Shapestuff Unit & Drawing Utilities for Apple Pascal.

Introduction

Apple Pascal provides the TURTLEGRAPHICS unit for high-res graphics, but there is nothing in it comparable to Applesoft's DRAW. These programs allow a programmer to create shape tables and to draw full-screen hi-res pictures. The SHAPESTUFF unit allows an applications program to load a table into otherwise unused memory, then draw shapes from that table, and to load or save high-res pictures. (These utility programs have help-screens on an 80-column display. While not required, simultaneous use of a color monitor and an 80-column text monitor can be desirable.)

Units:
SHAPESTUFF - an intrinsic unit to be installed in a library file and used by a program that declares USES SHAPESTUFF.
SHORTSHAPE - a reduced-size version of SHAPESTUFF that requires less memory and disk space.

Utility programs:
SHMAKE - A shape editor to create or modify shapes.
SHMERGE - A shape-table manager to merge shapes created with SHMAKE into multi-shape tables, or extract shapes from another table.
SHINDEX - A shape-table examiner for quick display of a shape-table's contents.
SHANIMATE - An interactive test program to display an animation sequence.
SKETCHIT - A hi-res drawing program.

Other files:
SHAPELIB.SHAP & SHAPELIB.SHDT - A sample shape-table. (A shapetable consists of a data file with suffix .shdt and a shape file with a suffix .shap.)
FAT.CHARSET - A version of SYSTEM.CHARSET (hi-res text characters) with vertical strokes having a two-dot width, to reduced color fringing.
ALTRNT.CHARSET - An alternate fat CHARSET with a modified uppercase font.
CHARS.SHDT - A shapetable datafile that corresponds to half of a charset, i.e. 64 shapes of size 8hx7w, useful for modifying charset fonts.

Demo files and programs:
DEMO.TEXT & DEMO.CODE - A demo program that uses SHAPESTUFF.
HILLSIDE.PICT - A picture created with SKETCHIT, used in DEMO.
MEN.SHAP & MEN.SHDT - A shapetable used in DEMO.
TRAIN.TEXT & TRAIN.CODE - Another demo program.
LOCOMOTV.SHAP & LOCOMOTV.SHDD - A shape used in TRAIN, modified from a shape in SHAPELIB.
TUNNEL.TEXT & TUNNEL.CODE - Another demo program.
LOCOM.SHAP & LOCOM.SHDT - A shapetable used in TUNNEL.
HILL.PICT - A picture used in TUNNEL, modified from HILLSIDE.PICT.

SKETCHIT

SKETCHIT is a drawing program to draw and save hi-res pictures. Pictures are saved in sixteen-block files with a filename suffix of .PICT.

Commands:

e - erase screen. (SKETCHIT does not automatically clear the hi-res screen.)
G - get (load) a picture from a diskfile.

P - set pencolor and draw that color at the cursor's location. Options are WT, W1, W2, BK, B1, B2, VT, GR, BU, OR, RV, NO, INV, and HOME. Most correspond to the TURTLEGRAPHICS pencolors. RV and NO are identical and stand for reverse and none. They do not draw a "reverse" line, but instead create a reverse, visible, non-drawing cursor. HOME sets the pencolor to RV and moves the cursor to the lower left corner of the current viewport. INV makes the cursor invisible. (Note that VT, GR, BU, OR, W1, W2, B1, and B2 draw a stroke two columns wide.)

L, R, U, D, arrow keys - move the cursor one dot left, right, up, or down, drawing the current pencolor at the new location.
J - jump the cursor to an edge of the screen and set the pencolor to RV. It will ask where, and you must answer T, B, L, or R for top, bottom, left, or right.

V - adjust viewport. It will ask if you want full screen. If you type Y, the viewport will be set to (0,279,0,191) and the cursor will be HOME'ed. If you type N, the viewport's outline will be highlighted. Then you may enter either <Space> to return to normal operation, or T, B, L, or R to adjust the top, bottom, left, or right boundary. If you choose a boundary, you then enter U for up (right) or D for down (left) or <Space> to fix that boundary and return to the "which boundary" option. When you are satisfied with the viewport, enter another <Space> to return to normal operation. The outline will disappear, the viewport will be fixed at the new position, and the cursor will be HOME'ed.
F - fill viewport with a color. The outline will be displayed briefly and it will ask if you really want the viewport filled. If you type Y, it will ask for a color. The options are the normal TURTLEGRAPHICS pencolors, except for RV for reverse. If you enter any other value, the fill will be aborted. (You may enter the Y before it asks if you want the viewport filled.)
T - write text using the current CHARSET. It will ask for the text to enter. If you enter <Space>, it will use the same text, mode, and direction as the previous entry. It will ask whether horizontal or vertical, and what mode (CHARTYPE). (To enter something erasable, use mode 6.) The cursor will return to its starting location, so that if you used mode 6, you can erase that text by just entering T and <Space>.
C - draw a circle centered on the cursor. It will ask for radius and color. You may get better result by moving the cursor one dot left or right and drawing the circle again.
X - mark an endpoint for a line segment. After that, you may enter only L, R, U, D, arrow keys, or J(ump, until you hit X or x to mark the other end of the line segment. It will ask for a color, then draw a straight line between the two points. If you completed the line with X, the pen will remain at its final postion. If you completed the line with x, the pen will return to the first point. The pencolor is left set to RV.
O - display the viewport's outline in reverse around the inside edge of the viewport, for reference, not a permanent part to the picture.
1 - load a shape table.
d - draw a shape from the currently loaded table. It will ask for the shape number in

that table and a mode. You can enter <Space> to repeat the last shape. If you use mode 6 for both draws, it will erase the shape.

<Space> - show the current cursor location, viewport, and pencolor.

h or H - display the help screen.

S - save the current screen to a diskfile. It will ask for a file name, and for verification if a picture file with that name already exists. Picture files occupy sixteen disk blocks.

K - Change any key command except L, R, U, D, arrow keys, H, h, or <space>. New keys can be any unused key and are valid only during this session.

q - quit SKETCHIT. It will ask if you want to Save the current picture.

SHMAKE

SHMAKE is a drawing program to create or modify shapes. Maximum shape size is 64x64 dots, which occupies 512 bytes (one disk block). The display shows the shape in normal and 3x size, the pen's location, current "pencolor" (white, black, or none), and the shape's size.

Commands:

W   - set pencolor to "white" to turn dots on, mark this dot.
B   - set pencolor to "black" to erase dots, mark this dot.
N,n - set pencolor to "none" to move pen without drawing.
w,b - mark this dot "white" or black, but leave pencolor "none".

Shapes are on/off bit patterns copied to screen, so turtle pencolor does not apply. For colors, dots must be on in odd or even columns only. The color will depend on the column and on the color bit for each screen byte. If the color (high-order) bit is off, odd-numbered columns will be green and even-numbered columns will be violet. If the color bit is on, odd columns will be orange and even columns will be blue. For white, two adjacent columns must be on.

When drawing black on the top or right edge, SKETCHIT may pause for a few seconds to update its size values. If you hit a key during this process, it will beep but not respond. If you must erase many dots on the top or right edge, you can prevent this annoyance by drawing an extaneous dot above and to the right of the shape, then remove that dot last.

L, R, U, D, arrow keys - move pen one dot left, up, right, or down, drawing current "pencolor".

O - display shape in "other" color, i.e. change the background color bits.
o - display color shifted by one column. The cursor will turn off and this image will remain until you hit any key. You cannot work on a shape with this shifted image displayed. SHMAKE creates its shapes beginning in column 1. To draw a shape in a program with the same colors, draw it in an odd-numbered column.

You cannot display a shape in SHMAKE with mixed color bits. However when a <u>program</u> draws a shape, it will use the color bits currently on-screen.

c - clear screen (erase current shape).
v - verify (i.e. re-draw) shape. Takes about one minute. Will display "Wait" until it is finished. Should never be needed.
f - fill screen with white (on). Takes about one minute.

S - save this picture. It will ask for a file name (and check whether a shapefile with that name already exists). It will ask if you want the top or right edge of the shape padded with black (see note below), display the shape's dimensions and how many bytes it occupies, then create .SHAP and .SHDT files with that filename.
G - get (load) a shape from a diskfile. Takes about one minute. Sets pencolor to None. If you give it the filename of a shape <u>table</u>, only the first shape is loaded. To modify any other shape in a table, use SHMERGE to extract the desired shape.
q - quit. It will ask if you want to save the current shape.
h or H or <Space> - display the 80-column help screen.

Maximum shape size is 64x64 dots, which requires 512 bytes (one disk block). A table may contain up to 99 shapes, and occupy 9 disk blocks (4.5 Kbytes).

Padding can be useful for animation. The smoothest animation is obtained by overlaying successive shapes using mode 10, with each shape sufficiently wide to cover the previous shape. A shape in SHMAKE always starts at the lower left, so the bottom and left can be padded with black by starting the "white" parts at some other point.

SHMERGE

SHMERGE is a program to combine shapes into a table. Files produced by SHMAKE contain only one shape (i.e., are a table of one). A program can use more than one shape without loading another file only if the shapes are combined into a table. A table may contain up to 99 shapes, for a maximum of 4.5 Kbytes.

SHMERGE will ask for the name of the new table. If that name already exists, SHMERGE will ask for verification. (The original will not be removed until the final write operation, so if you quit without writing the final file, the original will remain.) SHMERGE will then ask for the name of a shape table to load, from which to extract shapes. (If you gave an output filename that already existed, you may still load the original file, since the old one is not removed until the new one is written.) Next enter either a new filename to load or a shape number to transfer. If you enter a number, that shape is displayed for inclusion in the new table. Instead of a number, you may also enter = to transfer a group of shapes (all or between designated numbers), or ? for a sequential transfer of all loaded shapes with confirmation for each, or <Space> to transfer the next shape after the one just examined. For all except =, it will ask for confirmation.

SHMERGE will inform you when it writes each disk block. When the ninth (maximum) block has been written, it will warn you and give you the option to quit. You can create a larger table but it will not fit into normal shape table memory. If you quit, the last shape may not be complete.

When you are finished entering shapes, enter just <Return>. It will ask for verification, then create the .SHAP and .SHDT files and display all of the shape dimensions.

SHINDEX

SHINDEX is a program to examine a table's shapes and their widths and heights (in dots). It will display the first four shapes against a white background. You may then enter either <Return> to see the next four shapes or a shape number to display. If you enter <Space> followed by <Return> or when you reach the end of the table, it will ask for another file name. <Space> <Return> will return to the same table. To quit SHINDEX, enter <Return> instead of a file name.

SHANIMATE

SHANIMATE is a program for you to experiment with animation parameters. Two procedures in SHAPESTUFF allow a program to animate a sequence of shapes with a single procedure call. An animation sequence is consecutive shapes drawn in rapid succession. For a complete description, see SHAPESTUFF. SHANIMATE lets you experiment with the nine parameters.

SHANIMATE will ask for a background picture to load. (Enter <Return> for none). It will then ask for a shape table to load. If you enter just <Return> the program will terminate. Next it will step through the nine parameters and ask for new values. Enter just <Return> to keep the same value. Then the animation sequence will be displayed. When the sequence is finished, you may enter <Space> or <Return> to step through the parameters again, or q or Q to return to the initial picture-loading phase. Any other key will repeat the animation sequence.

SHAPESTUFF Unit

SHAPESTUFF is an Intrinsic Unit with 7 functions and 4 procedures. It should be installed in *SYSTEM.LIBRARY (or another library file for some p-system versions) with the program LIBRARY.CODE. Each unit in the library has a segment number, which must be unique. SHAPESTUFF was compiled with segment numbers 25 and 26. If these numbers conflict with another unit, contact me for a solution. A program may use these routines by declaring USES TURTLEGRAPHICS, SHAPESTUFF (it must delcare TURTLEGRAPHICS before SHAPESTUFF). (For more memory at run-time, you may use the No-load option. SHAPESTUFF invokes the Resident option for DRAWBLOCK.)

The following global identifiers are declared in SHAPESTUFF:

```
VAR SHAPEFILE: FILE;
    SHDTFILE: FILE OF STRING[2];
FUNCTION STRTOINT: INTEGER;
FUNCTION GETSCREEN: BOOLEAN;
FUNCTION PUTSCREEN: BOOLEAN;
FUNCTION LOADSHAPE: INTEGER;
FUNCTION LOADCHARS: INTEGER;
PROCEDURE DRAW;
PROCEDURE DRAWSHAPE;
PROCEDURE ANIMATE;
PROCEDURE DBLANIMATE;
FUNCTION SHWIDTH: INTEGER;
FUNCTION SHHEIGHT: INTEGER; .
```

FUNCTION STRTOINT(SS: STRING): INTEGER converts a string to its integer equivalent, scanning the input string from left to right, terminating at either the end or the first non-digit character. If the string length is zero, or if the first character is a non-digit, it returns a value of 0.

Example: I := STRTOINT('253M65') is equivalent to I := 253.

Example: READLN(SS); I := STRTOINT(SS) is almost equivalent to READLN(I). However READLN(I) will cause a runtime error if the user enters a non-digit character, while the STRTOINT form will accept any input.

FUNCTION GETSCREEN(SS:STRING): BOOLEAN loads into the hi-res screen the picture file named in SS. It returns a boolean value TRUE if the picture was loaded, FALSE if it wasn't. GETSCREEN automatically adds a suffix .PICT to SS. It looks on the current "prefix" (default) volume unless SS contains a volume designation.

Example: B := GETSCREEN('#4:HOUSE') will load HOUSE.PICT into the hi-res display and set B to TRUE, if the file HOUSE.PICT exists on the disk in drive #4:.

FUNCTION PUTSCREEN(SS:STRING; B:BOOLEAN): BOOLEAN saves the current hi-res screen into the file named in SS. It automatically adds the suffix .PICT to SS. SS plus the suffix must be a legal filename. If a file of that name already exists, B verifies whether to replace it, TRUE meaning replace, FALSE meaning do not replace. PUTSCREEN returns a boolean value TRUE if the save was successful, otherwise it returns FALSE. Failure is typically due to either insufficient disk space (it needs sixteen blocks), that file already existing with B set to FALSE, or an illegal filename.

Example: BB := PUTSCREEN('HOUSE',TRUE) will save the current hi-res screen in a file named HOUSE.PICT and set BB to TRUE (if there is sufficient disk space.) Any existing file named HOUSE.PICT will be erased.

FUNCTION LOADSHAPE(SS:STRING; I:INTEGER): INTEGER loads the shape table named in SS. LOADSHAPE adds the suffixes .SHAP and .SHDT for the shape and data files. It should be 0 to 8. It says where to load the table, i.e. how many 512-byte units above location 3584 (3.5k). Since the hi-res page starts at 8k, the memory available for the table is (9-I)*512 bytes. Normally you should use 0, but this parameter lets you reserve memory for some other use. LOADSHAPE will load only as many blocks as will fit between the starting location (3584 + I * 512) and the hi-res page (8192). If the load was successful, LOADSHAPE returns how many shapes it loaded. If LOADSHAPE could not find the .SHDT file, it returns a value of 0. If the load was incomplete (possibly the file was too large), LOADSHAPE returns a negative value, i.e. -1 times the number of shapes actually loaded. If it ever returns a value 0 or negative, the loaded shapes may be incorrect.

Example: II := LOADSHAPE('#5:MEN',0) loads the shape table in MEN.SHAP and MEN.SHDT, found on drive #5:. If the table had six shapes, it would set II to 6 if the load was successful.

PROCEDURE LOADCHARS(SS:STRING): INTEGER loads a CHARSET, replacing the current CHARSET in memory. The subscript .CHARSET is automatically appended to SS. If the CHARSET (2 blocks) loads properly, LOADCHARS returns a value of 128. If it does not load two blocks, it returns a value of -128. If the file is not found, LOADCHARS returns a value of 0.

Example: II := LOADCHARS('#5:ALTRNT') loads into memory the file ALTRNT.CHARSET found on drive #5, replacing the previouly loaded charset. II is set to 128 if the load is successful.

PROCEDURE DRAW(N,X,Y,M:INTEGER) draws shape number N from the current table, at the hi-res location X,Y, using mode M. If N is less than 1 or greater than the number of shapes loaded, it does nothing. It will accept any X and Y values, on screen or off, and only draw that part of the shape which extends into the viewport. The shape is drawn with the dimensions specified in the .SHDT file. X and Y become the lower left corner of the shape. To <u>center</u> a shape on a point X,Y, either use DRAWSHAPE or use X - SHWIDTH(N) DIV 2, Y - SHHEIGHT(N) DIV 2 unstead of X,Y (see below).

Mode is a DRAWBLOCK parameter. The most useful modes are 5, 6, 10, and 14. Mode 10 means copy the shape, both "black" (off) and "white" (on) dots. Mode 5 is the inverse, i.e. "off" dots written as "on" and "on" dots written as "off". Mode 14 means add the "on" dots to the screen, but do not force the "off" dots to black. Mode 6 means reverse the screen for the shape's "on" dots. Two successive mode 6 DRAW's will draw, then erase a shape, leaving the background untouched.

In Apple Pascal version 1.1, YSKIP supposedly did not work, so results may be unpredictable if Y is negative. There also appears to be one minor bug in the version 1.2 of DRAWBLOCK. Occasionally one column of left-to-right "wrap-around" can occur. If this is a problem, one solution is to reset the viewport when drawing that shape. Resetting the viewport can also be used to suppress parts of a shape.

Example: DRAW(2,20,44,10) draws shape 2 at 20,44, using mode 10.

PROCEDURE DRAWSHAPE(N,XSKIP,YSKIP,WIDTH,HEIGHT,X,Y,MODE:INTEGER) draws shape number N and allows you to specify other DRAWBLOCK parameters. (Note DRAWSHAPE does not require the ROWSIZE parameter.) DRAWSHAPE passes these parameters directly to DRAWBLOCK, so you must make sure that the values are legal. DRAWSHAPE makes one exception to this parameter handling: If you specify 0 for width or height, DRAWBLOCK uses the value in the .SHDT file and <u>centers</u> the shape in that direction over X or Y. For example, if WIDTH is specified as 0, width will be taken from the .SHDT file, and the X position will be set automatically to X - (width DIV 2).

Example: DRAWSHAPE(2,4,5,12,15,20,44,6) draws shape 2 at 20,44 using mode 6. It skips four columns on the left and draws a width of twelve, and it skips five rows from the bottom and draws a height of fifteen.

Example: DRAWSHAPE(2,0,0,0,0,20,44,6) draw the complete shape 2 centered on the point 20,44, using mode 6.

PROCEDURE ANIMATE(N1,N2,X,Y,DX,DY,R,W,M:INTEGER) animates a sequence of shapes N1 to N2, starting at X,Y and moving DX,DX for each successive draw. R is a repitition factor. W specifies time to wait on each picture. M specifies mode. X and Y may be anywhere, on screen or off (see DRAW). DX and DY may be positive, negative, or zero. Either N1 or N2 may be greater, or they may be the same. Each shape, not counting N1, is drawn R times so that the last shape to be drawn will be N2, at the location X + R * DX * (ABS(N2-N1)), Y + R * DY * (ABS(N2-N1)). The exact sequence is to draw one shape (so as to erase a pre-existing shape if mode 6 is used) then immediately draw the next. Thus, <u>except for N1, each shape is drawn twice,</u> R times. If R < 1, then N1 is drawn at X,Y and the sequence terminates. To animate a sequence with mode 6, first DRAW shape N1 at X,Y so that ANIMATE can erase it. Each sequence assumes that N1 is already on screen, and ends with N2 on screen; thus you can make successive calls to ANIMATE, each call starting with the previous N2 from where the last sequence ended.

In geneal, animation can be done either by erasing each shape before drawing the next, or by overlaying each shape with another sufficient to completely cover it. The first uses mode 6 and requires that the starting shape already be on screen. It produces some visible flicker, especially if the shapes are very large, but it allows a shape to pass over the background without disturbing it. For best results with mode 6, use larger DX or DY values and W of 4 or greater. Overlay animation works by using mode 10 to completely cover each shape with the next. It produces smoother results but destroys the background. If the shape is moving, there must be at least as much padding around each shape as the distance it moves. For either type, you will get smoother results if the shapes are smaller. A shape of 10x10 dots has only 100 dots, while a shape of 16x16 (seemingly not much bigger) has 256 dots, an increase of over 150%! (SHAPESTUFF is <u>not</u> intended to be a high-quality animation tool.)

Example: ANIMATE(1,6,10,10,2,0,2,0,6) assumes that shape 1 was

already drawn at 10,10, then animates 2 through 6. The exact sequence will be to erase shape 1, draw shape 2 at 12,10, erase 2, draw 2 at 14,10, erase 2, draw shape 3 at 16,10, etc., ending with shape 6 drawn at 30,10. It will move through background material without disturbing it.

PROCEDURE DBLANIMATE(N1,N2,X,Y,DX,DY,R,W,M, N1N1,N2N2,XX,YY,DXDX,DYDY,RR,MM:INTEGER); animates two sequences simultaneously. There are two sets of parameters equivalent to those of ANIMATE, except that the second set does not include a W value. DBLANIMATE animates both sequences N1 to N2 and N1N1 to N2N2, each with its own parameters, and terminates when the last sequence is finished.

Example: DBLANIMATE(1,6,10,10,2,0,2,0,6,11,11,10,50,2,0,1,10) will animate shapes 1 to 6 from 10,10 to 30,10 with mode 6 and draw shape 11 at 10,50 and 12,50 with mode 10.

FUNCTION SHWIDTH(N: INTEGER): INTEGER; returns the width of shape N in dots, as read from the .SHDT file. It will reflect the entire shape, padding and all, not just the width of the "white" portions. Since DRAW and DRAWSHAPE use the specified X,Y as the lower left corner of the shape, if you want to center a shape horizontally on an X location, you must DRAW it half of the shape's width to the left of X. Thus, one use of SHWIDTH is to DRAW a shape centered on a point X by specifying DRAW(N, X - SHWIDTH(N) DIV 2, Y, M). This is equivalent to specifying width of 0 in DRAWSHAPE (see above) but has the advantage that DRAW will work even if X or Y is negative. (DRAWSHAPE may execute slightly faster.)

Example: I := SHWIDTH(3) will set I to the width of shape 3.

Example: DRAW(3, 50 - SHWIDTH(3) DIV 2, 55, 6) will draw shape 3 with its bottom row on line 55, and centered horizontally on column 50, regardless of the shape's width.

FUNCTION SHHEIGHT(N: INTEGER): INTEGER; returns the height in dots of shape N, as read from the .SHDT file. (See SHWIDTH above.)

Example: I := SHHEIGHT(4) sets I to the height of shape 4.

Example: DRAW(3, 50 - SHWIDTH(3) DIV 2, 55 - SHHEIGHT(3) DIV 2, 6) will draw shape 3 centered both horizontally and vertically on the point 50,55, using mode 6.

SHORTSHAPE Procedures and Functions

SHORTSHAPE is a smaller version of SHAPESTUFF (using the same segment numbers). It requires 2 fewer disk blocks than SHAPESTUFF and uses slightly less memory. It is similar to SHAPESTUFF with the following exceptions: (1) The DRAWSHAPE, ANIMATE, and DBLANIMATE procedures have been removed. (2) LOADSHAPE no longer checks to see if a shape table will fit into the specified memory below the hi-res page. (3) The maximum number of shapes is 40 instead of 99. If a table contains more than 40 shapes, only the first 40 will be loaded. These changes save about 250 bytes in program stack and 700 bytes in code.

OVERVIEW and TUTORIAL:

There are two "creativity" programs - SHMAKE to create or modify shapes, and SKETCHIT to draw pictures. A shape is stored in two files with the suffixes .SHAP and .SHDT. Pictures are stored in files with the suffix .PICT. There are two "management" programs - SHMERGE to arrange (or re-arrange) shapes to/from shape tables, and SHINDEX to examine a shape table's contents. A shape table has the same form as a single shape, i.e. a single shape is just a table of one. A program can draw those pictures or shapes by using the procedures and functions in SHAPESTUFF. Shapes are identified by their position in the table.

Run DEMO (make sure your Prefix drive is the one containing HILLSIDE.PICT, MEN.SHAP, and MEN.SHDT). Hitting any key will terminate DEMO. Re-run DEMO and notice how the man figure is dressed in orange and blue when he moves up the hill, but green and violet when he moves across the bottom. This happens because, when HILLSIDE was drawn, the top of the screen was filled with Black2, while the bottom was left alone with its default Black1. This picture and animation were produced in the following steps:

1. SHMAKE was used to create four "man" shapes, each slightly different.
2. SHMAKE was also used to create two flag shapes.
3. SHMERG was used to combine the men figures into a 28-shape sequence. This sequence and the two flags were saved in a shape table named MEN. (Later, the four original men and the two flags were also copied into SHAPELIB and the originals removed.)
4. SKETCHIT was used to create the background picture named HILLSIDE. Before anything was drawn, the top of the screen was filled with Black2. One step in creating this picture was to load SHAPELIB and draw the flag onto the screen.
5. SHANIMATE was used to experiment with animation parameters of MEN on the HILLSIDE background.
6. DEMO was written, declaring "USES TURTLEGRAPHICS, SHAPESTUFF". It first does a GETSCREEN to load HILLSIDE, then does a LOADSHAPE to load MEN. It DRAW's shape 1 (the running man) at the bottom of the hill, DBLANIMATE's two men, ANIMATE's the man across the bottom, ANIMATE's the first man back to the bottom of the hill, and then repeatedly DRAW's the two flag shapes. (Actually, the DEMO.CODE supplied does not use SHAPESTUFF, but rather, has its own routines included, so you can run it before you install SHAPESTUFF.)

Now run SHINDEX to examine the contents of the shape table MEN.

Now run SHANIMATE and load the picture HILLSIDE and shapetable MEN. Try these parameters: N1 = 1, N2 = 28, X = 0, Y = 52, DX = 2, DY = 1, R = 1, W = 5, M = 6. This will make the man walk up the hill. Now change DX to 8, DY to 4, and W to 3. The man will run up the hill. These are examples of erase-and-draw (mode 6) animation. Now change the parameters to N1 = 1, N2 = 28, X = 0, Y = 0, DX = 2, DY = 0, R = 1, W = 0, M = 10. The man will smoothly walk across the bottom of the screen. Now change R to 4 so that more distance is covered before each shape is changed. This character is on skates. Finally, change DY to 1, R to 3, and W to 2. Since there are no padding lines across the bottom, the feet of the previous shape are not covered when each shape is moved up, leaving behind the tracks of a cross-country skier. To quit, enter q or Q, then <Return> twice.

Now run SHINDEX to see what is in SHAPELIB. Make note of what number the flag shape is. Then run SHMERGE to create a shape named FLAG: Load in SHAPELIB and copy into it only the first flag picture. Then quit SHMERGE and run SHMAKE to examine that flag shape more closely. From SHMAKE, enter G to get a shape, and specify the file name FLAG. This will load the first (and only) shape in FLAG. Notice how it is displayed in violet and green. These colors are obtained by turning on only odd or even columns, as seen in the 3x display. If you enter O, the top display will be re-drawn with a background of Black2, showing red and blue. Notice how the white stars and stripes were formed. If you now enter o, that shape will be drawn in column 0 instead of column 1. Notice how the colors are reversed. Entering any key will restore the original-column colors. Quit SHMAKE with q, then enter the Filer and remove the new FLAG files you made [enter r(emove flag.=]. Since this flag shape is in SHAPELIB, you don't need another copy hanging around.

Now run TRAIN, then examine the .TEXT files of both TRAIN and DEMO. TRAIN.TEXT has only 3 commands. DEMO.TEXT is larger and is annotated.

Finally, run TUNNEL, and then examine the contents of its .TEXT file.