FOR AGES
8 TO ADULT

SPRITEMASTER 64™

BY
ACCESS

COMPUTER
ANIMATION
PROGRAM

FOR USE WITH THE COMMODORE 64 COMPUTER

■ TAPE
■ DISK

Spritemaster® 64
by access

USER'S MANUAL

ACCESS SOFTWARE
925 EAST 900 SOUTH
SALT LAKE CITY, UTAH 84105

Table of Contents

1
Introduction

Program Description

The **Spritemaster 64**™ animation program is designed as a sprite generator and editor for use by programmers or as a fun and creative playmate for children or adults.

Sprites are moveable object blocks which, through the power of the new Commodore 64* computer, can be made to imitate almost any moving object.

Spritemaster™ helps you build your sprite, then copy and modify it to produce a sequence of pictures which represent the object in motion. The **Animate** command will then put each picture on the screen in rapid succession to create the animation. This is similar to what takes place in a movie projector where still photographs, each a slightly different picture, are rapidly projected one after another onto the screen to produce a "motion picture."

Spritemaster™ lets you modify your sprite picture sequence to produce colors, shapes and sizes which give the best effect. In addition, you may change the speed of animation (the rate at which the still frames are displayed) or the speed of travel (the vertical or horizontal movement across the screen).

Once you have created your animated figure, you can save the informaton to tape or disk for retrieval at a later time. You may want to refine several figures for use in a game program.

Although **Spritemaster**™ is not a game program, it is a program to create game objects. For example, it cannot display a pitcher throwing a baseball to a catcher. It can, however, be used to create the pitcher as a separate sprite, including the throwing motion, then the baseball as another sprite, and then the catcher as the final sprite. Each object is created separately and then transfered to other programs for interaction and movement as desired. The programmer's reference section provides guidance on how to manipulate sprites in your own programs.

As a learning companion for children, **Spritemaster**™ helps teach artistry and creativity. Children can experience the excitement of first visualizing the still picture sequence involved in motion and then seeing their creations spring to life on the screen.

Sprites and the difference between single and multicolor

The Commodore 64 has outstanding graphics resolution. The screen is layed out in a series of small dots called picture elements or "pixels." The screen area is 320 pixels horizontally and 200 pixels vertically. A sprite is a high resolution graphics object which in its normal size is 24 pixels wide by 21 pixels high. It can be expanded in either the vertical or horizontal direction to occupy 48 pixels wide and/or 42 pixels high. When a sprite is created and stored in memory it is always in the normal or 24 x 21 size. A single poke to a register in the video chip will expand it in the horizontal (X direction) while another poke will expand it in the vertical (Y direction).

A sprite may be either single or multicolored. In single color mode, each pixel may be 'on" or "off." When turned "on," the color of the pixel is the same as for all other "on" pixels in that particular sprite. The color for the entire sprite is set by poking a register with a value between 0 and 15 representing one of the 16 available colors.

In multicolor mode, the pixels are arranged in sets of two and work only in pairs. Either both are "on" or both are "off."

Because of this, the horizontal resolution is cut in half, but now each pair of pixels can represent one of three different colors in addition to off (the background color).

*Commodore 64 is a trademark of Commodore Business Machines

Coding sprite data is a tedious and complex process especially for multicolor sprites. The **Spritemaster**™ program does everything for you by automatically coding and decoding sprite data. You need only specify single or multicolor mode and then select the colors. Should you wish to use the sprites created with **Spritemaster**™ in your own program, however, you will need to know how to use and manipulate sprites and sprite colors. A thorough discussion on how to do this is given in the programmer's reference section of this manual.

Care of Diskettes

Always observe the following rules when handling diskettes to prevent damage and loss of program or data.

1. Always remove your diskette from the disk drive before applying power or turning off power to the drive.
2. Always return the diskette to its protective envelope when it is removed from the drive.
3. Keep diskettes away from magnetic fields, excessive heat, or sunlight.
4. Do not touch the diskette surface. Always handle by the label end.
5. Do not write on the plastic jacket with a lead pencil or ball point pen. Use a felt tip pen if you need to write on the jacket or label.

Care of Cassette Tapes

Rewind your tape to the beginning after each use to reduce the possibility of abrasion or contamination. Do not expose the tape to a strong magnetic field.

— CAUTION —

The **Spritemaster**™ program on disk has "write protect labels" installed which will prevent the use of the disk for storage of sprite data or the inadvertant destruction of the program or demo files.

Do not remove the "Write Protect Label." To do so may cause your program to become unusable.

Demonstration Sprites

The following sprite data files contain demonstration sprites.

Refer to the discussion on "Loading Demonstration Sprites" in Chapter 2 for instructions on how to access and use them.

Tape: On tape there is only one file containing two sprites, located right after the program.

Demo *#1* (walking man) enter sequence 1,2,3,4,5,6,7,8,9,0
Demo *#2* (flying bird) enter sequence 11,12,13,14,13,12,0

Disk: On disk there are two files which contain demo sprites. They are section 1 and section 2.

Section 1

Demo #1 (walking man) enter sequence 1,2,3,4,5,6,7,8,0
Demo #2 (flying bird) enter sequence 11,12,13,14,13,12,0

Section 2

Demo #1 (waving flag) enter sequence 1,2,3,2,0
Demo #2 (spinning disc) enter sequence 5,6,7,8,9,10,11,12,13,14,0

2
Getting Started

Loading The Program

Tape System

1. Placed tape in the recorder.
2. Rewind to the beginning and set the counter to zero.
3. Type: **Load** and hit **[Return]**.
4. The Computer Displays: **Press Play On Tape**
5. Press the 'Play' button on the recorder.
6. The screen will go blank while the computer searches for the program.
7. The Computer Displays: **Found Spritemaster**
8. Press the commodore key.
9. Again the screen will go blank for approximately 30 seconds.
10. The Computer Displays: **Ready**
11. Type: **Run** and hit **[Return]**.
12. The computer will display the program name for about 5 seconds and then go blank again while the main portion of the program is loaded. *This will take approximately 8 minutes* because of the large size of the program and the machine language routines that must also be loaded. Do not move the tape after the program is loaded.
13. If the program fails to load properly, try turning the computer "off" and then "on" and repeat steps 1 thru 12 above. There is a second copy of the program on the reverse side of the tape. Try both sides. Also, having your TV set or monitor too close to the computer or cassette unit can cause problems in loading programs. Try moving both about four feet away.

Disk System

1. Remove the diskette from its protective envelope.
2. Make certain that the disk drive has been turned on.
3. Insert the diskette into the drive with the label up.
4. Close the drive door.
5. Type: **LOAD"SPRITEMASTER",8** and hit **[Return]**.
6. The computer displays: **SEARCHING FOR SPRITEMASTER**
   LOADING
   READY
7. Type: **RUN** and hit **[Return]**.
8. The program will load in approximately one minute.
9. If the program fails to load properly, try turning both the computer and the disk drive (with the disk removed) "off" and then "on" and repeat steps 1 thru 8 above. If it still fails to load, contact your dealer.

HANDLING A SYSTEM CRASH

If the program should "lock up" for some reason, try pressing the **[RUN / STOP]** or the **[RUN / STOP]** and **[RESTORE]** keys at the same time. If the program breaks or if the screen

goes blue and displays **"READY"** then simply type **RUN**. Your sprites will not be lost, but be sure to answer **"NO"** when asked if you wish to clear storage areas. If you cannot get the computer to respond you will have to turn off the computer and reload the program. In this case you will lose any sprite data that was not "saved" to a data file.

If the program should terminate because of an error condition, the display will say **"READY"**. Simply type **RUN** to restart. Your Sprite will not be lost but again, *do not* clear the sprite storage areas.

Note: Whenever it is necessary to restart the program by typing **RUN**, be sure that the program disk is inserted in the drive. The program must access this disk to operate properly.

CLEARING OR NOT CLEARING STORAGE AREAS:

When you first enter the main program you are asked **"DO YOU WISH TO CLEAR STORAGE AREAS?"** Your response to this question depends on what you want to do.

1. If you intend to load a set of existing sprites from tape or disk there is no need to clear the areas because a "load" will write over what is presently there. You may clear them if you desire. It will take approximately 15 seconds.
2. If you intend to start building some sprites from scratch you should clear the areas. When the program first starts all areas will be full of "garbage" even if the storage map shows some areas as being empty. **Trying to use these areas without clearing them or loading over them can produce very strange program behavior or a system crash.**
3. As stated earlier, if the system has crashed for some reason and you have been able to restart the program, the storage areas are protected and will still contain any sprite data that was generated prior to the crash. In this case *do not* clear the areas.
4. If this is your first time and you are following the instructions as you go, you will want to answer **YES** and hit **[Return]**.

After the question on clearing the storage areas, the storage area map is displayed. You will notice that there are 16 areas in which different sprites may be stored in the computer. Areas which contain data would be marked with the word **"FULL'**. Empty areas are simply blank.

Any number of sprites may be stored separately on disk or tape, but only 16 are available for modification or animation at any one time. You will see this map again prior to a request to select an area for Building, Modifying, Copying, Deleting, etc. so there is no need to memorize or write down which areas are full or empty.

Hit **[Return]** to continue on to the main program menu.

THE COMMAND MENU

The command menu should now be displayed on the screen. The available commands are: **BUILD, MODIFY, VIEW, COPY, DELETE, PURGE, LOAD, SAVE, TRANSFER, ANIMATE**.

Each command is described in detail later in the manual, but let's proceed first to a demonstration.

LOADING DEMONSTRATION SPRITES

If this is your first time with **Spritemaster**™ it will be useful to load some sample sprites to demonstrate what they look like and how they are animated.

Step 1. Select the **LOAD** command by typing **"L"** and hitting **[Return]**.

Step 2. **Tape System**

The computer will say **"READY TO LOAD SPRITE DATA FROM TAPE".**

If you have not moved the tape since loading the program you are positioned right at the beginning of a sprite data file which contains some sample sprites. You may want to note the reading on the counter so that you can come directly to this file at some future time for loading this demonstration data.

If you have moved the tape, try setting it at location 184 from the beginning. (This may or may not work).

You can hit **[Return]** to continue or **[/]** if you wish to return to the command menu.

To continue the demonstration, hit **[Return]**. The computer then asks if you wish to protect any areas. This feature is explained later and since it is not needed here, simply enter **"N"** and hit **[Return]**. If the **"PLAY"** button was not down, you will be asked to press it. After doing so the screen will go blank for a few seconds and then flash on and off as the areas are loaded. Loading will take approximately 3 minutes and each area number will be displayed as it is loaded.

Disk System:

The computer will say:

"READY TO LOAD SPRITE DATA FROM DISK"
"ENTER SECT # (1-10) OR [/] TO EXIT"

The computer wants to know which section number is to be loaded. There are ten sections on disk, each of which will hold the entire 16 area working memory. Therefore the total disk storage capacity is 160 sprites.

For this demonstration, enter a **"1"** and hit **[Return]**

The computer then asks if you wish to protect any areas. This feature is explained later in this manual and since it is not needed here simply enter **"N"** and hit **[Return]**.

As each area is loaded, the computer will display both the section number and the area number.

Step 3. After loading, the storage map is again displayed. Notice now that some areas are marked as full. Hit **[Return]** to continue.

A DEMONSTRATION OF SPRITE ANIMATION

Now that the demonstration sprites have been loaded, there are several ways of looking at them. First let's get a close up look at areas 1 through 5 by using the **MODIFY** command.

The command menu should be displayed on the screen.

Step 1. Select the **MODIFY** command by typing **"M"** and hitting **[Return]**.

The computer will display the storage map and ask you to select the area to be modified.

Step 2. Select area **1** and hit **[Return]**.

The computer will display the full working copy of the selected area and, at the lower left, the actual sprite in expanded size.

Step 3. Since we really do not want to make any modifications, just hit the **[/]** key to exit.

Step 4. Repeat steps 1 thru 3 for areas 2, 3, 4, and 5.

You will notice that each picture is slightly different from the next. The five different "frames" or pictures represent five different views of a man as he is walking. To see the same sequence a little faster, let's use the **ANIMATE** command.

Again the command menu should be displayed on the screen.

Step 1. Select the **ANIMATE** COMMAND BY TYPING **"A"** and hitting **[Return]**.
The computer will then display another menu of four choices.

(1) . . . . ENTER SEQUENCE
(2) . . . . MANUAL DISPLAY
(3) . . . . AUTO DISPLAY
(/) . . . .EXIT

Step 2. Select choice **1** and hit **[Return]**.

The computer now displays a list of instructions which you should read carefully.

The cursor should be blinking in the upper right hand corner.

You need to enter the following numbers which represent the order in which you wish to view the demonstration areas. After typing each number, hit **[Return]**.

1
2
3
4
5
6
7
8

After the 8 the rest of the list should be zeros.

Hit the **[/]** key when finished.

You should now have returned to the small menu.

Step 3. Select choice **2** for manual display and hit **[Return]**.

The computer again displays some instructions which should be read carefully.

Notice that at the bottom of the screen is displayed area 1, which is the first picture or frame of the man.

Now press the cursor down key (down arrow). The picture will change to display area 2. Pressing it again will display area 3 and so on. The areas displayed will follow the sequence which you just entered and the sequence will repeat when the computer comes to the zero at the bottom of the list. hence, repeatedly pressing the cursor down key will show areas 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, ... and so on over and over again.

This is a great way to view each area as it is displayed directly over the previous area.

Hit the **[/]** key to exit.

Again you should have returned to the small menu.

Now for full speed animation.

Step 4: Select choice 3 for auto display and hit **[Return]**.

The computer now requests the initial time delay between frames (1-1000). Entering (1) will produce very fast motion while entering (1000) will produce very slow motion.

Enter **400** and hit **[Return]**.

The computer displays some more instructions and at the bottom of the screen you see the same sequencing that you just did manually only under automatic control. The 400 that you entered is the number of times that the computer counts before it changes the picture to display the next area.

If the man appears to be moving too fast, you can slow him down by holding down the [S] key (slower). You can speed him up by holding down the [F] key (faster).

Use [S] or [F] to get him moving at a good speed. Now hold down the cursor left (left arrow) key for a few seconds. The man should gradually start walking faster and faster to the left. If he gets going too fast, you can slow him down by holding down the cursor right (right arrow) key. If you hold the cursor right key down long enough he will stop walking to the left and start walking backwards to the right.

The up and down cursors have the same effect except that the travel is vertical. Hold the key down until the motion or travel reaches the speed you like and then release it. Experiment a little with the controls. It doesn't take long to get the hang of using them to produce some smooth animation. By the way, don't worry if he walks off the screen, he will re-appear on the other side. Hitting the **[Return]** key will stop all horizontal or vertical travel until you use the cursor controls again.

Hit **[/]** to exit to the small menu. Then type **[/]** and hit **[Return]** to go back to the main menu.

There is another demonstration display in areas 11, 12, 13, and 14. You may want to see what it looks like. You should enter the sequence 11, 12, 13, 14, 13, 12, 0 and proceed as before.

3

Making Your First Animated Sprite

In this chapter we will go thru all the steps necessary to develop an animated sprite. Our example will be quite simple but the procedure is the same for virtually any moving object you can imagine. One thing to keep in mind is that sprites are only so big. If you want to make a large moving object you will have to build several small sprites and piece them together in another program.

The only limit to the things that can be animated is your own imagination. In fact, part of the fun of **SPRITEMASTER**™ is thinking of new objects to create.

Now let's make your first sprite. If you intend to use the joystick make sure it is plugged into port #2.

BUILDING A SPRITE

For speed and convenience let's choose an object that is easy to build and to modify.

Let me introduce you to the HACMAN.

He is similar in many ways to the famous Arcade personality but is a little easier to create since he is square instead of round.

The main command menu should be displayed on the screen. If not, enter [/] as many times as needed to work your way back to the main menu.

In the remaining portion of the manual we will not bother to tell you to hit the **[Return]** key after an entry. Sometimes a **[Return]** is required and sometimes a single keystroke, such as **[/]** is all that is needed. It really depends on whether or not the computer thinks it is necessary to let you change your mind. The best approach is to make your selection or entry, and then if nothing happens immediately, hit **[Return]**. There is no problem if you do hit **[Return]** when it is not required.

First we will erase all the demonstration sprites.

Step 1: Select the PURGE command from the menu by entering **"P"**.

The computer asks if you wish to clear storage areas.

Answer **"Y"** or **"Yes"**.

The computer then clears all the areas.

Step 2: Select the BUILD command by entering a **"B"**.

The computer displays the storage map (which should now be empty) and asks you to select an area.

Step 3: Select an area number for your sprite.

You may enter any area number that is not already occupied. In this case area 1 would be a good place to start, so enter a **[1]**.

Step 4: Select single or multicolor mode

The computer will display some important information and asks you to select single [S] or multicolor [M]. Let's enter **[S]** for single color.

Step 5: Select a color or colors.

Now you must select a color. Although you can choose any of the 16 colors available, I suggest yellow (#7).

You make your selection by pressing the **[Return]** key several times until the blinking cursor moves to the color you desire. Then press the function key **[f1]** on the far right of the keyboard.

Because we selected single color, there is only one selection required. Had we chosen multicolor, there would have been three selections needed as [f1], [f3], and [f5].

When the correct color is displayed just to the right of "F1" on the screen, then touch the **[/]** key.

Step 6: Create your sprite.

The computer now displays your working grid. Notice the flashing cursor in the upper left hand corner of the grid.

At the upper left corner of the screen you will notice a yellow bar next to the letters "F1". This means that pressing the [f1] key will select the color yellow to draw with.

Had we been in multicolor mode, the letters F3 and F5 would also have colors adjacent to them. Select the color yellow by pressing **[f1]**. Notice that **"NOW USING:"** indicates the present active color.

Joystick Control

Move the joystick up, down, and sideways to see how the flashing cursor moves around on the grid. To "paint" a dot you must hold the button down and at the same time move off of the dot. Draw a few lines by holding down the button as you move.

To erase your mistakes or to modify an existing picture you need to press the [f7] key. "NOW USING:" will show the grid symbol. Simply paint over the colored dots with f7 selected to erase.

Keyboard Control

Keyboard control is accomplished in the same manner as the joystick. The shift keys serve the same purpose as the button, and the S, Z, X, keys and the space bar serve as up, left, right, and down respectively.

You may select [f2] (shift / f1) to change colors and you can exit this routine at any time using the [/] key.

Once you have exited the build routine you cannot re-enter using the BUILD command since only empty areas can be accessed through "BUILD". To finish a sprite or to modify one, you must use the MODIFY command.

Erase any garbage you have created and do your best to draw our friend HACMAN, who should look something like this:

After you have finished, press **[/]** to return to the main menu.

COPYING A SPRITE

Step 7: Make a copy of your sprite.

Now that we have one picture of HACMAN, we want to make a copy of him for modification.

Select the COPY command by entering a **[C]**.

The computer displays the storage map and asks you to select the area number to be copied from. If you created your first picture in area 1, then enter a **[1]**.

The computer then requests the area number that you wish to copy to. It is a good idea to keep sequential pictures in order, so enter a **[2]** for area 2.

A picture of area 1 is displayed and the computer asks if you want to continue or abort. Enter **[C]** to continue.

The computer will make the copy and then return to the main menu.

MODIFYING A SPRITE

Step 8: Modify the copy

We want to modify the copy you have made to represent the second picture in the motion sequence.

Select the modify command by entering **[M]**.

Again the storage map is displayed. Notice that area 2 is now full. Enter a **[2]** to modify area 2.

Now the computer takes you to the same routine used by the BUILD Command.

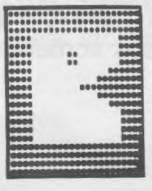
We need to modify HACMAN by closing his mouth slightly. Make him look something like this:

Press **[/]** when finished to return to the main menu.

MAKING MULTIPLE COPIES

Step 9: Make another copy

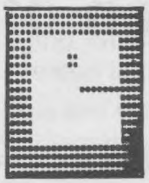
Proceed as before but this time copy area #2 into area #3. Then modify area #3 to close his mouth further to make him look something like this:



Hit **[/]** to exit.

Step 10: Make still another copy

Make the final copy from area #3 to area #4 and then modify area #4 to close his mouth completely, like this:



ANIMATING YOUR SPRITE

Step 11: ANIMATE your sprite

Select the **ANIMATE** command. From the small menu, select **[1]** for "entering sequence".

Remember, areas 1 thru 4 show HACMAN gradually closing his mouth. In the motion sequence however, his mouth would close and then open again in reverse order. Enter the following sequence. Don't forget to hit return after each number.
1,2,3,4,3,2,0

Recall that the computer will repeat the sequence when it comes to the zero at the end of the line.

Now exit by touching **[/]**.

Select **[3]** from the small menu for auto display.

When the computer asks for an initial time delay, enter 400.

Play with your sprite as you did in the demonstration to observe the effects of different animation and travel speeds.

When finished, hit **[/]** twice to get back to the main menu.

4
Description of Commands

BUILD

The build command is used to generate a sprite in an empty area and you cannot build an area that is occupied without first deleting that area. Refer to Chapter 3, page 8 for a description of how to use the build command.

MODIFY

The modify command is used to change shapes or colors of existing or unfinished sprites. It can also be used as a close up display of a single sprite. MODIFY is identical to BUILD except that the color and mode selections routines are not called.

VIEW

The view routine is very useful for examining sprites using various colors, positions, and sizes to see how they will appear under various conditions.

The contents of any three sprite areas can be displayed on the screen at any one time. If you have just started the program, areas 1, 2, and 3 will be displayed. If you have entered the view routine before, the last 3 areas being viewed will be returned to the display.

The sprites are displayed with the area number which represents the sprite shown to its left. A white dot between the area number and the sprite indicates that sprite is "active" and can be altered by changing the parameters located on the right side of the screen. Parameter changes only affect the view routine and are not permanent.

[Space Bar]

Pressing the [space bar] will move the dot to the next sprite in the group of three. That sprite now becomes "active" and the parameters to the right will change to represent that particular sprite.

[Up Cursor] or **[Down Cursor]**

Pressing [up cursor] or] down cursor] will move the blinking cursor up or down the list of parameters.

[Return]

Once the cursor has been positioned over the desired parameter, its value can be changed in one of two ways.

1. By entering the new value followed by [RETURN]
2. By entering only a [RETURN] which will cause the value to increment by one. By repeatedly pressing [RETURN] you can sequence thru all possible parameter values and come back to the original value.

[/]

Press [/] to exit this routine.

The following is a description of each parameter:

Primary Sprite Color (originally selected as F1)

This represents the primary color for multicolored sprites or the only color for single colored sprites (The F1 selection during Build or Modify). A multicolored sprite has 3 color

possibilities (actually 4 if you include the background color) but only one color is unique to that sprite. The other two colors must be the same for all sprites on the screen at that time. You will notice that changing the primary color only affects the "active" sprite.

Multicolor #1 or **#2** (originally selected as F3 and F5)

These represent the secondary colors. Changing either of these values will change all portions of the displayed sprites which were created under the F3 and F5 color selections.

It is important when creating multicolored sprites and when selecting colors to be sure that the F1 selection is the color which you wish to remain unique to that particular sprite.

Background Color and Border Color

Changing these parameters will change the background color or border colors. You will notice that some colors do not interact well together and the real purpose in changing these colors is to determine which color combinations are compatible.

Vertical Size and Horizontal Size

The view routine initially sets sprites to their expanded size. Changing these parameters will change either the vertical or horizontal size by a factor of 2.

Sprite Area Select

Changing this number selects a new sprite area to replace the sprite that is "active". The parameter list is updated to represent the new sprite.

Sprite Move

This parameter has no value. When the cursor is placed at this point and [Return] is pressed the cursor controls are activated. The three sprites on display can then be moved around on the screen to observe how they appear in different positions relative to each other.

Pressing [F1], [F3], or [F5] activates the top middle and bottom display sprites respectively.

Pressing [/] exits back to the main view routine.

Sequential Display

This is the same routine that is accessed by the ANIMATE command. Refer to chapter 2, page 6 and Chapter 3, page 11 for instructions on how to use the sequential display or "Animate" feature.

COPY

Copy is used to make copies of sprites. You cannot copy from an empty area, and you cannot copy to a full area. Use **DELETE** to remove an unwanted area prior to copying into it. Refer to Chapter 3, page 10 for copying instructions.

DELETE

The delete routine will erase a single sprite from the computer memory (not from disk or tape).

When **DELETE** is selected, the computer will display the storage map and ask you to select the area number to be deleted.

After your selection, the area will be displayed (whether empty or not) and you should enter a **[C]** to delete or **[/]** to abort the process. The delete routine is a quick way to get a storage map or a peek at a sprite area and then cancel by hitting a **[/]**.

PURGE

Purge will erase all sprites from computer memory (not from tape or disk). It is the same routine that is called at the beginning of the program.

LOAD

The load command will load sprites from disk or tape. In a disk system you have ten (10) sections on any one disk each of which stores 16 areas. In a tape system you have one section (16 areas) after the program with demonstration sprites in it. Sprite data can be stored anywhere on any tape. It is a good idea to record on the label or some other place where the data file starts (relative to the beginning of the tape) so that it can be located again for loads or saves.

On either a tape or disk system the computer asks if you wish to protect any areas. If you answer yes to this you will be given the opportunity as each area is about to be loaded, to protect the area currently in memory from being overwritten. This feature allows you to merge sprite data from two or more files. You are also permitted to exit the load routine after one or more areas have been loaded, if you have chosen the protect option.

Refer to Chapter 2, page 4 for instructions on loading sprite data.

SAVE

Save to Tape

(a) Position the tape at the place on the tape where you wish to save the sprite data which is currently in the computer. You will want to use a different tape since the program tape is "write protected".

(b) Hit **[Return]** or **[/]**

(c) The screen will go blank and will flash on and off and indicate that the areas are being saved.

Save to Disk

The disk has capacity for storage of 10 sections, each of which holds 16 sprites.

REMEMBER! You must use another disk to store your sprite data.

(a) The computer displays the last loaded section. The computer will display a (*) if no load has taken place. You are then asked to select one of the following.

(1) . . . . **SAVE CURRENT MEMORY AS SECTION _____**
(2) . . . . **SAVE AS ANOTHER SECTION**
(3) . . . . **EXIT**

(b) Enter a (1) if you wish to simply save the section you just loaded in its old location, together with any modifications. If you have not loaded an existing section the computer will not accept this response.

(c) Enter (2) if you wish to save computer memory into another location. This can be used to copy sections if you desire.

(d) Enter **[/]** to abort.

TRANSFER

This feature is used to transfer sprite data to another program in the form of Data Statements for use in games, etc. The program will first transfer four bytes representing the **mode** (0 for single color or 1 for multicolor), the **primary color, multicolor #1** and

multicolor #2. The mode has 128 added to it. Following these four bytes are 64 bytes representing the 63 bytes required for sprite description, plus 1 unused byte.

Data statement numbers are incremented by 10 and there are 5 elements per line. Transfer of one single colored sprite in the shape of a square would look like this:

```
1000 DATA 128, 7, 0, 0, 255
1010 DATA 255, 255, 128, 0, 1
1020 DATA 128, 0, 1, 128, 0
  .              .
  .              .
1120 DATA 1, 128, 0, 1, 255
1130 DATA 255, 255, 0
```

Once the data has been transfered to another program, it can be used to duplicate the motion you have achieved in this program. Refer to the programmer's section at the end of this manual for information on how to manipulate sprites in your own programs.

Note: Transfered data can only be added to the bottom of an existing program. You must be sure that the beginning line number which you specify is greater than any others in the destination program. If you do not know what the last line number is or have forgotten, you can give the computer a large number such as 50000 (it must be less than 60000) and then move them later via the screen editor in the normal manner.

(b) After selecting the transfer routine, the computer will ask you to enter the area numbers of the sprites to be transfered. Enter the area numbers and be careful as you make each entry because you have to start over if you make a mistake. When you have finished, enter a [/].

(c) The computer will then display your selections and ask if the above is correct. If it is, enter **[Y] [Return]**.

(d) The computer will think for a few seconds and then ask for the starting line number of the data statements in the destination program. Enter this number, then hit **[Return]**.

(e) The data statements will be displayed on the screen at the same time they are being transfered into a temporary storage area. As each page is displayed the display will stop and you must hit **[Return]** to continue.

(f) When all of the sprites you have designated have been assembled into temporary storage, you will be asked if you wish to clear the memory and lower basic back down to its normal location. The basic workspace (the programming area) has been raised to provide storage space for sprites. If you answer [Y] to this question the computer will say [are you sure]. If you say [Yes] again, the existing program will be erased, and the workspace will be set back to its normal starting point. You can then either start a new program or load an existing program into the workspace as you normally would. If you now want to retrieve the assembled sprite data and have it added to the bottom of your program, you must type the following exactly. Be careful not to make a mistake.

Type **SYS 32768** and hit **[Return]**.

The computer will respond with **[READY]**.
Then type **CLR** and hit **[Return]**.

You may now list the program to see that the Data Statements have been added to the end of the program.

Note: There must be at least one line of program in the workspace prior to typing the 'SYS' command to intiate the transfer. A single REM statement is sufficient.

ANIMATE

The animate command is used to give motion to your sprite sequences. Refer to Chapter 2, page 6 and Chapter 3, page 11 for a description of how to use this command.

5
Programmer's Reference Section

This section is intended to supplement Chapter 6 of the Commodore 64 User's Guide and to provide additional information and help in the use of sprites in your own programs.

If you are serious about learning how to program using sprites, we suggest that you read and study Chapter 6 of the User's Guide in addition to what we will teach you here.

The best way to become comfortable with sprite graphics is to read both manuals and then experiment to try the things you have learned. Then re-read everything again and experiment some more. We suggest that you read both Chapter 6 of the User's Guide and this chapter of the **Spritemaster**™ instruction manual at least three or four times. Each time you will better understand and retain more of what you read.

It is essential that you learn how to use binary numbers and how to convert from binary to decimal and vise versa if you want to be good at using Sprites.

THE VIDEO DISPLAY CHIP

The part of the Commodore 64* which handles the color video display is called the "Video Display Chip". Within the chip are 47 different registers which control the video display. The registers are numbered 0 through 46 and are arranged in consecutive locations in memory. Each register can be thought of as just a special memory location, and poking numbers into those locations make the display do special things. The first register (R0) is located at address 53248. You should memorize this number and then reference all other registers to it. It will be much easier this way to remember key register addresses. For example, if you define V = 53248 at the beginning of your program, then all registers can be addressed by adding their register number to the variable (V).

V + 0	=	address or register 0	(R0)
V + 1	=	address of register 1	(R1)
.		.	
.		.	
.		.	
.		.	
V + 46	=	address of register 46	(R46)

The following register map shows each register, its memory location and its function or functions.

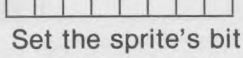
* Commodore 64 is a registered trademark of Commodore Business Machines.

Video Display Chip
Register Map

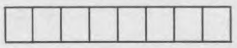
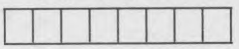
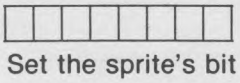
Register Number	Address in offset Notation (V = 53248)	Function
0	V + 0 (53248)	X coordinate of sprite 0
1	V + 1 (53249)	Y coordinate of sprite 0
2	V + 2 (53250)	X coordinate of sprite 1
3	V + 3	Y coordinate of sprite 1
4	+ 4	X coordinate of sprite 2
5	+ 5	Y coordinate of sprite 2
6	+ 6	X coordinate of sprite 3
7	+ 7	Y coordinate of sprite 3
8	+ 8	X coordinate of sprite 4
9	+ 9	Y coordinate of sprite 4
10	+ 10	X coordinate of sprite 5
11	+ 11	Y coordinate of sprite 5
12	+ 12	X coordinate of sprite 6
13	+ 13	Y coordinate of sprite 6
14	+ 14	X coordinate of sprite 7
15	+ 15	Y coordinate of sprite 7
16	+ 16	MSB for X coordinate

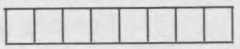
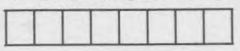
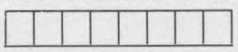
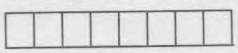
Bit locations

7 6 5 4 3 2 1 0 — Sprite #



Set the sprite's bit to "1" to move it into the screen locations from 256-511

Register Number	Address in offset Notation (V = 53248)	Function
17	V + 17	See Commodore 64 Programmer's Reference Guide.
18	+ 18	Raster Register
19	+ 19	Light Pen X
20	+ 20	Light Pen Y
21	V + 21	Sprite Enable Bit locations 7 6 5 4 3 2 1 0 — Sprite #  Set the sprite's bit to "1" to turn it on (0 = off)
22	V + 22	See Commodore 64 Programmer's Reference Guide.
23	V + 23	Sprite Expand Y Bit locations 7 6 5 4 3 2 1 0 — Sprite #  Set the sprite's bit to "1" to expand vertically
24	V + 24	Memory Pointers
25	V + 25	Interrupt Register
26	+ 26	Enable Interrupt
27	V + 27	Background / Sprite Priority Bit locations 7 6 5 4 3 2 1 0— Sprite #  Set the sprite's bit to "1" to make it pass behind the background

Register Number	Address in offset Notation (V = 53248)	Function
28	V + 28	Multicolor Sprite Select Bit locations 7 6 5 4 3 2 1 0 — Sprite #  Set the sprite's bit to "1" to enable multicolor mode
29	V + 29	Sprite Expand X Bit locations 7 6 5 4 3 2 1 0 — Sprite #  Set the sprite's bit to "1" to expand horizontally
30	V + 30	Sprite — Sprite Collision Bit locations 7 6 5 4 3 2 1 0 — Sprite #  Sprite bits will be set to a "1" when a collision occurs between them.
31	V + 31	Sprite — Background Collision Bit locations 7 6 5 4 3 2 1 0 — Sprite #  Sprite's bit will be set to a "1" when a collision occurs between it and the background.
32	V + 32 (53280)	Border color (0–15)
33	V + 33 (53281)	Background 0 color (0–15)
34	+ 34	Background 1 color (0–15)
35	+ 35	Background 2 color (0–15)
36	+ 36	Background 3 color (0–15)
37	V + 37	Sprite Multicolor #1 (0–15)
38	V + 38	Sprite Multicolor #2 (0–15)

Register Number	Address in offset Notation (V = 53248)	Function
39	V + 39	Sprite 0 Color (0–15)
40	+ 40	Sprite 1 Color (0–15)
41	+ 41	Sprite 2 Color (0–15)
42	+ 42	Sprite 3 Color (0–15)
43	+ 43	Sprite 4 Color (0–15)
44	+ 44	Sprite 5 Color (0–15)
45	+ 45	Sprite 6 Color (0–15)
46	+ 46	Sprite 7 Color (0–15)

In addition to the registers just described, there are eight other memory locations that are reserved for use with sprites. These are locations 2040 through 2047. Numbers must be placed in these locations to tell the video chip where to get its sprite data.

The video chip is smart, but it's not smart enough to know how you want your sprite to look unless you tell it.

First you have to design the sprite and then code the data. **Spritemaster**™ will do this for you. Next you have to place the data into memory and tell the video chip where you put it.

Since it takes 63 bytes to describe a sprite, let's add 1 to make it 64 (a nice round number) and call that a "sprite area." Then let's divide up the computer memory into sprite areas (instead of bytes). For example area 0 would be the first 64 bytes of memory, area 1 would be the next 64 bytes and so on.

A sprite area map would look like this:

Sprite Area Map

Sprite area number	Inclusive Memory Addresses	
0	0-63	Cannot use
1	64-127	.
2	128-191	.
.	.	.
.	.	.
.	.	.
.	.	.
10	640-703	Cannot use
11	704-767	Available
12	768-831	Cannot use
13	832-895	Available
14	896-959	Available
15	960-1023	Available
16	1024-1087	Cannot use
.	.	.
.	.	.
.	.	.
.	.	.
31	1984-2047	Cannot use
32	2048-2111	Basic work space
33	2112-2175	
.	.	.
.	.	.
.	.	.
.	.	.
255	16320-16383	
		Areas above this point can be made available.

Now here's where locations 2040 thru 2047 come in. Let's call these eight locations "area pointers" because they will point to the areas where sprite data is stored. Each of the eight sprites has its own "area pointer". Location 2040 is for sprite 0, 2041 is for sprite 1, and so on up

to 2047 which is the "area pointer" for sprite 7. First you place your sprite data in one of the **available** areas (63 bytes + 1 unused byte). Next you decide which of the eight sprites (0-7) will use that data, and then you place the area number into the selected sprite's "area pointer". For example, suppose we have stored our sprite data beginning at memory location 2048 which is area 32, and we want sprite 3 to access that data. We tell the video chip to look at area 32, for sprite 3 data, by doing a **[POKE 2043,32]**.

If you look carefully at the sprite area map you will notice that there is a problem with our example. Area 32 is located at the start of the basic workspace. If we placed our data there, any basic program in the workspace would be ruined. We can, however, raise the start of the workspace in memory to create room below it for our sprites. This is done as follows:

The workspace starts normally at page eight in memory (a page being 256 bytes) or location 2048 (8 x 256). Although the workspace can be raised by any amount (even one byte) it is easier to do by pages. Let's leave room for 10 sprite areas (areas 32 thru 41) below the workspace. The number of bytes needed is 10 x 64 or 640 bytes. We could allocate just 640, but let's go ahead and save three pages which is 3 x 256 or 768 bytes. First we must determine the new page at which the workspace will start. We do this by adding the number of pages we wish to save to the normal starting page of 8. In our example this is 8 + 3 or 11. Once this is done you must do the following. Be careful not to make any mistakes.

POKE 44,11: POKE 642,11: POKE 11*256,0: NEW

To come back down to the normal location, type:

POKE 44,8: POKE 642,8: POKE 8*256,0: NEW

Be sure there is no program in the workspace at the time or it will be lost.

Once you have raised the workspace, you can create, load, and save programs as usual and not disturb the protected sprite areas below.

Note: All of the above locations for sprite areas and area pointers can be changed. The video chip can be made to access sprite data throughout the entire 64K memory and can perform other complex display functions such as multicolor characters and full screen bit mapping for high resolution graphics. For information on how to implement these and many other changes, refer to the Commodore 64 Programmer's Reference Guide.

AN EXAMPLE OF HOW TO USE SPRITES IN YOUR OWN PROGRAM

Step 1: With **Spritemaster**™, create the sprite figures that you want to use in your program and use the TRANSFER command to assemble and transfer your sprite data. Specify a line number of 10000 for your data statements when requested by the transfer routine. When the computer asks if you want to reset the basic workspace, you should answer yes. (Caution: If you want to keep the sprites you generated for future modification by **Spritemaster,**™ be sure to save them to tape or disk before you reset the workspace.)

After you have reset the workspace (to page 8) and the computer responds with "Ready", you should enter at least one line of a program. Type in the following:

```
10 REM        THIS IS A SPRITE TEST
```

The reason for this is that the machine language transfer routine which you are about to execute will only append to an existing program. If the workspace is empty it will get lost.

Now type **SYS 32768** or **SYS 8*4096**

Then list the program to see your new data.

Step 2: Save the program to tape or disk.

Step 3: Raise the workspace as much as you need. If you have less than 10 sprites, you can use the following pokes to reserve three pages.

POKE 44,11: POKE 642,11: POKE 11#256,0: NEW

Step 4: Reload your program.

Step 5: Add a routine to store your sprite data.

For an example, let's assume that we have just transfered three sprites and we are going to store them in areas 32, 33, and 34.

The following routine will work nicely, but it does assume that there are no other data statements in your program other than sprite data, or that the other data statements have been previously read. It also will store in consecutive areas only.

```
1000  REM   SUBROUTINE TO STORE SPRITE DATA
1010  N = 3: REM NUMBER OF SPRITES
1020  A = 32: REM STARTING AREA NUMBER
1030  FOR I = 1 to 3: AD  =  A*64  +  (I-1) *64
1040  READ M: M( I )  =  M AND 1: REM SINGLE ( 0 ) or M / C ( 1 ) MODE
1050  READ PC( I ): REM PRIMARY COLOR,
1060  READ M1( I ): REM MULTICOLOR #1
1070  READ M2( I ): REM MULTICOLOR #2
1080  FOR K  =  0 to 63
1090  READ M: POKE AD  +  K, M
1100  NEXT K
1110  NEXT I
1120  RETURN
```

Note: If N is greater than 10 you will have to dimension M(N), PC(N), M1(N), M2(N) at the beginning of your program.

Notice that the first four data elements are the mode, primary color, multicolor #1, and multicolor #2 that were assigned to the sprite back when it was created with **Spritemaster**™.

Step 6: ADD A ROUTINE TO PUT YOUR SPRITES ON THE SCREEN.
DO NOT TYPE THE WORDS IN BOLD PRINT.

```
2000  REM ROUTINE TO DISPLAY SPRITES.
2010  V = 53248
```

ADDRESS OF VIDEO CHIP

```
2030  POKE 2040,A: POKE 2041,A + 1:POKE 2042, A + 2
```

Area Pointer — **Sprite Area Number**

```
2040  POKE V  +  0, 50          sprite 0 ( X ) position
2045  POKE V  +  1, 100         sprite 0 ( Y ) position
2050  POKE V  +  2, 100         sprite 1 ( X )
2060  POKE V  +  3, 100         sprite 1 ( Y )

2070  POKE V  +  4, 150         sprite 2 ( X )
2080  POKE V  +  5, 100         sprite 2 ( Y )
2090  POKE V  +  39. PC( 1 )    sprite 0 primary color
2100  POKE V  +  40, PC( 2 )    sprite 1 primary color
2110  POKE V  +  41, PC( 3 )    sprite 2 primary color
```

```
2120 POKE V + 28, M(1)*2↑0 + M(2)*2↑1 + M(3)*2↑2
```


```
2130  POKE V + 37, M1(1)      multicolor #1
2140  POKE V + 38, M2(1)      multicolor #2
2150  POKE V + 23, 0          normal size Y
2160  POKE V + 29, 0          normal size X
2170  POKE V + 21, 7          turn on sprites 0, 1, 2
2180  RETURN
```

The preceding routine will position three sprites at coordinates

```
        X    Y
Sprite 0, (50,  100)
Sprite 1, (100, 100)
Sprite 2, (150, 100)
```

The positions can be changed by changing the values that are poked in lines 2040 thru 2080. If the sprites are multicolor then the multicolor registers are set to display the colors of sprite 0. This can be changed by using M1(2) or M1(3) instead of M1(1) in line 2130, and M2(2) or M2(3) instead of M2(1) in line 2140.

Remember that only one color for M/C #1 and one color for M/C #2 can be displayed at one time, since there is only one video chip register for each.

To see the sprites in still form, type the following:

```
100  PRINT CHR$(147): REM CLEAR SCREEN
110  GOSUB 1000
120  GOSUB 2000
130  GET A$: IFA$ = "" then 130: REM WAIT FOR ANY KEY
140  END.
```

Then type **RUN**.

The left sprite should be displayed as it appeared in **Spritemaster**™. The others will be the same if they had the same multicolor 1 and 2 as did the first sprite.

ANIMATION TECHNIQUES

The secret to animation is to take the still frames that were generated and refined with **Spritemaster**™ and display them one after another at the same point on the screen.

There are two ways to accomplish this. One is to use a single sprite such as sprite 0, and change the area, mode, colors and sizes associated with it. This technique must be used when more than 8 frames are involved.

The second method assigns each area a sprite such as sprites 0, 1, & 2 for areas 32, 33, & 34. Then all sprites are positioned in the same location and enabled with register 21 in the proper sequence. If different multicolors are used, then as each frame is displayed, the appropriate multicolors and mode must be POKED into registers 37, 38, and 28.

When you are trying to sequence sprites and at the same time move them, you will quickly find that Basic is too slow for all but the simplest games. The animation routine in **Spritemaster**™ is written in machine language, and any sophisticated fast action program using animated figures will require machine language routines for smooth motion sequences.

CODING FOR MULTICOLOR SPRITES

The procedure for generating code for multicolored sprites is quite different than for single colored sprites.

Step 1: Enable multicolor mode in register 28 by placing a "1" in the bit position that corresponds to the sprite you are working with.

Step 2: Poke registers 37 and 38 with the color codes (0-15) you wish for M / C #1 and M / C #2 respectively.

Step 3: Poke the appropriate primary color register (39 thru 46) with the primary color code you desire.

Step 4: You must now break down the 24 x 21 sprite grid into pairs horizontally. Each pair of locations (Bits) will be used to represent one of four possible conditions:

1. (00) OFF
2. (01) MULTICOLOR #1
3. (10) PRIMARY COLOR
4. (11) MULTICOLOR #2

Since two bits and their corresponding picture elements (pixels) are required to define the color, the horizontal resolution is cut in half.

Step 5: Proceed to set up the remaining registers as you would for a single colored sprite.

RETRIEVING SPRITE DATA FROM TAPE OR DISK

The following routine will permit you to load sprite data directly from tape or disk, and will place the data in the array A(J,I).

J = 1 thru 16	**Area No.**
I = 0 thru 67	**sprite data bytes**

Tape

```
OPEN 1,1,0, "SPDATA"
```

Disk

```
OPEN 1,8,3,"SPRITEDATA" + STR$(S) + ",S,R"
```

NOTE: The variable "S" in the above open statement represents the section number (1 thru 10).

```
10  DIM A(16,67)
20  FOR J = 1 TO 16
30  FOR I = 0 TO 67
40  INPUT#1,A(J,I)
50  NEXT I
60  NEXT J
70  CLOSE 1
```

NOTES

SPRITEMASTER 64

SPRITEMASTER™ LETS YOU BUILD MOVEABLE OBJECTS (SPRITES) ON A FULL SCREEN GRID USING EITHER JOYSTICK OR KEYBOARD CONTROL. UP TO 16 SPRITES ARE STORED IN MEMORY AND THEN SAVED TO TAPE OR DISK. (DISK STORAGE CAPACITY IS 160 SPRITES)

CREATE OBJECTS IN 16 COLORS

PEOPLE
ANIMALS
CARS
CREATURES
SPACESHIPS
MACHINES
MONSTERS
TOYS
ANYTHING

CREATE ACTION SUCH AS

WALKING
JUMPING
RUNNING
CLIMBING
THROWING
FLYING
ROLLING
SPINNING
EXPLODING

UTILITY COMMAND MENU

BUILD
MODIFY
VIEW
COPY
DELETE
PURGE
LOAD
SAVE
TRANSFER
ANIMATE

- **EXCELLENT CREATIVE PLAYMATE FOR CHILDREN**
- **INDISPENSABLE TOOL FOR PROGRAMMERS — ELIMINATES TEDIOUS CODING**
- **AUTOMATICALLY TRANSFERS SPRITE DATA TO OTHER PROGRAMS**

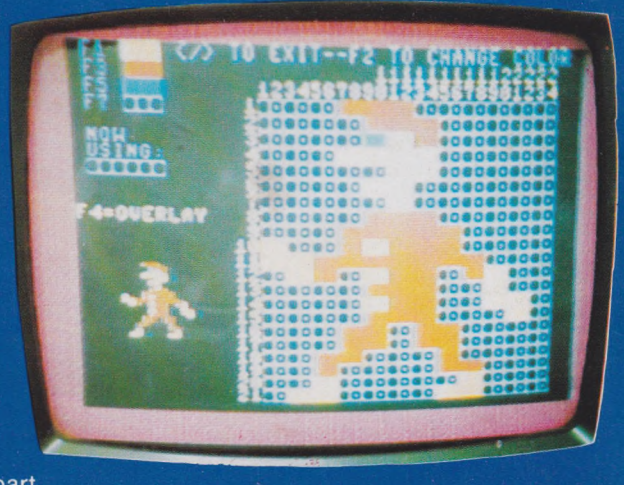
THIS PACKAGE CONTAINS

SPRITEMASTER™ PROGRAM (Tape or Disk) including samples of animated sprites

FULL INSTRUCTION MANUAL

LANGUAGE: Basic and machine language

MEMORY USAGE: 37K Bytes (Running)



Spritemaster™ 64

by access