The Graphics Magician Picture Painter™

by Mark Pelczarski and Steven Meuse

The Graphics Magician™ Picture Painter

by Mark Pelczarski and Steven Meuse

The enclosed product is supplied on a disk that is not copy-protected. It is our intent to make this product as useful as possible, and we feel that with applications software, the ability to easily make your own backup copies is a great asset.

We ask that you not abuse our intentions by making copies for others. The result of such activity only helps promote the unfortunate existing situation of most software being protected, and in our opinion, less usable. We hope that the commercial success of non-protected products such as this one will signal other publishers that copy-protection is not necessary for a product's survival and help reverse the trend in protected applications software.

At Penguin, through policies such as this and our pricing, we are trying to look out for your best interests. Please help us.

Sincerely,

[illegible] 49769

Mark Pelczarski
President, Penguin Software

penguin software™
the graphics people

P.O. Box 311, Geneva, IL 60134 (312) 232-1984

Table of Contents

Chapter 1 - Introduction

The Graphics Magician Picture Painter is a set of graphic editors and machine language routines that help you easily put professional, state-of-the-art graphics in your own programs. Included are routines that help you draw and recreate very compact, multicolored pictures (great for use in adventure games and educational software), and easily mix text with graphics on the screen. **The Graphic Magician Picture Painter** can be used just for drawing and saving pictures, or it can be used as a programmers' tool. The graphics routines included in this package are being used in dozens of commercially marketed software packages, produced by many of the most well-known publishers in the industry. And with **The Graphics Magician** available on several leading micro-computers, designers will find that most of the graphics work done on one machine can be easily transferred to several others, saving long hours of duplicated work, and in some cases making software portable in ways never before possible.

Also available from Penguin Software, for 48K Apple computers, is **The Graphics Magician**, which includes a standard graphics version of the Picture Painter, plus an Animator for use with standard graphics. Available for standard hi-res or for double-res is **The Complete Graphics System**, which is a set of graphics and drawing programs for non-programmers. Other graphics programs from Penguin include **Transitions**, a presentation and slide-show system, and **Paper Graphics**, which lets you print your graphics screens to your printer.

"Double-Res" on the Apple is an extended graphics mode available on Apple computers with 128K of memory. Whereas standard hi-res has 6 pure colors and a resolution (dots on the screen) of 280 by 192, double-res has 15 pure colors and a resolution of 560 by 192. In our standard hi-res graphics programs, we combine the existing six colors to form 108 different blended colors. With double-res, we combine the 15 pure colors to form 256 blended colors.

Interestingly, **The Graphics Magician Picture Painter** offers an easy solution to software developers who wish to take advantage of double-res, but still want their programs to work on all the Apple computers that only have standard hi-res capabilities. **Graphics Magician** pictures are stored by saving the artist's moves while drawing, thus creating sort of a "picture program", unbeknownst to the artist. By mapping our standard 108-color palette to purer colors in double-res, we've created an environment where these "picture programs" can be reconstructed in standard hi-res using colors like our blended-yellow (there is no real yellow in standard hi-res), or reconstructed in double-res, if the computer supports it, using the purer colors such as the real yellow in double-res. The results are dramatic. For examples, see the double-res editions of our adventure games, such as **Transylvania** and **The Quest**. For **Transylvania**, we created a special double-res version that is on the flip side of the standard-res disk. **The Quest**, however, is a two-sided adventure, so we use the exact same graphics files for both versions. When the adventure begins, a simple choice tells the program whether to draw the pictures in standard or double-res!

When learning to use **The Graphics Magician Picture Painter**, start simply. First of all, sit down with your manual AND your computer. This is computer software, not a book. You'll learn it best by using it while reading about it. Since **The Graphics Magician** has many features, it is tempting to want to understand them all immediately. To start, just try some scribbling with lines and fills. Test the "redraw" command to see how your picture is recreated. Then experiment with the different options, one by one.

Input Devices

The Graphics Magician Picture Painter can be used with joystick, mouse, paddles, touch tablet, track ball, Apple Graphics Tablet, or Houston instruments HiPad for input. As other input devices are supported, we will note so with inserts.

For devices with two buttons, we'll refer to **button 0** and **button 1**. For single button/switch devices, such as the mouse or graphics tablets, **button 0** is the switch on the device. **Button 1** is the **RETURN** key on the keyboard.

Backup Copies

The Graphics Magician is provided on a copyable disk, and all the routines are accessible by your own programs. We strongly recommend that the first thing you do is make a backup copy or two and store the original in a safe place. We feel the ability to make your own backups is of great benefit to you. Please don't misuse our trust by making copies for others.

A registration number is written on your **Graphics Magician** disk and stamped on the inside cover of your manual. If you call with questions regarding use of this product, please be prepared to provide this number, as it will help us identify your exact version. Also remember to send in your registration card so that we may notify you of any new versions or updates.

Licensing

Any of the routines enclosed may be freely used in your own programs. If the routines or facsimiles appear in another product for sale, there is no fee, but we do require that a license be obtained from Penguin Software stating that you have permission to use the copyrighted routines. Note that **The Graphics Magician Picture Painter** is available for several different microcomputers. In writing the different versions, the routines were made as compatible as possible, so much of the work you do on one computer can be easily transferred to another, if you desire. Some of the basic transfer routines are included in this package. Also, please consider Penguin Software as a possible publisher for your works.

Getting Started

First, using the copy program on your Apple master diskette, or any other copy program, make one or two copies of your **Graphics Magician Picture Painter** disk. Put the master away in a safe place, and use the copies in everyday work.

Next, initialize a blank disk or two for use as data disks. **The Graphics Magician Picture Painter** disk is close to full. The pictures you create should be saved on a separate data disk. To format a disk, make sure you start with a disk that contains nothing important already on it (it will be erased). Then "boot" your Apple master disk, and when you get the prompt "]", remove the master, insert a blank disk, type "NEW", then press **RETURN**, then type "INIT HELLO", and press **RETURN**. The disk will whir for a minute, and your disk is now initialized.

Now "boot" your **Graphics Magician** copy (put it in the disk drive, and turn on the computer). You will be presented with a list of choices in parentheses. This is the "menu" screen. When done using any of the programs listed in the choices, you will always be returned to this page. Note that throughout this manual, for ease of reading and understanding, single key choices will be listed with their meanings. Most of the choices in **The Graphics Magician** require only a single keypress, but expressing them in the form "(**P**)icture editor" instead of "P" helps one follow the meanings of each a little more easily.

The Graphics Magician Picture Painter comes ready to use on an Apple with one disk drive and at least 128K of RAM, with double-res capabilities. Your dealer can tell you if your computer can display double-res. If you have one drive, you will have to switch disks when you read and save your picture files to your data disk, and again each time you are done with a program and wish to return to the menu screen. If you have more than one disk drive, you can set **The Graphics Magician** so that it automatically uses drive 2 for the data disk. To set this, choose "(**R**)edefine Disk Access" from the menu. Then, when asked for master disk, type "D1", and for data disk type "D2". At the bottom of the menu screen the program will list where it expects the master and data disks.

Now you're ready to go. Remember, don't try too much too fast. Try simple pictures, then experiment with the options, one at a time.

Chapter 2 - Drawing Pictures

The picture drawing system is designed to let you create screen pictures that take a minimal amount of storage space. Double-res pictures always take 16K, approximately 16,000 bytes, of display space in your computer. However there are ways that allow you to take considerably less storage on disk. About 6 double-res pictures can normally fit on one side of a floppy disk. With **The Graphics Magician Picture Painter**, you can easily fit fifty to well over a hundred pictures on a single side of a disk.

Double-res pictures are stored as the values in the 16,384 bytes that make up the double-res graphics screen. With **The Graphics Magician Picture Painter**, instead of storing the results of your drawing as a screen image, the moves that you make in creating your drawing are stored. The moves for most drawings can be in hundreds of bytes instead of thousands. We call these "sequential pictures", since the sequence is remembered instead of the actual picture.

The effect this has is that the computer "remembers" what you do as you draw. Later, when you want to view that picture again, the computer simply reconstructs your moves, very quickly. If you've played any graphic adventure games, you probably will recognize what this looks like; the picture redraws very rapidly before your eyes. What you see recreated are the moves that the artist made while drawing the picture the first time. Most adventure games use this technique (many of them done with **The Graphics Magician**) since they demand that large numbers of pictures fit on a disk. There are also many educational products that use **The Graphics Magician** this way, since they too require a large number of graphic images.

There are four types of "moves" that you, the artist, can make. You may draw a line, fill an enclosed area with color, plot a computer "brush", or type a letter over your picture. In addition, you may choose from a palette of 256 color mixes, and select one of eight different brushes, ranging from a small, precision size to a large, airbrush effect.

Using the Picture Editor

From the menu, select the version of the Picture Editor that corresponds to your input device. Joystick, trackball, paddles, and touch tablets all use the joystick version.

You'll first be asked if you want to "(**L**)oad" a previous picture from disk, start a "(**N**)ew" picture, load a "(**B**)ackground", or see a "(**D**)isk catalog". For the first time through, choose "(**N**)ew".

Now you'll see a white screen, with a few text lines on the bottom, as in figure 2.1. If you move your input device you'll see a small cursor in the shape of a crosshair move around the picture. This is what you control for drawing your picture.

```
M N T Z SP ESC R DEL E S I C Q (H)ELP (560)
LINE (560)  FCOLOR:0  LCOLOR:0
picture instructions listed here
BYTES USED:1      SECTORS=2      X:  Y:
```

Figure 2.1 - Picture Editor Command Lines

Line Mode

When you first start, you are in line mode (the second command line says "line"). Pressing the **RETURN** key or **button 1** will put a "start line" command at the current location of your moving cursor. This is the starting point of the next line you draw. Pressing **button 0** draws a line from the start point to the movable cursor (ending point). The ending point also becomes the new starting point. Move your cursor around and try the effects of the **RETURN** key and/or buttons.

How Long is Your Picture?

When playing with line mode, note that three things happen on the bottom of the screen. The x, y position listed at the right side of the bottom line keeps changing as you move the cursor, and each time you press a button, the third line tells you what you just did ("Start Line at ---", or "Draw Line to ---"), and the byte count on the bottom line tells you how many bytes you've used for your picture and how many sectors the picture will take on disk. Each "start line" or "draw line" command takes 3 bytes. Every picture has a starting length of 1 byte.

Deleting Steps

The first nice thing to learn is that pressing the **DELETE** key will delete the last step. If you make a mistake, it's easy to back up as many steps as you want and try again. Note that each time you delete a step, you see how the redraw option works. The program remembers what you've done to that point and recreates everthing except the step you deleted.

If you have a lot of steps to delete, you can also press **Control -D** (hold down the control key and press "D"). You will be asked how many steps you want to delete. The program still deletes one step at a time, but it cycles through automatically the number of times you specify.

Fine Cursor Control

"(**Z**)ero" toggles the joystick input so the cursor moves only in a small range on the screen, allowing you to zero in on a specific point more easily. Pressing "(**Z**)ero" again puts you back into normal mode.

For even finer control, pressing "(**Control-Z**)ero" controls how tight of an area the controller will cover in "zeroed" mode. Press "(**Z**)ero" to zero in on an area, then try the effects of "(**Control-Z**)ero" while you are zeroed in.

Selection Page

Okay, now for some fun. Press the **SPACE BAR**. A graphic screen appears that shows your choices for modes across the top (line, fill, or brushes) and pictures of the eight brushes, a strip on the left that shows the sixteen possible line colors, and a palette showing the 256 possible fill and brush colors. Your joystick controls an arrow cursor that moves around the screen.

An arrow near the top should point to the line option, meaning that you are in line mode. Moving your cursor to point at the fill option or one of the eight brushes and pressing the button on your controller will select that option and put you in a different mode.

A second arrow points to the current line color on the left, black. This means that all your lines will be drawn in black. Other colors are selected by pointing at them and pressing the controller button.

The third arrow points to the current color, at the upper left corner of the palette. (This color is color #0. The palette colors are numbered from top to bottom in each column, starting with 0.) This is the current color used for filling and for brushes. Moving your cursor to any other color and pressing the controller button selects that color for future fills and brushes. The color will be displayed in a larger block at the top of the screen (above "fill") to give you a better look at it.

Select fill mode and a fill color other than white, then press the **SPACE BAR.** The **SPACE BAR** switches between the drawing screen and the selection screen. While viewing the selection screen, you may change any of the options that you want, or you may just check the options and colors and press **SPACE BAR** again to get back to the drawing screen.

Fill Mode

When in fill mode, you can fill any enclosed white area with the current fill color by positioning your cursor inside the area you want to fill and pressing your controller button. The area should be all white, with borders in black or the edge of the screen.

The fill routine is designed to be as fast as possible, so that pictures that are reconstructed in a finished program will appear very quickly. Most irregular areas will require two or three fill commands to fill the entire area, since with speed comes some compromise with completeness of fill. The fill routine used works the following way.

1) Scan directly up from the selected point until a border is found.

2) Move down one line, filling to the left and right borders.

3) Average the left and right borders to find the midpoint, and move down one line from there.

4) Check to see if the point moved down to is a border. If not, go back to Step 2.

The basic thing to remember is step 1. It means that the best place to position your cursor is anywhere directly below the uppermost point in the area to fill. Using this one trick will minimize the number of fill commands necessary for filling any area.

Brushes

From the selection page, choose any of the eight brushes by pointing to it and pressing your controller button. Go back to the drawing page and you'll see that your cursor is now in the shape of the brush that you selected. Each time you press the controller button, the brush will be plotted with the color you selected.

The brushes give you a very large amount of control over detail, shading, and effects that cannot be achieved with "line and fill" coloring-book graphics. You will probably want to start most pictures by laying down a background with lines, then adding most of the colors with fills, and finally adding the detail touches with brushes.

There is no "brush up/brush down" selection that lets you cover a wide area. Each time you press the button the brush plots just once. If you want to move across an area, you have to keep pressing and releasing the button. While strange-seeming at first, remember that the computer is remembering your moves. If the brush were constantly down, it would have to remember each and every point that you move over, wasting a lot of memory very quickly.

Other Quick and Easy Options, Including Saving Your Picture

While drawing, you have these other following choices available at all times. The letter commands are listed if you press "(**H**)elp".

The **ESC** key switches between graphics and text display (the normal mode, with the command lines at the bottom of the screen), and full-screen graphics mode. It has no effect on the picture itself, since the graphics area under the text is always available.

"(**R**)edraw" will reconstruct the picture as it would be seen from a program. This is handy if you are trying for some animation (explained later), and most people admit that it's fun just to see what you have drawn redone at blinding speed by the computer.

"(**S**)ave" allows you to save your picture in the special compact format created by **The Graphics Magician**. To view this picture later, you will have to load it back into the picture editor, or follow the instructions for using the DPICDRAW routine in chapter 4.

The "(**S**)ave" command only saves the displayed portion of the picture, which should be remembered when in edit mode, described below. This allows a way to extract parts of pictures, if desired.

"(**I**)mage" saves a memory image of the picture displayed on the screen. This saves the full 16K screen area as two 34-sector binary files, using the suffix ".DPM" for the main memory image and ".DPA" for the auxiliary memory image. It is normal for the displayed picture to look peculiar during an "(**I**)mage" save. This is because the auxiliary memory image has been moved into main memory, overwriting the image there temporarily. The "(**I**)mage" save may be useful in situations where you want to use the picture but not the DPICDRAW routine.

Note that there is NO way to edit a picture previously saved in 16K memory image format with this picture editor. Pictures created with other graphics editors cannot be converted to the special compact format of **The Graphics Magician**. Since it is the moves that are saved, the pictures must be created with **The Graphics Magician Picture Painter** to have this compact format.

"(**Q**)uit" lets you return to **The Graphics Magician Picture Painter** menu. You are asked if you want to "(**S**)tart over", in case you want to start a new picture, or go to the "(**M**)enu". Any other key returns you to the editor.

Adding Text

You can add text to your picture at any time by positioning your cursor where you want the text to start and pressing "(**T**)ext". Whatever you type now will be overlaid onto your picture at that point. The text mode takes control of the cursor, and you have the following options, which are listed at the bottom of the screen:

First, you may release the **CAPS LOCK** key if you wish to enter upper and lower case. When leaving text mode, you should re-engage the **CAPS LOCK** key.

"(**CONTROL-R**)everse" and "(**CONTROL-N**)ormal" let you choose between reversed letters and normal letters in color. Reversed letters flip whatever dots are already on the screen. Normal letters plot in the current brush/fill color. The current type used is listed at the bottom of the screen. Which type is better depends on the background color and letter color desired. Experimentation is the best way to judge. The sharpest results come from a combination of black and white for background and text.

ESC toggles to full-screen mode and back, just as before.

Back Arrow or **DELETE** allow you to backspace and delete the last character.

RETURN, or reaching the end of line, returns you to the normal drawing with cursor control in line mode. If you reach the end of a line, you'll hear a beep.

You may reenter text mode at the last text cursor position by pressing "(**CONTROL-T**)ext" while in normal drawing mode. Pressing "(**T**)ext" by itself always puts you in text mode at the current position of the controller cursor.

Edit Mode

Since the picture editor saves pictures as a set of moves, it is possible to go back and edit those moves, much like a computer program itself. Edit mode allows you to single-step forward and backward through a set of picture commands, displaying what the picture looks like at each point, and allowing you to delete or add moves at any time. If you decide later that you don't like a color, find where you set that color and set a different one. If a house needs a few extra lines before the color is filled in, backstep to before the color was added and put in the lines. Easy!

Pressing "(**E**)dit" while in normal drawing mode will clear the screen to white and position you in your "picture program" at the first move. Each time you press the **right arrow** the next move in sequence will be displayed in words at the bottom of the screen and performed to the picture. Pressing the **back arrow** will back up one step. Your entire picture is still stored in memory, and pressing "(**R**)edraw" will bring it all back, but the edit mode allows you to add to or delete things that you did early in drawing your picture and see, step-by-step, how it was constructed.

In edit mode, all drawing commands remain usable, and anything you draw while in edit mode will be inserted into your picture. You can also use the **DELETE** key just as before to remove commands. Saving while in edit mode only saves what is visible on the screen. To be sure that you have the entire picture, just press "(**R**)edraw".

It is also possible to "back into" edit mode from the end of a picture by using the **back arrow.** Stepping forward through a picture is faster, but if you have a long picture and want to edit one of the last moves, this makes it easier.

"(**R**)edraw" will always let you out of edit mode by redrawing the entire picture stored in memory. You will also get out of edit mode if you single-step through the entire picture and reach the end.

"(**S**)ave", when used in edit mode, will only save the part of the picture that is displayed. This is a convenient way to save only the first half of a picture, for instance. If you want to save the whole picture, you should use "(**R**)edraw" to get out of edit mode.

While single-stepping in edit mode, you can speed up by using the **TAB** key to tab forward 10 steps at a time.

Graphics Resolution Modes - Thick and Thin Lines

When you start, you are put in "560-mode", which means you are using the full capabilities of the double-res 560-dot resolution, including all 256 blended colors. The "560" in parentheses in the first command line tells you that you are in this mode. There are other options.

First, when using 560-mode, you've got two choices for line width. Thin lines use the full 560-dot resolution, but any point on the line is only considered "on" or "off", meaning a lighted dot, or black. Thick lines can truly be drawn in any of the 16 line colors. The lines are actually four dots wide. Why? Because in the color mode, there are only 140 color-dots across the screen, each defined by four screen dots. Use of the thick lines or thin lines depends only on how fine you want your lines, and if you want them in color. Use "(**C**)hange Line Mode" to toggle back and forth between the thin lines and the thick colored lines.

Graphics Resolution Modes - 280 vs. 560 Mode

Second, there is also a 280-mode in the double-res picture painter. This 280-mode corresponds to the resolution and colors of an Apple without double-res features. Use "(**M**)ode" to toggle between 280-mode and 560-mode.

In 280-mode, the color palette is trimmed to 108 colors, matching those from the standard hi-res **Graphics Magician.** The colors are not the same, however. The simulated yellow that you would see in standard hi-res is replaced with the pure yellow in double-res. The same with brown, magenta, pink, and light and dark shades of green and blue. We set up a color map that would interpret the blended 108 colors from standard hi-res as much purer double-res colors. The result is that you can take any picture created with the standard hi-res version of **The Graphics Magician**, read it into the 280-mode double-res **Graphics Magician**, and see the same picture redrawn automatically in the purer colors. Similarly, you can take pictures drawn in the 280-mode double-res **Graphics Magician**, and display them with the standard hi-res **Graphics Magician** on Apples that do not support double-res.

Changing between 280-mode and 560-mode

As stated, "(**M**)ode" toggles between 280 and 560-mode.

If you want to load a picture done with the standard hi-res **Graphics Magician**, or if you want to create a picture that can be used with the standard hi-res **Graphics Magician**, go to 280-mode. The files read and created in 280-mode have the suffix ".SPC" appended to their names. (SPC for Sequential PiCture).

If you want to take a picture created with the standard hi-res **Graphics Magician** and convert it to 560-mode to take advantage of the higher resolution and all 256 colors, you should load the picture, then switch to 560-mode. All the calculations and color-mapping to change the coordinates and color numbers are done for you. When you save the picture now, it will have the suffix ".DPC", (for Double-res sequential PiCture).

You may also take pictures created in 560-mode and convert to 280-mode, by loading, then changing modes, but the conversion isn't as flawless. Any of the colors from the 256-color palette that are not mapped to the 108-color palette are assigned a color number of 0. Because this can affect the picture, when switching to 280-mode you are always asked to verify the choice.

Drawing for Use in Double-res and Standard Hi-res

The editor has features that make it easy to create picture files that can be interpreted by the Double-res DPICDRAW and by the standard hi-res PICDRAW. This makes it possible to have one program and one set of picture files operate on any Apple, using the double-res features only if available on the particular computer. These pictures should always be created in 280-mode. While in 280-mode, you can look at how the picture would appear in standard hi-res by pressing "(**N**)ormal". This interprets and displays the picture using the standard PICDRAW routine in the standard hi-res mode. Any key returns you to 280-mode double-res.

Note that when creating pictures for use on both standard hi-res and double-res, you will have to deal with some of the limitations of standard hi-res. Because of color blendings, some fills will act differently in standard hi-res, and you may also have to contend with color-bleeding (see Appendix B on color). This is why the "(**N**)ormal display" feature is included: to help see these problems immediately while you are working on the picture.

A Fill Anomaly

There is a seldom occurring instance where the fill routine may seem to refuse to fill an area. Here are the circumstances, and the solution to the situation.

Figure 2.2 shows part of a drawing where the top and left side have been filled by a color. Since this is printed matter, the lines appear in inverse, but assume that the solid fill lines are white. A few colors in the palette are blended so that alternate lines are solid white, and those are the ones that can cause this occurence.

Now suppose that you try to fill the area on the right. The fill routine will search upward and find the bottom white line of the area already filled, assume it is also to be filled, and proceed. Since that line stretches much further to the left, when the routine averages the endpoints and attempts to step down to the next line to fill, it will step down on the left side, which already is filled!

The solution is to break the white line extending across the top of the fill area by placing a ~~single~~ black dot atop the triangle in that line. The fill routine will now average the endpoints so that the area that needs to be filled is filled.

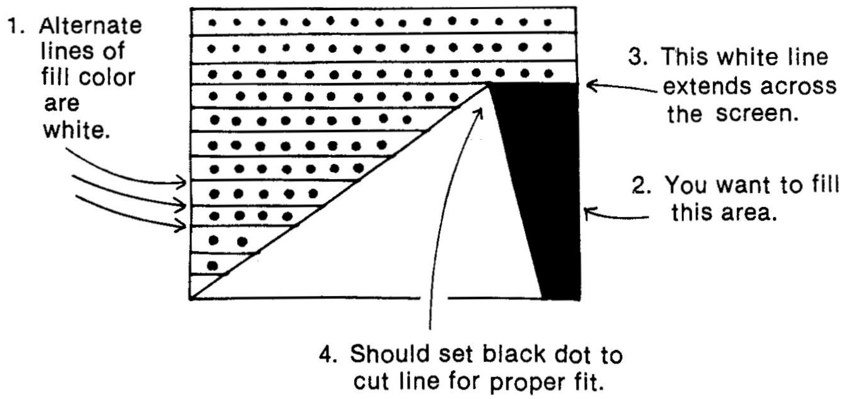

1. Alternate lines of fill color are white.
3. This white line extends across the screen.
2. You want to fill this area.
4. Should set black dot to cut line for proper fit.

Figure 2.2 - Fill Anomaly

Chapter 3 - Tricks with Pictures

Objects

One of the features of many programs that require compact pictures is the ability to move objects from picture to picture, or to draw a picture with an object sometimes appearing, sometimes not. The obvious example is an adventure game, where something will appear in a picture, you take it with you, and thus it should no longer appear in the picture. We'll call this type of thing an "object".

Objects with **The Graphics Magician** are actually the same as pictures. You create them in generally the same way as you would any other picture. However, in your own program, when you use the picture-reconstructing "DPICDRAW" routine, you tell it to draw that "picture" as an overlay at a certain coordinate. The picture thus becomes an object in the picture previously displayed.

The only requirement for objects that is not necessary for pictures is that the first command is a "Start Line" command. This is really a dummy "Start Line" command, and will later be interpreted as a "Start Object at" command for object positioning. If you actually need a "Start Line" command, you must use a second one.

With objects you should be careful about when you use a fill command, if one is used at all. Since the fill command requires a white background, you must be sure that the background picture will provide a white fill area. If you use a fill command you should first "white out" the area with one of the larger brushes.

For detail, most objects are done primarily with brushes and some careful use of lines. Some prefer to set down a white background with brushes first, no matter what, so that all the drawing commands can be used freely. It's usually a matter of style.

Creating your Object over a Background

To make it easier to choose colors and see how your object will look, you can draw it directly over a background picture. You may load a background by pressing "(**B**)ackground load" when first entering the picture editor. If you load a background, you may draw your object directly on top of it. The background does not become part of your object, nor does it affect it in any way. Each time the screen is cleared, the background is displayed in place of the blank screen that normally is shown.

Animation with Pictures

One of the effects discovered after the original **Graphics Magician** was completed was that of animation created with the picture editor. Suppose you draw an entire picture, and in the picture is a man. Your commands in creating that picture are saved, so each time you view the picture, you'll see it redrawn. Now, suppose that once the picture is complete, you draw more right on top of what you finished. For example, draw the man's eyes closed, then go back and draw them open again. When you press "(**R**)edraw", you'll see the man being drawn to completion, then his eyes will blink! You can extend it out further and keep drawing over and over the original picture and make all kinds of things in the room "animate".

This effect can be accomplished by drawing continuously on top of a picture, or it can be done by drawing a sequence of objects over your picture. The latter method has the advantage of allowing more control in timing, even tying it to user responses in your program. To do it with objects, you might load in the background of the man, then draw one object of his closed eyes, and another of his eyes open again. Or you might make one object his closed eyes quickly overlaid by his open eyes, since it's an immediate progression. In your program you'd draw the background picture, then whenever you wanted his eyes to blink, you would draw your blinking eye object(s). You could have it controlled by time, or have it happen every time someone touched a key. The flexibility is yours!

Chapter 4 - Using Pictures in your Programs

When you created your pictures in the picture editor, what was saved was your moves in drawing the picture. To display it, you need some way to tell the computer to recreate those moves. The machine language routines called PICDRAW and DPICDRAW do just that. DPICDRAW is the double-res version.

Put the DPICDRAW routine on your disk

First, you must move the DPICDRAW routines (they are split into two parts) from **The Graphics Magician Picture Painter** disk to your program and data disk. To do so, quit **The Graphics Magician** so that you get the "]" prompt, then use the following steps:

1) Put in your **Graphics Magician Picture Painter** disk, type

```
BLOAD DPICDRAWH,A$2000
```

and press **RETURN**. Then type

```
BLOAD DPICDRAWL
```

and press **RETURN**.

2) Put in your disk and type

```
BSAVE DPICDRAWH,A$2000,L$F00
```

and press **RETURN**. Then type

```
BSAVE DPICDRAWL,A$9500,L$62
```

and press **RETURN**.

Using DPICDRAW

After DPICDRAWL and DPICDRAWH are moved to the same disk as your picture files, you can write a short BASIC program to display the pictures, such as that in listing 4.1.

```
5 HIMEM: 32768
10 X = PEEK ( - 16254) + PEEK ( - 16255) + PEEK
   ( - 16255)
15 REM WRITE-ENABLE LANGUAGE CARD
20 PRINT CHR$ (4);"BLOAD DPICDRAWL"
30 PRINT CHR$ (4);"BLOAD DPICDRAWH,A$D000"
40 X = PEEK ( - 16254): REM WRITE-PROTECT CARD
50 PRINT CHR$ (4);"PR#3": PRINT
60 HGR : CALL 38202
65 REM MOVE BLANK MAIN MEM TO AUX MEM
70 PRINT CHR$ (4);"BLOAD PNAME.SPC,A32768"
80 POKE 0,0: POKE 1,128
90 CALL 38144
```

Listing 4.1 - Using DPICDRAW from BASIC in 280-mode

This example shows how to display a 280-mode (.SPC) picture. To display a 560-mode picture, change line 70 to use ".DPC" instead of ".SPC", and add:

85 POKE 38175,255

which tells the DPICDRAW routine that the color conversion routines for 280-mode pictures need not be used.

To go back to displaying 280-mode pictures, use

POKE 38175,0

The HIMEM command makes sure that the DPICDRAW routine and the area in which you loaded your picture will not be disturbed. The DPICDRAW routine starts at location 38144 ($9500), and we'll load the picture commands at 32768 ($8000). Line 10 sets the language area (the top 16K of main memory) to receive information. Lines 20 and 30 load in the two DPICDRAW files. Line 40 protects the language area from being written over. Line 50 activates the auxiliary 64K memory and 80-column display. Line 60 sets the standard hi-res mode, and then moves the contents of hi-res graphics page 1 into the hi-res data area in the auxiliary memory, using a transfer routine in DPICDRAW. Line 70 loads in the picture information. Note that the ".SPC" or ".DPC" was added to the name automatically by the picture editor, and that the ",A32768" tells the program to load the picture commands at location 32768.

Line 80 pokes the starting address of the picture commands into locations 0 and 1. The address is poked in Lo/Hi format, which is computed as follows:

Hi=INT(A/256)
Lo=A-HI*256

where A is the address. Listing 4.2 shows how we could have used this formula in line 80. Line 90 contains the call to the DPICDRAW routine at 38144 (or from machine language, use JSR $9500) that will cause the picture to be redrawn.

```
80 A = 32768:HI =  INT (A / 256):LO = A - HI * 256
    : POKE 0,LO: POKE 1,HI
```

Listing 4.2 - Computing the POKES

Putting an Object over a Picture

The same technique used for redrawing a picture is used to redraw an object. Once the background picture is shown, load the object picture, and use the same two pokes to tell the location of the object commands. Then you have a choice between two calls.

CALL 38147 (or JSR $9503) will draw the object over the picture at the same location in which the object was originally created. It is the same as drawing a picture, except the screen is not cleared beforehand.

CALL 38150 (or JSR $9506) will allow you to specify an x,y coordinate at which to have the object drawn. The x,y coordinate should first be put in locations 38153-38155 as follows:

```
POKE 38153,X-INT(X/256)*256
POKE 38154,INT(X/256)
POKE 38155,Y
```

Technically, "xlo" is put in 38153 ($9509), "xhi" is put in 38154 ($950A), and "y" is put in 38155 ($950B). **IMPORTANT:** for compatibility with programs written for the standard hi-res screen, the x,y coordinate is given based on a 280 by 192 screen when in 280-mode. If you are using 560-mode the x-coordinate can be from 0 to 559.

If the second call is used, the x,y coordinate must be such that the entire object will be shown on the picture. If any part of the object goes off the edges, it will not work properly.

Listing 4.3 can be used as a continuation of listing 4.1, and shows an example of drawing an object over a picture using alternate coordinates. If it should be drawn in its original location, line 130 could be omitted and the call in line 140 could be changed to CALL 38147.

```
100 PRINT CHR$ (4);"BLOAD ONAME.SPC,A32768"
110 POKE 0,0: POKE 1,128
120 X = 200:Y = 53
130 POKE 38153,X - INT (X / 256) * 256: POKE 38154
   ,X / 256: POKE 38155,Y
140 CALL 38150
```

Listing 4.3 - Adding an Object

Note that there is no restriction on mixing 560-mode and 280-mode picture files here, provided you let DPICDRAW know what you are doing. Just remember to POKE 38175,255 for ".DPC" (560-mode) files, and to POKE 38175,0 for ".SPC" (280-mode) files before you make the call to DPICDRAW.

Note about Graphics Pages

There is only one double-res graphics page. It consists of an 8K block of main memory from $2000 to $3FFF, and an 8K block of auxiliary memory, also mapped into $2000 to $3FFF. The standard hi-res **Graphics Magician** lets you switch between displaying your pictures on standard hi-res page 1 or standard hi-res page 2. DPICDRAW ignores this, since there is only one double-res page.

Technical Trivia

Before a picture is drawn, you must put the starting address of the picture file into locations 0 and 1, in Lo/Hi format. After the picture is drawn, locations 0 and 1 contain the first address after the picture commands.

Chapter 5
- Advanced use of DPICDRAW

Loading Groups of Pictures into One File

You can put several pictures in one long file, so that you only load one time, but can display several different pictures without going back to disk. This was done with the demo program on your disk so that all the pictures would load at once and the disk would not have to be accessed between each picture.

First, find the length of each picture you plan to use. This can be done by noting the "BYTES USED" in the picture editor for that picture. Suppose the results were as in Figure 5.1.

Figure 5.1 - Sample Picture Lengths

Name	Length	Load At
House .SPC	1254	16384
Tree .SPC	879	17638
Moose .SPC	2318	18517

You next need to choose a starting location for your group of pictures. For example, we'll start at location 16384 ($4000). Now, load each of your pictures sequentially in memory.

For the first, you would

```
BLOAD HOUSE.SPC,A16384.
```

which loads the picture at 16384.

For the second, add the length of "House" to 16384 to find the next available space. 16384+1254=17638, so we load the second picture, in this example, at 17638.

```
BLOAD TREE.SPC,A17638
```

Remember the location at which you load each picture, since that's the address you must poke into locations 0 and 1 before redrawing the picture.

The third file, MOOSE.SPC, is loaded at 17638+879=18517.

BLOAD MOOSE.SPC,A18517

Now, compute the length of the picture sequence by adding each of the lengths together (1254+879+2318=4451), and save the entire file with

BSAVE PICTURE GROUP,A16384,L4451

substituting the name you want for PICTURE GROUP, and the appropriate starting address and length for your picture.

Using a Group of Sequential Pictures

Listings 5.1 and 5.2 give examples of using picture groups in your programs. Listing 5.1 cycles through the pictures in order, waiting for a keypress between changes. Listing 5.2 uses the addresses we noted to allow you to select which picture is shown, and when. Pressing **1**, **2**, or **3** draws the appropriate picture. The programs should have names and addresses changed to match the ones you used before you try them, and lines 100 in listing 5.1 and 80, 90, and 100 in listing 5.2 should have the "3" changed to the number of pictures you used.

```
5 HIMEM: 8192
10 X =  PEEK ( - 16254) +  PEEK ( - 16255) +  PEEK
   ( - 16255)
20  PRINT  CHR$ (4);"BLOAD DPICDRAWL"
30  PRINT  CHR$ (4);"BLOAD DPICDRAWH,A$D000"
40 X =  PEEK ( - 16254): REM  WRITE-PROTECT CARD
50  PRINT  CHR$ (4);"PR#3": PRINT
60  HGR : CALL 38202
70  PRINT  CHR$ (4);"BLOAD PICTURE GROUP,A16384"
80  POKE 0,0: POKE 1,64
85  REM  LINE 80 CONTAINS 16384 PRE-COMPUTED IN L
   O-HI FORMAT
90  POKE 38175,255
95  REM  LINE 90 IS FOR 560-MODE PICTURES
100  FOR X = 1 TO 3
110  CALL 38144: GET A$
120  NEXT
```

Listing 5.1 - Cycling through a Group of Sequential Pictures

```
5 HIMEM: 8192
10 X = PEEK ( - 16254) + PEEK ( - 16255) + PEEK
   ( - 16255)
20 PRINT CHR$ (4);"BLOAD DPICDRAWL"
30 PRINT CHR$ (4);"BLOAD DPICDRAWH,A$D000"
40 X = PEEK ( - 16254)
50 PRINT CHR$ (4);"PR#3": PRINT
60 HGR : CALL 38202: POKE 38175,255
65 REM CLEAR SCREEN AND SET "560" MODE
70 PRINT CHR$ (4);"BLOAD PICTURE GROUP,A16384"
80 DIM L(3): REM L CONTAINS THE LOCATION OF EAC
   H PICTURE
90 FOR X = 1 TO 3: READ L(X): NEXT : DATA 16384,
   18425,20655
100 GET A$: IF A$ < "1" OR A$ > "3" THEN 100
110 V = L( VAL (A$))
120 POKE 0,V - INT (V / 256) * 256: POKE 1,V /
   256
130 CALL 38144
140 GOTO 100
```

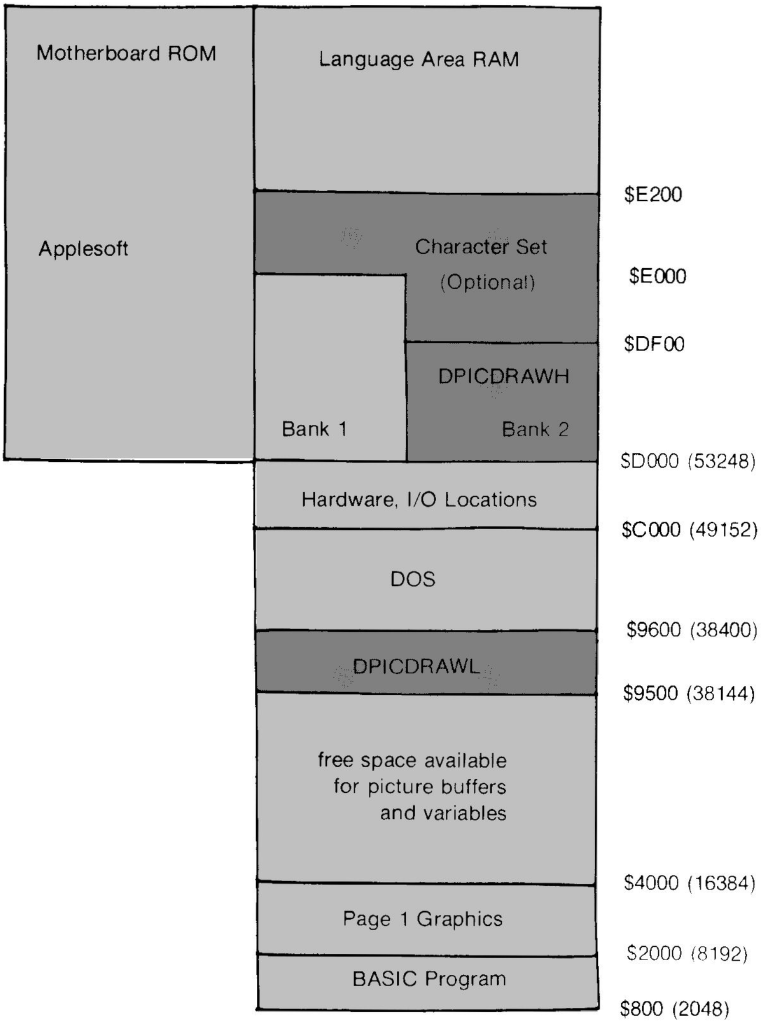
Listing 5.2 - Choosing from a Group of Sequential Pictures

Memory Usage

DPICDRAWL takes 98 bytes, and resides in locations 38144-38241 ($9500-$9561). DPICDRAWL serves as a scratchpad area for DPICDRAWH, and a place where your programs can pass values to DPICDRAWH. Also, DPICDRAWL switches the 16K language area in and out each time you call it, because DPICDRAWH is placed there to leave more free space for your programs.

DPICDRAWH is where all of the main routines are. It is 3840 bytes long, and resides at 53248-57087 ($D000-$DEFF). Figure 5.2 shows a memory map of where everything is located when using DPICDRAW.

Your picture buffer (where the picture commands get loaded) may be placed anywhere where there is room, but the usual place is directly below the DPICDRAW routine. To find the highest location you may use for the buffer, find the length of your longest picture (from the picture editor) and subtract it from 38144, the starting location of DPICDRAWL. Sometimes it is useful to actually use two picture buffers. You may want one for "room", or background, pictures, and another separate buffer for objects.



DPICDRAW also uses zero-page locations $0-$C,$F9,$FB-$FF (0-12,249,251-255).

DPICDRAW uses $2000-$3FFF, the double-res display area, in the auxiliary 64K of memory.

Figure 5.2 - DPICDRAW Main Memory Map

There is a second version of DPICDRAW available in case you do not want it to use the language area (for example, if you will be using ProDOS, which resides in the same area). The alternate version is called DPICDRAW.ALT, and it starts at $800 (2048) and is exactly 4K long (to $17FF, or 6143), not including the character table.

Listing 5.3 shows an example of using DPICDRAW.ALT from Basic. It is virtually the same as listing 4.1, except for the code in lines 10 and 20, and the CALL address is moved down to 2106. Lines 10 and 20 cause the Basic code to relocate itself to $4000, or 16384. The CALL, and all other access points in DPICDRAW.ALT, are 36096 ($8D00) less than those for the standard version of DPICDRAW, described earlier.

```
5  HIMEM: 32768
10  IF  PEEK (104) = 64 THEN 40
20  POKE 104,64: POKE 16384,0: PRINT  CHR$ (4);"RU
    N LISTING 5.3"
30  REM  MOVE PROGRAM OUT OF DPICDRAW.ALT'S SPACE
40  PRINT  CHR$ (4);"BLOAD DPICDRAW.ALT"
50  PRINT  CHR$ (4);"PR#3": PRINT
60  HGR : CALL 2106
65  REM  MOVE BLANK MAIN MEM TO AUX MEM
70  PRINT  CHR$ (4);"BLOAD PNAME.SPC,A32768"
80  POKE 0,0: POKE 1,128
90  CALL 2048
```

Listing 5.3 - Using DPICDRAW.ALT

Adding a Character Table

As it comes, DPICDRAW and its character set are in separate files. If your application requires text on the graphics screen, there are two ways to install DPICDRAW's character table:

When loading the DPICDRAW files, add:

```
25 PRINT CHR$(4) "BLOAD SYS.STS,A$DF00"
```

or add the table to the end of DPICDRAWH:

```
BLOAD DPICDRAWH,A$2000
BLOAD SYS.STS,A$2F00

BSAVE DPICDRAWH.WT,A$2000,L$1200
```

The ".WT" means "with text", and will help you remember that this version has a character table added.

To use text with DPICDRAW.ALT, you would use the following with listing 5.3:

```
45 PRINT CHR$(4) "BLOAD SYS.STS,A$1800"
```

or add the table to the end of DPICDRAW.ALT with:

```
BLOAD DPICDRAW.ALT
BLOAD SYS.STS,A$1800
BSAVE DPICDRAW.ALT.WT,A$800,L$1300
```

Changing the Character Table

The character table may be changed if you have a small font editor such as that available with **The Complete Graphics System** from Penguin Software or **The Applesoft Tookit** from Apple Computer. Several alternate text fonts are also available in **Additional Typesets** from Penguin Software.

To use an alternate text set, simply use the name of the character set you want in place of SYS.STS in the previous section, "Adding a Character Table". To change the character set on your editor disk, on a copy of the disk replace SYS.STS with your character set. Its name must be changed to SYS.STS for the editor disk.

Chapter 6 - The Picture Lister

Pressing "(**L**)ist picture" from the menu gets you to a picture listing utility. Normally, this will be of little use, but it does have a few interesting applications.

First, you can print out on your printer a list of commands that verbally describe a picture. These are the same commands that are listed individually at the bottom of the screen while you are editing a picture.

More importantly, the Lister lets you list your picture commands to a transfer file. Normally a picture's commands are stored in binary format for speed. When the commands are taken apart and listed to a transfer file, a few things can happen. First, it becomes possible to transfer your pictures to other types of computers, and for them to interpret the pictures in a way best suited for display on that system. Second, it becomes possible to reload the picture on your own Apple, altering various parameters in the translation back to sequential format.

Simple Options

Briefly going through the options, "(**L**)oad" and "(**S**)ave" let you load and save sequential pictures from **The Graphics Magician** to and from disk. "(**P**)rint picture commands" lets you print the instructions that make up a picture to your printer. "(**V**)iew picture", "(**D**)raw picture", "(**C**)atalog", and "(**Q**)uit" do just as they say.

"(**W**)rite transfer file" takes the current picture loaded in memory and puts its commands into a **Graphics Magician** picture transfer format. The commands are written out so that they can be interpreted by other Apples, and by other computers such as those from Atari, Commodore, and IBM.

Reading and Interpreting Transfer Files

"(**R**)ead transfer file" lets you read and interpret a picture transfer formatted file. This file can be created with an Apple, as in above, or it could be a graphics file created with another computer. In either case, you are first asked to enter a scaling factor. Standard Apple picture files are written out with a scaling factor of 100; that is, all x,y coordinates are multiplied by 100 before being written to the transfer formatted file. To read a standard Apple picture back in, you would want to divide each coordinate by 100, therefore enter 100 for the scale factor. This should be done with any 280-mode ".SPC" picture.

In a Double-Res picture, the x-coordinate can be from 0 to 559. The x-coordinate multiplier, then, is 50. The y-coordinate multiplier (100) stays the same. To read in Double-Res picture, you would divide the x-coordinates by 50, and the y-coordinates by 100. This should be used with all 560-mode ".DPC" pictures.

Other computers do not necessarily have the same 280 or 560 dot by 192 dot resolution of an Apple. Files written out by other computers may use different scaling factors, or require that different scaling factors be used for interpreting their files so that the end result is a picture that is proportional to a full screen on an Apple.

You may also use the scaling factor to play tricks with the physical size of pictures created with an Apple, making them larger or smaller. One thing to remember is that the brush and text sizes do not change, so whereas the line and fill commands will actually condense or expand properly in size, brushes and text will be the same size, but move apart or scrunch together.

For those interested in playing around with the actual code for interpreting the pictures, the instructions for reading and interpreting a picture file are in lines 500-600 of the program DPICDOC, with the specific commands for interpreting the coordinates at line 525. You can do some pretty weird and interesting things by playing with that part of the code.

Color Table

The other intepretation that occurs when reading a file is that of color. Different types of computers differ greatly when it comes to color availability, so a method is provided to interpret colors as best possible across systems. When the interpreter reads a color from the color table, instead of automatically assigning that color number, it looks in a color table to find the equivalent that should be used.

The existing color table uses 16 line color equivalences, and 256 brush/fill color equivalences, with each number equivalent to itself (in other words, color 12 is interpreted as a 12).

Pressing "(**E**)dit color table" will give you a set of choices that allows you to load or save a color table of your choice, print the current color table equivalences to a printer, load the Double-Res Apple equivalence color table (named AP560.CLT, which is loaded in automatically when the program is run), change any one of the color equivalences, or return to the main options.

If you choose to change a color equivalence, you'll be asked which number you want to change, and what you want it to be. If you choose to change fill color 5 to 17, whenever a picture transfer file is read in and interpeted, any fill color 5 will be interpreted as a 17. Besides being irreplaceable as a tool for transferring colored pictures between two different types of computers, this will also allow a quick way to change colors in any picture created in an Apple. It lets you bypass the need for going into the picture editor and single-stepping to find all the color commands that you want changed. The whole step is done automatically when you write a picture transfer file, change the color equivalences, then read and interpret the same file with the new colors. In fact it's so quick that is allows you to easily test several diffferent color combinations for the same picture very painlessly.

Customizing the 280/560 Mode Color Conversions

There is one other use to which the color table editor can be put. DPICDRAW uses a color lookup table to convert picture commands from 280 data mode to 560 data mode, and vice versa. The default color table for 280 mode to 560 mode conversion is AP280.CLT. DPICDRAWH contains a copy of this table, as does SYS.CLT. Everytime you enter the main menu, however, the contents of SYS.CLT are overlaid onto DPICDRAWH. Using DPICDOC, you can get, edit, and save SYS.CLT, and that way change how DPICDRAW interprets 280-mode files.

Here's an example: In the 280-mode palette, color number 2 (remember to start counting at color number zero) is defined as solid yellow, which is color 112 in the 560-mode palette. You decide that you'd prefer to have DPICDRAW interpret color #2 as a lighter yellow, and you pick color #126 (yellow and white) from the 560-mode palette as a substitute. From DPICDOC, you would load SYS.CLT, and choose "(**E**)dit color table". (Note that all ".CLT" files are kept on the master disk, usually in drive 1.) Then choose "(**C**)hange fill color table". DPICDOC will ask you which color to change. Enter a 2. Then DPICDOC will ask what the new value should be. Enter 126. That's it! To make these changes permanent, "(**S**)ave color table", and enter "SYS" as the name. This will save your new table as SYS.CLT. It's advised that you always do this onto a COPY of the master, so that your real master disk still has the original SYS.CLT.

If you want to make a lot of changes to the 280-mode palette, you may want to load the palette, and redraw it occasionally to be able to see what you're doing. The name of the palette is PAL.280.SPC. Just select "(**L**)oad picture", "(**2**)80 mode", then enter "PAL.280". If you have changed the lookup table, you will see the difference when the palette is drawn. You may make more changes with DPICDOC, then "(**D**)raw" the palette again to see the changes as often as you like.

If you ever want to restore SYS.CLT to its original state, "(**L**)oad color table" AP280, and "(**S**)ave" it as SYS. AP280.CLT is the backup copy of the standard table, so it's probably best not to change it or save over it.

Finally if you want to create a custom DPICDRAW with your color lookup table, from Applesoft type:

```
BLOAD DPICDRAWH,A$2000
BLOAD your table .CLT,A$2CF0
BSAVE DPICDRAWH.MC,A$2000,L$F00
```

The ".MC" stands for "modified colors", and tells you that this DPICDRAW will interpret 280-mode pictures in a non-standard way.

Chapter 7 - Extras

The Binary Transfer Utility

Choosing "(**B**)inary transfer" from the menu lets you use the binary transfer utility. On the Apple, there are four types of storage files, designated by a letter in front of the name when you do a disk catalog. "A" files are Applesoft BASIC files. "I" files are Integer BASIC files (not used much anymore). "T" stands for text file, and "B" is a binary file.

BASIC files are easy to move around; you just LOAD them and SAVE them. The computer does all the work with location and length. All the files you create with **The Graphics Magician**, though, are binary files. They either contain binary data or machine language code. To move a binary file, you must know its starting address and length (in bytes). The binary transfer utility lets you easily find these. It will also let you automatically load a binary file from one disk and save it to another disk.

The options from the binary transfer utility are to "(**L**)oad", "(**S**)ave", "(**D**)isk catalog", or "(**Q**)uit". Each time you load a binary file, its starting address and length are automatically displayed both in decimal and hexadecimal. Once you load a file, you don't necessarily have to save it. You would do so only if you want to put it onto another disk.

With a couple commands from Applesoft it is possible to change the starting location (and thus, loading location) of a binary file. Generally you should not do this with machine language program files, since these are usually dependent on location. You may easily make the change with binary data files, such as pictures, shapes, and paths, though. If you have a sequential picture called HOUSE, for example, that was 387 bytes long and you want to change its loading address to 24800, you would use the following from the Applesoft prompt (]):

```
BLOAD HOUSE.SPC,A24800
BSAVE HOUSE.SPC,A24800,L387
```

Demo

A demonstration of **The Graphics Magician Picture Painter** with some sample picture files is provided on your disk. To see it, choose Demo from the menu.

Double-res Text Generator

A small character generator has been included that will let you print text on the double-res graphics screen from your programs. It is called DHPRINT, and it loads just above DPICDRAWL.

The routine itself will simply plot an ASCII character in the range 32-127 at a specified x.y coordinate. Y may be anywhere in the range 0-184, X must be 0-39. To use the routine, first load it, then, given x,y coordinate and an ASCII value A, put the ASCII value in 38242, put X in 38243, put Y in 38244, and CALL 38245. (Hexadecimal addresses $9562-$9565). A sample is shown in listing 7.1, and an ASCII table can be found in your Applesoft manual. The numerals 0-9 have ASCII values of 48-57, and the letters A-Z have values 65-90.

```
5  HOME
10  REM  ASSUMES DPICDRAW AND A CHARACTER TABLE AR
    E LOADED AND READY
20  HGR : CALL 38202
30  PRINT  CHR$ (4);"BLOAD DHPRINT"
40  GET A$
50 A =  ASC (A$)
60  IF A = 13 THEN Y = Y + 1:X = 0: IF Y > 19 THEN
    Y = 0
65  IF A = 8 THEN X = X - 1: IF X < 0 THEN X = 0
70  IF A = 3 THEN  END : REM  CONTROL-C QUITS
75  IF A < 32 OR A > 127 THEN 40
80 X = X + 1: IF X > 39 THEN X = 1:Y = Y + 1: IF Y
     > 19 THEN Y = 0
90  POKE 38242,A: POKE 38243,X: POKE 38244,Y * 8
100  CALL 38245
110  GOTO 40
```

Listing 7.1 - Using DHPRINT

Note that DHPRINT uses the same character set and lookup tables used by DPICDRAWH. DPICDRAWH and a character set should be in memory before using DHPRINT.

DPICDRAW.ALT already has a version of DHPRINT built in. To use it, just subtract 36096 ($8D00) from the addresses given above.

Appendix A - Apple Double-Res Colors

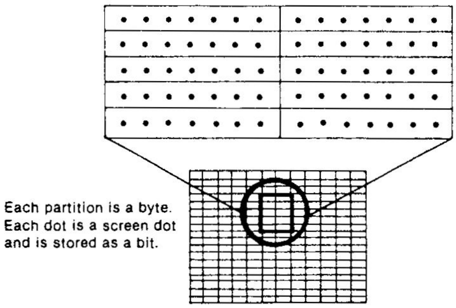


Figure A.1 - Section of Apple Graphics Screen

To fully take advantage of the double-res color graphics ability, it helps to know just how those colors are stored on the double-res screen. Figure A.1 shows a magnified portion of the graphics screen. Each dot corresponds to a dot that may be on or off and each is technically stored as a bit. The lines divide the basic storage units on the Apple, bytes. Each byte consists of seven bits displayed horizontally on the screen. The eighth bit in each byte, used as a color-select bit in standard hi-res, is ignored in double-res. When the 8K main memory is displayed simultaneously with the 8K auxiliary memory, we get a screen that is 80 bytes, or 560 dots, wide. As with standard hi-res, the screen is 192 bytes, and dots, tall.

Each dot may only be on or off. An "on" dot may be one of four colors, depending on its horizontal position on the graphics screen. For color purposes, the 560 dots going across the double-res screen are divided into groups of four dots. This unit of four dots, or "color group", is capable of 16 combinations, or 16 colors. See figure A.2.

	Color Group
Black	0 0 0 0
Magenta	0 0 0 X
Brown	0 0 X 0
Orange	0 0 X X
Dark Green	0 X 0 0
Grey 1	0 X 0 X
Light Green	0 X X 0
Yellow	0 X X X
Dark Blue	X 0 0 0
Violet	X 0 0 X
Grey 2	X 0 X 0
Pink	X 0 X X
Blue	X X 0 0
Light Blue	X X 0 X
Aqua	X X X 0
White	X X X X
X = on	0 = off

Figure A-2 - Double-Res Colors

This new color layout has a few interesting results. First, black and white mode is basically dot on and dot off. "White" means a dot is on, without regard to actual color, and "black" means a dot is off, and will always be black. The actual resolution of this mode is 560 by 192.

The second result is that, like standard hi-res, the color mode resolution is really only 140 by 192. There are 560/4, or 140 color groups across the screen, and each color group can only produce one color at a time. This is not as severe as it sounds, because most graphics you produce will be some mixture of the "black and white" 560-dot resolution and the color 140-dot resolution. Most computers require that you choose either one or the other display mode, with very limited mixture, if any.

The third result is that there is no such thing as color bleed in double-res. The color-select bit is not used, so you may freely place any color next to any other color without fear of affecting other colors within the byte boundaries.

The final result is that, unlike the standard hi-res version of The Graphics Magician, there are no "group A, B, and C" colors on the two color palettes. You may mix color fills any way you want. Of course, if you create a picture in 280-mode, and wish to use it with the standard hi-res PICDRAW, you should observe the limitations of standard hi-res, as described in Appendix B.

Appendix B - Apple Standard Hi-Res Colors

The diagram in Figure A.1 also corresponds to the standard hi-res graphics screen. Each dot corresponds to a dot that may be on or off on the screen, and each is technically stored as a bit (on or off). The lines divide the basic storage units on the Apple, bytes. Each byte consists of seven bits displayed horizontally on the screen. An eighth bit is used as a color flag, or switch, which we will find is the key to most color anomalies on the Apple. The screen is 40 bytes, or 280 dots wide. It is 192 bytes, and dots, tall.

Each dot may only be on or off. If a dot is off, it is black. An "on" dot can be one of four colors, depending on two conditions. "On" dots in even columns will be either blue or violet. "On" dots in odd columns will be either orange or green. If the color flag for that byte is on, the dots will be orange and blue, and if the flag is off, the dots will be green and violet. Got all that? Okay, now if two dots are "on" horizontally next to each other, they both become white. This gives the "eight" Apple colors, even though there are two versions of white, and two versions of black. See figure B.1.

	Group 1 Color Flag Off	Group 2 Color Flag On
Dots Off	Black #0	Black #4
Odd Dots On	Green #1	Orange #5
Even Dots On	Violet #2	Blue #6
All Dots On	White #3	White #7

Figure B.1- Color Construction

The color layout has a few interesting results. First, black and white mode is basically dot on or dot off, without regard to color flag or position. "White" means a dot is on, and it can actually be displayed as white, blue, violet, green, or orange. "Black" means the dot is off, and will always be displayed as black. The actual resolution in this mode is 280 by 192. (Note: the color flag also slightly affects the position of a dot, shifting it left or right a half position. By using this shift, it is possible in some instances to make the horizontal resolution appear to be 560 since there are 560 positions in which dots can appear. It is not a true resolution, however, since no matter what you do there are only seven dots per byte.)

The second result of the color layout is that in color mode, the resolution is really only 140 dots wide. An orange line across the screen has only the odd dots set; the even dots must be off. It only uses 140 dots and restricts the use of the others. This is part of the reason why most uses of color have a black background. Any color may be used against a black background, although in limited positions. Against any other background, even white, you are restricted to a much lower apparent resolution.

The third result is that some colors are difficult to use next to others because of the color flag. Orange and green dots, for example, cannot appear in the same byte, so it's usually impossible to use them next to each other horizontally. Looking at figure B.1, the colors in group 1 generally cannot be used horizontally next to the colors in group 2.

Blended Colors and the Color Palette

There are ways to combine colors so that you perceive different shades and textures, and even get fooled into thinking there are a few new colors. Part of it is just creating different color patterns horizontally. The other part is that colors that cannot be placed horizontally next to each other can be placed in alternate rows vertically. Green and orange, for example, when placed in alternate rows give a yellow color.

The 280-mode color palette available in the picture editor, when displayed in standard hi-res, actually has three groups of colors. Group A, the first and largest set, consists of blended colors such that alternate rows use group 1 and group 2 standard Apple colors (as in figure B.1). Group A is the first 5 columns of the palette and the top two colors from the sixth column, or colors 0 to 51. (The first color in each group is white).

Group B consists of blends of colors made only with the Apple colors 4-7, and is colors 52 to 76 on the palette. Group C is made of Apple colors 0-3, and is colors 77 to 107 on the palette. Lines from the matching set of line colors can be used over group B or C, but lines should not be drawn over group A colors. Instead, the lines should be drawn first, then colors added.

"Color bleed" is when color flags change around the border of two colors and cause the first color to change at the border. In the picture editor, color bleed may be minimized by avoiding putting two colors from different groups next to each other horizontally. Instead, work in horizontal zones that keep the conflicting colors apart horizontally, but allow them to share vertical borders (see figure B.2). Note also that several colors appear in two or three of the groups. They are constructed differently, but appear the same. There are three different whites, for example. You can use these different colors across the zones to make it appear that one color spans them (see figure B.3 for an example).

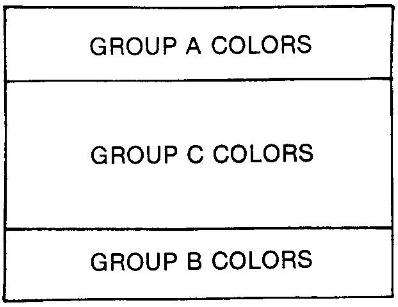


Figure B.2 - Color Zone Example

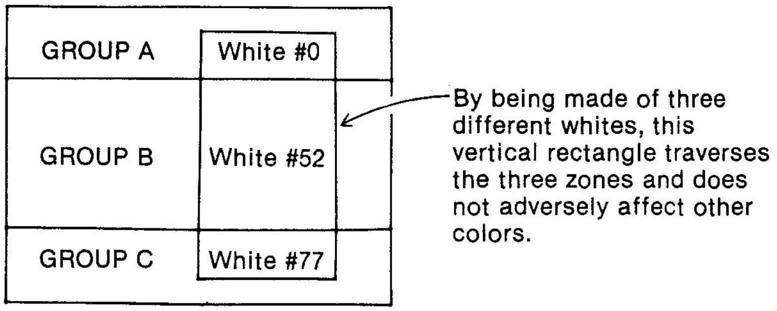

GROUP A
White #0
GROUP B
White #52
GROUP C
White #77
By being made of three different whites, this vertical rectangle traverses the three zones and does not adversely affect other colors.

Figure B.3 - Traversing Zones

Appendix C - Filename Suffixes

Below is a listing of the types of files that can be created or used with the Double-Res **Graphics Magician Picture Painter**. The suffixes will always identify the file type.

.CLT color table; a line and fill color lookup table used by DPICDOC and DPICDRAW for translating colors.

.DPA double-res picture/auxiliary memory; half of a double-res picture saved with the "(I)mage" save in the picture painter. This is an 8K exact memory copy of half the picture. The other 8K is in the corresponding ".DPM" file.

.DPM double-res picture/main memory; the other half of the ".DPA" file.

.DPC double-res sequential picture, created with the picture painter in 560-mode.

.FNT text character font created with early versions of **The Complete Graphics System**; small fonts can be used with the DPICDRAW and DHPRINT routines.

.PTX picture transfer file; a transfer-formatted picture saved with the picture lister.

.SPC sequential picture file created with **The Graphics Magician** or with the Double-Res **Graphics Magician Picture Painter** in 280-mode.

.STS small typeset, created with **The Complete Graphics System** or from **Additional Type Sets**, which can be used with the DPICDRAW and DHPRINT routines.

Appendix D - Command Reference

M	mode switch (280 <-> 560)
N	normal hi-res display (from 280-mode)
T	enter text mode at cursor position
Control-T	enter text mode at last text position
Z	zero in on area
Control-Z	control how tight "Zero mode" is
SPACE BAR	go to selection page, or back to picture page
ESC	full screen switch
R	redraw picture, or leave edit mode
DELETE	delete the last step
Control-D	delete multiple steps
E	enter edit mode at beginning of picture
<-	enter edit mode from end of picture
C	change line mode thick/thin (560-mode only)
S	save sequential picture
I	save screen image
Q	quit
H	help

Text Mode Commands

Control-R	reverse type
Control-N	normal colored type
ESC	full screen switch
DELETE or<-	delete last letter
RETURN	leave text mode

Edit Mode Commands

Same as normal commands, with these added:

->	advance one step
<-	go back one step
TAB	tab forward 10 steps

Other Unlisted Commands

J	reverse joystick orientation
L	line mode, without going to selection page
F	fill mode, without going to selection page
1-8	brush selection, without going to selection page