

Garry Kitchen's

# GameMaker™



*The  
Computer  
Game  
Design  
Kit™*



*For  
Commodore  
64™/128™  
Computers*

*A guide to creating and editing animation  
programs and computer games with GameMaker,  
including special comments and programming tips  
from Garry Kitchen.*

ACTIVISION  
CREATIVITY SOFTWARE™

# TABLE OF CONTENTS

GameMaker Overview .....	3
Loading GameMaker .....	4
Using the Joystick or Keyboard .....	5
Getting Started/Creating a Simple Animation Program .....	6
The GameMaker Editor .....	17
SceneMaker .....	26
SpriteMaker .....	36
SoundMaker .....	44
MusicMaker .....	52
Advanced Game Design .....	60
GameMaker Commands .....	72
GameMaker Helpful Hints and Cautions .....	93
Saving and Loading .....	96
Printing Your Program .....	98
Error Messages .....	98
Games, Scenes, Sprites, Sounds and Music Included on your GameMaker Disk .....	100
Color Chart for SceneMaker/SpriteMaker .....	104
Note Chart for SoundMaker .....	104
Instrument Chart for SoundMaker .....	105
GameMaker Positioning Coordinates .....	105

## GameMaker OVERVIEW

Now you can create your own animated computer graphics and games just like the best professional designers with GameMaker by Garry Kitchen.

It's an easy to use tool that lets you create background scenes, design characters, create music and sound effects and put them all together in a program. Or you can choose from backgrounds and characters already created and saved for you, plus a great selection of pre-programmed music and sound effects.

Everything you need to put together your animation programs or games is included in GameMaker. We've also included several pre-programmed games – **Chopper**, a brand new game designed with GameMaker by John Van Ryzin, and two old favorites -- **MegaMania™** and **Pitfall!™**, adapted by Dan Kitchen using the GameMaker program. Plus there are many more games and surprises included in GameMaker that guarantee you hours of playing and programming fun. (You will find a complete list of all the pre-programmed game elements at the back of the manual.)

This manual is organized into several parts. We recommend you begin by learning how to load GameMaker into your computer in the **Loading GameMaker** section. Then read **Using the Joystick or Keyboard** to learn how to use the joystick controller with GameMaker.

Next, go through the **Getting Started/Creating a Simple Animation Program** section. This section takes you step-by-step through creating your first program. Once you've gone through this section, you're on your way! The rest of the manual can then be used for more advanced programming ideas.

The **GameMaker Editor** section discusses the commands you'll use to put your program together.

The **SceneMaker** section tells you how to create computer graphics for scenes.

The **SpriteMaker** section explains how to create characters and objects and how to make them animate.

The **SoundMaker** section describes how to make and use a variety of sound effects.

And the **MusicMaker** section explains how to arrange creative musical scores for your programs or games.

The section on **Advanced Game Design** describes the various commands you'll use to connect all the parts of your program together. In the introduction to the section, Garry Kitchen offers some valuable ideas concerning what goes into designing a game. By looking at each of the commands used to create a game called **Archer™**, you will see how a complete game is created using GameMaker. For a description of all the commands available in GameMaker, consult the **GameMaker Commands** section. A **GameMaker Reference Card** is included in your GameMaker package for handy reference to all the programming commands

We recommend you play with GameMaker. It's a program designed for experimentation, creativity and fun.

## **LOADING GameMaker**

- Follow the manufacturer's instructions to set up your computer system.
- Insert the GameMaker disk into your disk drive and turn the drive on. Then turn on your computer.
- If using a Commodore 64, type:

**LOAD " \* ",8,1** and press **RETURN**.

- If using a Commodore 128, type:  
--**GO64** and press **RETURN**.  
--Then type: **Y** and press **RETURN**.  
--Then type: **LOAD "★",8,1** and press **RETURN**.

When the program loads into your computer, you'll see the GameMaker Title screen.

To start the program:

- Your joystick should be in the first joystick port.
- Press the joystick button. The GameMaker Editor screen will appear.

## USING THE JOYSTICK OR KEYBOARD

GameMaker uses a joystick plugged into the first joystick port for programming. In the games you create, however, you can use two joysticks. (We'll explain more later in the **Advanced Game Design** and **GameMaker Commands** sections.)

### *Selecting Commands*

To execute a command with GameMaker, you must first select it. In this manual, whenever you're instructed to select something, it means:

- 1) Move the joystick until the pointer is on the object to be selected. This can be a command or an area on the screen.
- 2) Press the joystick button.

Try it out! Move the arrow to the **menu** command at the upper right corner of the Editor screen. Now press the joystick button. You've just told GameMaker you want to go to the Main Menu by selecting the command. The Main Menu is now on your screen.

You don't need to type or memorize any programming commands. Using GameMaker is simply a matter of selecting commands or areas of the screen.

**NOTE:** *You can use the function keys on the right side of your keyboard instead of a joystick. The space bar substitutes for the button. For a complete explanation, see **GameMaker Helpful Hints** on page 94.*

## GETTING STARTED/ CREATING A SIMPLE ANIMATION PROGRAM

Designing a program is like building something: a model plane, a house, a chair—any object that must be designed and constructed in pieces, then put together. It involves having an idea of what you want the final product to look like, then putting the pieces together to make it look like you want it to look.

Let's create a simple animation program using GameMaker. If you've followed the instructions to the end of **Using the Joystick or Keyboard** section, you should be at the GameMaker Main Menu. If you're not there, use the joystick to move the pointer to the **menu** command. Press the button. The Main Menu will return to your screen.

To begin:

- Use the joystick to move the pointer to the **editor** command.
- Press the button. (Remember, this is called **selecting** a command.)
- You now see the GameMaker Editor screen. The program that you see listed on the screen is the program that makes the Title Screen of GameMaker.

Let's clear the screen. To do so:

- Select the **clr** command at the right of the screen.

- At the top of the screen, you'll see the message:

**clear the program area?**  
[yes]    [no]

- Move the pointer to **yes** and press the button.

The first thing we'll do in the program is use a "scene". Scenes are explained in detail in the **SceneMaker** section of the manual. For now:

- Move the pointer until it points to the small **tab** at the bottom right corner of the **Command Window** (where you see **add 0000 to score 1**).
- Press the button. The tab will start flashing.
- Pull back on the joystick until you see four or five commands displayed in the Command Window at once.

This will make it easier to find the commands you'll need for creating your animation program.

- Press the button again. The Tab will stop flashing.
- Move the pointer to the small **arrow** at the bottom left of the Command Window.
- Press the button. You will see the commands scroll up into the window, one at a time. (Push up or pull back on the joystick while pressing the button to scroll quickly.)

**NOTE:** To scroll in the reverse direction, point to the arrow at the top left corner of the Command Window and press the button.

- Scroll through the commands until you reach the command:  
**scene 1 is [    ]**

**NOTE:** The commands in the Command Window are listed alphabetically , so you always know in which direction to scroll to find a particular command.

- Release the button to stop scrolling.
- Use the joystick to move the pointer until it points to the beginning of the command.
- Press the button.

The command will appear in the **Program Area** of the screen. You've just written your first programming command.

The pointer will be pointing to the **1** in the command. Notice that the 1 is highlighted.

- Press the button. You'll notice a message at the bottom of the screen that says:

**loading catalog**

GameMaker is now loading all of the scenes that were previously created and saved with **SceneMaker** on this side of the disk. When the catalog is finished loading, you will be pointing to the name **Archer**.

- Move the joystick forward. You'll see the next scene in the catalog. Keep pushing up on the joystick until you reach **jungl2**.
- Press the button to load **jungl2** into your program.

There! See how easy it is! You've just completed your first programming instruction.

**NOTE:** If you select the wrong command by mistake, you can correct it in one of two ways. First, point to the beginning of the command in the Program Area and press the button. (If you're too far to the left of the command, you'll see a **label** number appear—I001. Don't worry about this for now. Move the pointer just to the left of the beginning of the command



and press the button. You know you've "selected" a command when it becomes highlighted.) Now, you can either 1) find the correct command in the Command Window, point to it, and press the button; the new command will automatically replace the old one. Or 2) Point to the **del** command at the far right of the screen and press the button; the command will be deleted from the Program Area.

Let's run your program. To do so:

- Select the **run** command.

You should now see a picture with three trees on a grassy plain on the top half of your screen and a black cavern on the bottom half. You are running your program, even though it has only one programming instruction!

- To get back to the Editor screen, press the button.

---

**NOTE:** From now on, we will be "selecting" various commands. Whenever we use the word "select," this will mean:

- Find the correct command by scrolling through the commands in the Command Window until the one for which you are looking appears.
- Point to the command you want.
- Press the button, thereby adding the command to your program at the location of the blinking cursor in the Program Area.

---

Now you're ready to add some characters or "sprites" to your program. You'll want your characters to appear in your scene.

The "catalog" of sprites that have already been created with **SpriteMaker** and saved for you are on the reverse side of the **GameMaker** disk. Before

proceeding, take out the disk, turn it over and insert the disk, label down, in your disk drive.

- Select the command:  
**sprite 1 is [    ]**
- Press the button to select sprite 1 and again to load the catalog.
- Move the joystick forward until you find **dog**.
- Press the button.

**NOTE:** Notice that, when you load the **dog** sprite, another command is automatically added to the Program Area that says **sprite 2 is -dog**. This is because the **dog** sprite is actually made up of two sprites that work together as one. This is explained further in the **SpriteMaker** section of the manual.

You've now added a command that will use the **dog** sprite in your program. Now we'll place the dog on the scene in the right place. To do so:

- Select the command:  
**sprite x position = 000**
- At this point, moving the joystick forward selects sprites one through eight. For now, select the **dog** sprite (sprite 1).

The pointer now will be on the numbers. To scroll through the numbers, push forward on the joystick.

- When you reach **040**, press the button. This will set the horizontal (x) position of the dog sprite (sprites 1 and 2). (An illustration of the screen's horizontal (x) and vertical (y) coordinates appears in the **GameMaker Positioning Coordinates** illustration on page 105.)

Now let's set the vertical position of the **dog** sprite.

- Select the command:  
**sprite y position = 000**

- Press the button to select the **dog** sprite.
- Scroll through the position numbers until you reach **154** and press the button.
- Now run the program. (Point to the **run** command and press the button.)

You should see the **dog** sprite placed on the ground just to the right of the first tree. Looks pretty good, huh?

Now we'll "animate" the sprite. We'll make the sprite move. But first, press the button to return to the editing screen.

- Select the command  
**Sprite animation speed = 000**
- Press the button to select the **dog** sprite.
- Select **30** for the animation speed.

**NOTE:** From now on we will be setting numbers, colors, etc. This means:

- *Select the command*
- *Fill in the highlighted areas by moving the joystick forward or back to select an appropriate value for that command.*
- *Press the button when you reach the value you want.*

- Now run your program.

You can see that the dog is running, but he's running in one place. We need to tell the program the direction in which we want the dog to run and how fast we want him to run.

- Select the command:  
**sprite dir = 000 000**
- Select the **dog** sprite and set a direction equal to **064 right**. This command sets the direction of the sprite.
- Now select the command:  
**sprite movement speed = 000**
- Select the **dog** sprite and set a movement speed of **45**. This is the speed at which the dog will move.
- Run the program.

The dog is now running accross the jungle. Let's add another sprite to the program. (You can have up to eight sprites appearing on your screen at any one time.)

- Select the command **sprite 1 is [ ]**.
- Move the joystick forward to select sprite **3** and press the button. Press again to load the catalog of pre-created sprites.
- Select **duck**.
- Select the **sprite x position = 000** command.
- Set **duck 3** as the sprite.
- Select an x position of **90**.
- Select the **sprite y position = 000** command.
- Set **duck 3** as the sprite.
- Select a y position of **145**.
- Select the **sprite animation spd = 000** command.

- Set the sprite to **duck 3**.
- Set an animation speed of **30**.
- Select the **sprite dir = 000 000** command.
- Set the sprite as **duck 3**
- Select a direction of **064 right**.
- Select the **sprite movement speed = 000** command.
- Set **duck 3** as the sprite.
- Select a movememnt speed of **45**.
- Run the program!

Now you have a program of a dog chasing a duck. It is easy to add a couple more ducks to the program by using the **copy** command. Then the dog will be chasing a flock of ducks!

- Point to the **copy** command and press the button.
- In the message line at the top of the screen, the program asks you to **select the first line**.
- Move the pointer until you reach the beginning of the **sprite 3 is duck** command.
- Press the button.

The command is highlighted by a black bar. The message in the Message Line now reads **select last line**.

- Move the pointer down the screen until you reach the beginning of the **duck 3 movement speed = 045**.

- An entire area of commands in your program is now highlighted by a large black rectangle.
- Press the button.

You now have four choices: to **quit**, to **delete** the highlighted area, to **move** the area or to **copy** the commands in the area to another place in the program.

- Point to **copy** and press the button.

The program now asks you to **select destination**.

- Move the joystick until you're pointing to the blank line directly under the black highlighted area.
- Press the button.

The entire highlighted area is now copied to the new location you selected.

- Point to the **scrolling arrow** at the bottom left of the screen and press the button. This allows you to scroll through the program to see different sections of your program listing. The **scrolling arrow** at the bottom right of the screen allows you to scroll in the opposite direction.

Now we need to change a few things. We want to change the sprite that says **duck 3** to **duck 4**. To do so:

- Select all of the instances in the new (copied) section of commands where it says **duck 3**.
- Move the joystick to change each sprite to **duck 4** (one command at a time, of course).
- Change the x position of **duck 4** to **110**.
- Change the y position of **duck 4** to **150**.
- Run the program.

- If you want to, you can add another sprite, **duck 5**, to your program in the same way you added the fourth sprite, **duck 4**. If you want to do this, change the x position of the new sprite to **115** and the y position to **155**.

Now let's add a little music to the program. Make sure the flashing blue square (the **cursor**) is positioned at the blank line directly under the last command in your program. (To do this, select the blank line under the last program line.)

- Select the command **song volume = 000**.
- Set the volume to **10**.
- Select the **song is [   ]** command.
- Press the button to load the catalog and select the **wiltel** song. (This is an abbreviation for the William Tell Overture.)
- Run the program.

Tallyho! The dog is now pursuing the ducks to the tune of the William Tell Overture. And there you have your first program! It's pretty simple, isn't it?

It's also very easy to change and modify things in your program. You can add more sprites to the scene. You can change the scene, change the sprites or change the music. And each time you do, you have a unique program!

Now let's save this program on your own library disk.

- Make sure you have a disk, either new or used (but no longer needed), which works with your computer. An extra blank disk is included in your GameMaker package.
- Remove the GameMaker disk and insert the disk that will now be your library disk.

- Select the **file** command.
- Select the **init** command.
- To confirm that you want to erase your library disk, select **yes** in the message Line. If you change your mind, select **no**.
- Type a new, up to six-letter name for your new library disk.
- When GameMaker is finished initializing your disk, point to the **file** command and press the button.
- Select **save**.
- Type in the name **chase** as the file you want to save. The name will appear at the top of the screen. Then press the button.
- Again, select **yes** to confirm that you want to save the file **chase**.

Now let's save the background scene for your game. You need to do this, because background scenes for games should be saved on the same side of the disk as the program you created. To do so:

- Select the **menu** command.
- Take out your disk and re-insert the GameMaker program disk, label up, into your disk drive.
- Select **SceneMaker**.

When SceneMaker is finished loading, follow these steps:

- You should see the scene for your program on your screen.



Take out the GameMaker disk and insert your library disk in the disk drive (the one on which you just saved the **chase** game).

- Select **file**.
- Select **save**.
- Type in the name **chase** and press the button.

The scene (named **chase**) is now saved on the same disk as your program. To get back to the Editor, select **quit** and answer **yes** to the confirmation question.

Now you can load the **chase** program into GameMaker and play with it at any time.

We've gone through just a few of the programming and animation ideas of GameMaker. If you want to learn more about creating complete games, or how to use any other part of the program, go on to the appropriate sections in the rest of the manual.

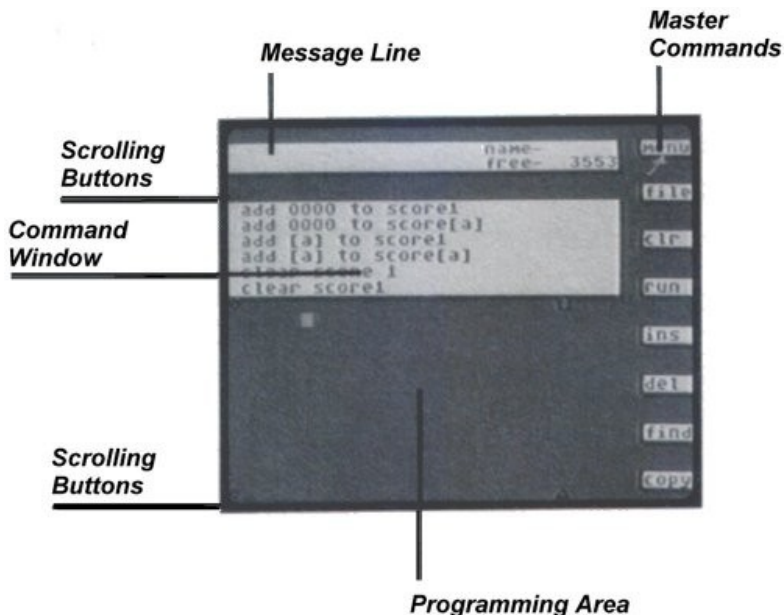
## ***GameMaker EDITOR***

### ***Introduction***

"One of the great things about the GameMaker program is that you can do everything with a joystick," says Garry Kitchen, creator of GameMaker.

"Usually when I'm programming a game, I sit hunched over the computer keyboard for months. With GameMaker, I can sit back in an easy chair, put my feet up and relax. Plus I don't have to remember any commands. They're all there for me."

The GameMaker Editor is where you actually put all of the pieces of your program together. The illustration below shows the Editor screen and its main parts.



### ***The Master Commands***

The eight small rectangles at the right border of the screen are the **master commands**. To execute one, point to it with the arrow and press the button (**select** the command). The Master Commands are:

**menu:** Takes you to the GameMaker Main Menu.

**file:** Presents another set of commands which handle various filing jobs. These are:

**quit:** Takes you back to the Editor.

**load:** Loads a program from disk. (See **Saving and Loading**, page 96.)

**save:** Saves a program on disk. (See **Saving and Loading**, page 96.)

**init:** Initializes a diskette so your computer can read and write on it. Initializing erases everything on a disk. When you select this command, GameMaker asks you to confirm your choice. Select **yes** to erase your disk. Select **no** to cancel the command. (**NOTE:** Make sure you have the disk you want to erase in your disk drive before using this command. Disks only have to be initialized once. Then you can save and delete files as many times as you want until the disk is full.)

*The only time you should re-initialize a disk is if you want to erase all the files on the disk or use the disk with another brand or model of computer.)*

**del:** Deletes a file saved on disk. To delete a file:

1. Select the **del** command.
2. Move the joystick to cycle through the files on the disk until you reach the one you want to delete.
3. Press the button again.
4. Select **yes** to delete the file.
5. Select **no** to select a different file name or cancel the command.

**make-a-disk:** Allows you to **create a version of your game that does not need GameMaker to operate.**  
To use the Make-A-Disk command:

1. Load the game you want to **create** into the GameMaker Editor. (See **Saving and Loading** on page 96.)
2. Insert an initialized disk into your disk drive. **Make-A-Disk command will not work if your disk is not initialized.**
3. Select the command **make-a-disk**.
4. Type in your name and press **RETURN** or the joystick button.
5. Press the button again to create the game. Or type in a new name for the game and press the button.
6. You'll see the message "**saving [your game name]**" at the bottom of the screen.
7. **When your friend gets your disk with your game, he can play it by typing LOAD"(your game name)",8,1, RETURN. He does not need the GameMaker program to enjoy your game!**

**NOTE:** *Make sure you include instructions for playing your game when you send it to your friend or keep it for your own game library.*

**prnt:** Prints a listing of the program shown in the Program Area of the GameMaker Editor screen. Make sure your printer is on and then answer the **yes/no** message to confirm or cancel your choice.

A list of printers you can use with GameMaker can be found on page 98.

**clr:** Clears the program area. Select **yes** to erase the screen. Select **no** to cancel the command.

**NOTE:** *Unless your program is “SAVED”, there is no way to retrieve it once the Program Area is cleared.*

**run:** Runs the program in the Program Area. To get back to the Editor from a running game, press the button on the joystick plugged into port one. If the program you've designed is using the first joystick button, press the space bar.

**ins:** Inserts a blank line at the blinking cursor in the Program Area. To insert a blank line:

1. Point to the line below which you want a blank line inserted.
2. Press the button to highlight the line.
3. Select the **ins** command.

**del:** Deletes a line at the blinking cursor in the Program Area. To delete a line:

1. Point to the line you want deleted.
2. Press the button to highlight it.
3. Select the **del** command.

**find:** Finds and places the cursor next to any command that's labeled. (See the **jump to label l001** command on page 77.) To find a label:

1. Select the **find** command.
2. Use the joystick to select the number of the label you want to find.

3. Press the button.

**copy:** Allows you to copy, move or delete a number of program lines in the Program Area. To copy:

1. Select the **copy** command.
2. Select the first program command you want to copy by pointing to it and pressing the button. A black rectangle highlights the line.
3. Move the point to the last program command you want to copy and press the button.
4. Point to the area on the screen where you want the copied lines to go. Press the button.

You'll see some additional commands in the Master Command area:

**copy:** To copy the lines you selected to the new location.

**move:** To move the lines you selected to the new location.

**delete:** To erase the lines from the Program Area.

### ***The Message Line***

The Message Line gives you messages to help you use some of the more complex commands. Also, anytime GameMaker feels you might accidentally erase or delete some of your work (for example when using the **del** command or saving a file), the Message Line asks you to confirm your selection. It does this by giving you a **yes/no** prompt.

To answer a question in the Message Line:

1. If using the **save**, **init** or **make-a-disk** commands type in the name of your choice.

Otherwise:

1. Move the joystick to cycle through a list of available selections.
2. When you come to your choice, press the joystick button.

At the right side of the message line is the **name** of the file that's on your Editor screen and the amount of **free** memory available for you to work with. (You start with 3553 spaces, or programming lines. If the memory is getting low, you can use various methods to change your program and conserve memory. See the **GameMaker Helpful Hints and Cautions** section on page 93.

### ***The Command Window***

The Command Window displays all of the commands available to you for making a program. The Command Window can be lengthened or shortened, and the list of commands can be scrolled forward and backward in the window. To increase the size of the Command Window:

1. Point to the small "tab" at the lower right corner of the window. (See the GameMaker Editor illustration on page 18.)
2. Press the joystick button. The Command Window tab will flash.
3. Use the joystick to move the pointer toward the bottom of the screen. The Command Window will expand.
4. To shrink the window back, repeat the above procedure and move the pointer toward the top of the screen.

To scroll through the commands:

1. Point to the small “up” arrow at the upper left corner of the Command Window.
2. Press the joystick button and hold it down. The commands will scroll from bottom to top through the Command Window. To scroll quickly, point to the Command Window arrow, press the button and move the joystick in any direction.
3. To scroll toward the bottom of the commands, repeat the same procedure while pointing to the “down” arrow at the lower left of the Command Window.

To select a command from the Command Window and place it in a program:

1. Place the blinking cursor where you want the command to go in the Program Area by locating the line on which you want the command to go and pressing the button.
2. Scroll through the command list to bring the command you want into view.
3. Select the command you want to use. The command will be copied into the Program Area.

Try it out! First point to the scrolling button at the bottom right of the Command Window and press the joystick button. Move the pointer down the screen to expand the window.

Make sure you can see the blinking cursor in the Program Area. Then point to the first command in the Command Window and press the joystick button. This command will become the first line in your program.



## ***The Program Area***

The Command Window and Program Area work closely together. All of the commands used to make your program are taken from those listed in the Command Window.

The way you create a game or program is by putting the commands together in the order you wish them to be executed. The order of the commands is important! (For more information, see the **Advanced Game Design** section which starts on page 60.) You can run your program as often as you like to see if it's working the way you want.

Many of the commands you choose need more information to be complete. For example, scroll the command list until you find the command "**Scene 1 is [   ]**" with the pointer. Press the joystick button. The command appears in the Program Area. To complete or change the information in the highlighted area of a command:

1. Select the highlighted area of the command. This tells GameMaker you're ready to fill in the box or change the information
2. Move the joystick to cycle through the list of choices for that area.
3. When you reach the choice you want, press the joystick button. That value will be fixed as part of the command. The values in the commands can be changed at any time while you are composing a program by repeating the above procedure.

## ***Scrolling in the Program Area***

To scroll through the list of program commands in the Program Area:

1. To move forward (top to bottom) through your program listing, point to the small "up" arrow at the bottom left of the Program Area and press the button.

2. To scroll backward through your program listing, point to the small “down” arrow at the bottom right of the Program Area and press the button. (See the GameMaker Editor illustration on page 18 to help you locate the Program Area Scrolling Arrows.)
3. To scroll quickly, point to either scrolling arrow, press the button and move the joystick in any direction.
4. To stop scrolling, release the button.

## **SceneMaker**

### **Introduction**

The SceneMaker program helps you draw your own computer graphics for your animation programs or games. Also there are many scenes already designed and saved on both sides of the GameMaker program disk. The scenes are listed in alphabetical order in the catalog displayed when you load a scene. (A complete list is also at the back of the manual on page 100.)

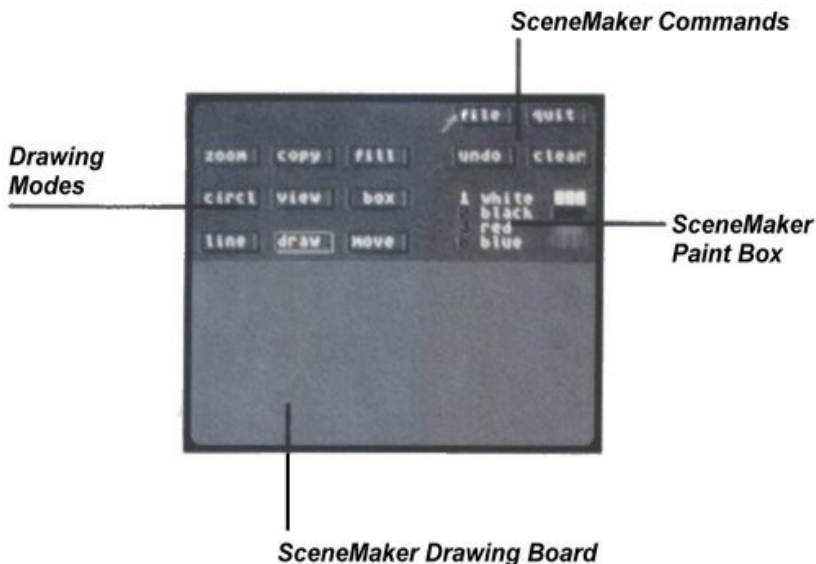
“It can be very time-consuming,” Garry Kitchen says, “to do detailed computer graphics. With SceneMaker, you’re given tools that make it extremely simple to make a professional drawing. It’s like making a computer painting. And it’s very easy to do.”

To load the **SceneMaker** program:

1. Make sure Side 1 of the GameMaker disk is inserted, label up, in your disk drive.
2. Select the **menu** command. Or select **quit** and then **menu**.
3. Select the **SceneMaker** command from the GameMaker Main Menu.

When the **SceneMaker** program finishes loading, you'll see the SceneMaker Screen.

### ***The SceneMaker Screen***



The **SceneMaker Commands** help you create the scenes.

The **SceneMaker Paint Box** lets you choose the colors for your scene.

The **Drawing Modes** offer options for drawing the scene.

The **SceneMaker Drawing Board** is where you actually draw your scenes.

### ***Loading a Scene***

You can create your own scenes from scratch for use in your game or program. Or you can use those already created and saved for you. To see how SceneMaker works, let's call up the scene that you used in the **Getting Started** section.

1. Select the **file** command.

2. Now select the **load** command. You will see the message **loading catalog**. When the catalog is loaded, move the joystick until you reach the **jungl2** scene and press the button.
3. Select **yes** to confirm that you want to load the **jungl2** scene.

You will see the bottom half of the scene on your screen. To see the whole scene, you must turn on the **view** command. (See **view** on page 33.)

### ***The SceneMaker Commands***

The commands in the small rectangles at the upper right of your screen are the master commands for the SceneMaker program.

- file:** Presents another set of commands which handle various filing jobs. These commands work in exactly the same way throughout the GameMaker program. See the **file** description under the GameMaker Editor Master Commands on page 17 for more information.
- quit:** Takes you back to the GameMaker Editor. Select **yes** to leave SceneMaker or **no** to cancel the command.
- undo:** Reverses the **last** major change you made in SceneMaker. **Undo** is an “oops, I didn’t want to do that” command. Use it immediately to correct an error.
- clear:** Clears the SceneMaker Drawing Board. You must confirm this choice by selecting either **yes** or **no**. If you haven’t saved the scene you are working on (see page 96), you will lose it if this command is selected.

### ***The SceneMaker Paint Box***

Select the background color and three additional colors for your scene in the Paint Box area. All the colors available to you are listed on page 104. To choose a color for the background:

1. Point to the color preceded by a “**b**” (signifies background) and press the button. The color becomes highlighted.

2. Move the joystick forward to cycle through the available colors. Move the joystick to the right to cycle quickly through the color selections. When you reach the color you want, press the button.

Choose the additional three colors for your scene in exactly the same way.

### ***The Drawing Modes***

SceneMaker provides you with all the tools you'll need to draw your scene. When you select a drawing mode or turn it **on**, your selection becomes highlighted. Select it again to turn it **off**.

**NOTE:** *Several SceneMaker modes can be used together. For example, you might want to turn on the **zoom** mode and then use the **circl** mode while in **zoom** mode. Any modes that are on will be highlighted or will flash.*

*If you select a mode that can't be used with one that's currently on, SceneMaker will automatically turn off the old mode for you.*

The available drawing modes are:

**draw:** Lets you draw lines in the SceneMaker Drawing Board area. To draw:

1. Select the color you want to use.
2. Turn the **draw** mode **on**.
3. Pull back on the joystick until you see a pencil in the Drawing Board area. Use the joystick to position the pencil on the Drawing Board.
4. Press the button. A small dot of color will be painted on the Drawing Board.

5. To draw, press and hold the button while moving the joystick.

**circl:** Lets you easily draw a circle. To use the **circl** mode:

1. Select the color you want to use.
2. Turn the **circl** mode **on**.
3. Position the pencil on the Drawing Board
4. Press the button. Move the joystick just a little. A shaded pencil will mark a point. This point will become the center of your circle.
5. Move the joystick to another point on the Drawing Board and press the button again. The distance between the pencil points will become the radius of the circle. A circle will be drawn using the radius.

**line:** Lets you quickly and easily draw a straight line. You can draw horizontal, vertical or diagonal lines. To use this mode:

1. Select the color you want to use.
2. Turn the **line** mode **on**.
3. Position the pencil on the Drawing Board and press the button. Move the joystick. A shaded pencil will mark the point.
4. Move the pencil to a new location.
5. Press the button. A line will be drawn between the two points.

**box:** Lets you quickly draw a rectangle or square. To use the **box** mode:

1. Select the color you want to use.
2. Turn the **box** mode **on**.
3. Position the pencil on the Drawing Board and press the button. Move the joystick. A shaded pencil will mark the point.
4. Move the pencil to a new location.
5. Press the button. A square or rectangle will be drawn using your two points as the two farthest diagonal corners of the box.

**fill:** Lets you fill a space with color. To use the **fill** mode:

1. Select the color you want to use.
2. Turn the **fill** mode **on**.
3. Position the pencil on the Drawing Board.
4. Press the button. Several pencils will appear on the Drawing Board and will fill in the open space with color. (Make sure the space you want to “fill” is appropriately marked off, or you’ll fill your entire screen. Remember, however, you can always **undo** your last choice. Refer to the section on **SceneMaker Commands**, page 28, for a further explanation of **undo**.)

**zoom:** Gives you a magnified view of what you’re drawing. It’s very useful for drawing detailed scenes. To use **zoom**:

1. Turn the **zoom** mode **on**.
2. Move the pencil into the Drawing Board area. You’ll see a flashing window on the Drawing Board and a checkerboard display in the upper left corner of your screen. The small flashing window on the Drawing Board acts like a magnifying glass. This magnified portion of the

Drawing Board is what appears in the checkerboard display in the upper left corner of your screen.

3. Use the joystick to move the flashing window to the area on the Drawing Board where you want to draw. In other words, what you want magnified.
4. Press the button. A pencil will appear in the zoom checkerboard area. You can now draw your object dot by dot. To do so, place the pencil in a square of the checkerboard and press the button. A block of color is placed on the board. You can see the real size of the picture on the Drawing Board in the flashing window.

**Zoom  
Checker-  
board**



5. To quit the **zoom** mode, move the pencil as far to the right as possible (until you reach the SceneMaker Paint Box).



6. Pull back on the joystick until the list of available modes appears again.

**copy:** Lets you “cut and paste” drawings in an area on the Drawing Board. For example, if you have just spent a lot of time drawing a tree and would like another tree in your scene, you can use the **copy** mode, to “surround” the tree with a box and “copy” it to a new location. To use the **copy** mode:

1. Turn the **copy** mode **on**.
2. Use the joystick to move into the Drawing Board area. You'll see a flashing corner cut mark.
3. Position this cut mark on the area you want to copy and press the button.
4. Move the joystick to expand the cut marks into a box. This box should surround the area you want to copy.
5. Press the button again.
6. Move the joystick. A copy of the “cut-out” box will move with you.
7. Position the box in a new location.
8. Press the button. Any drawings which appear in the original area will be copied to the new location.

**view:** The **view** mode “hides” the SceneMaker commands and lets you see and work on the entire Drawing Board area instead of just a part of the scene. To view the picture:

1. Turn the **view** mode **on**.
2. You will see the entire Drawing Board area. It takes up the entire screen and shows you what you are working on. You can draw anywhere on this screen as long as you have a drawing mode turned on.

To get back to the list of available modes, move the joystick forward until the pencil goes off the top of the screen. The SceneMaker screen will come into view.

When using the **view** mode, any mode that you turned on prior to selecting **view** will remain on.

**move:** To work on one piece of your scene at a time while still being able to easily reach the list of modes and SceneMaker commands, use the **move** mode. To do so:

1. Turn the **move** mode **on**.
2. Move the joystick forward or backward to bring new sections of the Drawing Board into view.
3. Press the joystick button to quit the **move** mode.

## ***Erasing***

There are several ways of erasing with SceneMaker. For example, we've already mentioned how you can select the **undo** command to erase the last thing you did on the screen. (See **SceneMaker Commands** on page 28).

To Erase using the **draw** mode:

1. Select the background color in the SceneMaker Paint Box (**b**).

2. Turn the **draw** mode **on**.
3. Position the pencil on the object you want to erase and trace over it with the pencil. The area will be erased.

To erase your picture dot by dot:

1. Turn the **zoom** mode **on**.
2. Position the flashing window in the area you want to erase and press the button.
3. Position the pencil on the dot of color you want to erase in the zoom area.
4. Press the button. The color will be lifted off the Drawing Board.

To erase using the **box** and **fill** modes:

1. Select the background color of your scene in the SceneMaker Paint Box.
2. Turn **on** both the **box** and **fill** modes.
3. Mark off the area of the scene you want to erase with a **box**. (See the **box** drawing mode instruction on page 30.)
4. SceneMaker will draw a box and fill it with background color, erasing what was previously there.

### ***Saving your Scenes***

Once you've drawn a scene, make sure you save it. (See **Saving and Loading** on page 96.) Remember! You must save your scene before you **quit** to the GameMaker Editor, or you'll lose what you were working on.

# SpriteMaker

## Introduction

You can make colorful animated characters with the SpriteMaker program. You can also use any of the sprites already pre-created for you. The pre-designed sprites are listed in alphabetical order in the catalog when you load a sprite. (The catalog of sprites is contained on Side 2 of your GameMaker disk. A list can also be found at the back of the manual on page 101.)

“When you’re drawing a detailed character for your game, it’s important that you get a ‘zoom’ view of it,” remarks Garry Kitchen. “SpriteMaker lets you do that. It also lets you experiment with colors, sizes and textures. Characters are an important part of any game, but they’re also fun just to create and play with by themselves.”

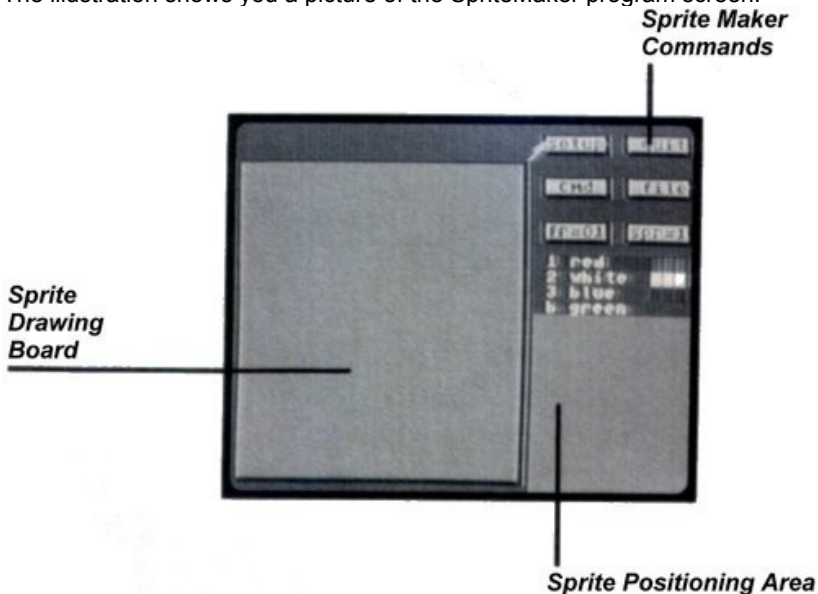
To load the SpriteMaker program:

1. Make sure Side 1 of the GameMaker disk is inserted, label up, into your disk drive.
2. Select the **menu** command. Or select **quit** and then **menu**.
3. Select **SpriteMaker** from the GameMaker Main Menu.

When the program finishes loading into your computer, you will see the SpriteMaker screen.

## *The SpriteMaker Screen*

The illustration shows you a picture of the SpriteMaker program screen.



The **SpriteMaker Commands** help you create sprites.

The **Sprite Drawing Board** is where you actually draw sprites.

The **Sprite Positioning Area** is where you position sprites when using more than one sprite working together to create a complete character.

### *Loading a Sprite*

To see how the SpriteMaker works, let's call up the **dog** sprite you used in the **Getting Started** section. To do so:

1. Turn the GameMaker disk over and insert the disk, label down, in your disk drive.
2. Select the **file** command.

3. Select the **load** command. **Loading Catalog** appears in the message line. When the catalog has finished loading, move the joystick until you reach the **dog** sprite. Press the button.
4. Select **yes** to confirm that you want to load the **dog** sprite.

After the sprite loads, you will see it on your screen. You will notice that the sprite appears in both the Sprite Drawing Board and the Sprite Positioning Area. You design sprites on the Sprite Drawing Board. The Sprite Positioning Area shows what the whole sprite will look like when it appears in a program.

### ***Setting Up A Sprite***

Before you can draw a sprite, you need to tell GameMaker a few things about what you want the sprite to look like. To do so:

1. Select the **setup** command. A box containing additional choices appears.
2. Select the **color** command. Here you choose whether you want the sprite to be a **single color** or **multi-colored**. To select **single** or **multi-colored**, press the button to make your choice. To select the opposite choice, press the button again.

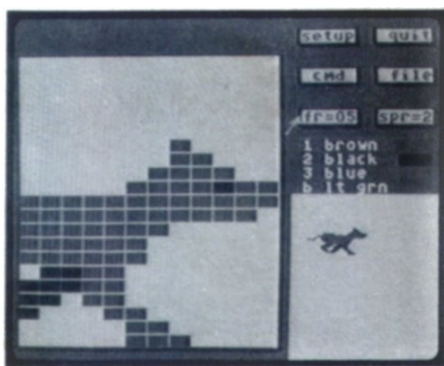
***NOTE:*** *Single color sprites allow you to make the sprite more detailed.*

3. Point to **v mag**. This lets you magnify your sprite vertically. Press the button to turn the vertical magnification **on**. Press it again to turn it **off**.
4. Point to **h mag** and select whether you want horizontal magnification **on** or **off**.

***NOTE:*** *Using **v mag** and **h mag** doesn't allow you as much detail and clarity when they're turned **on**, but they make your sprite appear larger.*

5. Point to **sprites** and press the button to select how many sprites you'd like working together as one. For example, look at the **dog** sprite. You'll notice that only half of the dog appears on the Sprite Drawing Board. This is because the **dog** sprite consists of two sprites that work together as one. The back half of the dog is one sprite, and the front half is another sprite. When you include the **dog** sprite in your animation or game program, two sprite numbers will always be taken up by the **dog** sprite. (See the section called **Drawing More than One Sprite** on page 40 for more information.)
6. Select **quit**. A message appears in the Message Line that says **position your sprites**. Depending on the number of sprites you've decided to use, you'll see up to four boxes in the Sprite Positioning Area, labeled 1 through 4. Press the button. Use the joystick to position sprite number 1 where you want it. Do the same for sprites 2 through 4, pressing the button each time you're ready to position a new sprite.

For example, to make a dog, the sprites were positioned like this:



### ***Drawing A Sprite***

Now you're ready to start drawing a sprite. First, you have to choose the colors you want to use for the sprite. The only colors used for the dog sprite are brown and black. The background color used is light green.

To change a sprite color:

1. Select the color (**1**, **2**, **3**, or **b**) you want to change. The color is highlighted.
2. Move the joystick to cycle through the list of colors available. (For a list of all available colors, see the chart on page 104.)
3. When you reach the color you want, press the button again.

You can use up to three colors per sprite and one background color.

To draw a sprite:

1. Choose the color with which you want to draw. To do this, point to the color and press the button **twice**.
2. Move the pointer to the left to get into the Sprite Drawing Board. The pointer will become a flashing dot of color.
3. To lay down the dot of color on the Drawing Board, press the button. Move the joystick. Another color dot will move with you.

See how easy it is! To use another color in a sprite:

1. Move the cursor back to the command area. The pointer will reappear.
2. Select the next color you want to use.
3. Move the pointer back onto the Drawing Board. The flashing color dot will be the new color you've chosen.

Any time you want to erase a color dot, simply move the cursor onto the dot you want to erase and press the button. The dot will be picked up by the cursor.

### ***Drawing More Than One Sprite***



Using more than one sprite together to make one larger sprite is like drawing separate pieces of a puzzle so that, when combined, they fit nicely together and look like one large object.

If you've set up a sprite so that more than one will be working together, you'll need to draw the other parts of the sprite. To do this:

1. Select the **SPR = 01** command in the Sprite Maker command area. (Notice the four flashing corners that surround the back half of the dog. This is the first sprite used in creating the dog.)
2. Move the joystick to select **SPR = 02**. (You will see the flashing corners surrounding the front part of the dog.)
3. Look at the Sprite Positioning Area. You will see the area of the sprite that you'll be working on.

Notice how the two sprites were drawn so that each half of the dog fits together like pieces of a puzzle. The first sprite consists of the back part of the dog's body. You can use up to four sprites together as one. (You specify this in the setup mode. See page 39.)

You could, for example, use four sprites to make a large dog. You'd simply divide the dog into four puzzle pieces and draw each separately.

The first sprite could be the dog's tail and upper half of his back end. The second sprite could be the upper half of the front of his body and his head. Sprite number three could be the bottom half of his back end and his hind legs. And the fourth sprite could be the lower half of the front of his body and the front legs.

### ***Creating a Sprite That Will Animate***

Often, you'll want to create a sprite so that the character or object will move once you put it in your program. For example, you may want the dog to run or the duck to flap its wings. To do this, you create the sprite using different **frames**. This is much like what a film cartoonist does when creating a cartoon character. Each frame becomes a step in the way the sprite moves.

If you've been following the instructions, you should have the **dog** sprite still on your screen. (If not, load it into the SpriteMaker now. See page 97 for loading instructions.) The dog sprite consists of two sprites fitted together like a puzzle to work as one. The dog animation sequence has six **frames**. The different frames work together to make the dog appear as if he is running. To see how the frames work:

1. Select the **FR = 01** command in the Sprite Maker Command Area. Look closely at the dog as it appears in the Sprite Positioning Area.
2. Move the joystick forward so that the command changes to **FR = 02**. Notice that, in the second frame, the dog changes a little. The dogs legs have started to stretch out. His front legs are up in the air, poised to take a forward leap.
3. Move the joystick forward to look at frame 3. The dog is fully stretched out in a running position now.

Each frame of the dog animation sequence must be drawn separately. And, if the sprite consists of more than one sprite working together, each part of the sprite must be drawn separately for each frame of animation.

Going through the sprite animation frame by frame will show you how the sprite will animate in slow motion. To see the sprite fully animated:

1. Select the **cmd** command. AN additional set of options will appear in the command area.
2. Select the **anima** command. In the message line you will see **animate—from 01 to 00**.
3. Select the **to 00** area of the command. Move the joystick forward until you reach **06** (since the sprite has six frames).
4. Press the button.

5. Now select the speed at which you want the sprite to animate. (You can select from 01 to 32. Experiment with the speeds to learn the effects of changing the animation value.)
6. Look at the Sprite Positioning Area. You can see the dog “animating”—moving from frame to frame at the speed you selected. To change speeds during an active animation sequence, pull back on the joystick to slow animation and push forward to speed it up.
7. To stop the animation, press the button again.

A sprite animation sequence can have up to 31 frames. It takes a little practice, but once you get the hang of it, frames are really easy to create. In addition to creating your own sprites, you can call up any sprite that's already on your GameMaker disk and modify it.

### ***Sprite Editing Commands***

There are various editing features available to make sprite animation easier. All of these commands are under the **cmd** command of SpriteMaker.

With the **dog** sprite still on your screen, point to **cmd**. A selection of other commands appears. These are:

- flip:** Turns the sprite in the direction you move the joystick. Select the command. Then move your joystick. **Flip** can turn your sprite upside down or around.
- anima:** When you select this command, you can see how the frames work to “animate” the sprite. Press the button again to turn the animation off.
- shift:** This command moves the position of the graphics within the area of the current frame.

- quit:** Takes you back to the previous set of commands.
- clear:** Erases the drawing on the Sprite Drawing Board. You must confirm that you want the sprite erased.

**NOTE:** *If you select this command it will erase **all** of the frames of the sprite, not just the one you're working on! Be careful! To erase just one frame, copy a blank frame over it.*

- copy:** Copies the sprite on the Drawing Board in another frame. (This is very useful in animating sprites.) When you select this command, you must indicate the frame you want copied and where you want it copied. For example, you could draw one frame of a bird and then copy the same drawing in the second frame. By changing the second frame just a little, you can “animate” your sprite without having to redraw the sprite every frame.

You must save a sprite *before* you **quit** to the GameMaker Editor, or you'll lose your sprite forever! (See **Saving and loading** on page 96.)

## **SoundMaker**

### **Introduction**

Using sound effects can help create the feeling of excitement in your programs and games. There are dozens of sound effects already created and saved on the second side of the GameMaker disk. They are listed in alphabetical order in the catalog when you load a sound effect. (For a complete list, see page 102 at the back of the manual.)

“Sounds are often used as rewards in a game,” says Garry Kitchen. “If someone does something good, they should get either a graphic reward or a sound reward. As far as conventional programming goes,” he continues, “sounds are a difficult concept to grasp, technically. You constantly wish you could just turn the knobs on your computer and see what happens. The SoundMaker program lets you do just that. You can just sit and turn the knobs and see how it sounds.”

## ***The SoundMaker Screen***

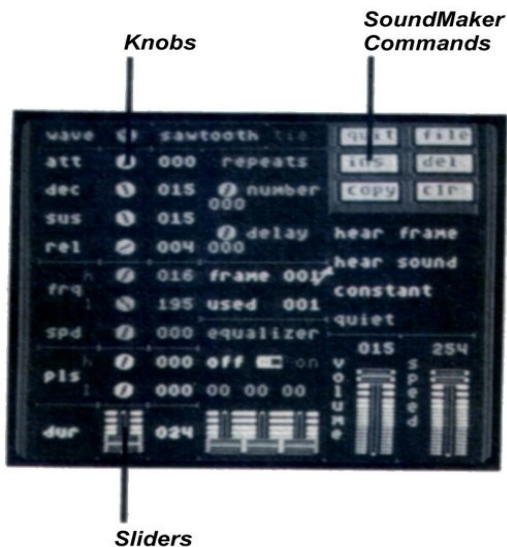
To load the SoundMaker program:

1. Make sure Side 1 of the GameMaker disk is inserted, label up, into your disk drive.
2. Select the **menu** command. Or select **quit** and then **menu**.
3. Select **SoundMaker** from the GameMaker Main Menu.

When the program finishes loading into your computer, you'll see the SoundMaker screen.

## ***The SoundMaker Screen***

The following is an illustration of what you will see on your screen:



The SoundMaker program closely resembles a “sound board” used by professional recording engineers.

The **SoundMaker Commands** help you create the sound effects.

The **Knobs** let you control the minimum and maximum strength of any sound attribute.

The **Sliders** act much the same way as knobs.

### ***Loading A Sound***

Let’s call up a sound to get an idea of how the SoundMaker program works. To do so:

1. Turn the GameMaker disk over and insert the disk, label down, into your disk drive.
2. Select the **file** command.
3. Select the **load** command. **Loading catalog** appears in the message line. When the catalog has loaded, move the joystick until you reach the **haryds** sound and press the button. (**Haryds** is the “losing” sound you hear in the **Pitfall!** game.)
4. Select **yes** to confirm that you want to load the **haryds** sound.

Let’s listen to this sound. To do so:

1. Select the **hear** command.
2. To stop the sound, select the **quite** command.

Each part of the sound effect sequence you just heard was created separately in a **frame**, then the **frames** were put together in a sequence. The **haryds** sound was created using 16 frames. Use SoundMaker to create single sounds in frames, then link them together to create one complete sound effect.

## ***Knobs and Sliders***

The SoundMaker program includes Knobs and Sliders to control the strength of the sound attributes. See the illustration on page 45.

To move a knob:

1. Point to it and press the button. The value that is controlled by the knob (usually to the right of it) becomes highlighted. Push forward or pull back on the joystick to increase or decrease the numbers. Move the joystick left or right to increase or decrease the numbers quickly.
2. When you reach the number you want, press the button again.

To move a slider:

1. Point to it and press the button. Move the joystick forward to move the slider up. The counter will change as you do so. Pull back on the joystick to lower the slider. Move the joystick left or right to increase or decrease the numbers quickly.
2. When you reach the number you want, press the button again.

## ***SoundMaker Commands***

The SoundMaker commands in the upper right corner of the SoundMaker board work in exactly the same way as the similar commands in the rest of the GameMaker program.

- quit:** Returns you to the GameMaker Main Menu.
- ins:** Inserts a frame in the middle of a series of sounds.
- del:** Deletes the current SoundMaker frame.
- copy:** Copies a sound from one frame to another.

**clr:** Clears the sound effects board completely. You must confirm that you want the board cleared. If you don't save a sound effect, clearing the board will erase it permanently.

**NOTE:** *Selecting this command erases **all** frames in a sound effect sequence.*

**file:** Presents another set of commands that handle various filing jobs. These work in exactly the same way as in the rest of the program. (See the **GameMaker Editor** section starting on page 17 for more information.)

## **Master Controls**

There are five Master Controls on your sound effects board. They are:

**repeats:** The **repeat** control repeats sounds. It's controlled by the **number** knob and the **delay** knob. Set the **number** knob to determine how many times the sound will be repeated. Set the **delay** knob to tell GameMaker how much time to pause between repeats.

**NOTE:** *If you select **hear sound** when **repeat** is set, SoundMaker will repeat the entire sound sequence over and over. If you select **hear frame** when **repeat** is set, you'll hear a single frame repeated over and over. If you set the **repeat** knob at 255, the sound will repeat indefinitely.*

**equalizer:** If you turn the equalizer **on**, it will remain on throughout the sound effect. The equalizer will filter the volume of any tone in the high, medium or low frequency range. When turned **on**, the volume level of the sound effect will lower. Use the sliders below the **on/off** switch to set the equalizer levels.



**volume:** Sets the volume for the entire sound effect.

**speed:** Sets the speed of the sound.

**frame:** Indicates which frame you're creating. The knob below it tells how many frames are used in the entire sound effect. You can have up to 511 frames to compose a single sound effect.

### ***SoundMaker Controls***

**wave:** Choose between four different sound waves—noise, square, sawtooth, or triangle. The noise wave creates a hissing sound. The square wave creates a tone of varying timbres based on the pulse width. The sawtooth wave makes a tinny sound. The triangle wave makes a hollow sound.

**att:** Raise or lower the speed of the increase in volume at the beginning of a frame.

**dec:** Raise or lower the speed with which a sound decreases after the attack.

**sus:** Determines the volume of the duration portion of a frame.

**rel:** Stands for release. Controls how long it takes a sound to fade away.

**frq:** Controls the high and low frequency, or pitch, of a sound effect. The low frequency levels have a fine-tuning effect on the pitch. The high frequencies have more dramatic effects.

**spd:** Raises or lowers the frequency of the sound effect at different speeds.

The **spd** control operates in two ranges.

The values of **1-127** control how fast the frequency of the sound is raised from the bottom of the frequency to the top of the frequency.

The values of **128-255** control how fast the frequency of the sound is lowered from the top of the frequency to the bottom of the frequency.

For example, setting the **spd** value at **127** creates a quick “sweeping” sound from the bottom of the frequency range to the top of the range.

Likewise, setting the **spd** value at **255** creates a quick “sweeping” sound in the opposite direction—from the top of the frequency range to the bottom of the range.

Setting the **spd** value at **0** will not change the frequency during any given frame.

**pls:** Stands for pulse. Works only when using the **square** wave. Determines the width of the square **wave**. Raising or lowering this knob changes timbre.

**dur:** Stands for duration. Determines the amount of time a sound level will be maintained after the **attack** and **decay**, and before the **release**.

**NOTE:** *If you've programmed a sound effect, but you do not hear anything when you select **hear sound**, check to see if one of the following are set to 0: **volume**, **duration**, **frequency**, or **pulse**.*

**tie:** Creates a smooth transition in sounds between frames.

## Hearing Sound Effects

The following commands let you hear the sound effects you create or load. They are:

**hear sound:** Lets you hear the sequence of frames that make up the entire sound effect.

**constant:** Flashes when you point to it and press the button. It repeats the sound or frame indefinitely until turned off. When this command is turned on, the **delay** command works with it to make the sound faster or slower. To turn the command off, select it again.

**NOTE:** Turning on **constant** is an editing feature only. If you save a sound effect with this control turned **on**, the sound will not repeat indefinitely when you retrieve it.

While in the **constant** mode, you can press the button to stop and edit any frame. The frame will appear on the screen, but the entire sound effect will keep playing.

**quiet:** Select this command to stop a playing sound.

We recommend you tinker a lot with the SoundMaker program. You'll be surprised at the sounds you can create! It also helps to look at the sounds already created to see how they were made.

Remember to **save** any sound effects you create, or you'll lose them forever! (See **Saving and Loading** on page 96.)

# **MusicMaker**

## **Introduction**

MusicMaker is used to create background music for programs and games. You can use the music already created and saved for you or create your own. The pre-programmed music is listed in alphabetical order in the catalog when you load a music piece. (The catalog of music is contained on Side 2 of your GameMaker disk. A complete list of the available music is on page 103.)

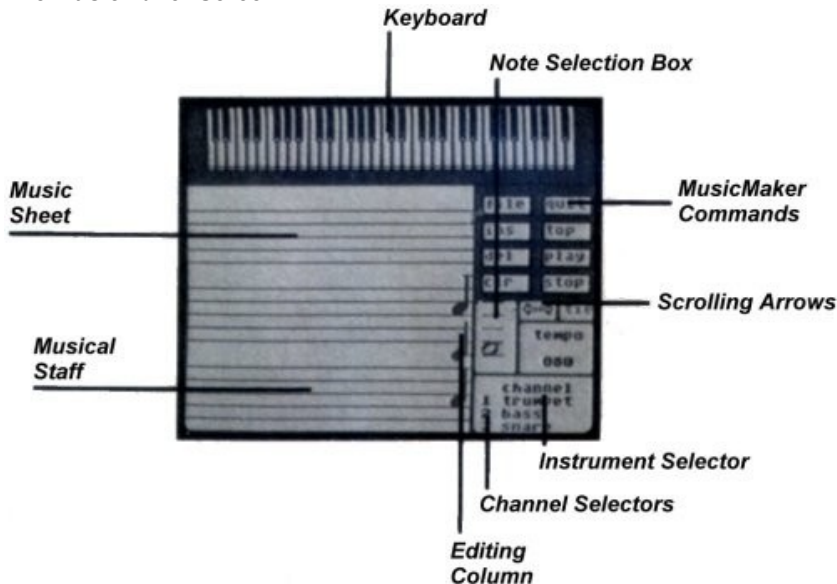
“The music in a game is very important,” Garry Kitchen says. “First of all, it lets the player know when a round of play has ended and another one is about to begin. It can also be used as a reward. It has an effect on people. Good music can get someone excited about playing a game. And it’s very easy to create with MusicMaker.”

To load the MusicMaker program:

1. Make sure Side 1 of the GameMaker disk is inserted, label up, into your disk drive.
2. Select the **menu** command. Or select **quit** and then **menu**.
3. Select **MusicMaker** from the GameMaker Main Menu.

When the program finishes loading, you’ll see the MusicMaker screen.

## The MusicMaker Screen



The **MusicMaker Commands** help you create and change music.

MusicMaker offers a **Music Sheet** with three **Musical Staffs** on which the music is written.

The **Scrolling Arrows** allow you to move forward and backward through the music, note by note.

The **Note Selection Box** is where the available notes are selected.

The **Editing Column** is where notes are entered and edited.

The **Channel Selector** lets you select the number of "voices" you want to use. The Commodore 64 or 128 computers have three voices. This means the computer can play up to three instruments at one time.

The **Instrument Selector** lets you choose the type of instrument that will be played in each channel (voice).

## **Loading Music**

To see how MusicMaker works, let's call up the song you heard in the program you created in the **Getting Started** section. To do so:

1. Turn the GameMaker disk over and insert it, label down, in your disk drive.
2. Select the **file** command.
3. Select the **load** command. **loading catalog** appears in the message line. When the catalog is loaded, move the joystick until you reach the **wiltel** song and press the button.
4. Select **yes** to confirm that you want to load the **wiltel** song.

After **wiltel** loads, you will see the first notes of the song in the Editing Column of the MusicMaker screen.

Let's play the song. To do so:

1. Select the **play** command. Notice how the notes are displayed on the screen as the song plays.
2. Select **stop** to stop the song.
3. Select the **edit** command.
4. To clear the screen of this song, select the **clr** command.
5. Select **yes** to confirm that you want to clear the song from the program. (This will not clear the song from your disk. A copy still remains there.)

## **The MusicMaker Commands**

The commands in the upper right of the screen are the **Master Commands** for the MusicMaker program.

**file:** Presents another set of commands which handle various filing jobs. These commands work in exactly the same way throughout the GameMaker program. See the **file** description under the GameMaker Editor Master Commands on page 17 for information.

**quit:** Takes you back to the GameMaker Editor. Select **yes** to leave MusicMaker or **no** to cancel the command.

**ins:** Inserts a note between two others. To use **ins**:

1. Use the **Scrolling Arrows** to locate the place in which the note will be inserted. (The note you're about to insert will be inserted **before** the one then in the Editing Column. See the section on **Scrolling Arrows** on page 59 for more information on scrolling.)
2. Select the **ins** command. The command flashes.
3. Move the pointer into the Editing Column and position the note you want to insert at the appropriate place on the staff.
4. Press the button. The note will be inserted. Each time you press the button while the **ins** command is flashing, a new note will be inserted into the music.
5. To turn off the **ins** command and advance (move the then current note to the left and bring the next note into the Editing Column), select the **ins** command again.

**top:** Takes you to the very first note in the song. Select the **top** command to place the first note of the song in the Editing Column.

**del:** Deletes a note in the song. To use **del**:

1. Use the **Scrolling Arrows** to bring the note you want to delete into the Editing Column.
2. Select the **del** command.
3. Select **yes** to delete the note from the song. Select **no** to cancel the command.

**NOTE:** Each time you delete a note, the note following the one you delete moves up in the Editing Column.

*All songs have to have at least one note. You can't delete the only note in a song.*

**play:** Lets you hear a song.

**clr:** Clears the music sheet of all notes. Remember to save the song you're working on before selecting this command, or you'll lose the work. You must confirm this choice.

**stop:** Lets you stop playing a song. To continue hearing the song, select **cont**.

## Selecting Channels

MusicMaker allows you to write music in three different "voices." To make arranging and editing your music as flexible as possible, MusicMaker provides a variety of different options for **hearing** the music and **seeing** the musical notation displayed on the screen for all three voices and all three instruments.

Displaying the musical notation (seeing each note appear first in the Editing Column and then advance across the Music Sheet) can be turned **on** or **off** with the **Channel Selector** located at the lower right corner of your screen. Turning the display for a channel on or off allows you to create each voice of your song separately.



To turn the display for a channel **on**:

1. Select the channel number (**1, 2** or **3**). A channel is **on** when its number is flashing. Turning a channel on means you can enter and edit notes in that channel.

To turn the display for a channel **off**:

1. Select the channel you want to turn **off**. A channel is off when it is not flashing. When a channel is turned off, notes may not be entered or edited in that voice, and the notes for that voice are not displayed on the Music Sheet.

Any channel may be turned on or off at any time.

**NOTE:** *Turning off a channel will not turn off the sound for that channel if you play the song. If you want to turn off the instrument for any channel (and therefore not hear the voice when the song is played), see **Changing Instruments**.*

## **Changing Instruments**

You can choose one of 13 instruments to be played on each channel. (The instruments that are available are listed in the back of the manual on page 105.)

To choose an instrument:

1. Select the name of the instrument that you want to change.
2. Move the joystick to cycle through the list of available instruments. When you reach the one you want, press the button again.

To turn an instrument **off**:

1. Select the instrument's name.
2. Move the joystick to the left to cycle through the list of choices until you reach **off**.
3. Press the button again.

When an instrument is turned **off**, you won't hear the notes on that instrument's channel when a song is playing.

As we explained earlier, if you turn a channel's display off, you'll still be able to hear the notes written for that channel. If you don't want to hear a particular channel's music, turn the instrument **off**.

### ***Entering Music on the Staff***

1. Move the pointer inside the **Note Selection Box** and press the button.
2. The note in the box flashes. You can only work on the music for one channel at a time. To select the channel number you want to work in, move the joystick left and right. The channel number and corresponding instrument flash together. The color of the flashing note matches the color of the channel on which you are working.
3. Move the joystick forward and back to cycle through the list of available notes. (A chart listing the notes is at the back of this manual on page 104.) When you find the one you want, press the button.

To enter the note on the staff:

1. Move the pointer into the MusicMaker **Editing Column**. The flashing note will follow you.
2. Move the joystick forward or back to move the note up and down the **Music Sheet**. When you reach the place on the staff where you want the note to go, press the button.
3. To advance and move to the next position on the staff, press the button again.

**NOTE:** When you're working in the Editing Column, you can move the joystick left to cycle through the available notes without having to return to the Note Selection Box.

## Scrolling Arrows

The Scrolling Arrows let you scroll forward and backward through a song, note by note. To scroll:

1. Select the **Scrolling Arrows**. The arrows start to flash.
2. Move the joystick left or right to move through the notes in a song.
3. To stop the scrolling, press the button.

**NOTE:** When scrolling, the next note in the selected channel moves into the Editing Column.

## Tie

To make your notes flow together smoothly, use the tie command. To turn on the **tie** command:

1. Select the **tie** command.
2. When the **tie** command is turned on, each note you enter on the staff is followed by a short quarter-moon symbol at the bottom of the note. This is the tie symbol.

## Tempo

Change the speed at which a song is played by changing the **tempo**. To change the tempo:

1. Select the **Tempo** command.
2. Move the joystick to increase or decrease the numbers. The lower the number, the slower the speed of the song. The higher the number, the faster the speed of the song.

3. When you reach the tempo you want, press the button again.

**NOTE:** *It's a good idea to experiment with the tempo of a song to see how it corresponds to the action in a particular game.*

When you create or modify any song, remember to save it. (See **Saving and Loading** on page 96.) If you don't save your music, your work will be lost forever!

## **ADVANCED GAME DESIGN**

### ***Introduction and Tips from Garry Kitchen***

Most game designers **program** their games in a computer language called **assembly language**. It's a very complicated and time consuming language that can take years to master.

With GameMaker, you can use the GameMaker commands to make the same kinds of programs created by professional game designers in assembly language. GameMaker takes the drudgery out of programming, so, instead of spending all your time writing computer **code**, you can spend your time playing and being creative. GameMaker designer Garry Kitchen offers some tips on the steps involved in creating a game:

"There are three important steps in creating a game. The first step is the **idea** for the game. The idea has to have a **theme** or focus. It has to have **characters**, and the game has to take place in a particular **setting**. Also, when coming up with an idea for a game, you have to think about **game play mechanics**. Game play mechanics concern what the person playing the game is going to have to do in order to play. In other words, how is the person going to interact with the game?"

"The second step in creating a game involves actually creating the program. With GameMaker, this is very easy to do. First you decide the **scene** for your game. Then you put up or create the main objects or **sprites** you'll be using in the game. Then you give the game some **movement**. You include in your program how you want each character to interact with other characters in the game."

“Next, you add **sounds** or **music** to the game. For example, you may want a firing sound to go off when you shoot a gun, or you might want the program to beep each time the score increases.”

“Finally, you need to add **game play** and **scoring** to your program. Game play has to do with the object of the game, what will happen when certain things occur, how the game will involve different levels of play, and things like that. Scoring, of course, is the number of points the player will get for playing the game well.”

“The third step in the process involves **working on the game play**. This is where the intangible part of creating a game comes in. For example, you need to fine-tune the **feel** of the game. This includes answering questions like: Is everything ‘in sync’? Do the sounds match the movement in the game? Are the character movements jerky or smooth? Are the graphics as good as they could be?”

“Then you need to include **depth**. *This is very important to game design.* You have to include surprises in the game. You have to make sure that the player doesn’t see everything in the game in the first five minutes of play. The player has to be continually surprised by new characters, new challenges, new graphics and new scenes.”

“The next component is what I call **addictiveness**. The game has to be good enough for the player to come back and play again. Again, all of this comes as a process of trial and error. You have to try different things. You have to experiment with your program until something works right.”

“The last 10% of game design is really what separates the good games from the great games. It’s what I call the **clean-up phase** of game design. Here’s where you make sure all the elements look great. The game should look good, feel good, sound good, play good.”

“In the end, the most important part of creating a game is time and effort. GameMaker makes that part of it very easy. Experimenting makes a good game. Keep trying things out. And don’t get discouraged if everything doesn’t work 100% right the first time. Practice makes for perfection. Find out what works and what doesn’t. That’s how all good game designers do it.”

## ***The Archer™ Game***

The short animation program you created in the **Getting Started** section of the manual introduced you to the concept of how you start an animation program. You simply choose the commands you want to use in your program from the Command Window in the GameMaker Editor and complete the commands, if needed.

Let's call up the game called **archer**. It's on Side 1 of the GameMaker disk. **Archer** is an "archery" game. It's not a very complicated game and was written solely for the purpose of explaining some of the most commonly used program commands. We'll take you through the **archer** program listing, line by line, so you can get an idea of how to use the commands. When you start to create your own game, you can even copy some of the procedures used.

A complete description of all the commands in GAmEmaKer is included both in the section titled **GameMaker Commands** and on the **GameMaker Reference Card**. Keep this card handy whenever you start to design a program.

Let's get started.

- Make sure Side 1 of the GameMaker disk is inserted in your disk drive.
- You should be at the GameMaker Editor screen. If you're not, go there now.

Let's load in the **archer** game. To do so:

- Select the **file** command.
- **Select the** load command. When the catalog finishes loading into your computer, move the joystick to find **archer**.
- Press the button.
- When **archer** has been loaded, you'll see the program listing in the Program Area of the screen.

Let's run the game to see how it works:

- Plug joysticks into port one and two of your computer.
- Select the **run** command.

As you can see, the **archer** game has a longbow at the bottom of the "archery field." There are three targets at the top of the screen. There are also some ducks flying by.

The object of the archery game is to move joystick #2 left and right until you line up with a target. Then press the joystick button to shoot an arrow. If the arrow hits a target, you get 1000 points. When you've hit all three targets, the game begins again.

The various parts of the **archer** game were created with the different programs in GameMaker. The targets, longbow and ducks are **sprites**. The archery field was made with **SceneMaker**. When you shoot an arrow, you hear a sound created with **SoundMaker**. And when you first start the game, music is played that was created with **MusicMaker**.

- Press the button on joystick number one to get back to the Editor.

Let's go through the commands to see how they are entered to create the **archer** game. You can only see a portion of the entire program in the Program Area of your screen at any one time. To scroll through the program listing:

- Select the **Scrolling Arrow** at the bottom left of your screen. (The one that's pointing up.)
- Press and hold the button to scroll more than one line at a time.

**NOTE:** If you need to refresh your memory on the different parts of the GameMaker Editor or how to scroll, review the **GameMaker Editor** section of the manual starting on page 17 before going on.

You will see the title of the **archer** game on the first line of the program. Notice how the word "archer" and the following few lines in the program are a light blue color. They are also preceded by a slash mark ( / ) at the very beginning of the line. These lines were created with the / **comment** command. The / **comment** command lets you type in notes to yourself or someone else about your program. Throughout the **archer** game, you'll see these comments. These were put there to help you understand each step in the program.

- Scroll the program listing until you see the comment:

```
    / define picture from  
    scenemaker.  
  
1002  scene 1 is archer
```

The number (1002) to the left of the first command in the program is called a **label**. Normally, instructions in a program are executed in order from top to bottom. However, sometimes you may want to tell your program to go back up and start over at a particular place (called loops; we'll get to them later.) Labels are used to tell your program what line to go back to.

The first command in the program tells the computer that the scene for this game is going to be the **archer** scene.



Look at the next set of commands in the program:

```
score 1 at row 01 column 07  
score 1 colr = 01 on 00  
clear score1
```

With the scene set for the game, a score needs to be placed on the scene. The first command in this set positions the score on the screen. The second command sets the color for the score (color 01 on color 00, values which have been set from the chart on page 82). The command **clear score1** sets the score equal to 00 and tells GameMaker to print the score at the specified location.

Let's continue to scroll through the program until you see:

```
sprite 1 is longbw  
sprite 2 is arrow  
sprite 3 is target  
sprite 4 is target  
sprite 5 is target  
sprite 6 is duck  
sprite 7 is duck  
sprite 8 is duck
```

These commands are used to define the sprites to be used in the program. You can have up to 255 sprites in a game, but only eight may appear on the screen at any one time. Each of the sprites was selected from the catalog when the command **sprite 1 is [ ]** was entered.

Scroll to the commands:

```
duck6 animation spd = 029  
duck7 animation spd = 029  
duck8 animation spd = 029
```

The only sprites in the **archer** program that will be animated are the ducks. (See **Creating a Sprite that Will Animate** on page 41 for more information.) These commands tell GameMaker at what speed the ducks will animate. Animation speeds for sprites range from 00 (still) to 32 (fast).

Scroll until you see:

```
duck6 dir = 064    right
duck6 movement speed = 020
duck7 dir = 064    right
duck7 movement speed = 032
duck8 dir = 064    right
duck8 movement speed = 026
```

These commands tell GameMaker in which direction each duck should move and at what speed. (The ducks would look very silly flapping their wings and staying in one place!)

There are 256 directions available for sprite movement.

**0 = up, 64 = right, 128 = down, 192 = left**

Any other value between these numbers will move a sprite in a direction between the directions of the numbers shown above (diagonally).

Movement speeds for sprites range from **0 (stopped) to 255 (fast)**.

Scroll until you see the commands:

```
longbw1 x position = 086
longbw1 y position = 225
arrow2 x position = 086
arrow2 y position = 000
target3 x position = 046
target3 y position = 150
target4 x position = 086
target4 y position = 150
target5 x position = 126
target5 y position = 150
duck6 x position = 180
duck6 y position = 075
duck7 x position = 180
duck7 y position = 085
duck8 x position = 180
duck8 y position = 095
```

These commands are used to position each of the eight sprites on the screen according to x (horizontal) and y (vertical) coordinates. The possible coordinates for positioning objects or sprites on the screen can be found in the **GameMaker Positioning Coordinates** illustration on page 105 at the back of the manual.

```
song is song  
pause for 10.0 seconds
```

The first command above tells GameMaker what song will be played at the beginning of the game. The song named “song” was selected from the MusicMaker catalog. The second command causes GameMaker to pause for ten seconds before proceeding to the next instruction.

```
1001  if joystick 2 is left  then  
        longbw1 dir = 192  left  
        set a = longbw1 x position  
        if a < 016 then  
            longbw1 movement speed = 000  
        otherwise  
            longbw1 movement speed = 032  
        end if  
    end if
```

These commands begin the main game **loop** of the program. The first command tells GameMaker to check and see if joystick 2 (the joystick used to play the game) is moving left. If it is, then the second command sets the direction of the longbow to 192 (going left). This tells the program that the longbow sprite will follow the movement of the joystick controller.

The next command says **set a = longbow1 x position**. This is a command that uses a **variable**. Put simply, variables are letters (a-z) that store numbers. Whereas a number itself cannot ever be worth a different numeric value (3 is always equal to 3), a **variable** can store a different number at different times throughout the program. You can even set the variable equal to the value of something else. That's what is happening with this command. The program is telling GameMaker to make the value of **a** equal to the longbow sprite's x position on the screen.

The next commands in this set tell GameMaker: If **a** (which is now equal to the longbow sprite's x position) is less than (<) 16 (the coordinate for the far left side of the screen), then don't move the longbow—it's far enough to the left already. Otherwise, set the movement speed of the longbow to 32.

**If** commands create a series of tests for the program to run. They simply tell the program: If this is the case, then do this. If this is not the case, do something else. Each **If** statement has to have an **end if** to it. The last two commands in this set (the two **end if** commands), finish the previous two if statements.

```
if joystick 2 is right then
    longbow1 dir = 064 right
    set a = longbow1 x position
if a > 156 then
    longbow1 movement speed = 000
otherwise
    longbow1 movement speed = 032
```

This set of commands does exactly the same thing as the previous set, except that the program checks to see if the joystick is moving right. (Notice that the variable **a** is used again.)

The next set of commands says:

```
if joystick 2 is off then
    longbw1 movement speed = 000
end if
```

This tells GameMaker: if joystick number two is not moving at all, then don't move the longbow sprite.

```
if f = 000 then
    if button 2 is on then
        set f = 001
        sound channel 1 = sho2s
        arrow2 y position = 225
        arrow2 dir = 000 up
        arrow2 movement speed = 064
        set a = longbw1 x position
        arrow2 x position = [a]
    end if
```

Now the program tells GameMaker to execute commands based on whether the button on the joystick is pushed. The first command in the set says **if f = 000 then**. The programmer used the variable **f** to keep track of whether or not an arrow is in progress. When an arrow is fired, he sets **f = 001**. Consequently, if **f = 000** then an arrow is not in progress, and the program should check the joystick button. If it is pressed:

- 1) The variable **f** will be equal to **001**, designating that an arrow has been fired.
- 2) Sound channel 1 will be turned on and the sound effect **sho2s** will be played (sho2s was created with the SoundMaker program).
- 3) The y position of the arrow will be set to 255, and the direction will be **up**.
- 4) The movement speed of the arrow will be 064.

- 5) Finally, the variable **a** is set to equal the longbow sprite's x position. The command **arrow2 x position = [a]** says: Make the position of the arrow line up with the longbow by setting the arrow's position equal to **a**.

```
otherwise
    set h = 000

if arrow2 hit target3 then
    target3 x position = 000
    set h = 001
end if

if arrow2 hit target4 then
    target4 x position = 000
    set h = 001
end if

if arrow2 hit target5 then
    target5 x position = 000
    set h = 001
end if
```

All of the commands above set up true/false situations to see which target (if any) is hit if an arrow is fired. The variable **h** is set to equal 0. (**H** will determine if a target has been hit.) The following three sets of **if** statements tell GameMaker: If the arrow hits a target, then remove the target sprite from the scene by setting its position to zero (the coordinate zero does not show up on the screen). Then set the variable **h** to equal **001**. This will mean that a target has been hit.

```
if h = 001 then
    sound channel 2 = hitsnd
    arrow2 y position = 000
    arrow2 movement speed = 000
    set f = 000
    add 1000 to score 1
end if
```

In the previous set of commands, the variable **h** was set to equal **001** to designate that a target had been hit. In this set the variable is used again. The commands tell GameMaker: If a target has been hit (specified by **h = 001**), then turn on sound channel 2 and play the **hitsnd** sound effect. Next, remove the arrow from the screen by setting its position to zero and set its movement speed to zero too. Finally, set the variable **f** to equal **000**. (Remember from earlier in the program that the variable **f** was established to designate whether an arrow was in progress.) Finally, add 1,000 points to the score.

```
if score 1 > 002000 then
    longbw1 movement speed = 000
    pause for 1.0 seconds
    sound channel 1 off
    sound channel 2 off
    sound channel 3 off
    jump to label 1002
```

This set of commands instructs GameMaker to start the game over if the score is greater than (>) two thousand. If the score *is* greater than 2,000, then the commands say to:

- 1) Set the movement of the longbow sprite to zero. In other words, even if the joystick is moved, don't move the longbow sprite.
- 2) Pause for one second.
- 3) Turn off all of the sound channels.
- 4) Then **jump** to label I002 to re-start the game. Remember in our discussion of labels we said you'll often want your program to go back and start executing commands over again. This is what the **jump** command is used for.

```
otherwise
    jump to label 1001
end if
```

These are the final commands in the game. They tell the program:

If the score is less than 2,000, jump back to the instructions which begin the main program loop (checking to see whether the joystick is left or right and continuing the game play.)

You'll notice that this game does not have a **stop program** command. Most games are like that. They have programs that continuously **loop**.

There you have it! Of course, there are many more complicated games included on the GameMaker disk, but this gives you an idea of how to begin programming your own games. You can even modify this one, if you want, to include additional game play instructions, different music, different sprites, etc. A complete list of games is included on page 100.

For a complete description of all the commands in GameMaker, see the **GameMaker Reference Card** or the **GameMaker Commands** section that follows.

## **GameMaker COMMANDS**

These commands are listed in alphabetical order, exactly as they appear in the Command Window.

**NOTE:** *that the majority of GameMaker commands require you to set a value (either a number, a letter, or both) when you select a command. The area(s) requiring a value will always be highlighted when you select the command. You can also change, at any time, the values you set for a command. When variable [a] is referred to, it is understood that [a] can be set to any variable from a-z. When number 000 is referred to, it is understood that 000 can be set to any number from 000 to 255.*

**add 0000 to score1**—add a value from 10 to 1000 to score #1 or 2 in increments of 10.

**add 0000 to score [a]**—add a value from 10 to 1000 to the score specified by variable [a].



**add [a] to score1**—add the value in a variable [a] to score #1 or 2.

**add [a] to score[a]**—add the value in a variable [a] to the score specified by a variable [a].

**clear scene 1**—Erases the information in the specified scene and clears the 4 colors (background and colors 1, 2, 3) to black. If this is the first instruction in the program, the screen does not turn on until after the clear is executed. This prevents a flash of old information on the screen from the previous time the program was run.

**clear score1**—Sets the specified score (#1 or 2 can be set) to 000000 and prints it on the screen at the specified location (see **score at. . .** and **score color. . .** commands).

**clear score[a]**—Sets the score specified by a variable [a] to 000000 and prints it on the screen.

**clear sprite**—Makes the designated sprite invisible by clearing its graphics and positioning it off the edge of the screen. This command should be used when changing a sprite from one design to another while moving it to a different location on the screen at the same time. Otherwise, you might notice a flash of old graphics at the new location. After a sprite is cleared, it must be completely set up again as if it was never used.

**/ comment**—Holds 25 characters of typewritten information. Comments are used to explain what a particular area of the program is doing or to pass along information to other people looking at your program. It is important to add comments to your programs so that later on you (or someone else) can understand what the program is all about.

**data table at l001**—Tells the computer the location of a data table. A **data table** is a list of numbers that you would like to refer to in your program. Numbers are read from the list using the **set a = value at data + [a]** command.

Following is an example of how and where to use data tables: in an adventure game in a house with 100 different rooms, each room could be made to look different by placing objects (furniture, pictures, etc.) in different places according to the room number. If you wanted a unique x position for a chair in each room, you might do the following: Assuming the variable (r) corresponds to the room number (0-99).

```
if r = 000 then
chair x position = 020
endif
.
.
.
if r = 099 then
chair x position = 089
endif
```

Programming in this way would require writing 300 lines of code. Using data tables will simplify this process dramatically. First enter 50 data value instructions containing the 100 x position values (2 per data value instruction). Then assuming your data table was at label 1001 and (r) is the room number, the following commands would handle the chair positioning:

```
data table at 1001
set a = value at data + (r)
chair x position = (a)
.
.
.
1001 data vlaue 020 080
```

Each value of r will pull out a unique number (chair position in the room) from the data table. For example, in room 000, the chair's x position is 020. This technique requires 53 instructions and much less work.

**data values—000 000**—Holds the values in a data table. Two values are entered per instruction. Data in a data value instruction is read from left to right.

**display other scene**—Instantly switches the screen from the current scene to the other scene (1 to 2 or 2 to 1). If this is the first instruction in the

program when the program is run, the screen will not turn on until after the switch to the other scene has occurred. This prevents a flash on the screen of one scene when the other scene was intended to be first.

**display scene 1**—Instantly switches the screen to the specified scene (may be set to 1 or 2). If this is the first instruction in the program when run, the screen does not turn on until after a switch to the correct scene has occurred. This prevents a flash on the screen of scene 1 when scene 2 was intended to be seen first.

**endif**—Marks the end of a logical if function. Analogous to the last parentheses in a mathematical expression. See the logical **if** explanation below.

**if . . . then**—Marks the end of a logical **if** function. The result of the expression in the **if** statement dictates the program flow until the next **endif** instruction. The program:

- 1) Executes all the instruction(s) after the **if** statement and before the next **otherwise** or **endif** if the expression is true.
- 2) Executes all the instruction(s) after the **otherwise** and before the **endif** if the expression is false *and* there is an **otherwise** before the **endif**.
- 3) Executes no instructions between the **if** and the **endif** if the expression is false and there is no **otherwise** statement.

The **if—otherwise—endif** logic becomes very simple to understand if you read the program like normal English—

**If a = 120 then set b = b + 001, otherwise set b = b - 001.  
(endif)**

Or—

**If joystick 1 is right, then man direction = 064 right and man movement speed = 020. (endif). If joystick 1 is left, then man direction = 192 left and man movement speed = 020. (endif). If joystick 1 is off, then man movement speed = 000. (endif)**

**NOTE:** You are allowed only 128 **if . . . then** statements in a program. See **skip next if** on page 87 to learn how to keep down the number of **if . . . then** statements in a program.

The following **if** statements are available:

**if a = 000 then**

**if a = [a] then**

**if a > 000 then**

**if a > [a] then**

**if a < 000 then**

**if a < [a] then**—Affects the program flow based on the comparison of variable **[a]** and a number “000” or a variable **[a]** and another variable “[a]”. The mathematical operators are = (equals), > (greater than), and < (less than).

**if button 1 is on then**—Affects the program flow based on whether the button on joystick #1 or 2 is on or off. If your program is using a **button 1** if statement, you will have to use the space bar on the keyboard to go from the game screen to the Editor screen.

**if joystick 1 is right then**—Affects the program flow based on the direction of joystick @1 or 2. The five possible conditions of the joystick are **up**, **down**, **left**, **right**, or **off**.

**if score 1 > 000000 then**—Tests if the current value of score 1 or 2 is greater than the number specified. The number may be set from **1000** to **100000** in increments of 1,000.

**if score[a] > 000000 then**—Tests if the current value of the score pointed to by a variable is greater than a number as explained above. In other words, if **a** is equal to **1**, the instruction is equivalent to **if score1 > 000000**. if **a = 2**, then it is the same as **if score2 > 000000**.

**if score 1 > score2 then**—Tests the two score values against each other. For example, **if score1 > score2 then print player 1 wins (endif)**.

**if sprite hit sprite then**—Tests if a selected sprite (1-8) is colliding with any other sprite or graphic on the screen at that point. The instruction can test for a collision with **sprites 1-8, anyone** (meaning any of the other sprites) or colors 2 and 3 of the scene picture (designated in the instruction by **clr2/3**). Because collisions between sprites and scene color 1 are not detected, a scene picture should be drawn with this in mind.

**jump to label l001**—Changes the flow of the program by jumping to the designated label. A label can be placed in the left column of any instruction simply by pointing to the far left of the command line and pressing the button. There can be up to 255 labels in a program. If the label designated in the instruction does not exist, the program will stop on this command.

**jump to label l[a]**—Jumps to the label designated by a variable **[a]**. For example, if **a = 23**, **jump to label[a]** is equivalent to **jump to label l023**.

**jump to subroutine at l001**—Jumps to the subroutine at the designated label. A **subroutine** is a group of instructions ending with a **return from subroutine** command. While a **jump to label** instruction permanently changes the flow of the program, a **jump to subroutine** remembers where the jump came from and returns back to the next immediate instruction automatically after the **jump to subroutine** is executed. The usefulness of subroutines can be explained by the following example.

Suppose you would like to execute the following instructions at 15 different places in your program:

```
sprite 1 is explos
sound channel 1 is hitsnd
add 0100 to score 1
```

Rather than having to enter these commands 15 times, you could put a **jump to subroutine at l001** at each place you want this sequence to be executed and put the following command at the end:

```
l001  sprite 1 is explos
      sound channel 1 is hitsnd
      add 0100 to score 1
      return from subroutine
```

Not only does this save the need to enter these instructions many times, it also saves two instructions (in this example) in your program area each time you call the subroutine. This can become significant if the **free** (memory) number at the top of the screen is getting low.

Every subroutine must end in a **return from subroutine** and a **return** can only be encountered after a preceding **jump to subrouting** command. Since the jump to the subroutine tells the computer where to return, executing a **return** without a **jump to subroutine** will cause a program to stop.

**jump to subroutine at l[a]**—Jumps to the subroutine at the label designated by a variable [a]. For example, if **a = 23**, **jump to subroutine at label[a]** is equivalent to **jump to subroutine at label l023**.

**otherwise**—Tells the computer to execute the commands after the **otherwise** and before the next **endif** if the expression after the **if . . .then** statement was *false*. An **otherwise** command must be preceded by an **if . . .then** and followed by an **endif**. See the explanation of **if . . .otherwise. . .endif** logic for more information.

**pause for 00.0 seconds**—Waits for the designated time interval to pass before moving on to the next command. Time intervals can be chosen from 00.1 to 25.5 seconds in 1/10<sup>th</sup> of a second increments.

**plot a dot at x =000 y = 000**—Plots a high resolution pixel (short for picture element) on your scene picture at the specified location. There are 160 dots horizontally by 200 dots vertically on the television screen. Visible horizontal locations lie in the range from **x = 012** to **x = 171**. Visible vertical locations lie in the range from **y = 050** to **y = 249**. Plot values outside these ranges will not plot a dot on the screen. These coordinate values correspond to the upper left dot on a sprite positioned at the same coordinates on the screen. (See the **GameMaker Positioning Coordinates** illustration at the back of the manual.)

**plot a dot at x = [a] y = [a]**—Plots a dot at coordinate values designated by two variables. If the variable for x = [a] is 100 and the variable for y = [a] is 200 then **plot a dot at x = [a] y = [a]** is equivalent to **plot a dot at x = 100 y = 200**.

**plot color 0 to scene 1**—Designates the color and destination scene used by the plot command. The selectable colors are **scene colors 0** (background or **b** in the SceneMaker), **1**, **2**, or **3**. A scene need not be displayed to plot it. The colors themselves are designated in the **SceneMaker** when the picture is drawn or can be set using the **scene 1 color** command.

**plot color[a] to scene 1**—Designates by a variable the color used by the plot command.

**NOTE:** *The plot commands can be used to draw objects on the screen. This command can also be used to save memory. For example, if you wanted to draw a small character, plotting the dot with the plot commands will take up less memory than designating the character with **SpriteMaker**.*

**print**—Prints 20 characters on the screen at the position selected by the **print at row 00 column 00** command in the colors selected by the **print color = 00 on 00** command. The print statement starts with 20 underline characters (the null characters). The program skips these characters when executing a **print** command. It is useful to understand the difference between a null character and a space. A space prints a blank box on the screen, while the null character prints nothing at all.

**print at row 00 column 00**—Tells the computer the destination of the beginning of the next **print** statement. The screen contains 25 rows by 20 columns. Therefore, the legal values for row are **00-24** and for column **00-19**.

**print at row[a] column[a]**—Designates the destination of the next **print** statement with variables.

**print character of [a]**—Prints the character specified by a variable [a]. There are 64 possible characters to print corresponding to **a = 000-063**. The characters are as follows:

<u>value of [a]</u>	<u>character</u>
00	space
01-26	A - Z
27	(
28	=
29	)
30	>
31	<
32	space
33	multiply sign ( * )
34	period ( . )
35-37	graphic characters
38	?
39	apostrophe ( ' )
40-41	graphic characters
42	degrees sign
43	+
44	up arrow
45	-
46	down arrow
47	/
48-57	0 – 9
58-63	graphic characters

The following program will demonstrate the use of the **print character of [a]** command as well as show you the entire character set on the screen:

```

clear scene1
scene 1 color 1 = white
print at row 00 column 00
print color 01 on 00
set a = 000
1001 print character of [a]
set a = a + 001
skip next if a = 064
jump to label 1001

```



**print color = 00 on 00**—Designates the foreground and background colors of the next characters to be printed. The first number selects the color of the character itself (foreground) and the second number selects the color of the background block on which it is printed. The selectable colors are **scene colors 0** (background or **b** in **SceneMaker**), **1**, **2**, or **3**. A scene need not be displayed to print on it. The colors themselves are designated by **SceneMaker** when the picture is drawn, or they can be set by using the **scene1 color** command.

**print color = [a] ob [a]**—Designates by variables the colors used by the print command.

**print on scene1**—Designates on which scene the program should print the next print statement. The choices are **scene1**, **scene2** or **both**. A scene need not be displayed to be printed on. The default (or backup scene if another is not selected) for printing is scene1.

**print value of [a]**—Prints the 3 digit numeric value of a variable. In other words, if **a = 001**, the program prints the characters **001**. In contrast, the **print character of [a]** command would print the single character corresponding to the value of variable **[a]** (see **print character of [a]** on page 80).

**return from subroutine**—Returns to the instruction immediately following the earlier executed **jump to subroutine at l001**. See **jump to subroutine** for a further explanation.

**scene 1 background = black**—Selects the background color (color 0) for a designated scene. This instruction overrides the background color designated in **SceneMaker**. There are 16 possible colors. See the color chart following the next command description for a list of the choices. (on page 82).

**scene background 1 = [a]**—Selects the background color as above based on the value of a variable **[a]**. The colors correspond to the color chart on page 82.

<b>COLOR CHART</b>	
0 – black	8 – orange
1 – white	9 – brown
2 – red	10 – light red
3 – cyan	11 – dark grey
4 – purple	12 – medium grey
5 – green	13 – light green
6 – blue	14 – light blue
7 – yellow	15 – light grey

**scene 1 border = black**—Selects the color of the border around the designated scene from the list of available colors (see color chart above).

**scene 1 border = [a]**—Selects the color of the border based on the value of a variable [a].

**scene 1 color 1 = black**—Selects color 1, 2 or 3 for the designated scene, overruling the colors used in **SceneMaker**.

**scene 1 color 1 = [a]**—Selects color 1, 2 or 3 for the designated scene based on the value of the variable [a].

**scene 1 is [ ]**—Loads a scene into the memory of the computer. The disk containing the desired scene should be in the disk drive before selecting this command. When this instruction is selected, the computer will load a catalog of all scenes available on the disk and allow you to choose one. It will then load that scene into memory for the specified scene (1 or 2).

When a program which contains a **scene 1 is [ ]** command is loaded into the computer, **GameMaker** will automatically attempt to load the desired scene (or scenes). If the picture(s) are not on the disk in the disk drive, a **file not found** error will be displayed. At that time you must point to the **scene 1 is [ ]** command to load the picture. It is always easier to keep your pictures on the same disk with the programs that use them.

**NOTE:** The **scene 1 is [ ]** command only serves the purpose of bringing a

*picture into memory during the process of writing a program and has no function during the actual running of your program. Only two scenes can be in memory at once regardless of the number of **scene 1 (or 2)** commands in your program. The last picture loaded will be the one that you will see for the designated scene.*

**scene 2 color 1 = black**—Selects color 1, 2 or 3 for **scene 2**, overruling the colors used in **SceneMaker**.

**scene 2 color 1 = [a]**—Selects color 1, 2 or 3 for **scene 2** based on the value of the variable **[a]**.

**score1 at row 00 column 00**—Designates the position at which score #1 will be displayed. See **print at row 00 column 00** for a description of the row/column layout of the screen.

**score2 at row 00 column 00**—Designates the position at which score #2 will be displayed. See **print at row 00 column 00** for a description of the row/column layout of the screen.

**score1 color = 00 on 00**—Designates the foreground and background colors for score #1. See **print color = 00 on 00** for an explanation of the color selection process.

**score2 color = 00 on 00**—Designates the foreground and background colors for score #2. See **print color = 00 on 00** for an explanation of the color selection process.

**score1 displayes on scene1**—Selects on which scene (or scenes) the designated score will be displayed. The options for this command are **scene1**, **scene2** or **both**.

**screen update on**—When writing a program that handles computer graphics, it is sometimes desirable to have certain changes to the screen occur at exactly the same time. In GameMaker, as in any computer language, instructions are executed one at a time. Commands such as **sprite 1 is [ ]** and **sprite 1 x position = 000** are executed and their results are seen on the screen as each one occurs. This may be undesirable if you would like to (for example) have eight sprites change at **exactly** the same time. **Screen update on/off** allows you to turn off the visible changes to the screen, execute as many screen changing instructions as you'd like, then turn the screen update on and have all the changes take effect at once, instantly.

For example, the program:

```
sprite 1 is duck
sprite 2 is duck
sprite 3 is duck
sprite 4 is duck
```

will change the four sprites into ducks one at a time (quickly, but still one at a time). However, the program:

```
screen update off
sprite 1 is duck
sprite 2 is duck
sprite 3 is duck
sprite 4 is duck
screen update on
```

will turn off the screen, execute the four **sprite is duck** commands without affecting the screen, and change the graphics of the four sprites simultaneously upon executing the **screen update on** command. It must be noted that all screen updates (including animation and movement) stop during the off stage of the screen. It is also useful to note that the GameMaker language executes noticeably faster with **screen update** set to off.

**NOTE:** *It is not necessary to understand the concepts explained above or to use the **screen update on/off** command to get excellent animated computer graphics. It is included as an available option for advanced users and programmers.*

**set a = 000**—Sets the value of a designated variable equal to a number from 000 to 255.

**set a = [a]**—Sets the value of a designated variable equal to the value of another variable. For example, if **c = 115**, after the completion of the instruction **set a = [c]**, the value of **a** would then also be equal to **115**.

**set a = a + a**—Adds a selected number to the value in **a**. Since a variable can only hold a number from 000 to 255, an answer greater than 255 will roll around to zero. For example, **255 + 001 = 000** and **255 + 002 = 001** etc.

**set a = a + [a]**—Adds the value of a variable to the value in **[a]**.

**set a = a-000**—Subtracts the selected number from **a**.

**set a = a-[a]**—Subtracts the value of a variable from **a**.

**set a = a \* 000**—Multiplies the value in the variable by the selected number.

**set a = a \* [a]**—Multiplies the value in a variable by the value in another variable.

**set a = a/000**—Divides the value in a variable by the selected number.

**set a = a/[a]**—Divides the value in a variable by the value in another variable.

**set a = rnd number from 0 to 000**—Sets a designated variable equal to a random value from 0 to the selected value. For example,

**set a = rnd number from 0 to 020** will only return values in a range from 000 to 020.

**set a = sprite x position**—Sets a designated variable equal to the current value of the selected sprite's x position on the screen. See the explanation of **sprite x position = 000** for a range of valid values.

**set a = sprite y position**—Sets a designated variable equal to the current value of the selected sprite's y position on the screen. See the explanation of **sprite y position = 000** for a range of valid values.

**set a = value at data + [a]**—Sets a designated variable equal to the value in the data table pointed to by [a]. The data table must have been designated previously with the **data table at 1001** command. The value in the second variable determines which number in the list will be taken. For example, if **z = 000**, the command **set a = value at data + [z]** will set **a** equal to the first value in the list pointed to by the previous **data table at 1001** command. Similarly, if **z = 010**, the command **set a = value at data + [z]** would set **a** equal to the **eleventh** value in that list.

**set a = value at ram + [a]**—Sets the designated variable equal to the value in the user RAM table. The RAM table is a 256 number data table in memory that you can save numbers to and read numbers out of. The RAM table works exactly like a user defined data table except that you can change the values in it during your program. A data table is fixed by your program. See **set a = value at data + [a]** command for an explanation of the use of data tables.

These commands are for advanced users and programmers.

**set value at ram + [a] = 000**—Sets the value in the RAM data table pointed to by a variable equal to a selected number. For example, if you would like to set the first 50 numbers in RAM equal to 100, you could use the following program:

```
set a = 000
1001 set value at ram + [a] = 100
set a = a + 001
skip next if a = 050
jump to label 1001
```

**NOTE:** *You should not expect the values in RAM to be equal to zero at the beginning of the program. You should initialize the RAM locations you will be using with a program similar to the one listed above.*

**set value at ram + [a] = [a]**—Sets the value in the RAM data table pointed to by the first variable equal to the value of the second variable. For example, if **z = 100** and **a = 255**, **set value at ram + [a] = [z]** would set the last RAM location equal to 100.

**skip next if a = 000**

**skip next if a > 000**

**skip next if a < 000**—Skips the next instruction in the program if the mathematical expression is true. This command can be used in place of an **if . . then** command for simple testing of variables. One advantage of this is the saving of one instruction, because a **skip next** does not need an **endif**. Another advantage is that you are only allowed **128 if . . then** tests in a program. Using **skip next** commands will keep down the number of **if . . then** commands you use.

**song is [ ]**—Loads a song into a program. The disk containing the desired song should be in the disk drive when the instruction is selected. GameMaker will display a catalog of all songs available on the disk and prompt you to select one. When selected, the song will load into memory. When the program is run and the instruction is executed, the song will play as it was heard in MusicMaker.

**song volume = 00**—Selects a volume for the music in a program. The values are from **00 (off)** to **15 (loudest)**. The volume of the music is run at a volume of 10 if a value is not set. To stop a song before it is finished, set the **song volume = 00**.

**sound channel 1 = [ ]**—Selects a sound effect to be executed on the designated channel. The disk containing the desired sound should be in the disk drive when the instruction is selected during programming. GameMaker will display a catalog of all sounds available on the disk and ask you to select one. When selected, the sound will load into memory. When the program is run and this instruction is executed, the sound will be heard on the specified channel as it was heard in SoundMaker. The Commodore 64 has three separate sound channels. Sounds on one channel will not affect another channel. However, if the program executes a

**sound channel 1 = [ ]** while another sound is executing on the same channel, the old sound will stop and the new sound will begin. It is wise to use different channels for different sounds that will be occurring at the same time. If music is playing when a **sound channel is [ ]** command is executed, the sound will take priority, and that channel of music will be lost until the sound is over. Experimentation is suggested when mixing music and sounds to achieve the best results.

**sound channel 1 off**—Shuts off the sound on the designated channel. It will not affect music playing on that channel.

**sprite 1 is [ ]**—Assigns a sprite file created by SpriteMaker to the selected sprite (1-8). The disk containing the desired sprite should be in the disk drive when the instruction is selected during programming.

GameMaker will display a catalog of all sprites available on the disk and ask you to select one. When selected, the sprite file will load into memory. When the program is run and this instruction is executed, the selected sprite (1-8) will appear as it was drawn in SpriteMaker. If the object or character was defined as a multiple-sprite object in SpriteMaker, GameMaker will automatically insert any additional **sprite 1 is [ ]** commands following the initial one in order to load the other parts of the sprite. Secondary sprites of a multiple-sprite object will be designated by a – as the first character in the name.

For example, the **horse** on the GameMaker disk is a four-sprite object. Selecting the **sprite 1 is horse** command while programming will result in the following commands listed in the program:

```
sprite 1 is horse
sprite 2 is -hors
sprite 3 is -hors
sprite 4 is -hors
```

After the **sprite 1 is horse** command is executed, any command that refers to sprite 1 will also affect sprites 2, 3 and 4. GameMaker will handle the four sprites as if they were one object.

**NOTE:** *It is suggested that every sprite of a multiple-sprite object fall within the same half of the eight sprites. In other words, a four sprite object like the horse should be used as sprite 1 or sprite 5. This is because GameMaker handles the sprites in groups of four (first 1-4, then 5-8). A multiple sprite object that is split between the two groups will work properly but might not look as clean as it could.*

It is not enough to execute a **sprite 1 is [ ]** to see a sprite. The sprite must also be positioned in the visible part of the screen. See the command description for **sprite x position = 000**.



**sprite animates always**—Selects one of two different types of animation for the designated sprite. If **sprite animates always** is selected, the sprite will cycle from frame one to the last frame in the animation sequence (as determined in SpriteMaker) and continue back to frame one as long as it has an animation speed greater than zero. If **sprite animates once** is chosen, the sprite will animate from the first frame to the last frame and stop on the last frame (automatically setting the animation speed at 0), until it receives another **sprite animation spd = 000** command.

**sprite animation spd = 000**—Selects a speed of animation for the designated sprite. The valid speeds are from **000** (no animation) to **032** (fastest) just as they are in SpriteMaker.

**sprite animation spd = [a]**—Sets the animation speed for the designated sprite equal to the value of a variable .

**sprite color 1 = black**—Designates a new color #1 for the selected sprite, overriding the color with which a sprite was drawn in SpriteMaker. See the color chart on page 104 for the color choices.

**sprite color 1 = [a]**—Sets color #1 for the selected sprite equal to the value in the variable, overriding the color with which the sprite was originally drawn in SpriteMaker. See the color chart on page 104 for the relationship between variable values and color.

**sprite dir = 000 000**—Sets the direction of the selected sprite equal to a value between **000** and **255**. The second number equates the selected value to a value in degrees of the compass (with 000 degrees = up). The direction numbers are as follows:

**000 – 000 degrees = up**  
**064 – 090 degrees = right**  
**128 – 180 degrees = down**  
**192 – 270 degrees = left**

**There are 256 possible directions (values 000-255)**

**sprite dir = [a]**—Sets the direction of the selected sprite equal to the value in a designated variable. The values correspond to the explanation above for the **sprite dir = 000 000** command.

**sprite movement speed = 000**—Sets the movement speed of the selected sprite. Valid movement speeds are from **000** (stopped) to **255** (fastest). The sprite will move in the selected direction (see **sprite dir = command**) until it is stopped with a sprite movement speed of **000**.

**sprite movement speed = [a]**—Sets the movement speed of the selected sprite equal to the value of a designated variable.

**sprite shared colr2 = black**—Sets the designated sprite shared color equal to the selected color. The Commodore 64 shares two out of the three colors of a multicolored sprite among all eight sprites. The only unique color a sprite can have is color 1. (See **sprite color #1** command). The other 2 colors will be the same for all the sprites on a screen at any given time. If this command is not executed, the shared colors will be the colors of the lowest numbered sprite in the program.

**NOTE:** *It is important to keep the shared color concept in mind when designing a sprite in SpriteMaker. The predominant color in a multi-colored sprite should be color #1. Also, if you know that many different sprites will be used in the same program, you should design them all using the same colors #2 and #3.*

**sprite under colrs 2/3**—Selects the display priority of the designated sprite with the scene graphics. A sprite can be designated to appear **under** or **over** scene colors two and three. Sprites always appear over the scene background color and scene color #1.

**NOTE:** *Sprites have decreasing priority with each other as their number increases. In other words, sprite 1 will always display on the screen in front of sprite 3, and so on.*

**sprite x position = 000**—Positions the selected sprite on the screen at the designated horizontal position. A sprite is first partially visible on the left edge of the screen at **x position = 001**, fully visible for the first time on the left edge of the screen at **x position = 012**, fully visible for the last time on the right edge of the screen at **x position = 160** and partially visible on the right edge for the last time at **x position = 171**.

**NOTE:** *it is important that, when moving a sprite to a new x and y position on the screen, the **sprite x position =** and **sprite y position =** commands should be sequential with no instructions in between them.*

**sprite x position = [a]**—Positions the selected sprite on the screen at the horizontal position designated by the value of the variable **[a]**.

**sprite y position = 000**—Positions the selected sprite on the screen at the designated vertical position. A sprite is first partially visible at the top of the screen at **y position = 030**, fully visible for the first time at the top at **y position = 050**, fully visible for the last time at **y position = 229** and partially visible on the bottom for the last time at **y position = 249**.

**sprite y position = [a]**—Positions the selected sprite on the screen at the vertical position designated by a variable **[a]**.

**NOTE:** *The perfect place to hide a sprite so that it is invisible is at **x position = 172, y position = 250**. The **clear sprite** command positions it at this location.*

**stop program**—Stops the execution of the program. A blank line in the program is equivalent to a **stop program** command. Note that all a **stop program** does is *stop the instruction execution*. Sprites, sounds and music will continue to do whatever they were doing at the point the program was stopped.

**trace of [a] on**—Selects the trace mode of program execution. Trace mode is useful to see the program flow and to track down unwanted features (bugs). This instruction can either turn **on** or **off** the trace mode. When a **trace of [a] on** instruction is executed, the program 1) stops at the next instruction, 2) displays the text of the command at the bottom of the screen, 3) places the program in single step mode, displaying one programming instruction at a time and 4) displays the current value of the variable designated in the trace instruction.

Pressing **S** on the keyboard will allow you to step through instructions one at a time. Pressing **V** will cycle from a-z for the variable being displayed. Pressing the **up/down arrow** on the keyboard will put the program into a slow-run mode with each subsequent press speeding it up a little. Pressing the **left/right arrow** on the keyboard slows down the run mode until subsequent presses cause it to stop and revert to single step mode.

Pressing **S** during slow-run brings the program back to single step mode. Pressing **T** will quit the **trace** mode and return to normal running of the program.

**Trace** mode can also be entered without an instruction in the program by hitting **T** on the keyboard at any time *while the program is running*.

**NOTE:** During **trace** mode, the sprites on the screen do not move as they would during full-speed execution of the program.

## GameMaker *HELPFUL HINTS AND CAUTIONS*

The following is a list of programming tips as well as other helpful hints and a few cautionary notes.

### *Programming and Editor Tips*

- \* Review the programs included on the GameMaker disk to learn techniques that professional game designers use.
- \* The program instructions are color-coded. Commands having to do with **if** . **.then** logic are coded yellow, **comments** are blue and all other commands are green. This coding makes it easier for you to look at a program and understand the program flow. Also, any instructions having to do with the flow of the program (**ifs**, **jumps**, **skips**) begin in a different column of the Programming Area than the rest of the commands
- \* Place a label (l001) at the beginning of a potentially long program in this way you can always get back to the top of the program by selecting **find** label.
- \* Comment your programs *well* so that you remember what you intended for your program code to do when you look at it later. Keep in mind the following memory information.
  - the **free** number tells you how many instruction spaces are left.  
(You're allowed 3553 total programming lines.)
  - all instructions except **print** and **comment** count for one
  - print statements count for **six** (6) instructions
  - comments count for seven-and-a-fourth (7 1/4) instructions.
  - sprites, songs, and sounds count for various amounts depending on their size and complexity. The amount can be determined by noting the change in the **free** amount at the top of the screen.
  - If you run out of space, cut one or all of the above.
- \* If you write part of a program that you would like to run as fast as possible and it doesn't use any sprites, turn off the **screen update** for the fastest execution possible.

- \* Save your program frequently while developing it, and make many backups. Disk drives and computer memory can surprise you at times.
- \* When you leave the GameMaker Editor to enter another part of the program, the program in progress will be lost. Save it before you leave the Editor
- \* Check out the GameMaker library on Side 2 of the disk before reinventing the wheel. We might already have what you are looking for. A complete list of all the games and game elements pre-programmed and saved on the GameMaker disk appears on page 100.
- \* Rather than trying to create sounds, sprites, pictures and songs from scratch, it is sometimes easier to load existing ones and modify them.
- \* If you see a flash when you run your program, use the **clear scene** instruction as the first one in the program.
- \* If you see flashes when you try to move sprites, read the sections on the **clear sprite** command, the **screen update on/off** command and the **sprite x position = 000** command.
- \* When using the **copy** feature in the **Editor**, be patient if you select a large area. The program can take a while to copy.
- \* Keep your pictures (scenes and sprites) on the same disk as the programs in which they are used to avoid **file not found** errors.
- \* Always double-check a command when you are asked to confirm your selection. You can always “escape” from the command by answering “no”.
- \* The function keys at the far right of your keyboard can be used in place of the joystick if your joystick happens to break. The space bar substitutes for the button.

To read the function keys in your program, use the **if joystick 1** command and the following chart:

joystick 1 up = F1  
joystick 1 left = F3  
joystick 1 right = F5  
joystick 1 down = F7

\* Program development goes more smoothly when you have two joysticks.

### ***SceneMaker and SpriteMaker Tips***

\* If the program in the GameMaker Programming Area includes a scene and you leave the Editor and immediately load SceneMaker, the scene will already be loaded into the SceneMaker memory. If you have a picture in SceneMaker to which you would like to add text, you can include the instruction in your program, leave the Editor, go directly to SceneMaker, and your text will be printed on your scene. Then you can save that scene and remove the text commands from your program.

\* In SpriteMaker, you can clear a single frame by copying a blank frame over it.

\* When assigning sprites in a game, keep in mind that sprite 1 has the highest priority and sprite 8 has the lowest.

\* If you seem to be getting an impossible collision, keep in mind that sprites that are off the screen can still collide with each other.

\* In SceneMaker, to make multiple copies of an area, press the button twice after a **copy** to reset the same boundaries.

### ***MusicMaker Tips***

\* In MusicMaker, pressing the joystick button twice when entering a note will automatically advance you to the next available note.

\* When copying music from sheet music sources, it's best to enter one entire channel at a time rather than notes from all three channels at the same time.

\* MusicMaker can only enter sharp or natural notes; use this chart to convert flat notes to their sharp equivalents:

A flat = G sharp  
B flat = A sharp  
C flat = B  
D flat = C sharp  
E flat = D sharp  
F flat = E  
G flat = E sharp

\* Musical notes located at the same relative positions on the three musical staves are not in octaves. When entering notes with MusicMaker, refer to the musical notation chart located on page 104 to find the note values you need for your song.

\* In MusicMaker, "hidden" rests can be added to a song. When entering a rest in the editing column, press the joystick forward until the rest appears to be on top of the piano keyboard. Then press the button. The rest is placed under the piano keyboard. When the song is played, a rest will occur at that point but will not be shown. Hidden rests were used in many of the songs on side two of the GameMaker disk.

\*For those interested in a program to help learn basic elements of programming, may we recommend **The Designer's Pencil™** by Activision.

## **SAVING AND LOADING**

All of your designs and programs can be **SAVED** to diskette. It's a good idea to save your programs and designs frequently and especially any time you change them. Once you turn your computer OFF or leave any of the various parts of the GameMaker program, your programs or designs will be lost unless you save them. By **SAVING** you can later **LOAD** your program or design back into the computer at any time.

Saving and Loading work the same throughout the entire GameMaker program, whether you're saving or loading programs, sprites, scenes, sound effects or music.



To **save** a program or design:

1. Make sure you have an initialized disk in your disk drive with extra room on it. If you've forgotten to initialize your diskette, you can do it at this point by:
  - Selecting the **init** command.
  - Selecting **yes** to *erase all the files* on the diskette currently in the disk drive. Remember, a disk only needs to be initialized once unless you want to erase everything on it or use it for another computer.
2. Select the **file** command.
3. Select the **save** command. You'll see the message **save { }** in the Message Line. Type in a name for the program (up to six characters) and press the joystick button or RETURN. Remember to name your designs or programs with unique names. You can save a program with the same name as many times as you want, but each time you save a program with the same name, the new file REPLACES the old one.
4. Select **yes** to save the file.
5. Select **no** to either change the file name or cancel the command.

**NOTE:** *You are not allowed to save any programs or games on the GameMaker disk. To save your games, designs or programs, insert one of your own disks (or the blank disk included in the GameMaker package). Remember, the disk has to either have extra room on it; be a blank; be an initialized new disk, or be an old disk that has been re-initialized to store new files on it.*

To **load** a program or design from diskette:

1. Make sure the correct disk (or correct side of the disk) is inserted in your disk drive.
2. Select the **file** command.

3. Select the **load** command. You'll see the message **loading catalog**. When the catalog is finished loading, move the joystick to cycle through the available list of names until you reach the one you want.
4. Press the button again.
5. Select **yes** to load the file.
6. Select **no** to either change the filename or cancel the command.

### ***Printing Your Program***

GameMaker allows you to PRINT a listing of your programs or designs. Printers that you can use with GameMaker are:

MPS 801  
MPS 802  
Commodore 1526

You may also use most other printers compatible with Commodore BASIC.

To PRINT:

- Point to the **prnt** command in the GameMaker Editor. Make sure your printer is READY and then press the joystick button. Your program will start printing.

### ***ERROR MESSAGES***

**Out of memory**—You have filled the memory area in the computer. The **free** number at the top of the screen should warn you when you are getting close. The total memory available at the beginning of a programming session (with nothing on your GameMaker Editor screen) is 3553.

**Out of labels**—You have used all 255 labels available.

**Too many files**—You have attempted to load more than 255 data files (sprites, sounds, songs and pictures).

**File too large**—The data file that you are attempting to load will not fit into the available memory. If you are trying to replace an already existing file, you might fit the new one into memory if you first delete the instructions that access the old one.

**Too many if thens**—You have more than 128 **if** . . . **then** combinations in your program. See the **skip next** instruction on page 87 for suggestions on how to avoid this.

**If then logic error**—The **if** statement displayed at the top of the screen either does not have an **endif** or else has two **otherwise**s. Each **if** must have an **endif**.

**Device not present**—A disk drive or printer is not attached or the power is not turned on.

**Write protect on**—The disk that you are attempting to write on is write-protected. (You can remove the write protect “tab” to get around this.)

**Disk full**—No more room on your disk. Delete some files or use another disk.

**File not found**—You probably have the wrong disk in the disk drive. If you were trying to get another part of the program, put the **GameMaker** disk in the drive (Side 1 up) and try again. If you just loaded a program, a picture needed for that program is probably not on the disk. Find the correct disk and go to the **scene is { }** instruction. Select the name of the picture from the catalog list to load it in. It is smart to keep the pictures on the same disk as the program in which they are used.

**Read Error**—Your disk has gone to disk heaven or it was never initialized. Remember, if you initialize a disk, all the files are erased, so make sure it is one that is expendable.

If you receive other I/O (input/output) errors, they usually pertain to the disk drive or printer. Descriptions for the majority of these errors are contained in your disk drive or printer manuals.

## **GAMES, SCENES, SPRITES, SOUNDS AND MUSIC INCLUDED ON YOUR GAMEMAKER DISK**

Included on Side 1 of the GameMaker disk are:

### **Games:**

pitfal -Pitfall!™  
choper -Chopper™ by John Van Ryzin  
archer -an archery game

### **Scenes:**

archer -an archery field scene  
jungl1 -a jungle scene  
jungl2 -another jungle scene  
shore1 -a seashore scene  
shore -another shore scene

Included on Side 2 of the GameMaker disk are:

### **Games:**

brthday -a birthday greeting card. Change the name in the print command to personalize it for friends and relatives.  
doggie -a dog racing game  
dpoker -a draw poker game  
grfity -a graffiti game in which you can paint on a cement wall  
hliday -a "Happy Holidays" animation program  
manger -an animation program featuring a Christmas Nativity scene  
megama -*MegaMania*™  
sphere -an animation program featuring electronic music and multicolored spheres  
textil -a video graphics illustration

### **Scenes:**

brthday -a birthday greeting scene  
dpoker -a poker slot machine scene  
firplc -a fireplace scene

manger	-a Nativity scene
mega1	-a scene used in <i>MegaMania</i>
mega2	-another scene used in <i>MegaMania</i>
racpic	-a racetrack picture
sidewk	-a sidewalk picture
space	-a space scene
store	-a store scene
wall	-a cement wall

### **Sprites:**

arrow	-an arrow
ball	-a ball
biplan	-a biplane
boat01	-a boat
bombs	-a bomb
boom	-a <i>BOOM!</i> message
brickw	-a brick wall
bug	-a beetle
bug1	-a ladybug
bug2	-a wasp
candle	-a burning candle
car	-an automobile
carier	-an aircraft carrier
clam	-a clam
ctitle	-the Chopper title
dog	-a dog
duck	-a duck
explo1	-an explosion
fire	-a glowing fire
firlog	-a burning log
gamovr	-the message GAME OVER
gator	-an alligator
godzla	-gorilla
goldb	-a gold brick
gun	-a pistol
harry	-Pitfall Harry™
harryj	-Harry jumping
harryrr	-Harry running

helio1	-a helicopter
helio2	-a moving helicopter
horse	-a horse
hotair	-a hot air balloon
leye	-a left eye
log	-a giant log
longbw	-a longbow
madbmr	-the "mad bomber"
man	-another man
man1	-man walking right
man2	-man walking left
megas1	-all are alien spacecraft
through	"
megas8	"
misla1	-a missile
moneyb	-a money bag
plan01	-a plane
quasar	-an exploding quasar
rabrac	-a merry-go-round rabbit
reye	-a right eye
ring	-a diamond ring
robot	-a robot
santac	-Santa Claus
scorepr	-a scorpion
ship	-a ship
shot	-a speeding shot
silverb	-a silver brick
snake	-a snake
target	-an archery target
toydog	-a toy dog
worm	-a worm
zeplin	-a zeppelin
zsoldr	-a soldier

### **Sounds:**

clicks	-clicking sound
conts	-an alien sound
demo1	-a robot sound

drop1	-a dropping sound
explo1	-an explosion
fade2s	-a high-pitched tone
fade3s	-a high-pitched "sweeping" sound
fuels	-a "taking off" sound
haryds	-a "doomsday" sound
haryfs	-a "falling" sound
harygs	-a "charge" sound
haryjs	-a "jumping" sound
haryys	-a Tarzan "jungle call" sound
helio1	-a chopper sound
hitbrs	-a "pulsing hit" sound
hitsnd	-a hit sound
hitwl	-a pulsing sound
music	-a short tune
rabits	-a short upward-sweeping tone
sailor	-a short sailor tune
sho2s	-a shooting sound
sho3s	-a "muffled" shooting sound
siren	-a siren sound

### **Music:**

anthem	-The Star Spangled Banner
chief	-Hail to the Chief
danube	-The Blue Danube
dorami	-DO RE MI
fellow	-For He's a Jolly Good Fellow
glory	-Battle Hymn of the Republic
happyb	-Happy Birthday to You
jingle	-Theme from Pressure Cooker™
marine	-Marine Corps Hymn
merry	-We Wish You a Merry Christmas
pomp	-Pomp and Circumstance
race	-Racing Theme
rowrow	-Row, Row, Row Your Boat
silent	-Silent Night, Holy Night
song	-a traditional melody
sphere	-an original Activision tune
wittel	-The William Tell Overture
















## COLOR CHART FOR SCENEMAKER/SPRITEMAKER

The following is a chart of all the colors you can use in the SceneMaker and SpriteMaker programs.

0 black	4 purple	8 orange	12 medium grey
1 white	5 green	9 brown	13 light green
2 red	6 blue	10 light red	14 light blue
3 cyan	7 yellow	11 dark grey	15 light grey

## CHART OF AVAILABLE MUSICAL NOTES

The following notes are available for writing musical scores with MusicMaker.

 WHOLE NOTE	 WHOLE REST	 HALF NOTE	 DOTTED HALF	 HALF REST
 QUARTER NOTE	 DOTTED QUARTER	 QUARTER REST	 EIGHTH NOTE	 DOTTED EIGHTH
 EIGHTH REST	 SIXTEENTH NOTE	 SIXTEENTH REST	 THIRTY SECOND NOTE	 THIRTY SECOND REST

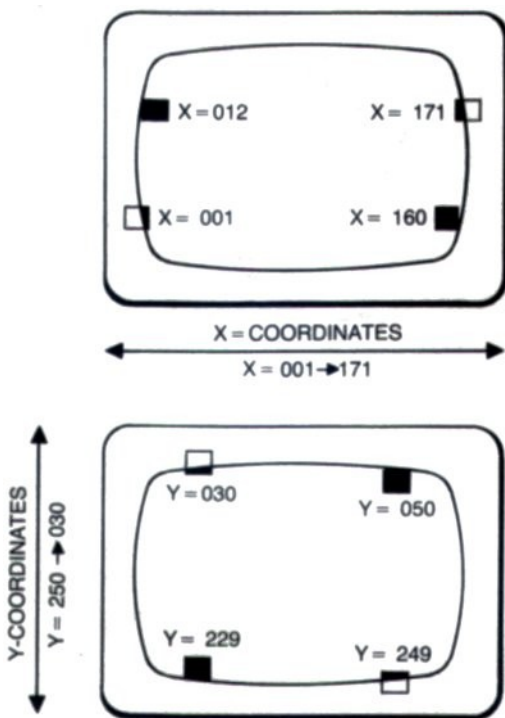


## CHART OF AVAILABLE MUSICAL INSTRUMENTS

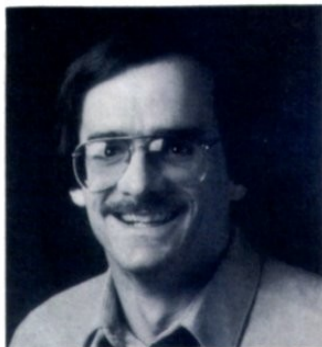
The following musical instruments can be used in any voice when arranging music with **MusicMaker**.

bass	saxophone
cowbell	snare
cymbal	synthesizer
flute	trumpet
guitar	violin
harpsichord	xylophone
piano	

## GameMaker POSITIONING COORDINATES



*Garry Kitchen*



## **CREDITS**

Garry Kitchen's GameMaker  
Conceived and Designed by Garry Kitchen  
SoundMaker and MusicMaker developed by Alex DeMeo  
Original graphics by Hilary Mills  
Musical Arrangements by Alex DeMeo and Stephen Gaboury  
Produced by Brad Fregger  
Manual Written by Teddi Converse  
"Happy Birthday to You" by Mildred S. Hill and Patty S. Hill  
© 1935 Summy-Birchard Music Division of Birch Tree Group, Ltd.  
Used by permission

Thanks to Jim Charne, Dan Kitchen, John Van Ryzin and Peter Patel  
for their contributions.

## **LET'S GET TO KNOW EACH OTHER**

We're working hard to design the kind of software you want, and we'd love to hear your comments. Drop us a note. We'll put you on our special mailing list. Also, if you'd like to find out about our newest computer software, call 800-633-4263 ANYTIME ON WEEKENDS.

In California, call (415) 940-6044/5 (WEEKDAYS ONLY).

Consumer Relations  
Activision, Inc.  
P.O. Box 7287  
Mountain View, CA 94039

## **ACTIVISION LIMITED 90-DAY WARRANTY**

Activision, Inc. warrants to the original consumer purchaser of this computer software product that the recording medium on which the software programs are recorded will be free from defects in material and workmanship for 90 days from the date of purchase. If the recording medium is found defective within 90 days of original purchase, Activision agrees to replace free of charge, any product discovered to be defective within such period upon receipt at its Factory Service Center of the product, postage paid, with proof of date of purchase.

This warranty is limited to the recording medium containing the software program originally provided by Activision and is not applicable to normal wear and tear. This warranty shall not be applicable and shall be void if the defect has arisen through abuse, mistreatment or neglect. Any implied warranties applicable to this product are limited to the 90-day period described above.

If the recording medium should fail after the original 90 day warranty period has expired, you may return the software program to Activision, Inc. at the address noted below with a check or money order for \$7.50 (U.S. currency), which includes postage and handling, and Activision will mail a replacement to you. To receive a replacement, you should enclose the defective medium (including the original product label) in protective packaging accompanied by: (1) a \$7.50 check, (2) a brief statement describing the defect, and (3) your return address. EXCEPT AS SET FORTH ABOVE, THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, WHETHER ORAL OR WRITTEN, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, AND NO OTHER REPRESENTATIONS OR CLAIMS OF ANY NATURE SHALL BE BINDING ON OR OBLIGATE ACTIVISION. IN NO EVENT WILL ACTIVISION BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGE RESULTING FROM POSSESSION, USE OR MALFUNCTION OF THIS PRODUCT, INCLUDING DAMAGE TO PROPERTY AND, TO THE EXTENT PERMITTED BY LAW, DAMAGES FOR PERSONAL INJURY, EVEN IF ACTIVISION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS AND/OR THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS AND/OR EXCLUSION OR LIMITATION OF LIABILITY MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

---

### **WARNING**

Any attempt to duplicate this product may damage it. Such damage is not covered by the warranty. U.P.S. or registered mail is recommended for returns. Please send to:

#### **WARRANTY REPLACEMENTS**

Consumer Relations

Activision, Inc.

2350 Bayshore Frontage Road

Mountain View, CA 94043

### **COPYING PROHIBITED**

This software product is copyrighted and all rights are reserved by Activision Inc. The distribution and sale of this product are intended for the use of the original purchaser only and for use only on the computer system specified. Copying, duplicating, selling or otherwise distributing this product without the express written permission of Activision are violations of U.S. Copyright Law and are hereby expressly forbidden.



