk, a Device Driver disk, and a Printer disk. Files from each of these disks must be put on to one disk. These files will be the basis for the software package. The programmer must figure out the device which is going to be programmed. The Artronics needs the device driver to access the Number Nine board. The file name is Halonine.dev and requires a drawing mode of thirteen. The locator must then be determined. The locators will activate all the different types of hardware being used by the system such as the tablet, pen and keyboard. Next, any printer output devices should be loaded onto the disk of the software package. There are many types of output devices provided which are optional to the programmer. The final files which must be loaded are the BASIC libraries and execution files. This BASIC software is an interpreted BASIC package. All files relating to the interpreted BASIC package must be loaded onto the software. This transformation input completes the transferring of all files for the software package. There is only one set of files left which must be transferred to the software package. The Halo software does not provide a BASIC editor. The Leading Edge BASIC Editor files must
be copied to the software package. It would be very convenient for the Artronics PCPAINT program to be on the software, but the disk does not have enough room. Now the Halo software and the Leading Edge BASIC Editor have been converted to one disk, which will be referred to as BasArt.

BasArt concluded five months of research and development in structuring a workable BASIC programing package for the Artronics. It would be simple to convert this BASIC software to a useable C software package. I find it amazing to realize the complexity involved in developing a software package for not only the Artronics, but any type of computer. The Artronics was finally ready to receive and run BASIC programs and eventually the Super Animation Program (SAP).
Creating an Animation Program for the Artronics

The Leading Edge BASIC editor has the same capabilities as any of the standard BASIC editors. There are no unusual differences between writing a Halo BASIC program and a standard BASIC program. It is important to plan an efficient BASIC program which will enable the program to run much smoother and create a quicker animation. The first test for animation was to see how fast a line could be drawn on the screen. The first program I wrote was a short linetest program (See Appendix B). This program will draw a line from one point to another. The speed of drawing was excitingly quick which led to another short line animation program. This program was developed to test the speed of drawing a large amount of lines and the color capabilities (See Appendix C). The speed again was very quick and gives the perception of realtime animation (See Figures 1, 2, 3). The color capability and control is wonderful. The programmer can access any of the 256 colors and even create their own color palette.

The next test program written was an animation of polygons transforming from one into another. This test program was to see the quickness of animating polygons and the ability to change colors. The actual animation decreased in speed, but was nothing to be concerned about for the time being. The test programs gave enough useful information to begin work on a major animation program in BASIC. The next program was Super Animation Program (SAP).

The program which I wrote combined a wide variety of operations. The fundamental program is a Three-Dimensional Wireframe Animation program with hidden line elimination. The program is frame-by-frame animation.
Realtime animation is impossible with BASIC. There is a strong possibility of realtime animation with C programming language. The three-dimensional animation program provides myself and the first year computer graphics students with the capabilities of animation on the Artronics. The first year majors are programming on the DEC PRO 350 with no color and a limited memory for creating objects. The Super Animation Program (SAP) will provide the students with 256 colors. Also, the students will be able to create more complicated shapes and a greater variety of objects. The program will enable me to study and create thumbnail drawings for my glass sculptures. The program will be able to XOR wireframe objects and areas on the screen. SAP will be an innovative program which will be a stepping stone to the development of bit map computer graphics on a personal computer.

The goal of the Super Animation Program is to make the program as user friendly as possible with enough flexibility for the designer to be creative. The program is developed around four major menus. A Shape Services, a Viewer Services, and Camera Services menus are included in SAP, while a fourth menu called Animation Services was created exclusively as a test menu for different types of animation (See Appendix E). Shape Services menu provides the user with such options as create shape, load shape, store shape and transform one shape to another. These options are the most difficult options to construct, but provide the most information as to the capabilities of the Artronics. The Viewer Services menu enables the user to change the viewing distance, move shape, scale shape, rotate shape, change any point to a new location, change line style, and change line width. These options all have frame-by-frame animation with straightforward and readable instructions. The Camera Services menu provides all the movements of a camera mov-
ing around an object on the screen. The camera movements have the ability to spin, pan, tilt, and zoom. All the menus have the options of changing the background and object color, clearing screen, drawing shape and the flexibility to interchange between menus.

The program provides a list of shapes already developed by the program such as a cube and cross. The program provides the user with a list of all objects in present memory. The screen can display up to five shapes at a time. Presently, the program only allows a maximum of 55 points to be displayed at one time on the screen. Every option has complete instructions and restrictions of operation. Presently, the first year graduate students are using SAP to create a three dimensional wireframe animation. The students are learning how to run SAP and are providing valuable feedback. The students are finding the program fairly easy to run.

The animation menu activates some very special options. The menu allows the user to save and restore images only created with SAP. These routines can not save and restore images created in PCPAINT. The user at this time can not store and restore color palettes. It seems Halo and PCPAINT color palettes are stored in two different locations. Another option in the Animation Menu is the exclusive OR of wireframe objects. The user is prompted to turn on and off the exclusive OR mode. There is only one problem with this feature. The object is drawn with planes and hidden line in SAP. A test is built into the program to determine whether or not a line should be drawn. The calculations of planes will cause one line to be drawn on top of another. This overlapping causes XOR to repeat the XOR mode one more time than it should, and therefore disrupts the XOR mode in animating lines. The other type of XOR is of an area. The user can define an area and move the area across the screen frame-by-frame. The XOR of an area works
in SAP over a black background. This malfunction is due to complex but solvable programming problems. One problem is the area being defined. The area must be properly defined after every frame of motion. Also, the object must be defined and redefined through the SET and RESET commands of Halo. These commands undoubtedly will slow the basic programs. C programming language would be crucial and may deliver realtime animation. Today, there are newer and faster computer chips in the marketplace such as the 8085, 8088/6, and 80186. These chips are easy to install on a board and rather inexpensive. These chips may produce realtime animation if programmed in C.

SAP allows the user to combine PCPAINT images and SAP images. It is possible to create an image in PCPAINT and then animate an object over top of the PCPAINT image. It is highly recommended to create palettes in PCPAINT before entering SAP. The user is able to specify a color by number and then draw with that color. There is a problem storing and recalling color palettes in SAP. This difficulty is a solvable problem which requires fundamental understanding of hardware and software programming.

SAP can be used for other creative arts besides animation. SAP could be used as a tool to design sculpture. I used SAP to design and study patterns which were used in designing sculpture. I was able to visually conceive line interactions within the piece, and then change a line or color in a matter of seconds. I believe this program could be a beginning of eventually combining computer graphics and other visual arts. The computer would act as a tool to the designer for visualization of a design before the actual design is created. SAP is designed to complement the designer. Animation has a definite future with the Artronics. The beauty of the package is the ease in which a program can be written.
Documentation

The following documentation will explain how to write and run a BASIC program on the Artronics. This documentation will not explain all the functions which are available in the Halo reference manual. The Halo reference manual provides information on more sophisticated programming techniques such as a complete explanation of all the functions description, usage, and synopsis; and information on adapting more input and output devices. I will document the booting of software and the software package SAP, running the basic editor, constructing a BASIC program for visual output on Artronics and the possibilities of animation.

A. Booting System and Software

The programmer must have in his possession three disks: PCPAINT Video version, BasArt or Halo software, and SAP programming disk or blank formatted disk.

There are two ways of booting the Artronics for programming. The booting of the Artronics depends on whether the programmer wants the Halo color palette or a PCPAINT color palette.

1. The first method involves the use of the Halo color palette. The computer must be activated and the BasArt or Halo software should be placed in disk drive A. The computer will respond with A] in which the user should type:

A] Halorbi (return)

This command will access the Halo library which enables the programmer to use any of the Halo functions. The com-
puter will respond with a message informing the user the library is available. The A] will return and the programmer must enter the following to activate the basic editor.

A] Basic (return)

A message will be displayed welcoming the programmer to the BASIC editor. The computer is now ready to be programmed.

2. The other way of booting the software is with an Artronics color palette. The programmer must first boot the PCPAINT program. Now, the programmer should Clearscreen and activate the proper color palette. The screen should be cleared of the menu, brush, and if desired the color palette. Next, the programmer should press the following keys at the same time:

(CTRL) (ALT) (DEL)

The computer will reboot and the A] will return. The same procedure takes place as in booting the Halo software. The programmer must replace the PCPAINT software disk with the BasArt disk. The SAP program disk or a blank disk should be placed in disk drive B. Enter the following:

A] Halorbi (return)
A] Basic (return)

This command will boot the basic editor and prepare the programmer to write his own program or run SAP.

B. BASIC Editor

The BASIC editor is very easy to use. The editor has all the standard procedures of a BASIC editor and some
extra helpful commands. The keyboard has a set of function keys on the left side numbered F1 to F10. These keys are very useful and help decrease editing time. For example, F2 is a run command which runs a program when activated. F3 is a load command and F4 is a save command. The functions are listed on the bottom of the screen with the corresponding key number.

The only problem with the editor is that a programmer can only edit one line at a time. This task becomes difficult when a small correction has to be made in a list of twenty lines and each line has to be edited individually. Another problem area may arise when storing and recalling shapes. The programmer must be sure to describe the proper disk drive. A (B:) must be used to store or recall a program in disk drive B. There is no actual compiler. The program will automatically compile and run with the RUN command. There should be no problems with the editor. There is a BASIC reference manual for the Leading Edge which will answer any questions.

C. Development of a Program Structure

Halo supplies a MEMORY.BAS program which identifies the values for the Clear and Def Seg statements to the interpreted BASIC package. These values will change depending on the hardware. The Def Seg and Clear values are needed to interface with the library. The Def Seg value is &H3800 and there is no value for the Clear statement.

The first and most important statement of the program is to define the offsets to the Halo routines. The Halo functions are all registered to specific locations in memory. These functions must be initialized before the program
can be activated. This initialization is done by placing all of the Halo functions into a subroutine in the program. (See Appendix E). The next statement must define Def Seg as &H3800. Then, the program must load the binary files for Halo and the device driver. All the statements must be activated before any type of program can be written. These statements are as follows:

10 Gosub 10000
20 Def Seg = &H3800
30 Bload "haloi.bin",0
40 Device$ = "haloine.dev"
50 Call Setdev(Device$)

This example program has now activated the functions of Halo, the binary files, and the device driver. Now, the graphics mode has to be initialized. There are two types of graphics modes. INITGRAPHICS command initializes the built-in Halo package, which clears screen to black and restores the Halo color palette. The STARTGRAPHICS initializes the graphics mode, but does not clear the screen black and leaves the present color palette. The STARTGRAPHICS command must be used when interfacing with the Artronics PCPAINT program.

60 Mode% = 0
70 Call STARTGRAPHICS(Mode%)

There is one more set of statements which must be incorporated to specify the parameters of the screen. These statements will set the viewport and window of the screen. The viewport parameters have been calculated to define the entire screen. The Border and Back variables are set to the current colors of the background and border.
The window or world coordinates are adjustable to the programmer's desire. The program will specify the center of the screen at 0,0. The maximum x and y coordinates are 201 while the minimum x and y values are -201. The reason for -201 is that the screen will not plot the last point number specified by window. The program actually plots an area of -200 to 200. Also, the YW = -151 is specified because the screen is a rectangle shape and there needs to be an adjustment to create a square image area. Otherwise, a cube would turn out looking like a rectangle.

80 xv=.02:yv=.04:x2v=-.95:y2v=.95
90 Boarder%=-1:Back%=-1
100 Call Setviewport(xv,yv,x2v,y2v)
110 xw=-210:yw=-151:x2w=201:y2w=151
120 Call Setworld(xy,yw,x2w,y2w)

The viewport, window and world features can be manipulated to the configuration of the programmer's needs. The following commands are the basic parameters which must be set for creating graphics. There is an automatic clipping of the screen. The programmer can also control the clip by using the Halo Clipping function.

The rest of the programming structure is fundamental programming logic for BASIC. It is good practice to always dimension arrays at the beginning of a program. The better organized program will run more efficiently. This organization is crucial for animation. Remark statements are also good practice and should be incorporated into a program. REM statements are not only helpful to the programmer, but to an individual debugging the program. There is sufficient memory available for storing programs and dimensioning arrays.
There is only one more area of programming structure which must be discussed. The calling of functions should not be a difficult task. The syntax of the functions are well documented in the Halo reference manual. The following code is an example of drawing a line from one point to another point.

130 rem set color to first position
135 rem
140 cl%=1
150 call Setcolor(cl%)
155 rem
156 rem calling the start point of line
157 rem
160 x1=100:y1=100
170 Call Movabs(x1,y1)
180 rem
190 x2=150:y2=150
200 call Lnabs(x2,y2)

The program draws a line from 100,100 to 150,150. The move and line functions are absolute numbers. There are a special set of routines for relative numbers. Color is called by location on the color palette up to 256. The color variables must have a % sign for the software to interpret. A list of all the functions available in Halo are listed in Appendix F. The first program has been written except for the closing statements. The program must close the graphics mode before exiting. The following statement will conclude this program.
D. Animation

There are two types of animation experimented with in SAP. The Halo software package has a Setxor command that will convert any program into an exclusive OR mode. This conversion worked fairly well as discussed in Creation of Animation Program. The other type of animation is defining an area on the screen and moving the area to another location. The process has some complications. Presently, the area being defined is stored in an array with Movefrom. The area is then saved and moved to its new position with Moveto. This process is repeated to remove the area and restore the background. Finally, the next position must be calculated and then the entire process is repeated. This repetition continues until the area has reached its final position. The area animation in SAP only works on a black background. There are some minor and major corrections needed to animate an area over a colored background. These changes have been discussed in Creation of Animation Program. This type of animation is possible and would be a wonderful future thesis.

The animation concludes a general documentation of writing a basic program for the Artronics. The only way to really understand the software package and its capabilities is to write programs and play with the BasArt software package.
Conclusion

The sole purpose of this thesis is to encourage future development not only of animation, but bit map graphics. There is no question that animation is possible on the Artronics. Frame-by-frame animation is demonstrated through the Super Animation Program (SAP). Realtime animation is possible if the correct programming language is used and the exclusive OR function is developed. The designer has the ability to create animation through programming on the Artronics. This project’s objective was not only to create an animation program, but to investigate the capabilities of bit map computer graphics. The personal computer market is an enormous growth area in the computer business. The declining cost of personal computers is providing the ability for the average household to purchase personal computers. The personal computer has more capabilities than ever before in the industry. According to my knowledge, not too many personal computers have the capability of 256 colors for two-dimensional animation. I feel that there could be a major market for individuals wanting to create this type of animation if the price is affordable. The Apple TAKE1 and other animation packages have proven to be successful in the marketplace. Vector graphics has made tremendous strides in the past ten years, but the hardware is still expensive and bulky. I think that vector graphics will be possible on a personal computer in the future. For the time being, I believe the personal computer can use bit map graphics to produce high quality and affordable computer animation.

This thesis poses many new questions and provides the basis for future research. Animation for the Artronics poses the most interesting questions. Most questions revolve around the possibilities of realtime animation. The
The most obvious question is whether or not C programming will produce realtime animation. I firmly believe that a well organized program written in C will produce realtime animation. All the problems with exclusive OR have been determined and are rectifiable. There is a limit to the amount of memory available, but the use of pointers in C would solve most of these problems. There is a problem with the location of color palettes in memory. The location of the PCPAINT color palette and Halo color palettes are in two different locations on the board. These locations become a problem when trying to load and save color palettes. There must be a way of storing color palettes through Halo by finding out the location of storing and loading routines from the Artronics Corporation. I would also suggest working with a computer scientist who specializes in software.

The technical aspect of this project should not overshadow the aesthetic values. The reason for programming the Artronics was to create more visually interesting forms which could be used by a designer. I think there are enormous possibilities with 256 colors to create beautiful controlled patterns (See figures 5,6). A designer could create an image in PCPAINT and combine that image with a personal program using BasArt. This possibility could be another thesis topic studying the combination of technical computer programming art and paint program art.

I would like to create a relationship between the crafts and computer graphics design. I believe that the computer graphics designer does not give enough thought to the applications of computer graphics in other areas of art such as the crafts. I incorporated scupltural glass and computer graphics. The general feedback was that either a person enjoyed the glass or was impressed with my computer
graphic images. A mere handful of people asked about the combination of glass and computer graphics. I therefore hope that the future will eventually draw more excitement to the combining of computer graphics and other media of art. This issue revolves around the fact that computer graphic imagery has not been accepted by the art community. Computer art is but still youthful in its stages. However, the major weakness in the industry is hardcopy output. Slide production quality is increasing with higher resolution and better film recorders. Hardcopy output is still expensive. It is difficult to match the quality of the image on a monitor to a print. Fortunately, the computer graphics industry is constantly improving hardware capabilities such as output.

I believe my thesis and other theses in the department are beginning to address these problems. I hope in the future the R.I.T.'s MFA program in Computer Graphics Design will tackle and solve some of these problems. I feel the success of this program depends on the students' willingness to solve the tough issues facing computer graphics in the future. This technical thesis is a start towards tackling the issues of bit map computer animation for the future. I truly hope that someone will follow up and expand upon my research of computer animation.
1. Superline1

2. Superline2
3. Superline3

4. Omniline
5. Omniline1
Bibliography


APPENDIX
### Sink Commands for the Artronics

<table>
<thead>
<tr>
<th>SINK</th>
<th>PARA</th>
<th>PITCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1fh</td>
<td>02h</td>
<td>40h</td>
</tr>
<tr>
<td>3ah</td>
<td>00h</td>
<td></td>
</tr>
<tr>
<td>64h</td>
<td>00h</td>
<td></td>
</tr>
<tr>
<td>0ch</td>
<td>7fh</td>
<td></td>
</tr>
<tr>
<td>03h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f0h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44h</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
10 REM
20 REM linetest
30 REM setup parameters
40 REM
50 REM GOSUB 1000
60 REM
70 DEF SEG = &H3800
80 BLOAD "B:HALO1.BIN",0
90 DEVICE$ = "B:HALONINE.DEV"
100 CALL SETDEV(DEVICE$)
110 MODE% = 0: CALL INITGRAPHICS(MODE%)
120 X = 0: YMAX = 200: XMAX = 200: Y = 0: CALL SETWORLD(X, YMAX, XMAX, Y)
180 REM
190 REM
200 INPUT "Enter the number of color - ", CL%
210 PRINT "*
220 INPUT "Enter the first point of line - ", X, Y
230 PRINT "*
240 INPUT "Enter the end point of line - ", XX, YY
250 REM
255 CALL SETCOLOR(CL%)
260 CALL MOVABS(X, Y)
270 CALL LNABS(XX, YY)
280 REM
290 INPUT "Do you want to draw another line - ", T$
300 IF T$ = "y" OR "Y" THEN GOTO 200
310 CALL CLOSEGRAPHICS
320 END
900 REM
1000 CLOSEGRAPHICS=&H2C3
1010 INITGRAPHICS=&H302
1020 LNABS=&H15
1030 MOVABS=&H23
1040 SETCOLOR=&H0
1050 SETWORLD=&H142
1060 SETDEV=&H261
1090 RETURN
APPENDIX C

Line Animation Program
linetest

setup parameters

GOSUB 1000
DEF SEG = &H3800
BLOAD "halo1.bin", 0
DEVICE$ = "halonine.dev"
CALL SETDEV(DEVICE$)
MODE% = 0: CALL INITGRAPHICS(MODE%)
X = 0: YMAX = 400: XMAX = 400: Y = 0
CALL SETWORLD(X, YMAX, XMAX, Y)

CLS: COUNT = 1
INPUT "Enter the number of color - ", CL%
PRINT "*"
INPUT "Enter the first point of line - ", X, Y
PRINT "*"
INPUT "Enter the end point of line - ", XX, YY
PRINT "*"
INPUT "Enter the x and y movement of first point - ", RX, RY

RR = 20
FOR I = 1 TO 140
CALL SETCOLOR(CL%) 
CALL MOVABS(X, Y)
CALL LNABS(XX, YY)
CL% = CL% + 1
IF CL% = 254 THEN CL% = 125
IF I < 70 THEN XX = XX + 4
IF I > 70 THEN XX = XX - 4
IF I = 140 THEN X = X + (RX * (-1))
IF I = 140 THEN Y = Y + (RY * (-1))
NEXT I
COUNT = COUNT + 1
IF COUNT < 8 THEN GOTO 210

INPUT "Do you want to draw another line - ", T$
CL% = 1
CALL SETCOLOR(CL%)
CALL CLR
IF T$ = "y" THEN GOTO 150
CALL CLOSEGRAPHICS
END

CLOSEGRAPHICS = &H2C3
CLR = &H2CA
INITGRAPHICS = &H302
LNABS = &H15
MOVABS = &H23
SETCOLOR = &H0
SETWORLD = &H142
SETDEV = &H261
RETURN
APPENDIX D

Test Animation Program
This is a test program for the purpose of finding the parameters of an animation program, and testing a small animation.

CLEAR
GOSUB 880
DEF SEG = &H3800
BLOAD "haloi.bin", 0
DEVICE$ = "halonine.dev"
CALL SETDEV(DEVICE$)
MODEZ = 0: CALL INITGRAPHICS(MODEZ)
X = 25: YMAX = 425: XMAX = 425: Y = 50
CALL SETWORLD(X, YMAX, XMAX, Y)

CLS: KEY OFF: LOCATE 10, 15
PRINT "welcome to A R T R O N I C S "
LOCATE 20, 15
PRINT "Written by Peter Franks"
FOR I = 1 TO 2000
NEXT I

CLS: LOCATE 10, 10
VV = 0
COUNT = 0
REM
INPUT "Input the number of points in shapes ", NP
DIM X1(NP), Y1(NP)
DIM X2(NP), Y2(NP)
DIM DX(NP), DY(NP)
DIM SX(NP), SY(NP)
REM
INPUT "Input the number of steps in transformation ", NS
REM
INPUT "Input the color of object by number ", CL%
REM
PRINT "Input the coordinates of the first shape"
FOR I = 1 TO NP
    INPUT "Input point ", X1(I), Y1(I)
NEXT I
REM
CLS
PRINT "Input the coordinates of the second shape"
FOR I = 1 TO NP
    INPUT "Input point ", X2(I), Y2(I)
NEXT I
REM
FOR J = 1 TO NS
    FOR K = 1 TO NP
        DX(K) = (X2(K) - X1(K)) / NS
        DY(K) = (Y2(K) - Y1(K)) / NS
        SX(K) = (X1(K) + J * DX(K))
        SY(K) = (Y1(K) + J * DY(K))
        CALL SETCOLOR(CL%)
    NEXT K
NEXT J
IF K=1 THEN GOTO 750
    CALL MOVABS(SX(K-1),SY(K-1))
    CALL LNABS(SX(K),SY(K))
    CL%=CL%+1
NEXT K
    WX=X1(1)+J*DX(1)
    WY=Y1(1)+J*DY(1)
    CALL MOVABS(SX(NP),SY(NP))
    CALL LNABS(WX,WY)
NEXT J
GOTO 860
COUNT=COUNT+1
IF COUNT=5 THEN GOTO 860
GOTO 630
CALL CLOSEGRAPHICS
END
'---------------------------------------------------------------
CLOSEGRAPHICS=&H2C3
CLR=&H2CA
INITGRAPHICS=&H302
LNABS=&H15
MOVABS=&H23
SETCOLOR=&H0
SETWORLD=&H142
SETDEV=&H261
RETURN
APPENDIX E

Super Animation Program (SAP)
This is it!! Super Animation Program

entering any code for chaining programs

setting up parameters for program

CLS:GOSUB 10000
DEF SEG = &H3800
BLOAD "haloi.bin",0
DEVICE$ = "halonine.dev"
CALL SETDEV(DEVICES$)
MODE%=0: CALL STARTGRAPHICS(MODE%)
XV=.02;YV=.04:X2V=.95;Y2V=.95
BORDER%=1;BACK%=1
CALL SETVIEWPORT(XV,YV,X2V,Y2V,BORDER%,BACK%)
XW=-201;YW=-151:X2W=201;Y2W=151
CALL SETWORLD(XW,YW,X2W,Y2W)

DIM AF(45),BF(45),CF(45)
DIM DF(45),FLAG(45),XSP(45),YSP(45)
DIM XI(45),Y1(45),Z1(45),X2(45),Y2(45),Z2(45)
DIM OBLIST(45),PL(45,45)
DIM X(45),Y(45),Z(45)
DIM XS(45),YS(45),SP(45),EP(45)
DIM DX(45),DY(45),DZ(45)
DIM XX1(45),YY1(45),ZZ1(45)
DIM XX2(45),YY2(45),ZZ2(45)
DIM CX(45),CY(45),CZ(45),BX(45),BY(45),BZ(45)
DIM ARRAY%(1253),OFFSET%(1253)
opening message to programmer before starting

CLS: LOCATE 7,20
PRINT "Three Dimensional Wireframe Animation"
LOCATE 10,20
PRINT " created by peter franks"
LOCATE 13,20
PRINT " Thesis-April 1986"
LOCATE 16,20
PRINT " S A P"
FOR I=1 TO 2000: NEXT I

first main menu

CLS: PRINT "*
PRINT -----------------------------
PRINT " Menu
PRINT -----------------------------
PRINT " Main Menu
PRINT -----------------------------
PRINT " 1) Shape Services Menu
PRINT " 2) Viewer Services Menu
PRINT " 3) Camera Services Menu
PRINT " 4) Animation Test Menu
PRINT " 5) Shapes in Storage Table
PRINT " 6) Quit
PRINT -----------------------------
LOCATE 15,1
PRINT "*
LOCATE 14,1
INPUT "Enter option by number: ", E
IF E=1 THEN GOTO 665
IF E=2 THEN GOTO 1065
IF E=3 THEN GOTO 865
IF E=4 THEN GOTO 3515
IF E=5 THEN GOTO 4015
IF E=6 THEN GOSUB 6930
GOTO 525

this the shape services menu

CLS: GOSUB 7565
LOCATE 1,1:PRINT "*
PRINT -----------------------------
PRINT " Menu
PRINT -----------------------------
PRINT " Shape Services Menu
PRINT -----------------------------
PRINT " 1) Create Shape 7) Clear Screen
PRINT " 2) Load Shape 8) Viewer Services Menu
PRINT " 3) Store Shape 9) Camera Services Menu
PRINT " 4) Transformation 10) Print Image(Ink Jet)
PRINT " 5) Color change 11) Text
PRINT " 6) Draw 12) Main Menu
PRINT -----------------------------
LOCATE 15,1
PRINT "*
LOCATE 14,1
INPUT "Enter the option by number: ", F
IF F=1 THEN GOSUB 7765
IF F=2 THEN GOTO 1265
IF F=3 THEN GOTO 1465
IF F=4 THEN GOTO 3015
IF F=5 THEN GOSUB 8015
GOTO 725
REM this is the camera services menu
CLS: GOSUB 7565
LOCATE 1, 1: PRINT "*
PRINT " ----------------------------------------
PRINT " Menu "
PRINT " ----------------------------------------
PRINT " Camera Services Menu "
PRINT " ----------------------------------------
PRINT " 1) Zoom " 6) Draw "
PRINT " 2) Tilt " 7) Shape Services Menu "
PRINT " 3) Pan " 8) Viewer Services Menu "
PRINT " 4) Spin " 9) Main Menu "
PRINT " 5) Color change 10) Quit "
PRINT " ----------------------------------------
LOCATE 14, 1
PRINT " ----------------------------------------
LOCATE 13, 1
INPUT "Enter the option by number: ", G
IF G=1 THEN GOTO 2120
IF G=2 THEN GOTO 2415
IF G=3 THEN GOTO 2515
IF G=4 THEN GOTO 2315
IF G=5 THEN GOSUB 8015
IF G=6 THEN GOSUB 7220
IF G=7 THEN GOTO 665
IF G=8 THEN GOTO 1065
IF G=9 THEN GOTO 525
IF G=10 THEN GOSUB 6930 ELSE GOTO 865

REM this is the viewer services menu
CLS: GOSUB 7565
LOCATE 1, 1: PRINT "*
PRINT " ----------------------------------------
PRINT " Menu "
PRINT " ----------------------------------------
PRINT " Viewer Services Menu "
PRINT " ----------------------------------------
PRINT " 1) Distance change 8) Print Image "
PRINT " 2) Move shape 9) Line width "
PRINT " 3) Scale shape 10) Line style "
PRINT " 4) Rotate shape 11) Camera Services Menu "
PRINT " 5) Change any pt. 12) Shape Services Menu "
PRINT " 6) Draw shape 13) Main Menu "
PRINT " 7) Clear screen 14) Change color "
PRINT " ----------------------------------------
LOCATE 16, 1
PRINT " ----------------------------------------
LOCATE 15, 1
INPUT "Enter the option by number: ", H
IF H=1 THEN GOTO 1615
IF H=2 THEN GOTO 1715
IF H=3 THEN GOTO 1815
IF H=4 THEN GOTO 2215
IF H=5 THEN GOTO 1965
IF H=6 THEN GOSUB 7220
IF H=7 THEN GOSUB 7515
IF H=8 THEN GOSUB 865
REM restoring shape
CLS: PRINT "*"
GOSUB 9015
LOCATE 2,1
INPUT "Enter the viewing distance of shape: ", D
PRINT "----"
INPUT "Enter the number of files to be restored: ", NS
PRINT "----"
PRINT "Enter filename in following form: b:filename.dat"
SPT=0: EPT=0: EPL=0: SPL=0
FOR S=1 TO NS
  INPUT "Enter shape filename: ", FILE*
  OPEN "I", #1, FILE*
  INPUT#1,NP:N=5*S
  FOR I=1 TO NP
    K=I+EPT: INPUT#1,A: INPUT#1,B: INPUT#1,C
    X(K)=A: Y(K)=B: Z(K)=C: NEXT I
  INPUT#1,NPL
  FOR I=1 TO NPL
    K=I+EPL: INPUT#1,PL(K,0)
    FOR J=1 TO PL(K,0)
      IF EOF(1) THEN GOTO 1347
      INPUT#1,E: PL(K,J)=E+EPT: NEXT J
    NEXT I
  SPT=EPT+1: EPT=EPT+NP
  SPL=EPL+1: EPL=EPL+NPL
  OBLIST(N+2)=SPT: OBLIST(N+2)=EPT
  OBLIST(N+4)=SPL: OBLIST(N+4)=EPL
  IF S=1 THEN PP*=FILE*
  IF S=2 THEN Q*=FILE*
  IF S=3 THEN R*=FILE*
  IF S=4 THEN S*=FILE*
  IF S=5 THEN T*=FILE*
  NEXT S
OBLIST(1)=1: OBLIST(2)=EPT
OBLIST(3)=1: OBLIST(4)=EPL
IF DICK=1 THEN GOTO 3050
IF DICK=2 THEN GOTO 3085
GOSUB 8017
GOTO 665
 FOR I=1 TO NFL
 PRINT#1,PL(I,0)
 FOR J=1 TO PL(I,0)
 E=PL(I,J):PRINT#1,E:NEXT J
 NEXT I
 CLOSE#1: GOTO 665

REM change distance
CLS
PRINT "*
PRINT "Changing Viewing Distance"
PRINT "-------"
GOSUB 7120
PRINT "-------"
INPUT "Enter the new viewing distance: ",VD
INPUT "Enter the number of steps: ",F
PRINT "-------"
GOSUB 9115
VD=VD-D:VD=VD/F
FOR FR=1 TO F
 D=D+vd:GOSUB 8320:GOSUB 7220
NEXT FR
STC=0:GOTO 1065

REM move shape
CLS
PRINT "*
PRINT "Moving object anywhere on screen"
PRINT "-------"
GOSUB 7120
PRINT "-------"
INPUT "Enter the amount of movement on the x-axis: ",MX
INPUT "Enter the amount of movement on the y-axis: ",MY
INPUT "Enter the amount of movement on the z-axis: ",MZ
INPUT "Enter the number of steps: ",F
PRINT "-------"
GOSUB 9115
FOR I=W1 TO W2
 FOR FR=1 TO F
  X(I)=X(I)+MX/F:Y(I)=Y(I)+MY/F:Z(I)=Z(I)+MZ/F
 NEXT I:GOSUB 8320:GOSUB 7220
NEXT FR
IF TIT=1 THEN GOTO 2460
IF PIN=1 THEN GOTO 2560
STC=0
GOTO 1065

REM change scale
CLS
PRINT "*
PRINT "Scaling object up or down"
PRINT "-------"
GOSUB 7120
PRINT "-------"
INPUT "Enter the scaling factor for the x-axis: ",SX
INPUT "Enter the scaling factor for the y-axis: ",SY
INPUT "Enter the scaling factor for the z-axis: ",SZ
INPUT "Enter number of steps: ",F
PRINT "-------"
GOSUB 9115
FOR I=W1 TO W2
 FOR FR=1 TO F
  X(I)=X(I)*SX:Y(I)=Y(I)*SY:Z(I)=Z(I)*SZ
 NEXT I:GOSUB 8320:GOSUB 7220
NEXT FR
IF TIT=1 THEN GOTO 2460
 IF PIN=1 THEN GOTO 2560
STC=0
GOTO 1065

Appendix E
Page 5 of 18
Appendix E
Page 6 of 18

REM change x,y,z
CLS
PRINT "*"
PRINT "Change any point on screen"
GOSUB 7120
PRINT "-----"
GOSUB 7520
HHT%=2:WTH%=1:PH%=0:MD%=1
CALL SETTEXT(HHT%,WTH%,PH%,MD%)
FORE=2:BACK=0
CALL SETTEXTCLR(FORE,BACK)
HGT=4:WIDT=2:BAC=0
CALL INITTCUR(HGT,WIDT,BAC)
FOR I=1 TO NP
XS(I)=(D*X(I))/(D+Z(I))
YS(I)=(D*Y(I))/(D+Z(I))
CALL PTABS(XS(I),YS(I))
CALL MOVTCURABS(XS(I),YS(I))
TXT=I: CALL TEXT(TXT)
NEXT I
INPUT "Enter the point number to change: ",PT
INPUT "Enter the amount of change in x, y, and z: ",XX,YY,ZZ
INPUT "Enter the number of steps: ",F
PRINT "-----"
GOSUB 9115
FOR FR=1 TO F
X(PT)=X(PT)+XX/F:Y(PT)=Y(PT)+YY/F:Z(PT)=Z(PT)+ZZ/F
GOSUB 8320:GOSUB 7220
NEXT FR
STC=0:GOTO 1065
REM
REM zoom
CLS
PRINT "*"
PRINT "Zoom screen in and out"
PRINT "-----"
PRINT "You will be asked to enter a zoom factor."
PRINT "The image will become smaller if the zoom factor is less than 1, and the image will become larger if the zoom factor is greater than 1."
PRINT "-----"
PRINT "Enter the zoom factor: ",ZM
INPUT "Enter the number of steps: ",F
PRINT "-----"
GOSUB 9115
WXM=XW/ZM:WXX=X2W/ZM
WYM=YW/ZM:WYX=Y2W/ZM
FACTOR%=0
GA=(WXM-XW)/F:GB=(WXX-X2W)/F
GC=(WYM-YW)/F:GD=(WYX-Y2W)/F
FOR FR=1 TO F
XW=XW+GA:X2W=X2W+GB
YW=YW+GC:Y2W=Y2W+GD
NEXT I
FOR FR=1 TO F
X(PT)=X(PT)+XX/F:Y(PT)=Y(PT)+YY/F:Z(PT)=Z(PT)+ZZ/F
GOSUB 8320:GOSUB 7220
NEXT FR
STC=0:GOTO 1065
CALL SETWORLD(XW,YW,X2W,Y2W)
GOSUB 8320:GOSUB 7220

NEXT FR
STC=0:GOTO 865

REM rotate shape
CLS:PRINT "*
PRINT "Rotation of object"
PRINT "------"
PRINT "This will rotate an object around its center"
PRINT "------"
INPUT "The object will rotate around which axis (x,y or z): ",AX$ INPUT "Enter the degrees of rotation: ",ANGLE
PRINT "------"
GOSUB 9115
IF AX$ = "x" THEN GOTO 8625
IF AX$ = "y" THEN GOTO 8690
IF AX$ = "z" THEN GOTO 8755
GOTO 1065

REM spin shape
CLS
PRINT "*
PRINT "Spin object around center of screen"
PRINT "------"
GOSUB 7120
PRINT "------"
INPUT "Enter the degrees of spin: ",SP
INPUT "Enter number of frames: ",F
PRINT "------"
GOSUB 9115
AZ=-1*SP
XSC=(XMIN+XMAX)/2:YSC=(YMIN+YMAX)/2
RS=(AZ*3.14)/180
FOR FR=1 TO F
    FOR I=W1 TO W2
        X1(I)=(X(I)-XSC)*COS(RS/F)-(Y(I)-YSC)*SIN(RS/F)+XSC
        Y1(I)=(X(I)-XSC)*SIN(RS/F)+(Y(I)-YSC)*COS(RS/F)+YSC
        X(I)=X1(I):Y(I)=Y1(I)
    NEXT I
    GOSUB 8320:GOSUB 7220
NEXT FR
STC=0:GOTO 865

REM tilt shape
CLS
TIT=1
PRINT "*
PRINT "Tilt camera around object"
PRINT "------"
GOSUB 7120
PRINT "------"
INPUT "Enter the amount of tilt: ",TT
MZ=0:MY=-1*TT:MX=0
GOSUB 1750
TIT=0
STC=0:GOTO 865

REM pan shape
CLS
PIN=1
PRINT "*"
PRINT "Pan camera around object"
GOSUB 7120
PRINT "-----"
INPUT "Enter the amount of pan: ", PN
MZ=0:MY=0:MX=-1*PN
GOSUB 1750
PIN=0
STC=0:G0T0 865
REM
REM transformation
GOSUB 8015
CLS:PRINT "*
PRINT "Transformation of two objects"
PRINT "-----"
PRINT "The shapes must have to same number of points"
INPUT "Enter the viewing distance of objects: ", D
INPUT "Enter the number of shapes being transformed: ", NS
PRINT "-----"
PRINT "The first shape can be entered one of two ways"
PRINT "  1) From disk  or  2) last shape drawn on screen"
DICK=1
INPUT "Enter 1 or 2: ", JJ
IF JJ = 2 THEN GOTO 3050
PRINT "  Enter the first shape 
GOSUB 1279
FOR I=1 TO EPT
  XX1(I)=X(I):YY1(I)=Y(I):ZZ1(I)=Z(I)
NEXT I

DICK=2
PRINT "-----"
PRINT "  Enter the second shape 
SPT=0:EPT=0:SPL=0:EPL=0
GOSUB 1279
FOR I=1 TO EPT
  XX2(I)=X(I):YY2(I)=Y(I):ZZ2(I)=Z(I)
NEXT I

INPUT "Enter the number of frames: ", F
PRINT "-----"
GOSUB 9115
FOR I=1 TO EPT
  CX(I)=(XX2(I)-XX1(I))/F
  CY(I)=(YY2(I)-YY1(I))/F
  CZ(I)=(ZZ2(I)-ZZ1(I))/F
NEXT I

FOR FR=1 TO F
  FOR S=1 TO NS
    N=S*S
    SPT=OBLIST(N+1):EPT=OBLIST(N+2)
    SPL=OBLIST(N+3):EPL=OBLIST(N+4)
    FOR I=SPT TO EPT
      BX(I)=XX1(I)+FR*C(I)
      BY(I)=YY1(I)+FR*CY(I)
      BZ(I)=ZZ1(I)+FR*CZ(I)
      X(I)=BX(I):Y(I)=BY(I):Z(I)=BZ(I)
    NEXT I
    GOSUB 7220
  NEXT S
NEXT FR
CLS: PRINT "*"
PRINT "Placement of text on screen"
PRINT "------"
PRINT "Text involves many variables which must be entered"
INPUT "Enter height and width of a letter in pixels: ", AH, AW
PRINT "------"
INPUT "Do you want a border (1) around type or not (0): 1 or 0: ", AB
PRINT "------"
PRINT "How do you want text to read? 1) normal left to right"
PRINT "2) 90 degrees 3) 180 degrees 4) 270 degrees"
INPUT "Enter number: ", AD
PRINT "------"
PRINT "Enter the foreground and background color of text: ", AF, AG
PRINT "------"
INPUT "Enter the height and width of text cursor attributes: ", AC, AV
PRINT "------"
INPUT "Enter the color of text cursor: ", AN
PRINT "------"
INPUT "Enter the x and y position to start text: ", AXS, AYS
PRINT "------"
INPUT "Enter the text: ", ATT$
HGHT%=AH: WDTH%=AW: PTH%=AD: MD%=AB
CALL SETTEXT(HGHT%, WDTH%, PTH%, MD%)
FORE%=AF: BACK%=AG
CALL SETTEXTCLR(FORE%, BACK%)
HHT%=AC: WTH%=AV: CLR%=AN
CALL INITTCUR(HHT%, WTH%, CLR%)
x%=AXS: y%=AYS
CALL MDVTCURABS(x%, y%)
TEXT$=ATT$
CALL TEXT(TEXT$)
INPUT "Do you want more text on screen? y or n: ", XT$
IF XT$= "y" THEN GOTO 3315 ELSE GOTO 665

REM animation test menu
CLS
LOCATE 1, 1: PRINT "*
PRINT "-----------------------------------"
PRINT " Menu"
PRINT " ------
PRINT " Animation Test Menu"
PRINT " -----------------------------------"
PRINT " 1) Save Palette"
PRINT " 2) Restore Palette"
PRINT " 3) Save Image"
PRINT " 4) Restore Image"
PRINT " 5) Convert Main Program to XOR"
PRINT " 6) Animate Area on Screen in XOR"
PRINT " 7) Main Menu"
LOCATE 16, 1
PRINT "-----------------------------------"
LOCATE 15, 1
INPUT "Enter option by number: ", K
IF K = 1 THEN GOTO 4315
IF K = 2 THEN GOTO 4215
IF K = 3 THEN GOTO 4412
IF K = 4 THEN GOTO 4512
REM setting xor
GOSUB 8915
CLS:PRINT "*
PRINT "Exclusive OR"
PRINT "-------"
PRINT "*
INPUT "Do you want turn on(n) or turn off(f) XOR: ",YY$ IF YY$ = "n" THEN GOTO 3755
SWITCH%= 0
PRINT "*
PRINT "XOR has been turned off...
FOR I=1 TO 1500: NEXT I
GOTO 3780
SWITCH%= 1
PRINT "*
PRINT "XOR is now being activated...
FOR I=1 TO 1500: NEXT I
CALL SETXOR(SWITCH%)
GOTO 3515

REM area movement in xor
GOSUB 8915
GOSUB 3950
CLS:PRINT "*
PRINT "Defining Area for Amination"
PRINT "-------"
PRINT "First, the area to be moved must be defined"
INPUT "Enter the upper left x and y coordinates: ",UX,UY
INPUT "Enter the lower right x and y coordinates: ",LX,LY
PRINT "---"
INPUT "Enter the amount of movement on x axis: ",MX
INPUT "Enter the amount of movement on y axis: ",MY
INPUT "Enter the number of frames: ",FRM
PRINT "---"
INPUT "Enter the mode 1, 2, 3<xor>, 4, 5, 6, 7, 8: ",MODE$
JAY=0
VX=MX/FRM:VY=MY/FRM
CALL MOVEFROM(UX,UY,LX,LY,ARRAY%0)
CALL SETCOLOR(CL%)
PX=UX:PY=UY
QX=LX:QY=LY
CALL BAR(PX,PY,QX,QY)
FOR FR=1 TO FRM
    CALL MOVETO(UX,UY,ARRAY%0,MODE$
    IF JAY=1 THEN GOTO 3894
    FOR I=1 TO 500:NEXT I
    JAY=JAY+1;GOTO 3890
    JAY=0
    UX=UX+VX
    UY=UY+VY
FOR H=1 TO 500:NEXT H
NEXT FR
PRINT "-------"
INPUT "Do you want to run XOR(x) and quit(q): ",F$
IF F$ = "x" THEN GOTO 3810
IF F$ = "r" THEN GOTO 3845
GOTO 3515

REM small sub for area

PRINT "A test for locating an area on screen"
INPUT "Do you want to plot any point for test (y or n): ", D$
IF D$ = "n" THEN GOTO 3995
CLS: INPUT "Enter the x and y coordinate: ", X, Y
CALL PTABS(X, Y)
INPUT "Do you want to plot any more (y or n): ", S$
IF S$ = "y" THEN GOTO 3980
RETURN
REM table for storing shapes
CLS: PRINT "*
PRINT "Table of available shapes on disk"
PRINT ":--------------------------:
PRINT " filename description     
PRINT ":--------------------------:
PRINT " 1) cubel.dat - cube in top right hand corner of screen."
PRINT " 2) cube2.dat - cube in lower left hand corner of screen."
PRINT " 3) cool.dat - an neat looking six sided figure."
PRINT " 4) cross.dat - a Swiss Red Cross Symbol in the middle of screen."
PRINT "
PRINT "-------"
INPUT "Hit (return) to return to Main Menu: ", B$
GOTO 525
REM restoring color palette
GOSUB 8915
CLS: PRINT "*
PRINT "Restoring Color Palette"
PRINT "------"
PRINT " This operation will only work if the user has"
PRINT " previously stored a color palette with this program."
PRINT "------"
INPUT " To proceed hit <return> else type (o): ", HJ$
IF HJ$ = "o" THEN GOTO 4295
PRINT "-------"
INPUT "Enter the name of color palette as b:filename.plt: ", PFILE$
PDES$ = "linear palette"
PRINT "*
PRINT "Restoring color palette..."
CALL PREAD (PFILE$, PDES$)
GOTO 3515
REM saving color palette
GOSUB 8915
CLS: PRINT "*
PRINT "Saving a color palette"
PRINT "------"
INPUT "Enter the name of color palette as b:filename.plt: ", PFILE$
PDES$ = "linear palette"
MODE% = 0
PRINT "*
PRINT "Saving color palette..."
CALL PWRITE (PFILE$, PDES$ MODE%)
GOTO 3515
REM saving image
GOSUB 8915
PRINT "Saving an image"
PRINT -------
INPUT "Enter the name of image as follows- b:filename.img: ",GFILE$
PRINT 
PRINT "Image is being saved...."
CALL GWRITE(GFILE$)
GOTO 3515

REM restoring image

GOSUB 8915
CLS:PRINT 
PRINT "Restore an image"
PRINT -------
PRINT " This operation will only work if the user has"
PRINT " previously stored an image with this program."
PRINT -------
INPUT "To proceed hit <return> else type (o): ",HJ$
IF HJ$ = "o" THEN GOTO 4590
PRINT 
INPUT "Enter the name of image as follows- b:filename.img: ",GFILE$
PRINT 
PRINT "Image is being restored...."
CALL GREAD(GFILE$)
GOTO 3515

REM ending program

CLS:LOCATE 15,15
PRINT "This is the end of the program"
LOCATE 17,15
PRINT " Thank you for playing with me"
LOCATE 19,15
PRINT " Go have a beer on me"
CALL CLOSEGRAPHICS
END

-----------------------------------
Subroutines
-----------------------------------
LOCATE 18,10
PRINT "This OPERATION is not in service"
PRINT "----Sorry Charlie !!! - Going crazy"
FOR I=1 TO 1500:NEXT I
RETURN

REM this will pick shape

CLS:LOCATE 20,1
PRINT "Enter number of shape for operation"
INPUT "(see shapes):Enter 0 for all shapes: ",S
MDR=1
IF S=0 THEN MDR=NS
FOR M=1 TO MDR:N=5*S
IF S=0 THEN N=5*M
NEXT M
SPT=OBLIST(N+1):EPT=OBLIST(N+2)
SPL=OBLIST(N+3):EPL=OBLIST(N+4)
IF S=0 THEN SPT=1
IF S=0 THEN SPL=1
W1=SPT:W2=EPT:RETURN
GOSUB 7115
CALL SETCOLOR(CL7.)
PRINT "Drawing...."
FOR S=1 TO NS
    N=5*S:SPT=OBLIST(N+1):EPT=OBLIST(N+2)
    SPL=OBLIST(N+3):EPL=OBLIST(N+4)
    XMAX=-1000:XMIN=1000:YMAX=-1000
    YMIN=1000:ZMAX=-1000:ZMIN=1000
    FOR I=SPT TO EPT
        IF X(I)>XMAX THEN XMAX=X(I)
        IF X(I)<XMIN THEN XMIN=X(I)
        IF Y(I)>YMIN THEN YMIN=Y(I)
        IF Y(I)<YMIN THEN YMIN=Y(I)
        IF Z(I)>ZMAX THEN ZMAX=Z(I)
        IF Z(I)<ZMIN THEN ZMIN=Z(I)
    NEXT I
    XC=(XMAX+XMIN)/2:YC=(YMAX+YMIN)/2:ZC=(ZMAX+ZMIN)/2
    FOR I=SPL TO EPL
        XF=X(PL(I,1)):YF=Y(PL(I,1)):ZF=Z(PL(I,1))
        A1=X(PL(I,3))-X(PL(I,2))
        A2=Y(PL(I,3))-Y(PL(I,2))
        A3=Z(PL(I,3))-Z(PL(I,2))
        B1=X(PL(I,1))-X(PL(I,2))
        B2=Y(PL(I,1))-Y(PL(I,2))
        B3=Z(PL(I,1))-Z(PL(I,2))
        AF(I)=(A2*B3)-(A3*B2)
        BF(I)=(A3*B1)-(A1*B3)
        DF(I)=(-AF(I)*XF)-(BF(I)*YF)-(CF(I)*ZF)
    NEXT I
    FOR N=SPL TO EPL
        TEST=(AF(N)*XC)+(BF(N)*YC)+(CF(N)*ZC)+DF(N)
        IF TEST>=0 THEN GOTO 7375
        AF(N)=-AF(N):BF(N)=-BF(N)
        CF(N)=-CF(N):DF(N)=-DF(N)
    NEXT N
    IF CENT=1 THEN XC=0 AND YC=0
    FOR I=SPL TO EPL
        FLAG(I)=0
        FOR J=1 TO PL(I,0)
            TESTA=(AF(I)*X(PL(I,J)))+(BF(I)*Y(PL(I,J)))
            TESTA=TESTA+(CF(I)*(Z(PL(I,J))+D))
            IF TESTA>=0 THEN GOTO 7415
            FLAG(I)=1:GOTO 7420
        NEXT J
    NEXT I
    FOR I=SPT TO EPT
        XSP(I)=(D*X(I))/(D+Z(I))
        YSP(I)=(D*Y(I))/(D+Z(I))
    NEXT I
    FOR I=SPL TO EPL
        IF FLAG(I)=1 THEN GOTO 7480
        FOR J=1 TO PL(I,0)
            IF J=1 THEN GOTO 7470
            CALL MOVABS(XSP(PL(I,J-1)),YSP(PL(I,J-1)))
            CALL LNABS(XSP(PL(I,J)),YSP(PL(I,J)))
        NEXT J
        CALL LNABS(XSP(PL(I,1)),YSP(PL(I,1)))
    NEXT I
    NEXT S:RETURN
REM this will clear the screen
LOCATE 21,1
INPUT "Do you want to clear screen:y or n ",P$ 
IF P$ = "n" THEN GOTO 7540 
PRINT "EXIT --- CM---- S --- COL"
LOCATE 5,50
PRINT "Shapes in memory"
LOCATE 6,50: PRINT "-----------------
LOCATE 7,50: PRINT "1) P"
LOCATE 8,50: PRINT "2) Q"
LOCATE 9,50: PRINT "3) R"
LOCATE 10,50: PRINT "4) S"
LOCATE 11,50: PRINT "5) T"
RETURN

REM
REM this will do the calculation for rotation
REM
LOCATE 22,1
INPUT "Enter the number of steps: ", F
AXA=((3.14*AX)/180)/F
AYA=((3.14*AY)/180)/F
AZA=((3.14*AZ)/180)/F
FOR FR=1 TO F
FOR I=W1 TO W2
X1(I)=X(I)
Y1(I)=(Y(I)-YC)*COS(AXA)-(Z(I)-ZC)*SIN(AXA)+YC
Z1(I)=(Y(I)-YC)*SIN(AXA)+(Z(I)-ZC)*COS(AXA)+ZC
X2(I)=(X1(I)-XC)*COS(AYA)+(Z1(I)-ZC)*SIN(AYA)+XC
Y2(I)=(X1(I)-XC)*SIN(AYA)+(Z1(I)-ZC)*COS(AYA)+YC
Z2(I)=Z1(I)
NEXT I
GOSUB 8320: GOSUB 7220
NEXT FR: RETURN

CLS: PRINT "*
PRINT "This is create shape. In this section"
PRINT "you will create your own three dimensional"
PRINT "shape. Once you create shape, make sure ":
PRINT "to save in Shape Services Menu."
PRINT "-----
INPUT "Enter the number of points in shape: ", NP
PRINT "-----
FOR K=1 TO NP
INPUT "Enter the x,y,z coordinates for point: ",X(K),Y(K),Z(K)
PRINT "-----
CLS: INPUT "Enter the viewing distance for shape: ", D
INPUT "Enter the number of connecting lines for shape: ", NL
FOR I=1 TO NP
XS(I)=(D*X(I))/(D+Z(I))
YS(I)=(D*Y(I))/(D+Z(I))
CALL PTABS(XS(I),YS(I))
NEXT I
PRINT "-----
FOR K=1 TO NL
INPUT "Enter two numbers to connect lines: ", SP(K), EP(K)
CALL MOVABS(XS(SP(K)),YS(SP(K)))
CALL LNABS(XS(EP(K)),YS(EP(K)))
NEXT K
CLS: PRINT "Now you must enter the planes in shape"
INPUT "Enter the number of planes: ", NP.
FOR I=1 TO NPL
  INPUT "Enter the number of corners pts. for plane: ", PL(I,0)
  K=PL(I,0)
  FOR J=1 TO K
    INPUT "For this plane, enter corner pts.: ", PL(I,J)
  NEXT J
NEXT I
NS=1
SPT=1: EPT=NPL
SPL=1: EPL=NPL
CLS: PRINT "You have created a shape in three dimensional form."
PRINT "If you would like to see shape in true three de. form,"
PRINT "then proceed to Viewer and Camera Services Menu. If "
PRINT "you are not happy with shape then type (r) and start "
PRINT "over. If you are delighted with shape then proceed to "
PRINT "Shape Services Menu and SAVE shape by just hitting return."
PRINT "----"
INPUT "Enter (r) to create new shape or hit return: ", Y$ IF Y$ = "r" THEN GOTO 7765
RETURN
CLS
PRINT "---"
CLS
PRINT "There are up to 256 colors to choose from"
PRINT "The colors will depend on the current palette"
PRINT "----"
PRINT "Enter the color by location on palette"
INPUT "Enter the color for object: ", CL7.
PRINT "----"
PRINT "To change background color; go to Clear Screen option"
INPUT "Enter the color for background: ", COL7.
CALL SETCOLOR(CL7.)
RETURN
CLS: PRINT "*"
PRINT "Line Width"
PRINT "----"
PRINT "The width of a line can between 1 and 1023, where "
PRINT "1 is the thinnest line. The number that is being "
PRINT "entered must be odd."
PRINT "----"
INPUT "Enter line width: ", WH7.
CALL SETLNWIDTH(WH7.)
RETURN
CLS: PRINT "*"
PRINT "Line Style"
PRINT "----"
PRINT "Once a certain line style has been choose, "
PRINT "the color of the object can not be changed "
PRINT "unless the line style is reset to Solid."
PRINT "There are three different types of lines styles"
PRINT "  1) Solid ------ "
PRINT "  2) Broken ----- "
PRINT "  3) Dots .... "
PRINT "----"
INPUT "Enter number of line style: ", STYLE7.
CALL SETLNSTYLE(STYLE7.)
RETURN
CLS: LOCATE 21,1
IF STC=1 THEN GOTO 8385
INPUT "Do you want to clear screen(y or n): ",M$
IF M$ = "n" THEN GOTO 8360
CALL SETCOLOR(COLX)
CALL CLR
INPUT "Do you want to change the color: y or n: ",CC$
IF CC$ = "n" THEN GOTO 8380
INPUT "Enter the new color for object: ",CLX
CALL SETCOLOR(CLX)
RETURN
INPUT "Do you want to clear screen:y or n: ",WW$
IF WW$ = "n" THEN GOTO 8391
CALL SETCOLOR(COLX)
CALL CLR
CLX = CLX + 1
CALL SETCOLOR(CLX)
RETURN
REM printer
CLS
PRINT "*
PRINT "Apple Imagewriter"
PRINT "-----
PRINT "This option will allow the image on screen"
PRINT "to be printed in color. This option is "
PRINT "not available at this time do to not having"
PRINT "the printer."
FOR I=1 TO 4000:NEXT I
RETURN
REM printer(ink jet)
CLS
PRINT "*
PRINT "Tektronic Ink Jet Printer"
PRINT "-----
PRINT "Instructions (please read the entire process before starting)"
PRINT "1) Insert MSDOS disk with Tektronic program"
PRINT "in drive A."
PRINT "2) Hit <CTRL>,<ALT>,<DEL> at the same time."
PRINT "3) Wait for A> and type TEK4596."
PRINT "4) The printer should be hooked up to back of PC."
PRINT "5) Follow instructions which will be given to you."
PRINT "It is very easy to use and can provide some visually "
PRINT "interesting prints."
INPUT "Hit (Return) to return to program: ",KK$
RETURN
REM new calculations for rotation
REM x
FOR DEGREES=(ANGLE/F) TO ANGLE STEP (ANGLE/F)
  RADIANS=(ANGLE*3.14159)/(180*F)
  NEXT I
GOSUB 8320
GOSUB 7220
NEXT DEGREES
STC=0
FOR DEGREES=(ANGLE/F) TO ANGLE STEP (ANGLE/F)  
RADIANS=(ANGLE*3.14159)/(180*F)  
FOR I=W1 TO W2  
   X1=(X(I)-XC)*COS(RADIANS)-(Z(I)-ZC)*SIN(RADIANS)+XC  
   Z1=(X(I)-XC)*SIN(RADIANS)+(Z(I)-ZC)*COS(RADIANS)+ZC  
   X(I)=X1:Z(I)=Z1  
   NEXT I  
GOSUB 8320:GOSUB 7220  
NEXT DEGREES  
STC=0  
GOTO 1065  

FOR DEGREES=(ANGLE/F) TO ANGLE STEP (ANGLE/F)  
RADIANS=(ANGLE*3.14159)/(180*F)  
FOR I=W1 TO W2  
   X1=(X(I)-XC)*COS(RADIANS)-(Y(I)-YC)*SIN(RADIANS)+XC  
   Y1=(X(I)-XC)*SIN(RADIANS)+(Y(I)-YC)*COS(RADIANS)+YC  
   X(I)=X1:Y(I)=Y1  
   NEXT I  
GOSUB 8320:GOSUB 7220  
NEXT DEGREES  
STC=0  
GOTO 1065  

REM warning subroutine  
CLS:PRINT "*"  
PRINT "Warning Message "  
PRINT "-------"  
PRINT " The following operation is a TEST. When using "  
PRINT " this operation beware that the computer may "  
PRINT " do some strange things and may even completely "  
PRINT " shut down. "  
PRINT " -------Sorry, I am not always perfect.......... "  
PRINT "*"  
INPUT "If you want to continue hit return else type (o): ",SD$  
IF SD$ = "o" THEN GOTO 3515  
RETURN  

REM menu for stored shapes  
LOCATE 10,55: PRINT "  Shapes Available  "  
LOCATE 11,55: PRINT "--------------------"  
LOCATE 12,55: PRINT "  1) b:cube1.dat  "  
LOCATE 13,55: PRINT "  2) b:cube2.dat  "  
LOCATE 14,55: PRINT "  3) b:cool.dat  "  
LOCATE 15,55: PRINT "  4) b:cross.dat  "  
GOTO 9090  
LOCATE 16,55: PRINT "  5) a. b:hip1.dat  "  
LOCATE 17,55: PRINT "  b. b:hip2.dat  "  
LOCATE 18,55: PRINT "  c. b:hip3.dat  "  
LOCATE 19,55: PRINT "  d. b:hip4.dat  "  
RETURN  

REM cycle color through palette  
STC=0  
PRINT "Do you want to run through a specific part "  
INPUT "of the color palette:y or n: ",RR$  
IF RR$ = "n" THEN GOTO 9170
APPENDIX F

Offset for Routines
Example.int  Sample program using Halo from Basic interpreter

Running the example program:
1: MODIFY the example program, as necessary, according to the comments in the program.
2: EXECUTE the resident driver HALORBI.EXE
3: EXECUTE the program MEMORY.BAS to determine the correct values for the CLEAR and the DEF SEG statements below.
4: EXECUTE the program EXAMPLE.INT.

NOTE: The following HALO files should all be in the current directory -
MEMORY.BAS  HALORBI.EXE  HAL0I.BIN  EXAMPLE.INT
and a device driver. i.e. "HALOIBM.DEV"

The CLEAR and DEF SEG statements may have to be modified according to your hardware. Run the program MEMORY.BAS to determine the proper values.

The statements beginning at line 8000 are necessary. They define the offsets to the Halo routines.

The call to SETSEG below must be included if you are using the SCION board or the Virtual Rasterization Interface with Halo. See the Halo manual for information on setting ADDRESSX.

addressX = 7
The SCION board is at segment 7
call setseg(addressX)

MAIN PROGRAM

Install the device driver for MULTIHALO

DEVICE$ = "HALONINE.DEV" :REM Use the IBM color graphics card
CALL SETDEV(DEVICE$)

Init graphics and write message
MODEX = 0: CALL INITGRAPHICS(MODEX)
X = 0: YMAX = 199: XMAX = 319: Y = 0: CALL SETWORLD(X, YMAX, XMAX, Y)
TEXT$ = "This was made with Halo and Basic"
TY = (YMAX + 1) / 20
CALL MOVTCURABS(TX, TY): CALL TEXT(TEXT$): CALL DELTCUR

MOVE TO CENTER AND DRAW PIE

X = (XMAX + 1) / 2: Y = (YMAX + 1) / 2
CALL MOVABS(X, Y)
RADIUS = (XMAX + 1) / 4: ARC1 = .5: ARC2 = 2: CLR% = 1
CALL PIE(RADIUS, ARC1, ARC2, CLR%)
STYLE% = 2: CALL SETHATCHSTYLE(STYLE%)
ARC1 = ARC2: ARC2 = 3.5
CALL PIE(RADIUS, ARC1, ARC2, CLR%)
STYLE% = 3: CALL SETHATCHSTYLE(STYLE%)
ARC1 = ARC2: ARC2 = 5.78
CALL PIE(RADIUS, ARC1, ARC2, CLR%)
XREL = 20: YREL = 0
CALL MOVREL(XREL, YREL)
STYLE% = 4: CALL SETHATCHSTYLE(STYLE%)
ARC1 = ARC2: ARC2 = .5
CALL PIE(RADIUS, ARC1, ARC2, CLR%)

Wait for keypress before exiting

K$ = INKEY$: IF K$ = "" THEN 740
CALL CLOSEGRAPHICS: END

ARC = &H85
BAR = &H22B
BOX = &HA1
CFREEZE = &H2B5
CGRAB = &H2BC
CIR = &H77
CLOSEGRAPHICS = &H2C3
CLR = &H2CA
COMLINE = &H467
CSMALUT = &H2DB
CSETLUT = &H2D1
CSNAP = &H2DF
DEFHATCHSTYLE = &HBD
DEFLNSTYLE = &H516
DELBOX = &H21B
DELCIR = &H222
DELHCUR = &H292
DELLN = &H229
DELCUR = &H299
DISPLAY = &H2E6
ELLIPSE = &H70
FCIR = &H93
FCLR = &H2ED
FILL = &H9A
FLOOD = &HAB
FLOOD2 = &H103
FTCOLOR = &HAF
FTTEXT = &H10A
FTINIT = &H111
FTLOCATE = &H11F
FSIZE = &H1B9
GPRINT = &H1D5
GREAD = &H26F
GSCAN = &H1DC
GMWRITE = &H46E
HALLOC = &H475
HFREE =
8169 SETPRN
8170 SETPWD
8171 SETRGB
8172 SETSCAN
8173 SETSCATTR
8174 SETSCREEN
8175 SETSEG
8176 SETSEG2
8177 SETSTANG
8178 SETSTCLR
8179 SETTEXT
8180 SETTEXT
8181 SETTEXTCLR
8182 SETVBW
8183 SETVIEWPORT
8184 SETWINDOW
8185 SETWORLD
8186 SETXOR
8187 SETXPAL
8188 SHIFT
8189 STARTGRAPHICS
8190 TEXT
8191 TEXT
8192 VFAN
8193 WORLDOFF
8194 ZOOM
9000 RETURN

= &H1C7
= &H4DE
= &H423
= &H27D
= &H2B4
= &H43D
= &H46
= &H436
= &H4C9
= &H4B4
= &H4AD
= &HCB
= &HD2
= &H444
= &H134
= &H149
= &H142
= &H1F8
= &H42F
= &H44B
= &H452
= &H4BB
= &HD9
= &H459
= &H150
= &H460