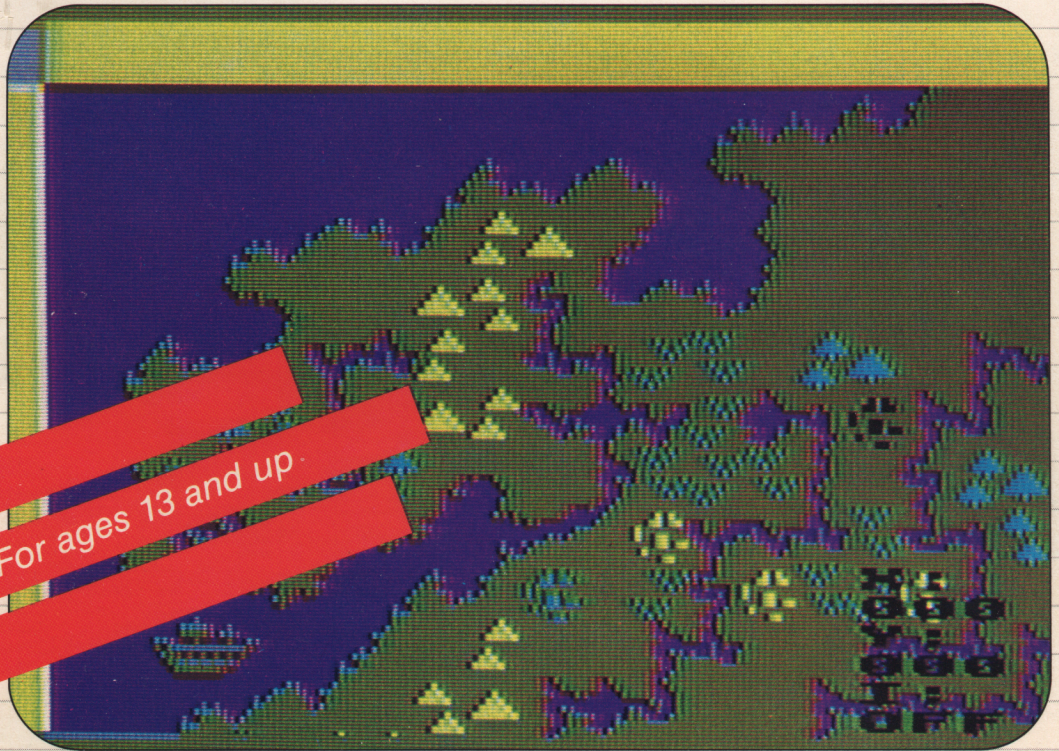


MAPMAKER

SYSTEMS/TELECOMMUNICATIONS

Create multiscreen, fine-scrolled map displays



For ages 13 and up

CONSUMER-WRITTEN PROGRAMS FOR

ATARI®

HOME COMPUTERS

APX

ATARI Program Exchange

MAPMAKER

by
Stephen W. Hall

Program and manual contents © 1982 Stephen W. Hall

Copyright notice. On receipt of this computer program and associated documentation (the software), the author grants you a nonexclusive license to execute the enclosed software. This software is copyrighted. You are prohibited from reproducing, translating, or distributing this software in any unauthorized manner.



Distributed By

The ATARI Program Exchange
P.O. Box 3705
Santa Clara, CA 95055

To request an APX Product Catalog, write to the address above, or call toll-free:

800/538-1862 (outside California)
800/672-1850 (within California)

Or call our Sales number, 408/727-5603

Trademarks of Atari

ATARI is a registered trademark of Atari, Inc. The following are trademarks of Atari, Inc: 400, 410, 800, 810, 820, 822, 825, 830, 850, 1200XL.

Limited Warranty on Media and Hardware Accessories. Atari, Inc. ("Atari") warrants to the original consumer purchaser that the media on which APX Computer Programs are recorded and any hardware accessories sold by APX shall be free from defects in material or workmanship for a period of thirty (30) days from the date of purchase. If you discover such a defect within the 30-day period, call APX for a return authorization number, and then return the product to APX along with proof of purchase date. We will repair or replace the product at our option. If you ship an APX product for in-warranty service, we suggest you package it securely with the problem indicated in writing and insure it for value, as Atari assumes no liability for loss or damage incurred during shipment.

This warranty shall not apply if the APX product has been damaged by accident, unreasonable use, use with any non-ATARI products, unauthorized service, or by other causes unrelated to defective materials or workmanship.



Any applicable implied warranties, including warranties of merchantability and fitness for a particular purpose, are also limited to thirty (30) days from the date of purchase. Consequential or incidental damages resulting from a breach of any applicable express or implied warranties are hereby excluded.

The provisions of the foregoing warranty are valid in the U.S. only. This warranty gives you specific legal rights and you may also have other rights which vary from state to state. Some states do not allow limitations on how long an implied warranty lasts, and/or do not allow the exclusion of incidental or consequential damages, so the above limitations and exclusions may not apply to you.

Disclaimer of Warranty on APX Computer Programs. Most APX Computer Programs have been written by people not employed by Atari. The programs we select for APX offer something of value that we want to make available to ATARI Home Computer owners. In order to economically offer these programs to the widest number of people, APX Computer Programs are not rigorously tested by Atari and are sold on an "as is" basis without warranty of any kind. Any statements concerning the capabilities or utility of APX Computer Programs are not to be construed as express or implied warranties.

Atari shall have no liability or responsibility to the original consumer purchaser or any other person or entity with respect to any claim, loss, liability, or damage caused or alleged to be caused directly or indirectly by APX Computer Programs. This disclaimer includes, but is not limited to, any interruption of services, loss of business or anticipatory profits, and/or incidental or consequential damages resulting from the purchase, use, or operation of APX Computer Programs.

Some states do not allow the limitation or exclusion of implied warranties or of incidental or consequential damages, so the above limitations or exclusions concerning APX Computer Programs may not apply to you.



Contents

Introduction	1
Overview	1
Required accessories	1
Optional accessories	1
Using <i>Mapmaker</i> with other programs	1
Getting started	3
Duplicating the files onto another diskette	3
Loading <i>Mapmaker</i> into computer memory	3
Demonstrating <i>Mapmaker</i>	5
Loading in the demonstration map	5
Moving around the display	5
Making a map	6
The <i>Mapmaker</i> menu	6
Changing hue	7
Changing intensity	7
Changing character sets	7
Creating a new map	8
Saving a map	8
Using <i>Mapmaker</i>	9
Background	9
Program capabilities	10
EDIT mode	11
General information	11
Scrolling	11
The cursor	11
Character editing	12
Exiting EDIT mode	12
<i>Mapmaker</i> menu	13
General information	13
EDIT MAP	13
CHANGE FONT	13
RUN FONT EDITOR	13
CHANGE COLOR REGISTERS	13
SAVE MAP TO DISK	14
QUIT	14

Technical information	15
Understanding redefined text displays	15
Character sets	17
Option 3: RUN FONT EDITOR	18
Color registers	19
MAP files	20
Fine scrolling	21
Bibliography	22
Appendices	23
SCROLL source listing	23
Character set table for ALLPRPS.SET	28
Character set table	29

Overview

Mapmaker is an easy-to-use utility to help you create multiscreen displays using redefined, BASIC Graphics mode 2 characters. To create a map, you first define the dimensions of the total display area, and then you enter characters from the computer keyboard. *Mapmaker* acts as an easel upon which you can try out different combinations of shapes by displaying them on the screen. The program's fine scrolling feature allows smooth movement from one part of your map to another by using either a Joystick Controller or the directional arrows on the computer keyboard. *Mapmaker* comes with an all-purpose character set of map elements. However, you can use a separate character editor to create your own mapping characters and then produce an infinite variety of displays with *Mapmaker*. The program supports four character colors, plus a background color, and the package contains an easy-to-use facility for changing the playfield color registers. When you complete a map, the program saves your file on diskette.

Required accessories

- 32K RAM
- ATARI 810 Disk Drive
- ATARI BASIC Language Cartridge

Optional accessories

- *Instdit* (APX-10060 or APX-20060) or other compatible character set editor
- One ATARI Joystick Controller

Using Mapmaker with other programs

Mapmaker is written in ATARI BASIC, with liberal use of machine language routines accessed by the USR function. In particular, *Mapmaker* uses page six of memory. Therefore, don't use *Mapmaker* with another program using page six. (If you're not familiar with the concept of pages in memory, you probably have nothing to worry about.)

Duplicating the files onto another diskette

Mapmaker is disk-based, and uses or creates four separate diskette files in the process of creating a screen display. For this reason, you need to copy the files MAPMAKER.BAS and SCROLL.OBJ onto another diskette before using the program. First-time users should duplicate the entire diskette to transfer the character set file (ALLPRPS.SET) and the demonstration file (DEMO.MAP) along with the program files.

Loading Mapmaker into computer memory

1. Insert the ATARI BASIC Language Cartridge in the cartridge slot of your computer.
2. If you plan to use the Joystick Controller, plug it into the first controller jack of your computer console.
3. Turn on your disk drive.
4. When the BUSY light goes out, open the disk drive door and insert the *Mapmaker* diskette with the label in the lower right-hand corner nearest to you.
5. Turn on your computer and your TV set.
6. When the READY prompt displays on your TV screen, type

```
RUN "D:MAPMAKER.BAS"
```

and press the RETURN key. The screen will clear, and then the message

```
PROGRAM INITIALIZATION
```


displays. The computer will access the diskette several times.

7. If this is your first look at *Mapmaker*, no messages relating to character set files will display. Later, when you have more than one character set file on diskette, the program will ask you which character set you want to load. When only one character set is on the diskette, the program loads it automatically.
8. Then the prompt

```
  Create a new map, or  
  Load an existing map?
```

displays. Type C to create a new map or L to load an existing one into memory.

(Note. If you're a new user of *Mapmaker*, go to the next section, titled "Demonstrating Mapmaker", at this point.)




Responding L tells the computer to scan the diskette for MAP files. If it finds just one, it loads the file automatically. If it finds two or more, the program asks you which file you want to load.

Mapmaker then loads in your specified file and goes into EDIT mode.

Responding C tells the computer to set up memory for a new map. The program asks you for the dimensions of the display area, measured in characters. Width can range from 22 to 128 characters and must be an *even* number (odd numbers are automatically rounded down). Height can range from 13 to 255 characters. The only other limitation on height is that it can't exceed a value which, when multiplied by width, equals more than 4K (4096 bytes). On systems with 40K or more of computer memory, the maximum display area is 8K (8192 bytes). The computer will substitute legal values for any values entered that aren't within the acceptable ranges. If these legal values aren't adequate for your purposes, reconsider the layout of your map project.

Upon receiving the display dimensions, the program switches to the display and draws a border around it. *Mapmaker* loads in the scrolling routine from diskette and then goes into EDIT mode.



Demonstrating Mapmaker

Loading in the demonstration map

Be sure you duplicate all the *Mapmaker* files onto another diskette before proceeding. Then, following the instructions in "Getting Started" up to step 8, you should now be looking at the screen displaying the messages "PROGRAM INITIALIZATION" and "Create a new map, or Load an existing map?" At this point, press the "L" key to instruct *Mapmaker* to load a MAP file from diskette that demonstrates a map created with *Mapmaker*. The program then automatically loads and displays the DEMO.MAP. The upper left corner of the map fills the screen, and the entire map is read into display areas not currently visible. Next, *Mapmaker* reads in a machine language file (named SCROLL.OBJ) containing the fine scrolling routine. A cursor is blinking in the upper left corner of the display. The lower right of the screen contains a set of X and Y coordinates and the status of the I (INVERSE video) flag, which is currently OFF. The current cursor coordinates are 0,0.

Moving around the display

First we'll move the cursor. If you want to use the Joystick Controller to move the cursor across the display, move the joystick in the direction of the desired cursor movement. Or, you can use the arrow keys (pressing the CTRL key isn't necessary) on the keyboard to move the cursor. The cursor shifts position at the same time that the cursor coordinates change to reflect the new location of the cursor. Don't move the cursor off the edge of the screen just now.

You can turn the cursor off, and incidentally remove the readout in the lower right. Press the SELECT key. Now the display is "clean." This allows viewing without the distraction and clutter of the editing aids.

Now, let's look at the whole map. Press and hold down the trigger button on the joystick, and pull the joystick toward you. This moves the screen "window" down the display. (You could also say that the display moves up. For scrolling, we always pay attention to the direction of window movement, not display movement.) To move the window under keyboard control, hold down the SHIFT key and press the down-arrow. Now, push the joystick to the right, or press the right-arrow to move the window to the right. You can move the window in any direction under joystick or keyboard control. This is called scrolling. The smooth movement of the window is called "fine" scrolling. The scrolling stops when the window arrives at the edge of the display. Examine the map. You're looking at a map of a mythical kingdom. With *Mapmaker*, you can make a map just like this.

Making a map

Let's find out how it's done. Press the SELECT key again. Do you see the cursor? If it isn't on the screen already, it will be just outside the screen. Move the cursor until it appears on the edge of the screen. You may have noticed something else. The coordinate readout is now white. Press SELECT twice more; now it's black again. This black/white shift allows you to read the coordinates whether the background color is light or dark.

As you move the cursor around, notice how the cursor highlights the shape of individual map characters. This points out how a map is a set of redefined characters placed side by side. The concept is quite simple once you get used to it. Instead of letters on the screen, we have map characters—trees, rivers and mountains. The shapes are contained on a diskette file called ALLPRPS.SET, which loaded into the computer during PROGRAM INITIALIZATION.

Now, let's see how these characters are put on the screen. Position the character in an open space. The cursor is now just a blinking square. Press the START key. The cursor stops blinking. Next, press the semicolon (;) key. Voila! A tree. Press the START key again. The tree stops blinking. Now press the equal sign (=) key. Presto! Two mountains. Move the cursor away. The mountains remain, but they are blue. We want gold mountains, so position the cursor on the mountains and press the START key again. You see by now that this tells *Mapmaker* you want to enter a new character at the cursor location. Before you press a key, press the ATARI logo key. Now press the "." key. The I (INVERSE video) indicator says ON. Move the cursor away, and !Shazam! the mountains are gold. That's all there is to it. Plug in characters at will. To remove those mountains, enter a blank character by pressing the space bar. To learn more about character sets and colors, read the sections titled "Character Sets" and "Color Registers".

The Mapmaker menu

Let's look at some more *Mapmaker* features. Press the OPTION key to display the *Mapmaker* menu. We're back to the standard character set. The MENU contains six options. It also tells you how to use the MENU. You move the pointers by pressing SELECT. To choose an option, set the pointer on the option and press START. Note that SELECT and START have different functions, depending on whether you're in EDIT mode or MENU mode. If you were to press START now, you would return to the map. Go ahead and press it to verify what happens. Now, press OPTION again to return to the MENU.

Press SELECT a few times, so the pointers are on CHANGE COLOR REGISTERS. Then press START. Now you're looking at the color palette routine. The color in the large screen area is the color of register 1. All the characters on the map that are this color are entered as register 1 characters. Press SELECT several times; as the text window indicates, this rotates the color registers. Next, we will change one of the colors. Select register B, the background color register. Right now, it's a rust orange. Let's make it white, so the map will look like winter. To change colors, we need to change both hue and intensity.

Changing hue

To change the hue, you use either the joystick or the arrow keys (you need to use CTRL with the arrow key in this instance). Push the joystick left or right, or press the left-arrow or right-arrow key. The color changes in the big window. Keep pushing the joystick or pressing the arrow key until the gray color appears.


Changing intensity

Now, change the intensity, which is the brightness of the color. We are currently looking at white, so a bright white is really white, while a dim white is gray, and a "dark" white is black. To change the intensity, push the joystick forward or back, or use the up-arrow or down-arrow. Get a nice bright white. Once you have it white, press OPTION to return to the MENU and then press START to return to the map. The background is now white, a field of snow. Or did you change the wrong register? Maybe you have snowy mountains instead. Go ahead; experiment with the colors. You're getting used to using *Mapmaker*.

Changing character sets

After you've tried out the color changing routine, go back to the MENU and select the CHANGE FONT option. In *Mapmaker* you work from a set of 64 characters. However, to make a complex display, we sometimes need more than 64 characters. The full character set has room for 128 characters. You can swap to the second set of 64 by using the CHANGE FONT option of the MENU. You still can't use more than one of the 64 character subsets on any one line of the display, but skilled programmers can divide the screen and use different sets at different map "latitudes." With CHANGE FONT, you can swap the character base, or load an entirely new font of characters. This can make the map look pretty strange.

Let's try it. Press START with the pointer on the CHANGE FONT option. Then, press the "S" key for "swap". Now, return to EDIT mode. The map is no longer a map, but a field of text characters. Go back to the MENU again. Now, we want to exit from *Mapmaker* briefly. Set the pointer on the QUIT option. Make it a practice never to leave *Mapmaker* except through the QUIT option. This time, we don't want to alter the demonstration file, and so we just QUIT. Now, press START and read the warning. This protects you from losing a lot of work by accident later on. This time, we press START again, and the program terminates after locking DEMO.MAP on the diskette.




Creating a new map

Let's look at how to initiate a new map. Type RUN and press RETURN. The program returns you to PROGRAM INITIALIZATION. This time, type "C" in response to the Create...Load prompt. The program requests the dimensions of your new map, width first. Choose any even width between 22 and 128 (*Mapmaker* automatically rounds an odd number down by 1) and press RETURN. Using an even number is a requirement of the fine scrolling routine. Now choose the height; the range can be from 13 to 255. However, width times height cannot exceed either 4K (4096) on a 32K system, or 8K (8192) on a 40K system. If you pick a height that is too large, *Mapmaker* will correct your number to an acceptable one.

After entering height, *Mapmaker* starts creating your map display by laying out the workspace. It draws a border around your map area, and assigns a set of default colors to the color registers. ("Default" just means that these are the standard colors. Black is the ATARI default background color.) You can check these colors through the color palette routine. After drawing the border, you are ready to edit. Go to it. You've already done some editing. An appendix contains a complete listing of the all-purpose (ALLPRPS) character set. Read the sections titled "Character Sets" and "Color Registers" to understand better how to use the characters.

Saving a map

To save your handiwork, go to the MENU and select the SAVE option. This will write your map to the diskette. Next time you run *Mapmaker*, you can recall your map for further editing, or for just showing off. It will be named FILE1.MAP on the diskette. The section on the MENU options tells you more about saving maps.



Be sure to read the rest of the manual before getting too far into using *Mapmaker*. It's full of useful information.

Background

Anyone who has experimented with the extensive graphics capabilities of the ATARI Computer is familiar with its high-resolution, multicolor graphics mode. Using this mode, a programmer can draw quite detailed pictures as long as he or she is willing to plot out the graphics commands needed to produce an elaborate display. Various hardware and software aids are available to help non-professional users create impressive hi-res screen displays. No matter what assistance is obtained from these products, however, hi-res displays are significant consumers of computer memory. A single screen of GRAPHICS 7 display requires about 4K of memory. While this is not too taxing for a BASIC program run on a 40K, disk-based system, few programmers are satisfied with a single hi-res picture. The alternatives for adding additional displays are either frequent disk access or significant additional memory commitment.

A much more serious problem arises for programmers wanting to create and use a multiscreen display with viewing of the display through scrolling. At 3200 bytes per screen, a modest ten-screen display leaves no room for program logic. Obviously, this situation is unacceptable. Fortunately for ATARI Computer owners, the computer comes to the rescue with two advanced features that in combination make creating a hi-res, ten-screen display possible in just 2.4K of memory. These features are the five-color text modes and the ability of the ATARI Computer to use redefined character sets.

Most users have seen displays using the larger multicolor text modes. In BASIC, these are the modes GRAPHICS 1 and 2. By using GRAPHICS 2, in particular, a single text character can fill an 8x8 pixel area of the screen. By redefining text characters so that they become portions of a picture or map display, a small allotment of screen memory can hold a truly impressive multiscreen display. Anyone who has seen the program *Eastern Front (1941)* can attest to the remarkable detail obtainable from a small amount of screen display memory. The multiscreen map, plus extensive game logic, of *Eastern Front* requires only 16K of memory.

Mapmaker lets you make multiscreen displays using redefined GRAPHICS 2 characters. Without *Mapmaker*, to create such a display, you'd have to arduously prepare a string of character names which, when displayed through an appropriate display list, would result in the desired map appearing on the screen.

Mapmaker does not include a character set editor. There are a number of good text character editors on the market, including *Instedit*, available from APX. (See the section on "Character Sets" for a further discussion of character editors.) *Mapmaker* is equipped with an all-purpose mapping character set so that users without a character editor can still use the program. This character set is further described in the section titled "Character Sets". The map file included with the program demonstrates what can be done with *Mapmaker* using the all-purpose character set.

Mapmaker was originally developed to facilitate the creation of maps to be used in simulations of historical conflicts, in other words, for wargames. Wargame designers using *Mapmaker* will be encouraged to improve the quality of graphics in professionally designed simulation games. Whether you're a professional simulation game designer, or are simply interested in getting the most personal enjoyment out of your ATARI Computer, you'll find using *Mapmaker* a pleasurable and rewarding experience.

Program capabilities

The standard dimensions of a GRAPHICS 2 full screen display (really GRAPHICS 2+16) is 20 characters wide by 12 lines high. The smallest display *Mapmaker* supports is a 22x13 character display. This limitation is because the program uses a fine scrolling routine that cannot properly manage a single screen display. The limitation shouldn't be considered a liability, however. You'll quickly discover that a small display area gives you insufficient range for an interesting display.

By comparison to the minimum display size, the maximum display supported by *Mapmaker* is truly awesome. The program makes up to 8K of memory available for screen display area. At 240 bytes per screen, 8K provides room for a single display comprising over 34 screens of total display area. Display width can vary from 22 to 128 characters; height can range from 13 to 255 character lines. The only limitations are that the product of width times height cannot exceed 8K (8192 bytes) and the width must be an even number (to simplify the fine scrolling routine).

(If you have a 32K system, your maximum display area is limited to 4K of memory, rather than 8K. The full 8K is available only on systems with 40K or more of memory. However, a 4K area still leaves room for more than 17 screens of display.)

GRAPHICS 2 is a five-color mode. That is, the characters can be any of four colors, and the background can be a fifth color. *Mapmaker* supplies a set of default colors (which differ from the standard ATARI default colors) and provides an easy-to-use facility for altering the "playfield" color registers to allow for any combination of five colors on the display. These color choices are stored with the diskette MAP file so that they are restored automatically upon recalling the map file.

Your primary method of communicating with *Mapmaker* is through the keyboard. However, you can also use a joystick to control the display. The joystick provides for faster scrolling and cursor movement, particularly in diagonal motion. Other features include (1) the ability to swap character sets during editing, (2) a facility for accessing character editor programs directly from the *Mapmaker* program, (3) BREAK key disablement to protect against accidental loss of edit work, and (4) automatic file selection and naming.

Creation of a multiscreen display is an unavoidably time-consuming task. Be prepared to spend several sessions creating a single display, and be prepared for a certain amount of tedious repetition entering characters for portions of displays, such as bodies of water. Clever choice of character shapes, as well as careful selection of field versus ground, will ease your task considerably.

If you're not very familiar with the concept of a hi-res display created from redefined character sets, read the section titled "Understanding Redefined Text Displays" before using *Mapmaker*.

General information

You use EDIT mode to edit your displays. While in EDIT mode, you can move the display window and the cursor using either the joystick or the arrow keys on the keyboard. You can alter the character under the cursor through the keyboard. The OPTION, SELECT and START keys all have important functions in EDIT mode.

Scrolling

Mapmaker's ability to create multiscreen displays is useless unless you can view different portions of the display with ease. This is accomplished with a fine scrolling routine, which moves the screen across the display like a window. The routine is described in detail in the section titled "Fine Scrolling". To scroll the display, hold down the SHIFT key and press any of the four arrow keys. The window moves in the direction of the arrow pressed. For example, to see portions of the display below the bottom of the screen, use the down arrow.

You can also scroll the display using a joystick. Press and hold the red trigger button on the joystick while pushing the joystick in the direction of desired window movement. You can scroll diagonally with a joystick, whereas you can't with the keyboard arrows. The display will not scroll past the borders which define the length and width of the display.

The cursor

When *Mapmaker* first goes into EDIT mode, the screen shows the upper left portion of the display. In the top left corner of the display, on the border, is a blinking cursor. The cursor keeps track of the current location of the character editor. In the lower right of the screen, the current coordinates of the cursor display. Starting from position 0,0 in the upper left, the X axis extends to the right along the width of the display. The Y axis runs down the screen matching the height of the display. As you move the cursor, the program constantly updates the coordinate information. The other information in the lower right concerns the status of the INVERSE flag. When this is OFF, character entry is in the colors of color registers 1 and 2. When the flag is ON, characters appear in the colors of registers 3 and 4. The flag flips each time you press the ATARI (INVERSE video) key prior to entering a new character onto the display. For more discussion of the INVERSE flag, see the section titled "Color Registers".

You can move the cursor in several ways. You can move it by pressing an arrow key. Hold down an arrow key for repeated cursor movement. Or, you can move the cursor by pushing the joystick in the direction in which you want cursor movement. As with scrolling, you can move diagonally only through the joystick. The cursor also moves automatically under program control during scrolling. As the window is moved, the cursor remains in place. However, as the cursor disappears off the edge of the screen, it moves under program control so that it lurks about the edge of the display. This is indicated by the coordinates being updated during fine scrolling. When you finish scrolling, if the cursor isn't visible, a few cursor moves will bring it into view. Also, if you move the cursor off the screen, the cursor will move back to the screen as soon as you stop issuing cursor directions.

The cursor indicates the current location for character insertion onto the display. This is further discussed in the following section, "Character Editing."

You can turn off the cursor to view the display without the distraction of a blinking cursor. Press the SELECT key to remove the cursor and the information display in the lower right of the screen. Press SELECT again to redisplay the cursor. The SELECT key also alters the "colors" of the coordinate data. Originally displayed in black, the readout will display in white when the cursor is turned off and then turned on again. This is useful when you alter the background color from the default choice of white to a dark intensity color. Press SELECT twice more to redisplay the readout in black.

Character editing

You use the keyboard and the START key to enter and change characters on the display. Position the cursor at the location desired for a new character. Next, press the START key; the cursor stops blinking. Now press the key corresponding to the character desired. That character appears on the screen at the cursor location, and the cursor blinks again. The START key in EDIT mode signals to *Mapmaker* that the next key pressed should be translated into a character to be displayed at the current cursor location. You may press the ATARI key to change color registers after pressing the START key but before you press the key for the character. All keys are interpreted as characters, even RETURN, TAB, and SHIFT-CLEAR. The BREAK key is disabled. Any key you press is converted into characters. Therefore, you can't lose the display no matter what key you press (see the warning about the SYSTEM RESET key below, however). The keyboard is in lower case, so use the SHIFT key to enter uppercase characters. You can still use the CAPS-LOWR key to lock in upper case or the CONTROL (CTRL) key when you need to enter a series of such characters.

The SYSTEM RESET key has *not* been disabled or reprogrammed. *Never* press this button while using *Mapmaker*, unless you're certain you don't wish to save your map-work. To exit the program, use the MENU.

Exiting EDIT mode

Use the OPTION key to leave EDIT mode and return to the MENU, where you can access *Mapmaker's* special functions. You can also return to EDIT mode from the MENU. The next section describes the MENU functions.

General information

In addition to the ability to save and recall edited map files, *Mapmaker* contains several special functions, which you can access through the MENU. The options are described below.

EDIT MAP

Use this option to exit from the MENU and enter EDIT mode. The screen returns to the map display at the same location that was on the screen when you pressed the OPTION key.

CHANGE FONT


Use this option to load a different character set file. Or, you can swap the character base from the uppercase/punctuation set to the lowercase/control character set, and back. The alternatives are displayed when you press START while the pointers are set on this option. The choices are to press "S" to swap the character base, or "L" to load a new font. Be sure that the diskette containing the font to be loaded is already in the drive before choosing the "L" option. If there are several character sets on the diskette, the program gives you a choice of files to load. After loading or swapping a character set, you return to the MENU.

RUN FONT EDITOR

This selection isn't active in this version of *Mapmaker*. Choosing this selection has no effect other than to reset the MENU pointer at EDIT MAP. It's included because the option to move back and forth quickly and easily between *Mapmaker* and a character editor program is often desirable when you're designing the characters as they're needed, rather than all at once before map editing commences. The option is inactive, in part, because its exact form depends on the character editor program in use, and in part because alterations are usually required in the editor program to optimize the rotation between it and *Mapmaker*. See the section titled "Character Sets" for further information.

CHANGE COLOR REGISTERS

Use this option to activate a mechanism for altering the colors in the five "playfield" color registers. The screen shows a split-screen display. The color of register 1 fills the upper area. In the lower text window are instructions for altering the registers.




You can vary both the hue and the intensity of colors, using either the keyboard or joystick. To change the hue, move the joystick right or left, or hold down the CTRL key and press the left-arrow or right-arrow. As the text window indicates, the first register displayed is register 1. To select registers 2 - 4 or the background "B" register, press the SELECT key. The color of each register displays in turn. The colors selected apply only to the map display, but the changes are permanent, and are stored with the map file on diskette. You can alter them again by returning to this option. Press the OPTION key to return to the MENU. Instructions for using the color changing facility display in the text window.

SAVE MAP TO DISK

Use this option to save your entire map on the diskette currently in drive 1. Use SAVE often; it protects your work. The file will contain the values of the five color registers for the map at the time you select this option. *Mapmaker* includes an automatic file naming facility. If you're creating a new map display, *Mapmaker* assigns the name FILE1.MAP to your map the first time you save it. If a file already exists on the diskette named FILE1.MAP, *Mapmaker* overwrites it, *unless* it is currently locked. If *Mapmaker* encounters such a locked file, the program names the new map file FILE2.MAP. Subsequent files are named FILE3.MAP, and so on.

At PROGRAM INITIALIZATION time, if you're loading an existing MAP file, that file is unlocked, and *Mapmaker* uses that filename when you select the SAVE option. You may freely change filenames through the DOS menu. The only requirement is that the extender of the filename must be ".MAP" (for example, "USA.MAP"). The automatic filenaming facility is efficient and safe, but it requires you always to exit *Mapmaker* by way of the QUIT option. **Never leave Mapmaker by pressing the SYSTEM RESET key.** Once the MAP file has been saved, control returns to the MENU. You can return to EDIT mode or terminate the session using the QUIT option. See the section on MAP FILES for further information regarding the contents of saved files.



QUIT

Use this option to leave *Mapmaker*. Using QUIT right after using the SAVE option locks your MAP file before you exit the program. Otherwise, when you select QUIT, the program displays a warning that no SAVE has occurred since the last time you were in EDIT mode. If you forgot to save your display, press OPTION to return to the MENU. Otherwise, pressing START again locks any currently open MAP file and returns you to BASIC.

Understanding redefined text displays

To understand the technique involved in creating a text-based, high-resolution display, you first need to understand the difference between a text mode and a bit-map mode. The computer generated display on the television screen is composed of many small dots, called "pixels" (for "picture element"). A given pixel appears as a dot of color on the screen, or as a gray or white dot. Screen displays are made up of patterns of these dots. A bit-map display uses computer memory to store the status of individual pixels using one or two bits of memory. Two-color, bit-map graphics modes use one bit per pixel ("on" or "off"), whereas four-color modes use two bits per pixel (to choose one of four colors). Although these modes let a single byte of memory store the color data for four or eight individual pixels, higher resolution displays can have thousands of pixels on the screen.

Text modes, on the other hand, use relatively little screen memory, because each byte now names a text character. Individual text characters on the screen comprise an area of 64 pixels (an 8x8 area); hence much less memory is needed to fill the screen. When operating in a text mode, the screen display system reads each byte of screen memory and uses the value found to locate an eight-byte shape in a table of character shapes. The 64 bits of this byte group describe the appearance of the text character on the screen. For example, Figure 1 shows the shape of the letter "R" in standard ATARI text format.



Figure 1 "R" in standard ATARI text format

The standard text character shape table is stored in a 1K area of the 10K ROM Operating System. One byte in RAM, address 756 (\$2F4 hex), and referred to as CHBAS, is a "pointer" to the table of text shapes. This means that location 756 contains the high byte of the two-byte address where the shape table begins. The shape table starts at location 57344 (\$E000) and so the value stored in location 756 is 224 (\$E0), which is 57344 divided by 256. If you were to change the address stored in CHBAS, you would be telling the computer to look elsewhere to find the shape table. This lets you create a table of alternate character shapes, store that table in a 1K area of RAM, and tell the ATARI to use that set of shapes for displaying text on the screen.

Suppose you've redefined the shape of the letter "R" to look like the shape in Figure 2.



Figure 2 Redefined "R"

Now when you call for the letter "R" on the screen, you see in its place a tree. To create a forest on the screen, you would enter several R's in close contact. As characters are put on the screen, they stand shoulder to shoulder without visible gaps or seams. Standard characters maintain separation by not filling the entire 8x8 area they are allocated. In redefining text shapes, you can run right up to the borders of the character grid. Rivers and coastline are generated by making characters which meet at common locations on the edges of the character grid. Figure 3 shows two shapes of "North Coast" characters which, when placed side by side, display an irregular pattern of seacoast.



Figure 3 North Coast Characters

To accomplish the *Mapmaker* technique, you create a variant character set with symbolic and cartographic shapes, then load that shape table into RAM, point to it, and start plugging in characters in a multiscreen GRAPHICS 2+16 (i.e., full-screen) display. *Mapmaker* provides an all-purpose mapping character set (the file is named ALLPRPS.SET) for you to experiment with. However, the full power of *Mapmaker* comes into play when you use a character editor program to create your own redefined character set.

Mapmaker uses GRAPHICS 2 for two reasons. First, GRAPHICS 1 and 2 are five-color modes, compared to GRAPHICS 0, which is only a 1 1/2-color mode. Second, *Mapmaker* uses GRAPHICS 2 rather than 1 because GRAPHICS 2 characters appear square on the screen whereas GRAPHICS 1 characters appear rectangular. The GRAPHICS 2 characters fill a larger area of screen per character, and thus use the least amount of memory per screen display.

In GRAPHICS 2, only 64 of the 128 character shapes are available for use because two bits of the byte that stores the character "name" value are used to denote the color the character should be displayed in. The remaining six bits provide for a range of values from 0 to 63. Thus, a given map in *Mapmaker* is limited to 64 different shapes. Usually, this is more than enough.

For more information about redefined character sets, see the BIBLIOGRAPHY. A thorough understanding of this facet of the ATARI Computer's design is critical to optimal use of *Mapmaker*.

Character sets

The magic ingredient in *Mapmaker* displays is the redefined character set. The section titled "Understanding Redefined Text Displays" briefly explains the technique. You can obtain more information from some of the literature mentioned in the BIBLIOGRAPHY. This section addresses facets of redefined character sets of particular importance in using *Mapmaker*.

You can't create new character sets with *Mapmaker*. For that you need a character set editor. *Instedit*, the character set editor available through APX, creates files with a filename extender of "SET" that *Mapmaker* accepts as character sets. Another good editor is *Fontedit*, which is part of IRIDIS #2, published by The Code Works. *Mapmaker* accepts files created by *Fontedit*. These use "FNT" as a filename extender. You may prefer yet another character editor. To be usable by *Mapmaker*, a character set file must have the following characteristics:

1. The filename must have the extender "SET" or "FNT".
2. The file must be a 1K binary format file of character shapes in the same order as they appear in the ROM OS.

When using a character set in large text graphics modes, only 64 characters of the 128 character set are available since the one-byte character "name" must include the color register value for displaying that character. Two bits identify the color; the other six bits name the character. *Mapmaker* initializes with the uppercase/punctuation subset of the character set. When redefining a set, use this subset rather than the lowercase/control character subset. You may use the latter, provided you also use the CHANGE FONT option in the MENU to swap the character base.

A blank font table is provided in the back of this manual to help you in creating new fonts. As you create characters, enter their descriptions into the appropriate areas of the table. You may make photocopies of this table for your personal use only.

The table also indicates the keys corresponding to the uppercase/punctuation set for generating characters in registers 2 and 4. For example, "A" generates a character in color 1, while "a" displays the same character in color 2. Of course, the same correspondence applies for registers 3 and 4. The difference is that the INVERSE flag must be ON to obtain registers 3 and 4. The status of the INVERSE flag is provided along with the cursor coordinate readout.

When creating a new character set, a few guidelines should be followed. Virtually every character set needs a "fill" character and a "null" character. *Mapmaker* expects the first character in the set to be the "null" character, obtained by pressing the space bar. This is ASCII value 32 (\$20) and internal value 0. This is the same as the standard character set. If you use the lowercase subset, you must redefine the CONTROL-COMMA character to be blank. *Mapmaker* also expects to find a "fill" character at ASCII 42 (\$2A). This is value 10 (\$0A) in internal code and it corresponds to the asterisk (*) character. The standard character set lacks a "fill" character, that is, a character with all 64 pixels "on". This corresponds to the inverse of a blank. The "*" was chosen because it's close to both the START key and the arrow keys. You'll find many occasions to enter multiple fill characters. Since the map border consists of register 4 fill characters, the automatic border drawing routine creates a useless edging if the "*" character isn't properly defined to be "fill." The other 62 characters can be any shapes you desire.



A disciplined approach to creating a new map begins with a plan for the colors and shapes you'll need to interpret the image you wish to display. Keep in mind that 64 shapes are not really all that many. Avoid too many unique shapes and get as much use out of all 64 as possible. If you're using *Instedit*, you'll see the standard character set displayed in a table. Your first step will be to change the "*" character to a fill character, all pixels on. Then start constructing your other shapes by selecting each character in turn, reconfiguring it to suit your needs. Remember that you must use either the uppercase/punctuation half of the set or the lowercase/control character portion. With *Instedit* you can try out different character combinations as you create them. You can always go back and revise individual characters if they don't look quite right in your map, but you'll save some time by trying them out as you create them.

After each character is edited, enter a description of the new character in the Character Set Table next to its corresponding key character. You can make photocopies of the table in the back of this manual to help you keep track of your character sets.

The SAVE command in INSTEDIT is the proper method for storing your character set on diskette for *Mapmaker's* benefit. You can put several .SET files on a single diskette; *Mapmaker* will give you a choice of files to load if it encounters more than one.

The file ALLPRPS.SET is designed to be an all-purpose mapping font. It contains many standard mapping characters, and provides good examples of how irregular terrain can be created through artful character design. Note, for example, that all river characters terminate at the same point on corresponding sides of each character. Coastline can be varied by one pixel for its endpoints as this fits with the irregularity of the coast in any event. At least two characters of each type of terrain should be made to provide for variety and to enhance irregularity. For extended coastlines, four different characters provide a more authentic look.



In sum, a map is only as good as its character set. The more creative your character set, the more original and appealing will be your map. *Mapmaker* encourages experimentation because it's so easy to use.

Option 3: RUN FONT EDITOR

In the description of the MENU, the third option, RUN FONT EDITOR, is inactive. Inquisitive users will discover that lines 7190 and 7195 of MAPMAKER.BAS contain REM lines with code for activating this MENU item. You can implement them by removing the REM's and inserting the name of your character editor program in line 7195. If it were this easy, however, the choice would have been made active. In reality, the task is more difficult. Some modification will probably be necessary to your character set editor program before this transition works smoothly. The goal is to have the editor program come up with the current redefined font already accessible for further editing. The editor program should also provide a quick return to *Mapmaker*. The solution isn't in this manual. You'll need to devise the proper code for linking the two programs.



Color registers

Careful selection of color mixtures is critical to an attractive display. *Mapmaker* uses BASIC's GRAPHICS 2 mode to display characters. This is a five-color text mode, meaning that each character can be displayed in one of four colors against a background that can be a fifth color. A single character can't contain more than one color. There are other graphics modes in the ATARI Computer that do allow for multicolor text characters. *Mapmaker* doesn't currently support these other modes, however. This means that any one square of map area should not contain more than one map feature. If you want a river to wind through a forest, put tree characters next to the river characters. The only way to put tree and river in the same character is to have them both the same color. If you like green rivers or blue trees, go ahead. Another problem with mixing features in a single character is that the character becomes too specialized. With only 64 characters in your palette, you have to get a lot of use out of almost all of them. Creating too many unique characters limits your display.

Recall that the character "name" selects both character and color. This feature deserves more explanation. Each character on the screen corresponds to one byte in RAM. The position of those bytes determines the position of the characters in the display. In reality, the display is one-dimensional in RAM. If the display is, for example, 80 characters wide, then the first 80 bytes of display RAM hold the top row of the display, the next 80 bytes hold the next row, and so on. In each byte is a value corresponding to the character to be displayed. Suppose a particular byte is to display the "fill" character in the color of register 4. This is, in fact, what the display border is composed of. The byte will then hold this binary value: 11001010. The two high bits, 11, designate the color. This is read as a binary value of 3 in decimal. A two-bit value can range from 0 to 3. For ease and clarity, we number the color registers 1 to 4, but internally they are numbered 0 to 3. Hence, 11 binary refers to color register 4, i.e., the fourth of four. The other six bits "001010" are read as value 10 decimal, the internal code value for the "*" character, which we redefined as the "fill" character. Since six bits can range from 0 to 63, we can name 64 characters in this graphics mode. So we see that the binary value "11001010" or 202 decimal (\$CA) "names" the "fill" character in color 4.

Color mixing is a process of trial and error. The ATARI Computer has a lot of colors to choose from, in part so that you can obtain pleasing color combinations. Intensity of the different color registers plays a key role in color mixing. Too many dark colors make a dim, muddy display that doesn't show good contrast. Too many bright colors can be unpleasantly dazzling. The default colors were chosen to provide some basic shades that work acceptably well together. Your displays don't have to keep to these color choices.

One guideline for selecting colors that makes mapping easier and lets *Mapmaker* work better for you is as follows. Registers 1 and 3 should be of contrasting intensity, and the same for registers 2 and 4. You'll want to maintain some contrast anyway, and this scheme works better in *Mapmaker* because the cursor blinks by shifting colors between registers 1-3 and 2-4. If these colors are too similar, the cursor won't have enough contrast to provide a noticeable blink. If you have changed colors and find the cursor hard to spot, readjust the colors for more contrast. You can always change the colors again when you've completed the map.



One final note that applies to both characters and colors: as you create a map with *Mapmaker*, you are constructing a "field/ground" display. By this is meant that you're designating a background color and then overlaying shapes on the background in other colors. It's more usual to see the shapes as "field", that is, covering the background with their shape content. It's entirely possible, however, to create shapes from the "ground", that is, to outline the shapes, fill all the character area around the shape, and let the background color show through as the representation of the shape. While the more customary approach is recommended, certain maps may call for a field/ground reversal that could result in a stunning display. The point is not to let your imagination be bounded by what you've seen. There are many paths to originality.

Map files

If you ever plan to do anything with the maps you create other than to revel in how clever they are, you must understand the structure of the files in which they are saved on diskette. The map data files must have the extender "MAP" in the filename. *Mapmaker* recognizes no other data file. The program also contains an automatic file naming system described earlier in this manual.

When *Mapmaker* saves a display, it writes a diskette file, which contains three subelements. The first two bytes of the file are the width and height of the display in numbers of characters. These values are always less than 256 and thus are one byte in size. When a file is read in, these two values are used to determine the amount of display RAM required for the display. This is particularly critical considering that a display block cannot cross a 4K boundary in RAM, while *Mapmaker* displays can range up to 8K.

The next five bytes on the file are the values of the five color registers, first the four text registers and then the background color. This enables *Mapmaker* to restore the color choices made during earlier editing sessions. The third and largest subsection of the file is the map data. This is written by the simple yet effective method of defining the entire display area in RAM as the buffer for a CIO call for disk output. I am indebted to the publication *De Re ATARI* for describing the function and use of the IOCB in relation to the multiple GET CHARACTERS function. This little subroutine string (CIO\$) in *Mapmaker* does yeoman service in quickly reading in font and map files and in writing out display data. Five simple POKEs and a USR call, and you can read or write any data file. Lines 8000-8080 in *Mapmaker* write the MAP file. If you can understand this, you can construct the necessary code to read the map into another program application. Making maps is fun, but using them in other programs is rewarding. Read *De Re ATARI* and examine *Mapmaker*; then write your own program around a map display of your own creation. That is what *Mapmaker* was really created for.



Fine scrolling

Face it, you really have to know something about assembly language to implement fine scrolling. This technique requires not only a wholesale modification (actually a complete rewrite) of the display list, but also the ability to update the entire display list during the course of a single vertical blank. Furthermore, my experiments with the fine scrolling registers suggest that trying to manipulate them with POKEs simply results in a jumpy, unstable screen display. In short, if you don't know a lot about display lists, USR routines and vertical blank interrupts, get someone to teach you before you try to implement fine scrolling.

Assuming you are at ease with these subjects, the assembly listing of SCROLL.OBJ is provided for your information. It is read into the unused area of the player-missile graphics region, which physically resides just below the display area in *Mapmaker*. (You didn't realize *Mapmaker* has PM graphics? What do you think the cursor coordinate readout is?) Once read in, it is activated by a short USR routine in line 11601 that resets the immediate vertical blank vector to the address of SCROLL. Henceforth, each time the screen is drawn, this routine is executed. It looks to see if the proper keyboard or joystick conditions exist to command scrolling, and if these are tripped, it executes the proper steps to accomplish the scrolling function.

The assembly listing is pretty well documented. If you intend to use a map of your creation in another program, you'll have to use a routine similar to SCROLL to enable you to see the whole display. Note that SCROLL uses some page zero of RAM and also stores some values in page six. SCROLL used to live in page six. That was when it was only joystick-driven. When I added keyboard drive, it outgrew one page of RAM. I moved it, but the data addresses stuck. There is so much going on in *Mapmaker* anyway, what with vertical blank interrupts and multiple display lists, that I thought it best to ban any other routines from page six while *Mapmaker* was running. Note, by the way, that *Mapmaker* supports three separate display lists simultaneously.

De Re ATARI has something to say about fine scrolling, but you have to see it in action to really learn how to use it. SCROLL is in this manual because you have to understand the technique before you can apply your maps to other applications. I hope it becomes a little clearer to those of you who choose to delve more deeply into your computer's capabilities.

Bibliography

The following works can help users wanting to learn more about the concepts used in *Mapmaker*. Not included are the many fine articles that have appeared in periodicals that focus on the ATARI Computer. Sources are still scanty, but the list is growing.

Chen, Amy *et al.*, *De Re ATARI*, ATARI Program Exchange (APX-90008), 1981.

Crawford, Chris, *Source Code for Eastern Front (1941)*, ATARI Program Exchange (APX-20095), 1981.

Inman, Don and Kurt Inman, *The ATARI Assembler*, Reston Publishing Company Inc., 1981.

Lock, Robert, ed., *COMPUTE'S First Book of ATARI*, Small System Services, Inc., 1981.

Moore, Herb, Judy Lower and Bob Albrecht, *ATARI Sound and Graphics*, John Wiley & Sons, Inc., 1982.

Poole, Lon, Martin McNiff & Steven Cook. *Your ATARI Computer*, Osborne/McGraw-Hill, 1982.

0000
0000

```
10 .TITLE "SCROLL.SRC ASSEMBLED AS SCROLL.OBJ"
20 .OPT NOEJECT
30 ;
40 ;
50 ;
60 ;
70 ;
80 ; *****
90 ; *
100 ; * SCROLL.SRC *
110 ; * by *
120 ; * STEPHEN W. HALL *
130 ; *
140 ; * COPYRIGHT (C) 1982 *
150 ; * by *
160 ; * PALANTIR PRODUCTS CORP *
170 ; * *****
180 ;
190 ;
200 ;
210 ;SCROLL is an integral part of a program called
220 ;MAPMAKER. It provides the capability to fine
230 ;scroll a multi-screen display.
240 ;
250 ;SCROLL operates under either keyboard or
260 ;joystick control. It takes the joystick value
270 ;(or the keyboard arrow key value converted to
280 ;its equivalent joystick value) and interprets
290 ;that value into display movement actions. Fine
300 ;scrolling is a combination of incrementing the
310 ;fine scroll registers and adjusting the LMS
320 ;((Load Memory Scan) addresses in the display
330 ;list to accomplish a coarse scroll each time
340 ;the fine scroll register wraps around its
350 ;maximum value.
360 ;
370 ;The lines preceding label EXIT establish that the
380 ;conditions for scrolling are met. They take the
390 ;joystick value (or the converted key value) into
400 ;the actual scrolling routines which follow.
410 ;
420 ;Four essentially similar sections accomplish the
430 ;scrolling. Each section directs scrolling in a pri-
440 ;mary direction.
450 ;
460 ;SCROLL is a vertical blank interrupt routine. It
470 ;executes every vertical blank. Once loaded into RAM,
480 ;it is enabled by a USR routine in MAPMAKER.
490 ;
500 ;To accomplish fast scrolling with the keyboard, SCROLL
```

SCROLL.SRC ASSEMBLED AS SCROLL.OBJ

```

0000      0510 ;          .PAGE
          0520 ;has to defeat the keyboard debounce feature. This is
          0530 ;done by storing a 1 in the debounce system time. The
          0540 ;following System Vertical Blank routine decrements
          0550 ;the timer to 0, clearing the debounce.
          0560 ;
          0570 ;MAPMAKER can disable scrolling in SCROLL by storing
          0580 ;a 0 in location FLAG. This is done during character
          0590 ;entry and MENU mode.
          0600 ;
          0610 ;The following documentation completes the description
          0620 ;of SCROLL.
          0630 ;
          0640 ;
          0650 ;
          0660 ;
          0670 ;
0000      0680      *= $6F0          ;PAGE 6 VARIABLES
06F0      0690 HLEN   = *           ;MAX. HORIZ. COARSE SCROLL VALUE
06F1      0700 VLEN   = **1        ;MAX. VERT. COARSE SCROLL VALUE
06F2      0710 INCRA  = **2        ;DISPLAY WIDTH FOR COARSE VERT. SCROLL
06F3      0720 DECRA  = **3        ;NEGATIVE OF INCRA IN TWOS COMP.
06F4      0730 FLAG   = **4        ;START-STOP FLAG
06F5      0740 VBLIN  = **5        ;INDIRECT JUMP ADDRESS OF VBLANK ROUTINE
06F7      0750 LINEOUT= **7        ;CURRENT VALUE OF HORIZ. COARSE SCROLL
06F8      0760 PAGEOUT= **8        ;CURRENT VALUE OF VERT. SCROLL
06F0      0770      *= 203          ;PAGE ZERO VARIABLES
00CB      0780 INDA   = *           ;DISPLAY LIST BASE ADDR.
00CD      0790 HSDAT  = **2        ;SHADOWS HORIZ. FINE SCROLL REG.
00CE      0800 VSDAT  = **3        ;SHADOWS VERT. FINE SCROLL REG.
D404      0810 HSCROL = $D404       ;HORIZ. FINE SCROLL REGISTER
D405      0820 VSCROL = $D405       ;VERT. FINE SCROLL REGISTER
0278      0830 STICK  = $278        ;STICK 0 REGISTER
0284      0840 STRIG  = $284        ;STICK 0 TRIGGER REGISTER
022B      0850 SRTIMR = $22B        ;SYSTEM TIMER FOR KEYBOARD DEBOUNCE
02FC      0860 CH     = $2FC        ;LAST KEYBOARD CHAR. PRESSED
00CB      0870      *= $6000       ;DUMMY INIT. ADDRESS
6000 ADF405 0880      LDA FLAG      ;GET THE VALUE OF FLAG
6003 F03C   0890      BEQ EXIT      ;IF FALSE, EXIT SCROLL
6005 AD8402 0900      LDA STRIG     ;GET TRIGGER STATUS:
6008 D008   0910      BNE CHAR     ;IF TRUE, TEST FOR ARROW
600A AD7802 0920      LDA STICK     ;TRIGGER IS PUSHED: GET STICK VALUE
600D AA     0930      TAX           ;STORE IT IN X
600E C90F   0940      CMP #15      ;IS THE STICK OFF CENTER?
6010 D035   0950      BNE SCRL     ;YES: GO TO SCROLL ROUTINE
6012 ADFC02 0960 CHAR  LDA CH      ;GET LAST KEY PRESSED
6015 AA     0970      TAX           ;STORE IT IN X
5016 29B0   0980      AND **B0     ;TEST FOR SHIFT BIT ONLY
6018 D027   0990      BNE EXIT     ;NOT A SCROLL VALUE
601A 8A     1000      TXA          ;RETRIEVE ORIGINAL VALUE

```

SCROLL.SRC ASSEMBLED AS SCROLL.OBJ

```

601B      1010      .PAGE
601B 2940      1020      AND**40      ;TEST FOR SHIFT BIT
601D F022      1030      BEQ EXIT      ;IF OFF, NOT A SCROLL VALUE
601F 8A        1040      TXA          ;RETRIEVE
6020 290F      1050      AND **$0F      ;STRIP OFF BITS 4-7
6022 AA        1060      TAX          ;STORE IN X
6023 2906      1070      AND *6        ;THE FOUR ARROW KEYS ALL HAVE
6025 C906      1080      CMP *6        ; BITS 1 AND 2 ON, THIS TESTS
6027 D018      1090      BNE EXIT      ; FOR THESE FOUR KEYS
6029 A9FF      1100      LDA **FF      ;CLEAR THE KEYBOARD REGISTER
602B 8DFC02    1110      STA CH       ; BY POKING A 255 INTO IT
602E 8A        1120      TXA          ;RETRIEVE
602F C90F      1130      CMP *15      ;KEY VALUE 15 EQUATES TO STICK
6031 D005      1140      BNE BYE      ; VALUE 13. IF EQUAL TO 15, THEN
6033 A90D      1150      LDA *13      ; CONVERT TO 13 AND GO TO SCROLL
6035 AA        1160      TAX
6036 D00C      1170      BNE SCRLK
6038 C906      1180      BYE      CMP *6        ;IT WASN'T 15. IS IT 6? IF NOT, THE KEY
603A D008      1190      BNE SCRLK    ; VALUE IS THE SAME AS THE STICK VALUE
603C A90B      1200      LDA *11      ;CONVERT 6 TO 11 STICK VALUE
603E AA        1210      TAX          ;STORE IT
603F D003      1220      BNE SCRLK    ;SKIP A LINE
6041 60F506    1230      EXIT      JMP (VBLIN)  ;EXIT POINT FOR NO SCROLL
6044 0980      1240      SCRLK     ORA **$80    ;ON AN ARROW SCROLL, SET BIT 7
6046 AA        1250      TAX          ;STORE IT
6047 D8        1260      SCRL     CLD          ;SCROLL DISPLAY LEFT STARTS HERE
6048 2908      1270      AND *8        ;TEST FOR BIT 3
604A D031      1280      BNE NEXT2    ;ON? GO TO NEXT DIRECTION
604C A5CD      1290      LDA HSDAT    ;GET CURRENT HORIZ. FINE SCROLL VALUE
604E 290F      1300      AND **$0F    ;TEST BITS 0-3
6050 D023      1310      BNE PASS1    ;ANY ON? FINE SCROLL ONLY
6052 ADF706    1320      LDA LINEDUT ;GET THE HORIZ. COARSE COUNTER
6055 CDF006    1330      CMP HLEN     ;COMPARE TO HORIZ. COARSE LIMIT
6058 F023      1340      BEQ NEXT2    ;REACH LIMIT? THEN DON'T SCROLL
605A A004      1350      LDY *4        ;BEGIN COARSE SCROLL ROUTINE
605C B1C3      1360      LOOP1     LDA (INDA),Y
605E 18        1370      CLC          ; THIS SECTION GETS THE LOW BYTE
605F 6902      1380      ADC *2        ; OF EACH LOAD MEMORY SCAN ADDRESS
6061 91CB      1390      STA (INDA),Y ; IN THE DISPLAY LIST AND
6063 C8        1400      INY          ; ADS 2 TO IT, THEN STORES IT BACK
6064 B1CB      1410      LDA (INDA),Y ; IN PLACE, THEN IT GETS THE HIGH
6066 6900      1420      ADC *0        ; BYTE AND ADDS THE OVERFLOW,
6068 91CB      1430      STA (INDA),Y ; IF ANY. WHY 2? BECAUSE FINE
606A C8        1440      INY          ; HORIZ. SCROLLING RANGES OVER 2
606B C8        1450      INY          ; CHARACTERS IN LARGE TEXT MODE.
606C C02B      1460      CPY *43
606E D0EC      1470      BNE LOOP1    ;END OF COARSE SCROLL LOOP
6070 EEF706    1480      INC LINEDUT  ;ADD 1 TO THE HORIZ. COUNT
6073 A5CD      1490      LDA HSDAT    ;GET THE HSCROL VALUE
6075 18        1500      PASS1     CLC
6076 69FF      1510      ADC **FF      ;SUBTRACT 1

```

SCROLL.SRC ASSEMBLED AS SCROLL.OBJ

```

6078          1520          .PAGE
6078 85CD      1530          STA HSDAT          ;STORE IT IN THE SHADOW, AND IN
607A 8D04DR   1540          STA HSCROL         ; THE HORIZ. FINE SCROLL REG.
607D 8A       1550 NEXT2   TXA              ;NOW REPEAT THE PROCESS FOR BIT 2
607E 2904     1560          AND #4           ; THIS SECTION MOVES THE DISPLAY RIGHT
6080 D030     1570          BNE NEXT3
6082 A5CD     1580          LDA HSDAT
6084 290F     1590          AND #$0F
6086 C90F     1600          CMP #15
6088 D020     1610          BNE PASS2
608A ADF706   1620          LDA LINEOUT
608D F023     1630          BEQ NEXT3
608F A004     1640          LDY #4
6091 B1CB     1650 LOOP2   LDA (INDA),Y
6093 18       1660          CLC
6094 69FE     1670          ADC #$FE
6096 91CB     1680          STA (INDA),Y
6098 C8       1690          INY
6099 B1CB     1700          LDA (INDA),Y
609B E900     1710          SBC #0
609D 91CB     1720          STA (INDA),Y
609F C8       1730          INY
60A0 C8       1740          INY
60A1 C02B     1750          CPY #43
60A3 D0EC     1760          BNE LOOP2
60A5 CEF706   1770          DEC LINEOUT
60A8 A5CD     1780          LDA HSDAT
60AA 18       1790 PASS2   CLC
60AB 6901     1800          ADC #1
60AD 85CD     1810          STA HSDAT
60AF 8D04D4   1820          STA HSCROL
60B2 8A       1830 NEXT3   TXA              ;MOVE THE DISPLAY UP HERE
60B3 2902     1840          AND #2
60B5 D034     1850          BNE NEXT4
60B7 A5CE     1860          LDA VSDAT
60B9 290F     1870          AND #$0F
60BB C90F     1880          CMP #15
60BD D024     1890          BNE PASS3
60BF ADF806   1900          LDA PAGEOUT
60C2 CDF106   1910          CMP VLEN
60C5 F024     1920          BEQ NEXT4
60C7 A004     1930          LDY #4
60C9 B1CB     1940 LOOP3   LDA (INDA),Y
60CB 18       1950          CLC
60CC 6DF206   1960          ADC INCRA      ; FOR UP-DOWN COARSE SCROLLING,
60CF 91CB     1970          STA (INDA),Y   ; ADD OR SUBTRACT THE SIZE
60D1 C8       1980          INY           ; OF THE DISPLAY WIDTH.
60D2 B1CB     1990          LDA (INDA),Y
60D4 6900     2000          ADC #0
60D6 91CB     2010          STA (INDA),Y
60D8 C8       2020          INY

```

SCROLL.SRC ASSEMBLED AS SCROLL.OBJ

```

60D9          2030          .PAGE
60D9 C8       2040          INY
60DA C02B     2050          CPY #43
60DC D0EB     2060          BNE LOOP3
60DE EEF806   2070          INC PAGEOUT
60E1 A5CE     2080          LDA VSDAT
60E3 18       2090 PASS3   CLC
60E4 6901     2100          ADC #1
60E6 85CE     2110          STA VSDAT
60E8 8D05D4   2120          STA VSCROL
60EB 8A       2130 NEXT4   TXA          ;MOVE THE DISPLAY DOWN HERE
60EC 2901     2140          AND #1
60EE D02F     2150          BNE EXIT2
60F0 A5CE     2160          LDA VSDAT
60F2 290F     2170          AND #$0F
60F4 D021     2180          BNE PASS4
60F6 ADF806   2190          LDA PAGEOUT
60F9 F024     2200          BEQ EXIT2
60FB A004     2210          LDY #4
60FD B1CB     2220 LOOP4   LDA (INDA),Y
60FF 18       2230          CLC
6100 6DF306   2240          ADC DECRA
6103 91CB     2250          STA (INDA),Y
6105 C8       2260          INY
6106 B1CB     2270          LDA (INDA),Y
6108 E900     2280          SBC #0
610A 91CB     2290          STA (INDA),Y
610C C8       2300          INY
610D C8       2310          INY
610E C02B     2320          CPY #43
6110 D0EB     2330          BNE LOOP4
6112 CEF806   2340          DEC PAGEOUT
6115 A5CE     2350          LDA VSDAT
6117 18       2360 PASS4   CLC
6118 69FF     2370          ADC #$FF
611A 85CE     2380          STA VSDAT
611C 8D05D4   2390          STA VSCROL
611F 8A       2400 EXIT2   TXA          ;GET THE VALUE ONE MORE TIME
6120 1005     2410          BPL EXIT3          ;IF A STICK SCROLL, GET OUT
6122 A901     2420          LDA #$01          ;ON A KEYBOARD SCROLL, WE WANT TO
6124 8D2B02   2430          STA SRTIMR          ; DISABLE DEBOUNCE, THIS DOES IT.
6127 F8       2440 EXIT3   SED          ;RESET THE DECIMAL FLAG FOR BASIC
6128 6CF506   2450          JMP (VBLIN)
612B         2460          .END

```

FILENAME: ALLPRPS.SET (.SET)(.FNT)

	REG 1	REG 2	REG 3	REG 4	REG 1	REG 2	REG 3	REG 4
--BLANK--	SPACE	C(,)			--CITY--	@	C(,)	
WNS RIVER	!	C(A)			_N COAST	A	a	
NS RIVER	"	C(B)			_N COAST	B	b	
NW RIVER	*	C(C)			_N COAST	C	c	
WNE RIVER	\$	C(D)			_NE COAST	D	d	
SW RIVER	%	C(E)			_NE COAST	E	e	
NSE RIVER	&	C(F)			_E COAST	F	f	
EW RIVER	'	C(G)			_E COAST	G	g	
WSE RIVER	(C(H)			_E COAST	H	h	
WNE RIVER)	C(I)			_SE COAST	I	i	
--FILL--	*	C(J)			_SE COAST	J	j	
_R MOUTH	+	C(K)			_S COAST	K	k	
LG RIVER	,	C(L)			_S COAST	L	l	
L-S RIVER	-	C(M)			_S COAST	M	m	
_FACTORY	.	C(N)			_SW COAST	N	n	
--TOWER--	/	C(O)			_SW COAST	O	o	
LT CASTLE	0	C(P)			_W COAST	P	p	
RT CASTLE	1	C(Q)			_W COAST	Q	q	
--FLAG--	2	C(R)			_W COAST	R	r	
SWASTIKA	3	C(S)			_NW COAST	S	s	
MLT CROSS	4	C(T)			_NW COAST	T	t	
--STAR--	5	C(U)			_LT SHIP	U	u	
LT BSHIP	6	C(V)			_RT SHIP	V	v	
CT BSHIP	7	C(W)			_LT TANK	W	w	
RT BSHIP	8	C(X)			_RT TANK	X	x	
HILO TREE	9	C(Y)			_CAVALRY	Y	y	
LOHI TREE	:	C(Z)			_SOLDIER	Z	z	
_LG TREE	;	ESC		RETURN	LG GRASS	[C(;	
HILO MNT	<	C(-)		SH DEL	LOHI GRAS	\		
LOHI MNT	=	C(=)		SH INS	HILO GRAS]	SH(<)	C(2)
LG MOUNT	>	C(+)		C(TAB)	_LAKE	^	BACK S	C(DEL)
_VOLCANO	?	C(*)		SH TAB	_GEAR	_	TAB	C(INS)

NOTE: "C" means CONTROL (CTRL) KEY and (character).
 "SH" means SHIFT KEY and (character).
 "DEL" means DELETE KEY.
 "INS" means INSERT KEY.
 "REG" means COLOR REGISTER.

LT=left CT=center RT=right LG=large MT=mountain
 "River mouth" is a West Coast character
 "L-S River" means large-to-small conversion
 "+", ",", and "-" are meant to be a trio

FILENAME: ALLPRPS.SET (.SET)(.FNT)

REG 1	REG 2	REG 3	REG 4	REG 1	REG 2	REG 3	REG 4
SPACE	C(,)			@	C(,)		
!	C(A)			A	a		
"	C(B)			B	b		
#	C(C)			C	c		
\$	C(D)			D	d		
%	C(E)			E	e		
&	C(F)			F	f		
'	C(G)			G	g		
(C(H)			H	h		
)	C(I)			I	i		
*	C(J)			J	j		
+	C(K)			K	k		
,	C(L)			L	l		
-	C(M)			M	m		
.	C(N)			N	n		
/	C(O)			O	o		
0	C(P)			P	p		
1	C(Q)			Q	q		
2	C(R)			R	r		
3	C(S)			S	s		
4	C(T)			T	t		
5	C(U)			U	u		
6	C(V)			V	v		
7	C(W)			W	w		
8	C(X)			X	x		
9	C(Y)			Y	y		
:	C(Z)			Z	z		
;	ESC		RETURN	[C(;		
<	C(-)		SH DEL	\			
=	C(=)		SH INS]	SH(<)		C(2)
>	C(+)		C(TAB)	^	BACK S		C(DEL)
?	C(*)		SH TAB	_	TAB		C(INS)

NOTE: "C" means CONTROL (CTRL) KEY and (character).
 "SH" means SHIFT KEY and (character). "DEL" means DELETE KEY.
 "INS" means INSERT KEY.
 "REG" means COLOR REGISTER.

Permission is granted to copy this page only.



ATARI Program Exchange
P.O. Box 3705
Santa Clara, CA 95055

Review Form

We're interested in your experiences with APX programs and documentation, both favorable and unfavorable. Many of our authors are eager to improve their programs if they know what you want. And, of course, we want to know about any bugs that slipped by us, so that the author can fix them. We also want to

know whether our instructions are meeting your needs. You are our best source for suggesting improvements! Please help us by taking a moment to fill in this review sheet. Fold the sheet in thirds and seal it so that the address on the bottom of the back becomes the envelope front. Thank you for helping us!

1. Name and APX number of program.

2. If you have problems using the program, please describe them here.

3. What do you especially like about this program?

4. What do you think the program's weaknesses are?

5. How can the catalog description be more accurate or comprehensive?

6. On a scale of 1 to 10, 1 being "poor" and 10 being "excellent", please rate the following aspects of this program:

- _____ Easy to use
- _____ User-oriented (e.g., menus, prompts, clear language)
- _____ Enjoyable
- _____ Self-instructive
- _____ Use (non-game programs)
- _____ Imaginative graphics and sound

7. Describe any technical errors you found in the user instructions (please give page numbers).

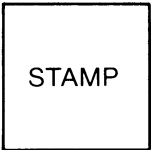
8. What did you especially like about the user instructions?

9. What revisions or additions would improve these instructions?

10. On a scale of 1 to 10, 1 representing "poor" and 10 representing "excellent", how would you rate the user instructions and why?

11. Other comments about the program or user instructions:

From



APX ATARI Program Exchange
P.O. Box 3705
Santa Clara, CA 95055

[seal here]

MAPMAKER

by Stephen W. Hall

- Create multiscreen maps using color, fine-scrolling, and redefined character sets
- Fit large maps into small amounts of memory
- Store completed maps on diskette

Whether you design computerized simulation games or just want to get the most from your ATARI Computer, you'll find *Mapmaker* a great programming tool. This easy-to-use utility can help you create multiscreen displays that capitalize on the ATARI Computer's outstanding multicolor, fine-scrolled redefined character sets. The large display that can fit into a small amount of memory is truly impressive. On systems with at least 40K of memory, *Mapmaker* makes up to 8K available for displays compris-

ing as many as 34 screens.

To create a map with *Mapmaker*, you define the dimensions of the total display area, and then enter characters from the computer keyboard. *Mapmaker* can act as an easel upon which you can try out different combinations of shapes by displaying them on the screen, and with fine scrolling you can move smoothly from one map area to another. *Mapmaker* comes with an all-purpose, map element character set and a sample eight-screen map. You can also use a separate character set editor (such as *Instedit*, available through APX) to create your own mapping characters and then produce an infinite variety of displays with *Mapmaker*. *Mapmaker* supports four character colors, plus a background color, and comes with an easy-to-use facility for changing the playfield color registers.

Requires:

- ATARI BASIC Language Cartridge

Diskette

- (APX-20144)
- ATARI 810™ Disk Drive
- 32K RAM

Optional:

- ATARI Joystick Controller
- *Instedit* or other Compatible character set editor



About the author

STEPHEN W. HALL

The idea for *Mapmaker* was born when Stephen Hall, of Arlington, Virginia, met Chris Crawford, author of APX's award-winning *Eastern Front*, at an ATARI users' seminar. As a war games enthusiast, Stephen was fascinated by the game's intricate screen display. He reasoned that anyone who was going to create war games would need a good utility

—and in writing one, Stephen could learn the technique for himself. His hobby has led him to a new career as a program developer for a software retailer. As computer games editor for *Fire and Movement*, a war games magazine, Stephen also writes regular reviews (including one praising *Eastern Front* as a ground-breaking achievement).