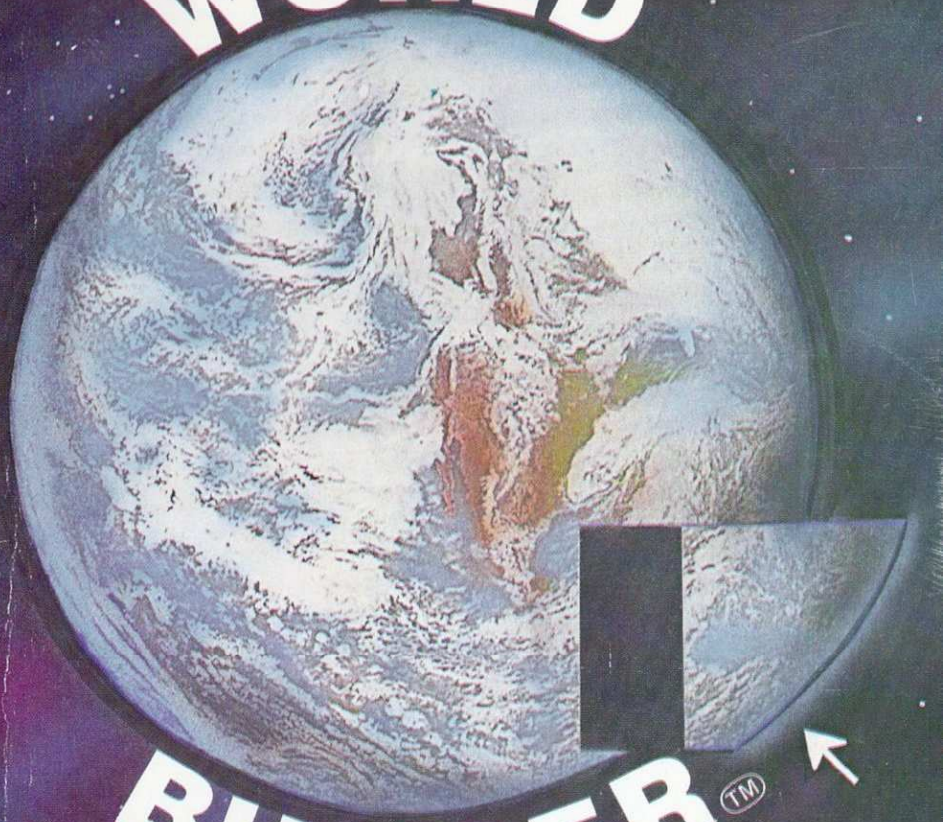
WORLD BUILDER™



SILICON BEACH SOFTWARE™ Inc.

CREATE YOUR OWN INTERACTIVE PROGRAMS

For the Macintosh

Copyright

Warranties

Silicon Beach Software, Inc. warrants the disk to be free of defects for 90 (ninety) days from the purchase date. If the disk is found to be defective in this 90 day period, return it to Silicon Beach Software for a free replacement. After 90 days, return it with $5.00 for a replacement. To obtain this warranty, you must send in the registration card.

World Builder™

Programmed by W. C. Appleton

Manual by W. C. Appleton and Charlie Jackson

Special thanks to:
Bill Atkinson, Mari Hughes
Robert Eberhardt, Richard Kaapke

Silicon Beach Software, Inc.
P.O. Box 261430
9770 Carroll Center Road, Suite J
San Diego, CA 92126
(619) 695-6956

Table of Contents

Preliminaries

Making Backup Disks

Before using World Builder, make a backup copy of the World Builder master disk. Put the original away in a safe place and use the copy. The World Builder disk is **not** copy-protected, so you may make backups from the Finder. If you're unsure how to do this, consult the user's guide that came with your Macintosh. Remember that **backups are for your use only.** Please do not copy this program for your friends, no matter how much they beg you. Have them buy their own copy.

If you use a sector-copy program to make your backup, you will continue to get the message that you're using your master. Use the Finder to make your working copy of the disk and you will not get the message anymore.

Becoming a Registered Owner

Please take the time to fill out the enclosed owner registration card. The World Builder disk is warranted to be free of defects for a period of 90 days from the purchase date. Please see the registration card for warranty details.

Also, if World Builder is updated for any reason, your name will be on file and you will be notified.

Commercial Use of World Builder

World Builder creates stand-alone applications. Although World Builder itself will not run on a 128K Macintosh, the programs created by it will.

There are no restrictions on the use of programs created with World Builder. It is your choice what you do with them. You may give them away, release them as Shareware, or even sell them. If you are going to sell one and you want to lock your program such that it cannot be altered with World Builder, contact Silicon Beach Software, Inc. for assistance.

Silicon Beach Software, Inc. cannot guarantee that programs created with World Builder will be compatible with future Macintosh architectures.

Playing Demo World

The application Demo World is a simple adventure game that has been created with World Builder. It consists of nine scenes and contains a couple of nefarious characters that you'll meet sooner or later. We've included it to show you what World Builder can do and how a World Builder game is played. After trying this game, you can use it as a springboard to build a more complex adventure. To begin Demo World, simply double-click on its icon.

Two Windows

When Demo World opens, two windows appear: the **scene window** shows where you are, and the **text window** describes what is happening. If the text window is full, you can scroll the text up or down by pressing the scroll bar.

You explore a World Builder adventure by selecting menu items, clicking on things in the scene window, or entering text commands.

Five Menus

The menus contain frequently used commands that allow you to move around, take a look inside your pack to see what items you're carrying, use weapons, and so forth. Try all the commands in each of the menus to see what they do.

There are five menus: Apple, File, Edit, Commands, and Weapons. The desk accessories are under the Apple menu. Most all accessories work with World Builder, but we recommend not using those that take up a lot of memory.

File Menu -- The File menu allows you to begin a new adventure, open an old adventure, close the current game, save the current game, revert to the last saved version, or quit.

Edit Menu -- You can cut, copy, or paste text with the Edit menu. Use this menu, for example, to capture an important passage of text, and place it in the Scrapbook for later retrieval.

Commands Menu -- The Commands menu contains frequently used directives. If you choose "North", "South", "East", "West", "Up", or "Down" then you will go in that direction, if possible. The "Look" command describes the current scene. The "Rest" command passes time and restores strength. The "Status" command describes the condition of you and your armor. The "Inventory" command lists the contents of your pack. "Open" and "Close" can be used to operate doors or boxes.

Weapons Menu -- The Weapons menu is used during battle. All of the weapons at your disposal are included in the Weapons menu. If a hostile character appears, choose a weapon (quickly!) to fight back.

Objects

Demo World contains two objects that you find by **Searching** different rooms. Clicking on an object has different results depending on the type of object. In Demo World, clicking on an object adds that item to your pack. In games you create, clicking on objects can lead to a totally different result. It's up to you.

Keyboard Entry

The keyboard is used to enter commands not included in the menus. After typing in a command, press the Return key to execute it. Here are some examples of some of the commands used in Demo World. A complete list of commands can be found at the end of this chapter.

Get and **Drop** -- If you want to add an object to your pack, or drop an object, type "Get" or "Drop" followed by the name of the object (don't type in the quotes, of course). Example: If you see a laser gun on the ground, type "Get Laser"; to drop the gun, type "Drop Laser."

Offer -- Angry characters can be bribed with the "Offer" command. If a giant monster attacks, type "Offer Laser" in hopes that the monster will accept the offer and go away.

Aim -- The "Aim" command lets you aim your attack at the head, side, or chest. Type "Aim Head" to direct your attack at the most vulnerable area.

In addition to the standard commands, there may be special commands that are only appropriate for a certain scene, character, or object. In another game, for example, if you find a lantern you might type "Light Lantern." Or if you find a book you might type "Read Book." The use of special case commands is up to you when you write your adventures.

Fighting Characters

Sooner or later you will meet a hostile character. If a fight gets started you can do four things:

- Fight with a weapon;
- Cast a magical spell;
- Offer any object you own in exchange for peace;
- Try to run away.

All of the weapons and spells that you have in your possession are listed under the Weapons menu. To fire your laser gun, for example, choose the item "Fire Laser" (or type it from the keyboard). If you think your opponent is greedy and you have something of value, offer it. If your attacker looks a bit slow you could try to run in an open direction by choosing the commands "North", "South", etc. from the Commands menu.

Summary of Standard Commands

NORTH	Moves you to the north, if possible
SOUTH	Moves you to the south, if possible
EAST	Moves you to the east, if possible
WEST	Moves you to the west, if possible
UP	Moves you up, if possible
DOWN	Moves you down, if possible
LOOK	Describes the current scene
REST	Restores strength and passes time
STATUS	Indicates your condition
INVENTORY	Lists the contents of your pack
OPEN	Opens a door or a box, if possible
CLOSE	Closes a door or box, if possible
GET *object*	Adds an object to your pack
DROP *object*	Drops an object from your pack
WEAR *armor*	Exchanges one piece of armor for another
OFFER *object*	Offers the given object in exchange for peace
AIM *body part*	Aims a weapon for the head, chest, or side
op-verb weapon	Uses the given weapon

Remember to try special commands specific to a given object, character, or scene!

Introduction

World Builder is primarily a game creation tool, but it can also be used for many other applications, such as training materials, storyboarding, etc. The manual takes the approach that you are creating games, in order to show all its capabilities.

If you will be using the program for non-game applications, try to create a small game first. Going through Chapter 1, which is a step-by-step tutorial, should be sufficient. Having done this, you'll understand how programs are created. You can then adapt these techniques to your particular needs.

Welcome to a Brave New World

Have you ever wished you could stop the hero of a novel from making a terrible mistake? Or perhaps you're the sadistic sort and wish that some of the less lovable characters would make a few more mistakes than they do. Computerized role-playing interactive fiction games put you in the driver's seat; you get to make all the decisions. When it comes time to choose a course of action, you take full command. The computer keeps track of your moves, accompanying you on your quest as a combination storyteller, scorekeeper, confidant, and referee. Interactive fiction games combine the intelligence and speed of the computer with the wit and entertainment of a good novel. It's no wonder they're so popular.

If you enjoy **playing** interactive fiction games, imagine **writing** your own. World Builder lets you do just that. This powerful program provides everything you need to make your own professional-quality stand-alone adventure games on the Macintosh. World Builder games combine text, graphics, even sound to create realistic and exciting adventures. Most of the difficult programming chores are done for you automatically, so you can concentrate on the more important aspects of game writing -- creating interesting characters, plot, and action.

World Builder is more than a simple game maker, it's a serious programmer's tool. With it, you can develop games that rival even the best commercially available adventures. Players of your World Builder adventures can:

- Explore up to 2,500 different scenes.
- View each scene in a detailed graphics window.
- Encounter any number of characters or objects.
- Battle monsters.
- Hear realistic digitized sounds while exploring or fighting.
- Save unfinished games for completion later.
- Plus lots more.

The Basics of Creating Your Own Adventures

World Builder is used to generate the scenes, characters, and objects that your planets will need. Unlike the normal Macintosh application that creates documents, World Builder generates stand-alone adventure game applications. Two examples of World Builder games are Demo World (included with World Builder), and Silicon Beach's Enchanted Scepters.

The application World Template is an adventure game just like Demo World except that it contains no scenes, characters, or objects. World Template is used as a source of new, empty programs that can be edited with World Builder.

Step One -- Design the Game

The first step in designing a new adventure is to decide what kind of people, places, and things exist in your world. World Builder can be used to create games from any place or time period. You might consider themes such as outer space, modern times, World War II, the wild west, knights in shining armor, dinosaurs, gangsters, monsters, or fantasy.

Step Two -- Populate the World

The next step is to draw the scenes, characters, and objects that you will need in your world. These things can be drawn in World Builder or brought in over the clipboard from other applications like MacPaint or MacDraw. You might want to copy a few of your scenes, characters, or objects from other World Builder adventures like Enchanted Scepters.

Step Three -- Define Characteristics

After this, you decide how your planet -- the scenes, characters, and objects -- behave by filling in a few dialog boxes. Some scenes will only allow movement in a certain direction. Different characters can be placed at different locations, and be given particular strengths, weaknesses, weapons, and strategies. There are various types of objects, and each type has different characteristics. Some objects are simple props, like a candle or a book. Other objects can be used as weapons or magical spells. You may also specify an object to be a piece of clothing or armor. All objects have value, and can be offered to angry characters in exchange for peace.

Step Four -- Play the Game

Once you have drawn some scenes, characters, and objects, and filled in the dialog boxes, you have done everything necessary to create an exciting adventure game! Try out the game to see how it plays. Give it to friends and see how they like it. Sell it! It's up to you.

Getting Used to World Builder

The best way to get started using World Builder is to go through Chapter 1. This chapter contains a step-by-step tutorial on how to create a complete game. It introduces you to the fundamental aspects of World Builder. After that, read the following chapters that are reference sections covering all the details of World Builder. You may want to examine Demo World with World Builder to get some ideas on how a planet is put together. If you've purchased Enchanted Scepters, be sure and take a look through it with World Builder, as it is full of detailed examples of how to do things. Be creative! Your world is at your fingertips!

About This Manual

This manual consists of a tutorial chapter (Chapter 1) and five reference chapters. Chapters 2 through 6 expand upon the information presented in the tutorial and act as reference guides for different aspects of word building.

Here's a quick rundown of what you'll find in each one:

Chapter 1 -- Using World Builder. A step-by-step guide to creating your own world.
Chapter 2 -- Creating New Worlds. Designing scenes.
Chapter 3 -- Populating Your World. Adding characters and objects.
Chapter 4 -- Using the Graphics Editor. Drawing graphics for scenes, characters, and objects.
Chapter 5 -- Writing Scene Code. Enhancing your games with adventure code.
Chapter 6 -- Advanced World Building. Using sound, advanced features of adventure code.
Chapter 7 -- Working with Sound. All about digitized sounds.

Appendix A provides details on World Builder specifications (including Switcher considerations), and Appendix B provides a summary of adventure code variables and instructions.

Chapter 1
Using World Builder

This chapter is going to take you through the step-by-step creation of your first World Builder adventure, "Midnite Snack." The player's goal in Midnite Snack is to get something out of the refrigerator in the kitchen and eat it. But, between the bedroom and the kitchen lurks a Wuggly Ump that escaped from an Edward Gorey story. The player must be wearing a robe to get past the Wuggly Ump; if the Wuggly Ump hasn't been given a ball of twine to keep it happy, it steals all the food from the refrigerator and the player dies of starvation. As in all adventure games, the player isn't told about these twists, but is left to deduce or otherwise discover them.

The Midnite Snack world is a simple one that consists of:

4 scenes: Bedroom, Hallway, Kitchen, and Outside.
2 characters: the player and the Wuggly Ump.
2 objects: the robe and the ball of twine.

Getting Started

The first thing you have to do is make a "shell" in which you can build your world:

- On the DeskTop, duplicate the icon named World Template and name it "Midnite Snack."
- Open World Builder by double-clicking on it.
- Select and open Midnite Snack from the Open box that appears.

Four windows open on your screen: Scene Map, Character List, Object List, and Sound List.

Creating Scenes

Each scene, or location, in an adventure has four components that you control:

The Design: The picture that appears when the player enters the scene.
The Text: The words and descriptions that appear when the scene is entered.
The Data: Information about which directions are blocked, and which sounds are automatically linked to the scene.
The Code: The list of instructions that tells the program how to handle the player's actions in the scene.

Making a Map

First, you're going to map out the four scenes needed for Midnite Snack. Working in the Scene Window, which is at the top of the pile:

- Press the Create button.
  > A black box labeled Untitled appears in the window.
- Press the Name button; type "Bedroom" in the dialog box that appears.
  > The box in the window is labeled Bedroom.
- Press the Create button three more times.
  > Three more boxes appear in the window.
- Click in the middle box in the top row and press the Name button; name the scene "Hallway."
- Select and name the right box in the top row "Kitchen"; select and name the box in the second row "Outside."

The positions of the squares in the Scene Window correspond to the relative positions of the locations in the game; right now, the Kitchen is east of the Hallway, and Outside is south of the Bedroom. You can change scene positions by dragging the boxes with the mouse.

- Rearrange the boxes so Outside is alone in the top row, and the other three boxes are in a row beneath it, in their original order.

The player now has to go North to go out the window; two moves east bring her to the kitchen.

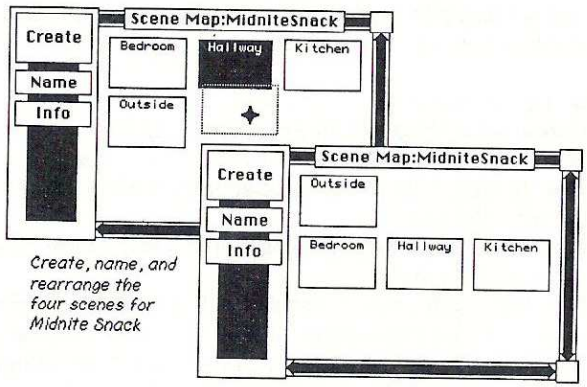


Fig. 1-1. Scene Map Window

Bedroom Design

With the Bedroom box selected, choose **Open Scene Design** from the Window menu. This is the window in which you create the picture for the scene. You can paste in a graphic that you've imported from another program, but you can also use the tools that World Builder supplies.

The Design window appears during the game exactly the way you create it in World Builder, including its size and placement. To move a World Builder window, drag it by its title; to re-size it, drag in the box in its lower right corner.

- Move and re-size the Design window so it takes up about two-thirds of the screen vertically.

To draw the bedroom with World Builder:

- Select **Rectangle** from the Tool menu, and draw a rectangle in the upper third of the Design window. When you release the mouse button, a black rectangle appears.

A graphic object can be moved or re-sized in the Design window as long as it is selected. A selected object has a frame around it that has small black squares in its corners. Drag on any of these corner handles to re-size the object; move the object by dragging it from within the frame.

- Select a pattern from the Fill menu to turn the black rectangle into a wallpapered wall.

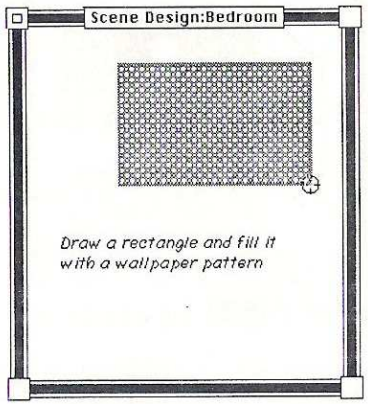


Fig. 1-2. Starting a Scene

- Select the **Rectangle** tool again. Draw a rectangle within the existing one to serve as a window; select **Black** from the Fill menu to turn it black.
- Draw a moon. Select **Oval** from Tool menu, and **White** from the Fill menu. Holding down the Shift key to make the oval a circle, draw the circle.

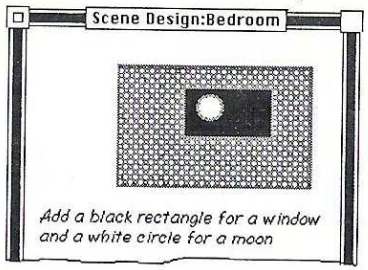


Fig. 1-3. Adding to the scene

- Draw the left wall with the **Polygon** tool from the Tool menu. Click on the upper left corner of the back wall; click again at a spot down and to the left to draw the ceiling line; click another line down to the floor. Place the cursor on the bottom left corner of the back wall, and double-click to complete the polygon. Select the wallpaper pattern from the Fill menu.

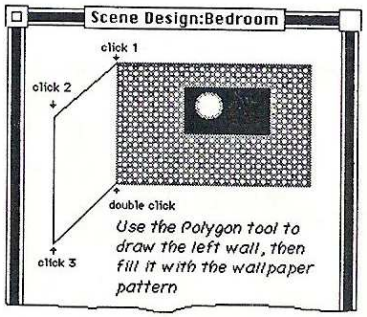


Fig. 1-4. Adding a Polygon

You can continue drawing the bedroom now or at another time, but that's enough to get the idea. Now it's time to create the text.

Close the Design window by clicking in the box in its upper left corner. A dialog box appears asking if you want to save the changes before closing; click Yes. Close the Design window, saving it by way of the dialog box alert.

Bedroom Text

Scene text is the text that appears in the game whenever the location is entered, and whenever the player uses the Look command in that location.

- Select **Open Scene Text** from the Window menu.
- Size the Text window to take up the remainder of the screen next to the Design window, and type the text as shown in Fig. 1-5.

You can use the Font, Fontsize, and Style menus to adjust the text the way you want it.

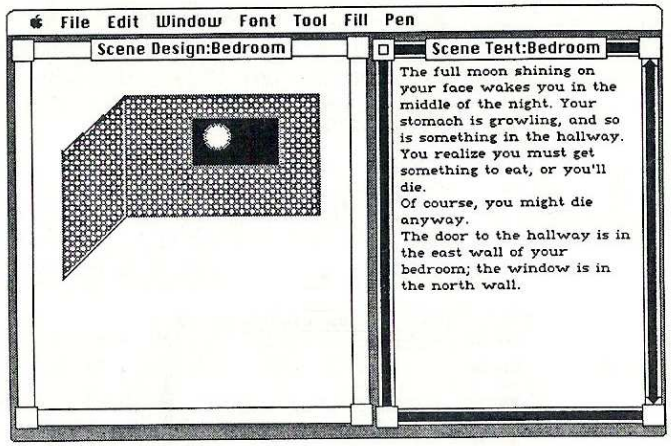


Fig. 1-5. Entering Text in the Scene Text Window

Bedroom Data

Data for a scene tells the game which directions a player is not allowed to move, and defines the sounds that will accompany the scene.

- Select **Open Scene Data** from the Window menu.
- Click in the South and West checkboxes to block those directions.

When a player tries to move in a blocked direction, the game will tell her "You can't go that way." If you want the game to say something else instead, you enter that information here.

- Type the comments as shown in figure 6.
- Click in the OK button to put the Data window away.

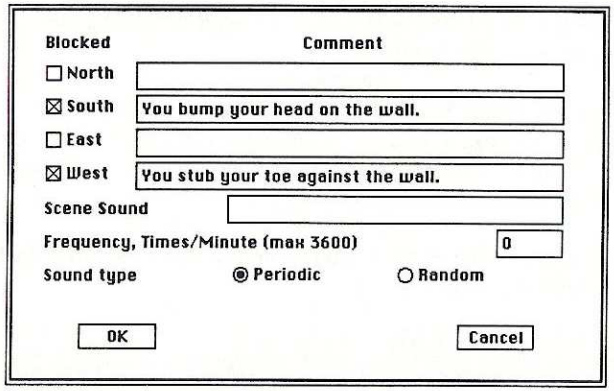


Fig. 1-6. The Scene Data dialog box

You'll find information about sound effects in Chapter 7; we're not going to put any in Midnite Snack. Notice the heavy lines on the blocked sides of the Bedroom box in the map that correspond to the information you entered in the Data window.

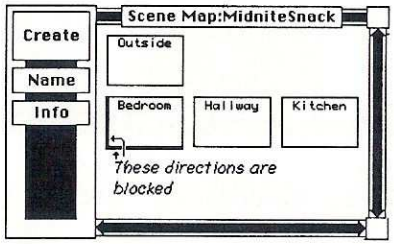


Fig. 1-7. Blocked directions

Bedroom Code

The code is the list of instructions that tells the game how to respond to a player's actions in a scene. If the player types a directional move, like North, the game automatically moves her to the appropriate scene. If she types "Swing Fist" and you've defined Fist as a weapon, the game will take care of the fight. You don't have to program these general activities.

You will, however, want to control a lot of what happens when the player does something in a scene. She may type a command in the text window, or select one from the Command window--the commands could range from eating something to casting a spell; or, she might click on an object that appears in the scene. These are all actions you'll want to respond to in a special way.

Midnite Snack allows only a limited number options; here in the bedroom, the only option besides leaving it is to pick up the robe or the ball of twine. (We'll be adding those objects to the scene later.)

Writing code in World Builder is programming; if you're familiar with BASIC, you won't have any trouble. If you're not familiar with BASIC, you still shouldn't have much trouble; BASIC is basic because it looks a lot like English.

- Select **Open Scene Code** from the Window menu.
- Enlarge the window to the width of the screen.
- Select the font and size you want to work with from their menus.
- Type the following in the Code window:

```
IF{CLICK$=robe}OR{TEXT$=Get robe}THEN
   PRINT{The robe is shabby, but you slip it on to ward off the
   night chill.}
  PRINT{With luck, it will ward off even more.}
EXIT
```

Make sure you put the braces where indicated and type in capitals where shown. World Builder will automatically indent the PRINT commands after you type the line. Don't leave any spaces between the command and the braces.

This piece of code tells World Builder what to do if the player clicks on the robe or issues the command "Get robe." CLICK$ stands for the action of clicking on an object; TEXT$ stands for whatever the player types or selects from the Command menu. In this example, if the object clicked on is the robe, or the phrase typed is "Get robe," World Builder will do everything listed before the EXIT statement--it will print the two statements.

You have the option in World Builder of changing what appears in the Command menu. The MENU statement is used to do this. For example, you might want to add "Get" to the Command menu for this scene. Normally "Get" works automatically in World Builder, but this would be a way of giving the player a strong hint or clue.

- Add this statement to the code window:
  MENU{Get/G}

The "/G" after the word adds the command key option to the menu, too. Once you use the MENU command in a scene's code, the normal Command menu is replaced with the commands you list. To put some of the standard commands back in the menu, you have to include them in your MENU statement.

- Change the menu statement to:

MENU{Get/G;North/N;South/S;West/W;East/E;Inventory/I}

One more important thing has to be added. If the player gets the robe, you'll want the program to keep track of the fact that the robe is now in her possession, and remove it from the bedroom scene. Since this can happen only if the player gets the robe, you must add another statement to the block of statements before the EXIT command.

- Click in front of the EXIT statement.
- Type:
  MOVE{robe}TO{PLAYER@}
- Press Return to move the EXIT statement to the next line.

The word PLAYER@ is always used to indicate the character in the game that represents the player. The MOVE command in this case moves the robe from wherever it is into the player's inventory of items.

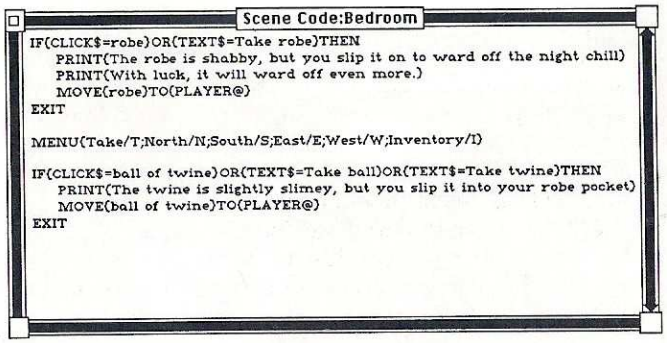


Fig. 1-8. Scene Code

- Close the Code window, by clicking in the box in its upper left corner.
  > A dialog box appears asking if you want to save the changes.
- Click in the Yes button.

If you have made any mistakes--put in extra spaces, or forgot to use capitals--you'll get an alert letting you know there's been a mistake, and what line it's in; fix the mistake and try closing the window again.

- Close the Design and Text windows; click Yes to save the changes.

Now, on to the other scenes.

The Outside Scene

If the player goes out the window, she falls to her death. The graphics are simple, since it's dark out, and the text lets the player know what happened.

- Double-click in the Outside box in the Scene Map; this opens both the Text and the Design windows.
- Set up the windows as shown in Fig.1-9.
- Open the Code window and type:
  MOVE{PLAYER@}TO{STORAGE@}

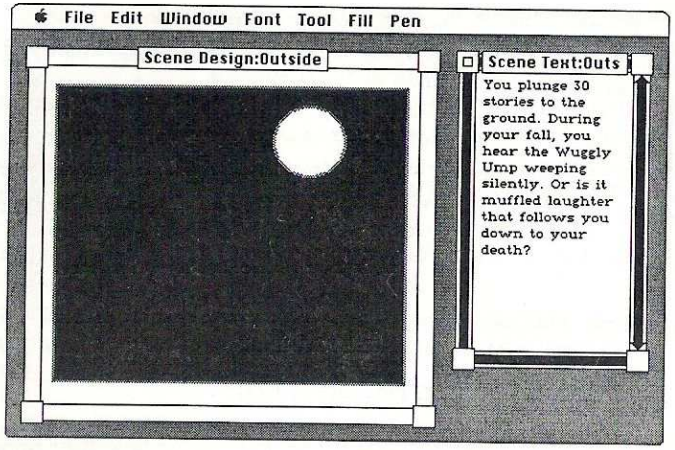


Fig. 1-9. Outside

STORAGE@ is another special World Builder word. When you move characters or objects to storage, they go into a temporary oblivion from which they can be recalled when you need them; this way, a player won't encounter the character or object as she wanders around the world. When you move the player character to storage, however, the oblivion is permanent, and the game ends.

Since moving outside ends the game, there's no need to set up a Data window for the Outside scene. Close all the windows, saving their information. On to the hallway!

The Hallway Scene

Setting up the Hallway design is simple, since the lights are off.

- Open the Design and Text windows and set them up as shown in Fig. 1-10.
- Close and save the Design and Text windows.

Don't worry about drawing in the eyes mentioned in the text; the Wuggly Ump will be designed separately and placed in the Hallway later.

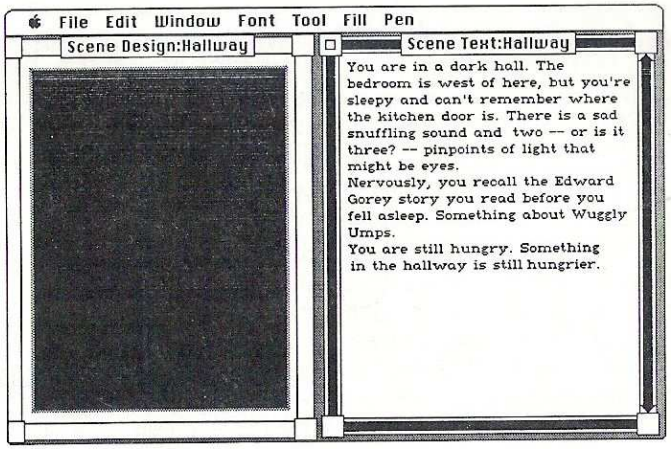


Fig. 1-10. The Hallway

- Open the scene Data window.
- Block the north and south paths, and type the comments as shown in Fig. 1-11.
- Click the OK button to put the Data window away.

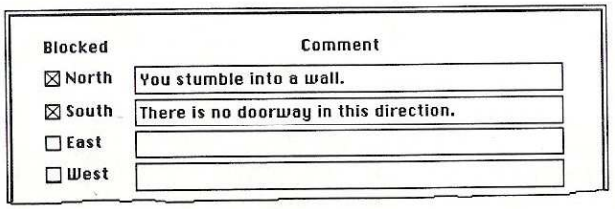


Fig. 1-11. Blocking directions

The hallway code has to take into account that unless the player has her robe, she can't do anything at all--except go back to the bedroom and get the robe. The only action allowed will be the command "west."

- Open the scene Code window.
- Type in the Code as shown in Fig.1-12.
- Save the code when you close the window.

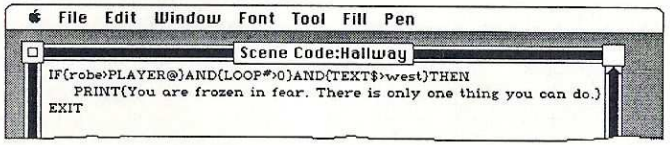


Fig. 1-12. Scene Code for the Hallway

There are three things that this piece of code checks: the player's possession of the robe, if she has issued a command in this scene, and the command that she uses.

To check if a character owns an object, the phrase "*object* = *character* " is used. To check if a character doesn't own an object, the phrase "*object* > *character* " is used. Since we want the PRINT statement executed only if the player doesn't own the robe, the first phrase, or condition, in the IF statement is "robe > PLAYER@."

The second condition in the IF statement has to do with whether or not the player has given a command yet. We don't want the "frozen in fear" comment printed unless she tries to do something. The word LOOP# stands for the number of commands a player has used in a scene. If LOOP# is zero, she hasn't done anything yet. Using {LOOP#>0} makes sure the number of commands is greater than zero; the ">" sign is used in its mathematical sense when it is used with numbers in the code window.

The third condition in this IF statement has to do with just what command the player uses; if she decides to go back to the bedroom, she will be allowed to do so without any statements regarding her emotional state. The phrase "TEXT$>west" means "the command does not equal west."

Using multiple phrases in an IF statement with AND between them tells the game to check that all conditions are true before executing the block of commands. There are three conditions here--all of which must be present--for the PRINT statement to be executed: the player does not own the robe, she has issued at least one command, and she has not used the command "west" (the bedroom direction).

In true adventure game tradition, the player is given no clue as to what she *can* do; she'll just be told she can only do one thing every time she tries the wrong command.

- Close the Code window and save the changes.

World Builder has a lot of built-in code to handle general commands. You don't have to write code for the hallway scene to take care of the ball of twine transfer. If the player uses the command "offer," the game takes over if you don't have specific code for the situation. Here in the Hallway, it will check the Wuggly Ump's character traits (which we'll define later) and calculate the likelihood of its accepting the twine. If it accepts, the game will inform the player, and the transfer of the twine from her possession to the monster's will also be taken care of automatically.

The Kitchen Scene

When the player reaches the kitchen, she'll want to open the refrigerator to get some food. According to the original scenario, she'll only find food there if she has given the ball of twine to the Wuggly Ump in the hallway.

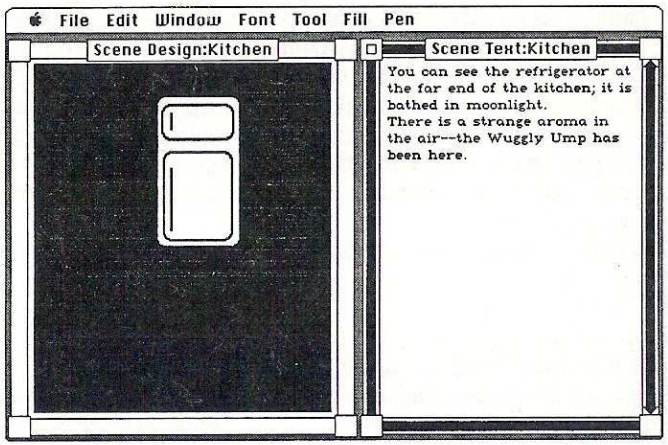


Fig. 1-13. The Kitchen

- Open the Design and Text windows for the Kitchen; set them up as shown in Fig.1-13.
- Open the Code window and type in the code shown in Fig.1-14.

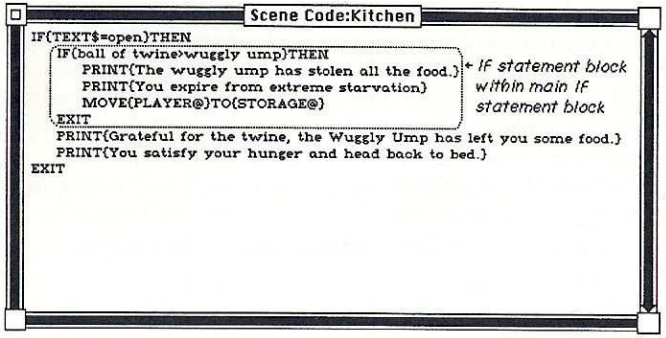


Fig. 1-14. Scene Code for the Kitchen

The code in the Kitchen scene must check for one major thing: the player's opening the refrigerator. After that happens, the game must do different things depending on whether or not the player has given the Wuggly Ump the twine.

The first IF statement in the code checks if the player's command is "open." (In a complete adventure, you'll want to be more specific, but for now, as long as the command contains the word "open", we'll assume the player said "open refrigerator.") If the player issues that command, the game looks at the rest of the code in the IF block.

Within the first IF block is another IF statement. If the Wuggly Ump does not own the ball of twine ("ball of twine > Wuggly Ump"), the game prints the bad news and ends the game by moving the player off to the storage area. Because this IF block is inside the main one, the game won't even check the Wuggly Ump's possessions unless the player has issued an Open command.

If the Wuggly Ump does indeed own the twine at this point, this inner IF block is skipped altogether, and the two PRINT statements are executed instead. Although this is the end of the Midnite Snack scenario, the player is not moved to storage -- she can go on exploring this limited world.

- Close the Code window and save the changes.

Creating Characters

It's time to create the two characters needed in Midnite Snack: the Wuggly Ump and the player. Click in the Character List window to bring it to the front.

The Wuggly Ump

- Click in the Create button, then the Name button; name the character "Wuggly Ump."
- Double-click on the Wuggly Ump name to open the Design window.
- Use the **Oval** tool, with **White** from the Fill menu, to draw three tiny ovals--this is all the player will see of the Wuggly Ump.
- Close the Design window and save the changes.

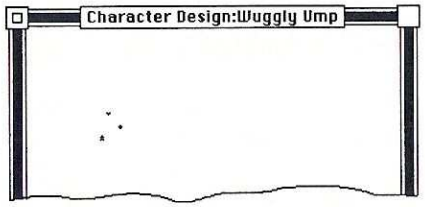


Fig. 1-15. The Wuggly Ump

Characters in World Builder are defined by attributes: how strong they are, how likely they are to fight, how open they are to offerings, and so on. These attributes are assigned through a series of dialog boxes.

- Select **Open Character Data** from the Window menu.
  >The first box appears.
- Fill in the information as shown in Fig.1-16.

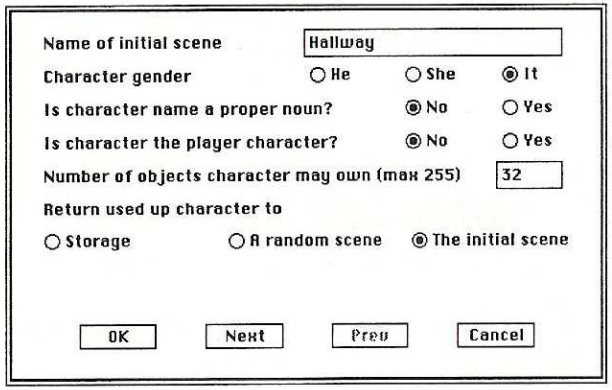


Fig. 1-16. The First Character Data dialog box

The first box contains basic character information. The Wuggly Ump's initial scene is the Hallway; its gender is neuter. (As far as we can tell.) You can leave the rest of the items in this box at their default settings, except the last. Click in the Initial Scene button so the creature stays put in the Hallway no matter what happens.

- Click the Next button to go to the next box.

The second box lets you set physical and spiritual characteristics of a character; the Wuggly Ump can use the default settings, so you don't have to change anything.

- Click the Next button to go to the next box.
- Click on the extreme left edge of each rating bar.

The thin rectangles partially filled with black are used to rate character attributes. Clicking in a bar changes the filled area to the click point; clicking on the left edge empties the bar. By setting these three attributes at the lowest point possible, you have made the Wuggly Ump extremely slow, very likely to accept any offer, and unlikely to follow the player around.

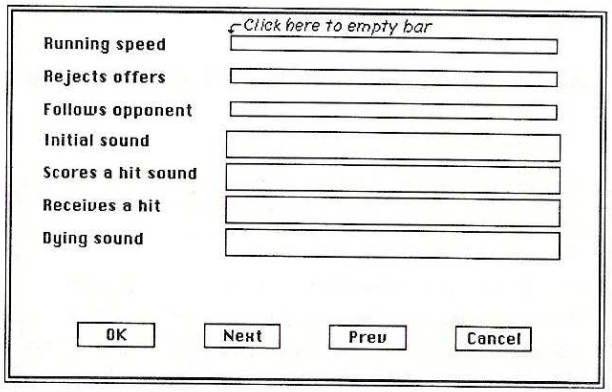


Fig. 1-17. More Character Attributes

- Click in the Next button. (We're not adding sound to Midnite Snack.)
- Erase the contents of all the edit fields in this box.
- Set both rating bars to minimum.

"Native weapons" are things like hands and feet, or claws and fangs. The Wuggly Ump is basically a passive and pacifist creature; not defining any native weapons for it will keep it from attacking the player when they meet.

- Click the Next button. Empty all the bars by clicking on the left edge of each one.

This box lets you define the character's tendencies to do something when she is winning or losing a fight. Since the Wuggly Ump won't fight, you can set these traits all to minimum.

- Click the Next button.
- In the first field, type:
  The Wuggly Ump is thinking about you.

This last box lets you fill in the text that will appear when a certain situation occurs. The initial comment when the player meets the Wuggly Ump is all that is needed.

- Click the OK button to close the box and save all the Character Data.

The Player Character

Every world must have a character in it who is the player's alter ego. You don't have to worry about the character design, because the player is seldom seen; however, you do have to set his or her attributes. (In a full adventure, these attributes can change as the game progresses--a character can become weaker, or more likely to run away, and so on.)

- Click the Create button in the Character window.
- Name the character "Sleepyhead."
- Select **Open Character Data** from the Window menu.
- Fill in the first four items of information as shown in Fig.1-18--set the gender at whatever you prefer.
- Click the OK button to close the box.

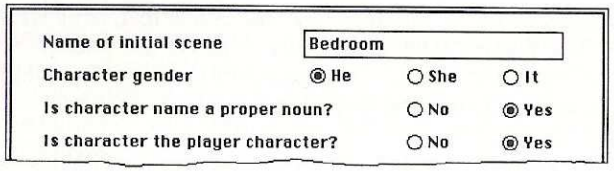


Fig. 1-18. The Player Character

The player character can use all the default settings for his character attributes, so there's no need to go through the other boxes.

Now, on to objects!

Creating Objects

Objects in World Builder can be of different types, and different types are assigned different attributes. Armor and weapons, for instance, need strength attributes assigned; magical objects need spell accuracy ratings.

Midnite Snack needs only two objects--the robe and the ball of twine. These are simple movable objects that need only a minimum of information entered.

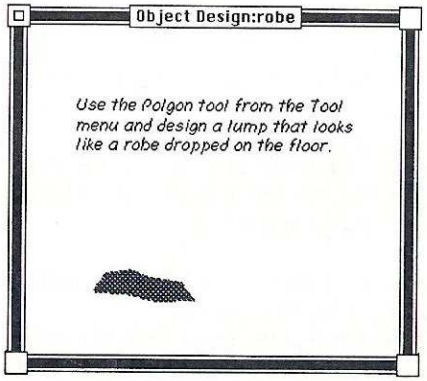


Fig. 1-19. The Robe

Creating the Robe

- Click on the Object window to bring it to the front.
- Click the Create button; click the Name button and name the object "robe"
- Double-click on the name "robe" to open the Design window.

- Use the **Polygon** tool to draw a lump that looks like a robe dropped on the floor. Select a pattern from the Fill menu.
- Close the Design window and save the changes.

Now, you have to assign a few attributes to the robe.

- Select **Open Object Data** from the Window menu.
- Type "Bedroom" as the initial scene. Leave all other attributes at their default values.
- Close the Data window, saving the changes.

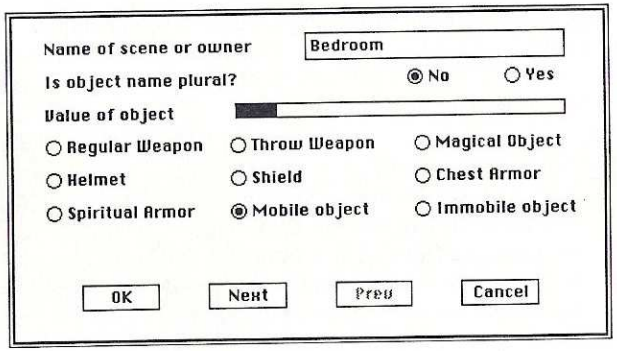


Fig. 1-20. An Object's Data dialog box

Creating the Ball of Twine

- Create and Name an object "ball of twine."
- Open the Design window and draw the twine: draw a circle with the **Oval** tool and fill it with a striped pattern. You can add a trailing string by selecting a striped pattern and a thick line from the Pen menu, then drawing with the **Freehand** tool.

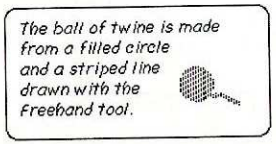


Fig. 1-21. The Ball of Twine

When you draw an object in a Design window, its position in the window controls where it appears in a scene during the game. To keep the robe and twine from appearing on the bedroom ceiling, make sure you draw them near the bottom of their Design windows.

- Close the Design window and save its changes.
- Select **Open Object Data** from the Window menu.
- Type "Bedroom" for the initial scene.
- Close the data box with the OK button.

Ending the Session

It may not have taken seven days, but you just created a whole world. Select **Quit** from the File menu. The Midnite Snack icon on the DeskTop is now a self-contained adventure game.

Exploring Your World

- Double-click on the Midnite Snack icon to start the game.
  > The initial bedroom scene appears, with the robe and the twine on the floor. The Command menu contains the commands you wrote in the Code window for this scene. Notice that the text refers to the robe and twine, although you did not code that.

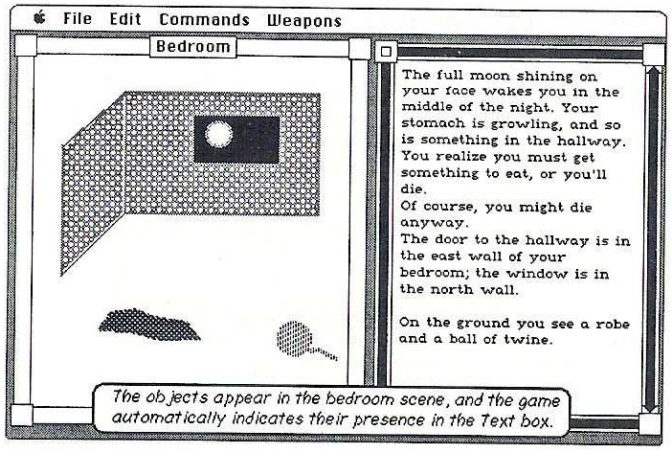


Fig. 1-22. Starting in the Bedroom

- Click on the robe.
  > The robe disappears (as a result of the "MOVE{robe}TO{PLAYER@}" statement you put in the code). In the text window, the text you specified is printed.
- Click on the twine.
  > The twine disappears, and the text window describes it.
- Type "east"
  > You are shifted to the Hallway scene. The Wuggly Ump's eyes are superimposed on the background. The "encounter statement" is put in by the program; the last statement is the one you entered in the character data for the Wuggly Ump.

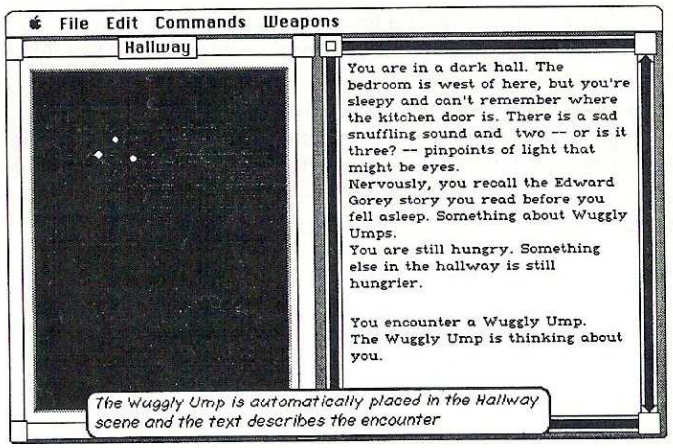


Fig. 1-23. In the Hallway

- Type "east"
  > You are moved to the Kitchen.
- Type "open fridge"
  > Since you didn't give the twine to the Wuggly Ump, you just died of starvation.
- Click OK in the Game Over box and select **New** from the file menu.
  > You're back in the bedroom again, at the start of a new game.
- Pick up the robe and twine, and go east into the hallway.

- Type "offer ball of twine"
  > The game responds "Your offer is accepted" and the Wuggly Ump disappears.
- Type "east"
  > Now you're in the Kitchen
- Type "drop robe"
  > This is just to see what happens. The robe appears on the kitchen floor, even though you did no coding for this situation.
- Type "open fridge"
  > This time you are saved from starvation.

That's the game as you designed it, but there's more to it than you wrote yourself, as you saw when you dropped the robe. Just for fun, pick up the robe and go back to the Hallway (west); the Wuggly Ump is probably there with his ball of twine. See what he does with it.

Try some of the things that you did write code for: try jumping out the window, or going into the hallway without the robe. (Remember that you wrote code that only lets the player go back to the bedroom to get the robe. If you leave the robe in the kitchen, and move into the hallway, you'll never get out.)

Chapter 2
Creating New Worlds

World Builder can be used to modify existing adventure games or to create new adventures. World Builder features:

- A graphics editor to draw the scenes, characters, and objects.
- A text editor to describe the scenes in words.
- A series of dialog boxes to create or change the characteristics of your world.

This chapter explains how to use these various features of World Builder.

Editing Existing Worlds

If you want to work on an existing adventure game, such as Demo World, open World Builder and choose the file you want to edit in the dialog box that appears (if the game is on another disk, click the Drive button or insert the new disk into the Mac's drive). Click Open.

Creating New Worlds

If you want to create a new world, make a duplicate of the World Template icon. Give it any name you like. Start World Builder and choose your new world in the dialog box that appears. World Template is locked so that it can't be renamed, thrown away, or edited directly with World Builder.

Note: When you are done editing, you must **close** all the edit windows and select "Close" or "Quit" from the File menu. If you just eject the disk or turn the machine off, your adventure game will not be saved.

Caution: You should **ALWAYS** keep a backup copy of the world you are editing or creating on a separate disk.

To create a new scene, first open your adventure game with World Builder as described above. Four windows appear:

- Sound List
- Character List
- Object List
- Scene Map

The Scene Map exhibits the relative location of all the scenes in the current world. In the example in Fig.2-1, the "Shack" is west of the "Rocky Slope", and the "Forest Path" is north of the "Forest Clearing." The scroll bars on the right and bottom of the window can be used to scroll around on the map. The total area provided is 50 by 50 scenes for a maximum world size of 2500 places. All of the scenes wrap around, so if a character moves off one side of the map, he'll arrive at the other side.

Making A Scene

Press the Create button and a new "Untitled" scene will appear on the map. The Create button is inactive if the window is full, in which case you should use the scroll bars to find a clear space for a new scene. You select a scene by clicking on it. On the map below, the scene "Forest Clearing" happens to be selected.

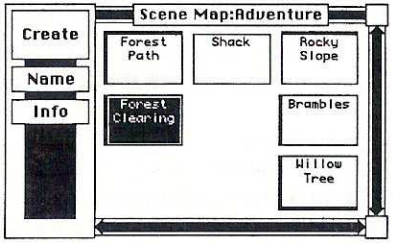


Fig. 2-1. Scene Map Window.

Moving Scenes

You can drag the selected scene to a new location with the mouse. The selected scene can be cut or copied with the Edit menu. Scenes can be pasted into any clear part of the window if there is room.

Naming Scenes

Select a scene to change, and click the Name button located on the side of the Scene Map window. A dialog box will appear. Type in the new name. World Builder accepts just about any combination of characters as scene names, but short names are easier to work with.

Scene Information

Use the Info button to find out how much disk space the selected scene takes up. Scenes will take much more disk space if they contain a lot of bit-map graphics than if they contain mostly objects.

Scene Design

The Window menu is used to open the design, text, code, or data window for the selected scene. You may open any number of windows at any time, and cut, copy, and paste between them with freedom.

Open Scene Design

The scenes in a World Builder game act as a background for characters and objects. The menu command Open Scene Design displays the design window for the currently selected scene (double-clicking on a scene in the Scene Map opens both the Scene Design window, and the Scene Text window, described below). The design window is the canvas where you draw the scene using World Builder's drawing tools. Chapter 4 explains how to use the drawing tools to create or edit scenes.

The Scene Design window can be resized and moved around the screen. World Builder remembers the position of the Scene Design window for game time, so you might want to arrange some windows in a special way for some scenes.

The Scene and Text windows can also be positioned using screen coordinates. Use the Align Window... command in the Edit menu when the window is selected. If you want all your windows to be in exactly the same place, jot down the coordinates you use and make sure they are the same for each Scene Design window or Scene Text window.

When you are done with the design window, close it by clicking in the upper left corner, or by selecting the Close Scene Design command from the Window menu. A dialog box will ask if you want to save your work.

Open Scene Text

The menu command Open Scene Text displays the text window for the currently selected scene. The text window should give the player some information about the scene, including which directions are open and which are blocked. You can type text into the window and select text with the mouse. You may change the font and size of the text with the Font menu. The Edit menu lets you cut, copy, or paste text.

The Scene Text window can be resized and moved around the screen. Use the Align Window... command to give it a position with screen coordinates. Close the window when you want to save your work.

Open Scene Code

If you would like your world to respond to special commands, or if you want to do some simple animation, you'll need to add adventure code to your scenes (worlds run just fine without using any code at all). The menu command Open Scene Code displays the code window for the currently selected scene. The use of adventure code in World Builder games is discussed in Chapters 5 and 6.

Open Scene Data

The menu command Open Scene Data displays the data window for the currently selected scene. This window, shown in Fig. 2-2, lets you fill in some more information about the scene.

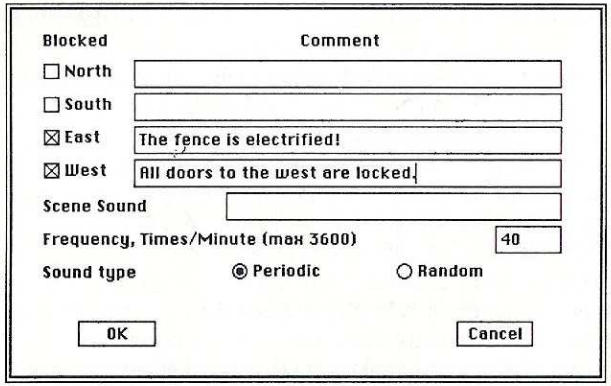


Fig. 2-2. Scene Data Window.

Blocked Compass Directions

Click in the North, South, East, or West box to block movement in that direction. Ordinarily, the game responds with a "Can't go in that direction" comment when the player tries to exit the scene, but you can change the comment by typing a new one in the comment box.

For example, you might check the "North" box and type "The path to the north is blocked by a steep cliff." in the comment field. When the player tries to go to the north, she will not be allowed to do so and she will get your message.

Scene Sounds

Scenes can be accompanied by digitized sounds, such as the sound of dripping water for the bottom of a well, or the sound of hoot owls for the dense woods behind a haunted house.

To include a sound for playback during any scene, type the name of the sound in the Scene Sound box (for more information on working with sounds, see Chapters 6 and 7).

Using World Data

At the bottom of the Window menu is the command Open World Data. This command opens a dialog box with three fields.

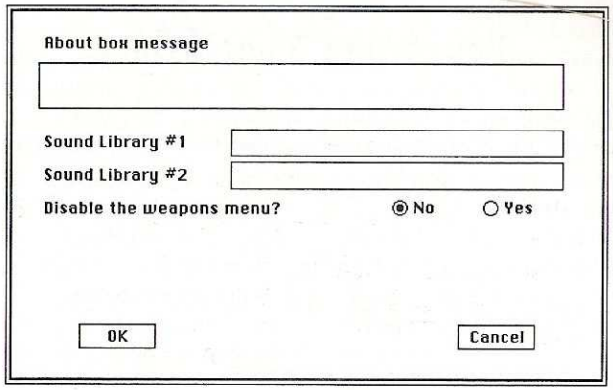


Fig. 2-3. The World Data dialog box.

Field 1 asks for an "About Box Message." This message will be displayed in your adventure game in the "About Box" under the Apple menu. Use it to put your name or some other identifying remark in your games. The radio buttons allow you to specify that you do not want the Weapons menu to appear, in case you are working on something other than a game.

Backup

World Builder is a resource-based program, very similar in nature to Apple Computer's Resource Editor. The biggest advantage to this is that you can make changes, save them, quit World Builder, immediately go double-click on your newly created game and the changes are there.

The one disadvantage to this is that if something goes wrong while you're editing (e.g. a power failure, or an incompatible desk accessory causes a system bomb), the file containing the game that you are working on can be irretrievably corrupted.

It is absolutely **essential** that you periodically use the Save Backup... command in the File menu while you're working. As long as your game is still small, you can simply save to the same disk under another name. If you do this, at the end of your work session, back up the file to another disk.

When the file is too big for the same disk, save it to another disk. When you finish your work session, do one last Backup and you will already have your backup file on another disk.

Please get into the habit of making backups frequently!!

Chapter 3
Populating Your World

No world is complete without characters. World Builder lets you create all sorts of interesting characters -- bank robbers, monsters, ghouls, vampires, robots, wizards, spies, you name it. World Builder characters are made with the help of the Character List window and the Character commands in the Window menu.

This chapter deals with creating characters for your adventure games, as well as adding objects that the player and your characters can find and use.

Editing Existing Characters

Characters present in a complete or partially finished game are shown in the Character List window. To look at or edit a character, select it and choose one of the Character commands in the Window menu.

Creating New Characters

Click the "Create" button located beside the Character List window to create a new character. The scroll bar on the right becomes active if there are too many characters to fit in the window. To begin work on a new character, click on it.

Naming Characters

Select a character to change, and click the Name button located on the side of the Character List window. A dialog box will appear. Type in the new name. World Builder accepts just about any combination of letters, numbers, and symbols as character names, but short names are easier to work with.

Character Information

Click on the Info button to find out how much disk space the selected character takes up.

Character Design

The Window menu opens the selected character's design window or data window. You may open any number of windows at any time, and cut, copy, and paste between them with freedom.

Open Character Design

Characters in a World Builder game appear in front of the scene backgrounds (discussed in Chapter 2). The menu command Open Character Design displays the design window for the currently selected character (double-clicking on a character in the Character List window opens the Character Design window as well). Chapter 4, "Using the Graphics Editor," goes into more depth on how to design characters using World Builder's drawing tools.

Open Character Data

The Open Character Data command displays a main dialog box revealing important data about the currently selected character. The behavior of each character is defined in the six character data dialog boxes. You can change the behavior of any character at any time simply by changing the values in the character data boxes.

To advance through the boxes, click the Next button. To go back, click the Previous button. When you're done defining the character, click OK. If you don't want to make any changes, click Cancel.

Boxes with slide bars: To increase or decrease the settings, put the cursor in the slide bar and click or press the mouse button. Clicking to the right makes the black indicator slide further right. Clicking to the left makes it slide back. By setting the bars farther to the right you indicate a higher level of the given attribute.

Boxes with fields for sound names: Enter the name of the sound. See Chapter 6, "Advanced World Building," for more information on using sounds.

Box One

Name of initial scene -- The character's location when the game is started. The default value is **RANDOM@**, which means that the character will start out at a random scene. You can also enter **STORAGE@** as the initial scene, in which case the character will start out in storage, which is like limbo or no-man's-land. A character in storage will never be found at a scene unless you use adventure code to bring him to a scene. If you are not using code, don't use the **STORAGE@** option. If the name of the startup scene is invalid or spelled incorrectly, the player will be assigned to storage.

Character gender -- Click He, She, or It, as desired.

Is character a proper noun? -- Click Yes or No. This and the preceding field helps World Builder put together accurate sentences about the character.

Is character the player character? -- Click Yes or No. The person who plays the game must have a character to represent him. If you assign more than one character as the player, one will be chosen at random as the player character. The player may start at a particular scene or at a random scene depending on the startup field. If no player character is specified, a random character is selected to be the player.

Number of objects character may own -- The maximum number of objects the character (including the player) may carry at once. A wild tiger, for example, will not own any objects, while a miser might own a great many. No character may own more than 255 objects.

Return used character to -- Characters that are killed in battle are returned to this scene -- storage, a random scene, or their initial scene. Generally, you will want to recycle some characters while others will only live once. The game ends if the player character dies.

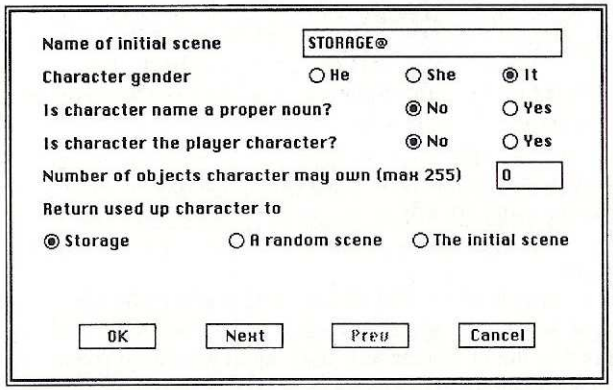


Fig. 3-1. Box One -- Character Data.

Box Two

If your character is a magic user, he or she will tend to have high spiritual settings. If your character is a fighter, he or she will tend to have high physical settings.

Physical strength -- The measure of the ability to deliver physical punishment.

Physical hit points -- The level of physical well being of the character -- as battle continues, hit points will fall, and if the physical hit points reach zero, the character will die.

Natural armor -- The natural resistance which the character has to physical damage.

Physical accuracy -- The tendency of the character to use weapons successfully.

Spiritual strength -- The measure of the ability to deliver magical punishment.

Spiritual hit points -- The level of spiritual well being of the character -- as battle continues, hit points will fall, and if the spiritual hit points reach zero, the character will die.

Resistance to magic -- The natural resistance which the character has to spiritual damage.

Spiritual accuracy -- The tendency of the character to use magic successfully.

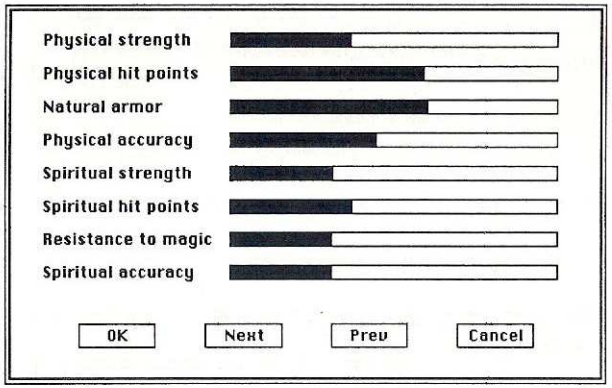


Fig. 3-2. Box Two -- Character Data.

Box Three

Running speed -- The speed at which characters can run away when attacked. If Running speed is high, the character will tend to escape characters who follow him and to catch characters he runs after.

Rejects offers -- The tendency to turn down offers when bribed. Each object has a different value and this is used to determine if offers are accepted or rejected. If the value of an offered object is greater than the Rejects offers field, the character will accept the offer. Otherwise, the offer will be rejected.

Follows opponent -- The tendency to follow when the player character runs away.
Initial sound -- The name of the sound the character makes when it is first encountered in a scene.
Scores a hit sound -- The name of the sound the character makes when it strikes an opponent.
Receives a hit -- The name of the sound the character makes when it is struck.
Dying sound -- The name of the sound the character makes when it dies.

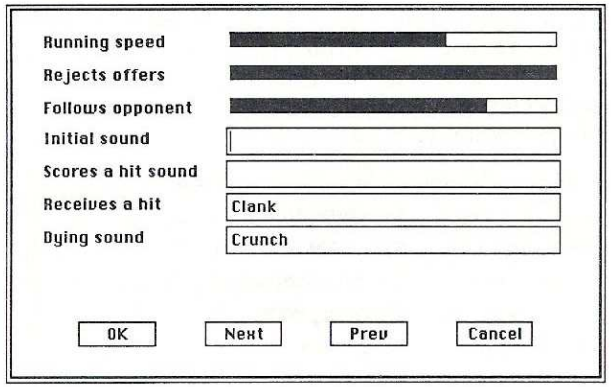


Fig. 3-3. Box Three -- Character Data.

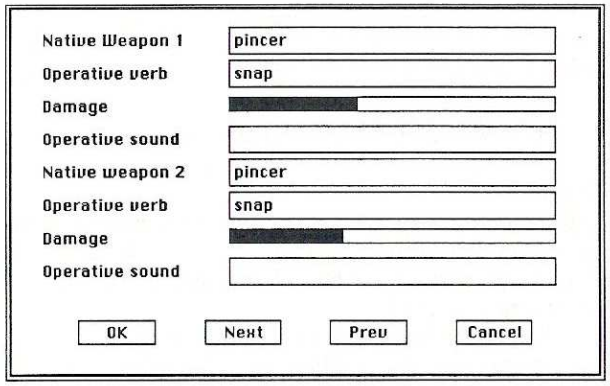


Fig. 3-4. Box Four -- Character Data

Box Four

Native weapon -- Native weapons are things like fists or teeth which are a physical part of the character. Native weapons cannot be traded or dropped. For example, a giant scorpion could have the native weapon "stinger."
Operative verb -- The verb used to describe the action of the native weapon. The operative verb for the scorpion's stinger might be "swing."
Damage -- The level of relative damage inflicted by the native weapon.
Operative sound -- The name of the sound the native weapon makes when it strikes an opponent.

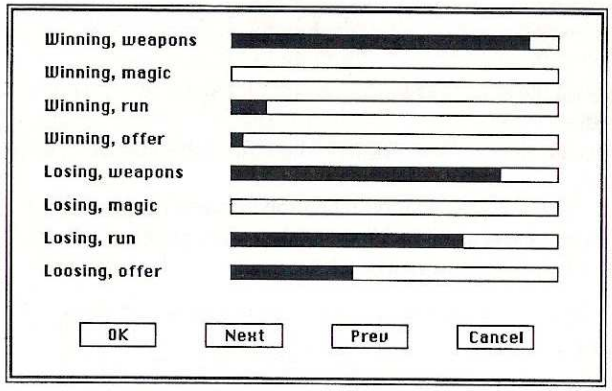


Fig. 3-5. Box Five -- Character Data.

Box Five

Winning, weapons -- The tendency to use weapons when winning a fight.
Winning, magic -- The tendency to use magic when winning a fight.
Winning, run -- The tendency to run away when winning a fight.
Losing, weapons -- The tendency to use weapons when losing a fight.
Losing, magic -- The tendency to use magic when losing a fight.
Losing, run -- The tendency to run away when losing a fight.

Battle strategy does not need to be set for the player character. The average character will tend to fight and use magic if he is winning; run or offer objects if he is losing. The native weapons are automatically used if the character doesn't have a weapon or spell to fight with, an object to offer, or a direction to run.

Box Six

Initial Comment -- Text printed when the character is first encountered.

Scores a hit -- Text printed when the character strikes an opponent.

Receives a hit -- Text printed when the character is struck.

Makes an offer -- Text printed when the character makes an offer.

Rejects an offer -- Text printed when the character turns down an offer.

Accepts an offer -- Text printed when the characters accepts an offer .

Dying words -- Text printed when the character dies.

Comments are not printed for the player character, who can make his own comments.

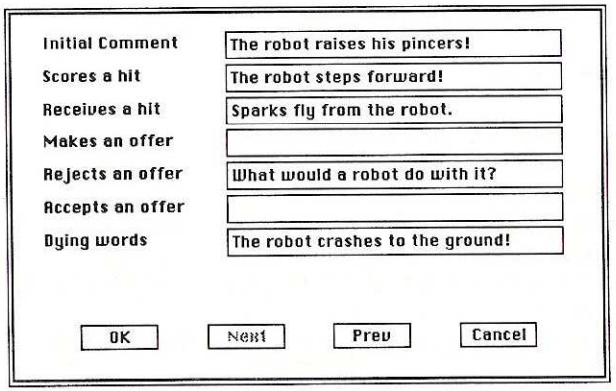


Fig. 3-6. Box Six -- Character Data.

Editing Existing Objects

Objects present in a complete or partially finished game are shown in the Object List window. To look at or edit an object, select it and choose one of the Object commands in the Window menu.

Creating New Objects

Click the Create button located beside the Object List window to create a new object. The scroll bar on the right becomes active if there are too many objects to fit in the window. To begin work on a new object, click on it.

Naming Objects

Select an object to change, and click the Name button located on the side of the Object List window. A dialog box will appear. Type in the new name. World Builder accepts just about any combination of characters for object names, but short names are easier to work with.

Object Information

Click on the Info button to find out how much disk space the selected object takes up.

Object Design

The Window menu opens the selected object's design window or data window. You may open any number of windows at any time, and cut, copy, and paste between them.

Open Object Design

Objects in a World Builder game appear in front of the scene backgrounds (discussed in Chapter 2). The menu command Open Object Design displays the design window for the currently selected object (double-clicking on an object in the Object List window opens the Object Design window as well). Chapter 4, "Using the Graphics Editor," goes into more depth on how to design objects using World Builder's drawing tools.

Open Object Data

The Open Object Data command displays a main dialog box revealing important data about the currently selected object. The type, use, and property of each object is defined in the two (and in one case, three) object dialog boxes. You can change the property of any object at any time simply by changing the values in the object data boxes.

To advance through the boxes, click the Next button. To go back, click the Previous button. When you're done defining the character, click OK. If you don't want to make any changes, click Cancel.

Box Two varies, depending on the type of object selected. Box Three is activated only when the Magical Object button is pushed.

Box One

Name of scene or owner -- The location of the object when the game first begins. The object can be placed at a scene, or can be owned by any character. The object is encountered when the character is encountered. You may enter **RANDOM@**, which starts the object out at a random scene, or **STORAGE@** which starts the object out in storage.

Is object name plural? -- Click Yes or No, as desired.

Value of object -- The relative value of an object, for use when offering objects to other characters. Use the mouse to increase or decrease the setting. By setting the bar farther to the right you indicate a higher level of value. If the value of an offered object is greater than a character's tendency to reject offers, the offer will be accepted.

Object type radio buttons -- The object can be one of nine types: regular weapon, throw weapon, magical object, helmet, shield, chest armor, spiritual armor, mobile object, or immobile object.

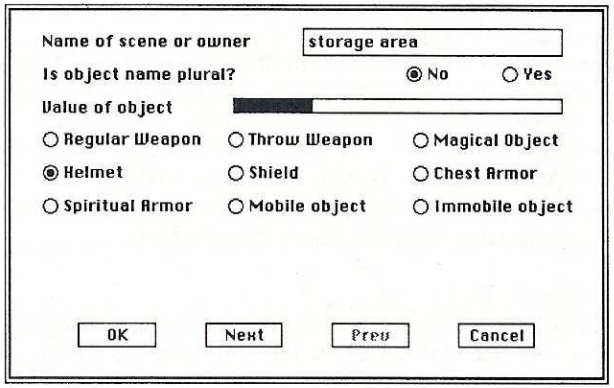


Fig. 3-7. Box One -- Object Data.

Box Two

Number of uses -- The maximum number of times the object can be used. (Does not apply to mobile or immobile objects.) Enter number or click Unlimited.

Return used object to -- The location that the object is sent to after it has been used the maximum number of times. Some objects can be re-circulated while others will be used only once. Click Storage or Random Scene, as desired. (Does not apply to mobile or immobile objects.)

Damage -- The amount of relative damage inflicted by regular and throw weapons.

Accuracy -- Tendency for regular and throw weapons to strike opponents. A cross bow, for example, might tend to do a lot of damage but have low accuracy. A sword might be very accurate but do less damage.

Pickup message -- Text printed when a mobile object is picked up by the player.

Operative verb -- The verb used to describe the action of the a weapon (not mobile or immobile objects). This verb is used in front of the object name, like "swing sword" or "fire gun."

Failure message -- Text printed after an object is used the maximum number of times (does not apply to mobile or immobile objects).

Operative sound -- The name of the sound that the object makes when used (does not apply to mobile or immobile objects).

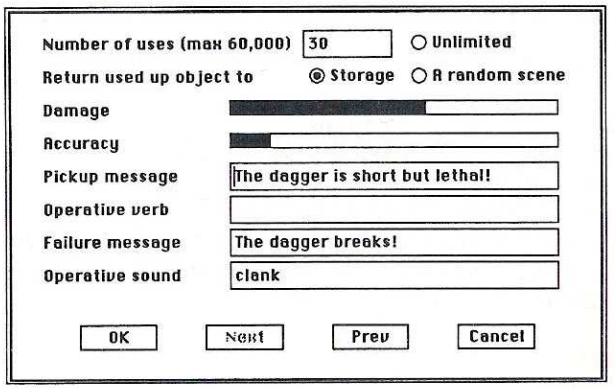


Fig. 3-8. Box Two -- Object Data for Weapons.

Box Three (Magical Objects Only)

Spell Power -- The relative magical strength of the object.

Spell Accuracy -- Tendency for magical objects to have an effect on opponents.

Cause physical/spiritual damage -- Click one or the other to indicate the kind of damage inflicted by the magical object.

Cause physical and spiritual damage -- Click if the object causes both physical **and** spiritual damage.

Heal physical/spiritual damage -- Click one or the other to indicate if the object heals physical or spiritual damage.

Heal physical and spiritual damage -- Click if the object heals both physical **and** spiritual damage.

Freeze opponent -- Click if the object freezes opponent. The duration depends on the power of the spell.

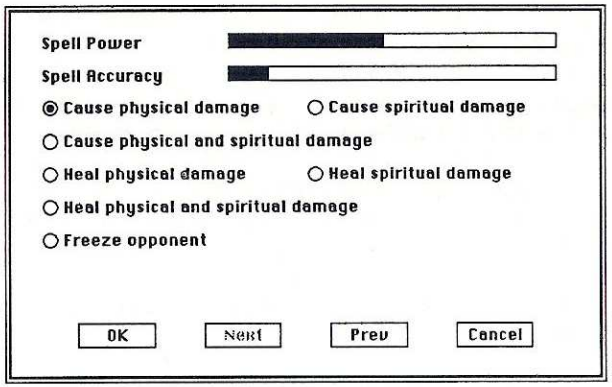


Fig. 3-9. Box Three -- Object Data (Magical Object Only).

Regular and Throw Weapons

Regular and throw weapons are used to inflict damage on an opponent. A regular weapon is held in the hand, like a sword or a gun. Throw weapons can be used only once, after which they are moved to the current scene where they must be picked up to be used again (opponents can also pick up thrown weapons and use them against the player).

Apparel

Helmets, armor, and shields are worn, but characters can wear only one of each at a time.

Mobile Objects

Mobile objects can't be used as weapons but can be carried by characters. They are often given a special purpose with adventure code, discussed in chapters 5 and 6.

Immobile Objects

Immobile objects can't be moved or picked up, and they are not described when the player arrives at a scene. Immobile objects are often things like doors or a heavy desk that stays at a certain scene. Adventure code must be used to move immobile objects back and forth from storage to create simple animation, such as doors opening and closing, etc.

Sometimes you will want to place an immobile object at a certain position in a scene. You might, for example, want to put a door on a building. To do this, just draw the object on the Scene Design window (as described in Chapter 4) in the desired position, then cut and paste it onto an Object Design window. Set the "startup scene" for the appropriate scene, and click the Immobile object button.

Chapter 4
Using The Graphics Editor

Once you have opened a scene, character, or object design window you can draw in the window with World Builder's graphics editor. The editor allows you to create individual graphic objects like rectangles, round rectangles, ovals, polygons, freeforms, and bit boxes.

These simple elements can be combined to form more complex pictures. Individual objects can be selected and reshaped with the mouse. You can change the patterns and line widths of selected objects with the Pen and Fill menus. The Edit menu allows you to work on selected objects in a variety of ways, as you'll see in this chapter.

Opening a Design Window

There are three types of design windows: scene, character, and object. Open a new or existing design window as described in chapters 2 and 3. The design window below shows a character being edited.

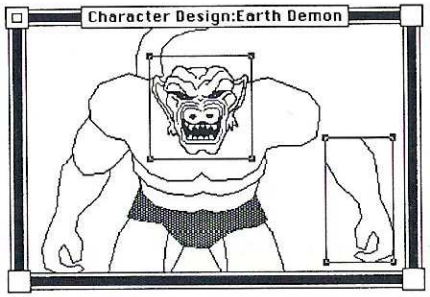


Fig. 4-1. Editing a Character in the Design Window.

Selecting an Object

You can select different parts of the picture with the mouse. Selected objects appear bordered with a solid line that has square "handles" on each corner. The head and arm of the Earth Demon happen to be selected in the example in Fig 4-1. Holding down the shift key and clicking allows selection of multiple objects. You can drag a selection rectangle around a group of objects and select many objects at once. Once selected, you can manipulate the group as a whole (some editing functions won't work when more than one object is selected at the same time).

Moving Objects

Selected objects can be dragged to a new location with the mouse.

Resizing Objects

Objects can be resized by dragging the little handles at the four corners of the selection rectangle when an object is selected. Simply position the mouse over a corner handle and drag the handle to a new location. The object will be resized to fit inside the new rectangle.

The Tools Menu

New objects are created with the Tool menu. Normally the Select item is active (shown by the checkmark beside the command), which lets you select objects in the drawing windows.

Rectangle, Round-Rectangle, Oval

Choose the desired tool from the Tools menu, and move the mouse over to the design window. The mouse will change into cross-hairs. Drag the mouse over the part of the window where you want to create the object. Release the mouse button when the object is the desired shape and size.

Polygons

To make a polygon, select the polygon tool and position the mouse where you want the object to start, then click and drag the mouse. A line will extend from the start point. Keep clicking to continue lines in other directions. Click back to the start point or double-click to close the polygon.

Freehand

To make a freehand object, select the freehand tool, and drag the mouse around the window to construct the shape. The shape is complete when you release the mouse button.

Bit Box

The bit box tool creates a floating plane of bits that can be painted and individually edited like a MacPaint document. The head of the Earth Demon in Figure 4-1 is a bit box. Bit boxes are discussed in more detail below.

The Fill and Pen Menus

The Fill menu lets you change the pattern of the selected objects. The Pen menu allows you to change the pen width and pen pattern of the selected objects. Notice that most of the Earth Demon above has a white fill pattern, a black pen pattern, and a pen size (or thickness) of one pixel. The last pattern in the lower right of each menu is the clear pattern.

The patterns in World Builder and World Template are stored as PAT# resources so that they can be edited with the Resource Editor.

The Edit Menu

Among other things, the Edit menu lets you cut, copy or paste graphic objects within World Builder, and with most other Macintosh applications. The Edit menu also includes a number of commands that allow you to manipulate objects in a variety of ways.

Undo

Cancels the very last action.

Cut

Removes an object from the window and places it in the Clipboard.

Copy

Copies an object and places the copy in the Clipboard.

Paste

Places the contents of the Clipboard onto the drawing.

Using the Cut, Copy, & Paste Commands

The Cut, Copy, and Paste commands are used most often to trade graphics from one window to another. You may want a character or object to appear at a certain position in the scene window. To do this:

1. Draw the character or object in the scene window at the spot you want it.
2. Select all of the character or object.
3. Choose Cut from the Edit menu.
4. Open a character or object design window.
5. Paste.

Whenever the character or object is called to the scene, it will appear exactly where you originally drew it.

You can import graphics from other applications and you can export graphics. World Builder is compatible with the standard Macintosh Clipboard. But because World Builder uses its own internal format, most objects will be converted to bits when pasted. Squares, rectangles, rounded rectangles, cirlces and ovals are retained as objects.

One helpful shortcut when designing World Builder games is to keep standard graphic elements in the Scrapbook. You could, for example, keep trees, desks, and tables in the Scrapbook and retrieve them when drawing new scenes.

Send to Back

World Builder stacks objects . The Send to Back command sends the selected objects to the very back of the stack.

Bring to Front

Brings an object to the front of the stack.

Round Edges

Smooths the angles of polygons and freehand shapes.

Sharpen Edges

Unsmooths the angles of polygons and freehand shapes.

Flip Polygon

Flips a polygon side to side.

Rotate Polygon

Rotates a polygon any number of degrees clockwise or counterclockwise. A handle appears beside the polygon; grab it and drag back and forth to rotate the objects.

Reshape Polygon

Reshapes the sides of a polygon. When you choose the Reshape Polygon command, a handle appears on every vertex of the object. Drag a handle to the new location where you would like that vertex.

Bit Commands

When a single bit box is selected, you have the option to "paint bits," "zoom bits," or "capture bits." A bit box is a floating field of bits, like a little MacPaint document. Bit boxes are mainly used for details like faces and hands, and are limited in size because they take up a great deal of memory.

Capture Bits -- The Capture Bits command copies any object (or portion of any object) under the bit box and transforms it into a series of bits.

Paint Bits -- Choosing Paint Bits forms a rectangle around the currently selected bit box, indicating that these bits can be painted. Select various brush patterns and pen sizes from the Pen menu, position the cursor over the bit box, and draw on the bits directly. The Paint Bits command is often used for getting the bits in correct proportion to the rest of the drawing. Once the bit box looks good, use Zoom Bits to clean up the details.

Zoom Bits -- The Zoom Bits command brings up a large window which allows you to edit individual pixels. At the bottom of the window is a bar, which you can slide back and forth to vary the amount of zoom magnification. When you get the drawing right, click OK; to cancel changes, click Cancel.

After clicking OK, you can use the Undo command to undo everything you did in Zoom Bits.

Each bit box has a mask drawn under it. If you draw a closed shape on a bit box, the shape itself will be black, the interior of the shape will be white and the bits outside the shape will be clear. This is very useful for objects where you need a white center. If a small break is made in the bits, the clear area will "leak" in, just like the lasso in SuperPaint or MacPaint. Zoom bits can be used to patch leaks in your bit boxes.

World Builder is Clipboard compatible. Bit boxes and other graphic objects can be imported from other programs and can be exported to other applications just as well. Certain types of objects that World Builder does not recognize will be pasted in as bit boxes.

Disk Space Considerations

An example of pasting in graphics is using digitized images. You can capture the image of a friend or family member's face with a digitizer and paste that into a scene (whether to make them a monster or hero is up to you). This would be a bit box. But you pay a price for using bit boxes, and that price is disk space.

Bit boxes take up **considerably** more disk space and memory than objects do. The best approach is to use objects as much as possible and use bit boxes only when you need to do detailed painting. Remember that you can check how much memory a scene, character, or object takes by clicking on the Info button of the selected window.

Chapter 5
Writing Scene Code

Once you create scenes, characters, and objects in World Builder, the world you've created seems to take on an existence of its very own; the player character can wander around the world, fighting its inhabitants and collecting treasures. World Builder is designed to take care of these normal aspects of adventuring.

When you create a scene, however, it is usually part of an overall story that you have in mind, and you may want the "normal" game rules to be altered or enhanced in some way. By writing "scene code," you can program your world to behave the way you want.

Scene code is entered in the Code window for any scene in your game map. It consists of a limited, though versatile, set of statements that tell the game what to do. Many of the statements you put in a Code window will be "conditional"; that is, they will only be carried out if certain conditions exist. This gives you the opportunity to run the game according to what the player has already done--where she's been, what she owns, who she's met so far, what physical condition she's in, and so on.

Syntax

Every programming language, like every spoken language, has its own "syntax"--a set of grammatical rules that control how commands must be written. These rules include certain constraints on spelling, capitalization, punctuation, and the order in which words appear.

The syntax of World Builder's code is relatively simple:

- Commands must be typed in capital letters
- Commands must be separated from other items in the statement by braces
- There can be no space between a command word and a brace
- Names of scenes, characters, and objects must be spelled exactly as they are named when you create them
- Variable names must use capital letters

The simple statement:

MOVE{PLAYER@}TO{STORAGE@}

gives a lot of opportunities for incorrect syntax--a missing brace, words in lowercase letters, extra spaces--any of these will make the statement unacceptable.

Luckily, World Builder will catch most of your syntax errors when you try to close the Code window; you'll be shown which line the error is in so you can correct it.

World Builder doesn't check the syntax of MENU statements, though, so be especially careful if you want to get fancy there. Another thing World Builder can't catch is a misspelled scene, object, or character name. If you've defined a Wuggly Ump character and refer to the "Wuggy Ump" in the code window, the mistake won't be detected.

Using MENU

You can configure the Commands menu to your liking in any scene by using the MENU statement in the Code window for that scene.

As soon as you use the MENU statement, all the usual contents of the Commands menu disappear, although the commands themselves can always be typed into the text window. To put the command "Jump" in the menu, use:

MENU{Jump}

When you want to provide a command-key equivalent for a menu command, you can do it like this:

MENU{Jump/J}

The Edit menu uses the letters Z, X, C, V, and B, so don't use them in your Commands menu.

Multiple menu items in a MENU statement are separated by semicolons:

MENU{Jump/J;Fly/F;Yell/Y}

When you want an item to appear dimmed in the menu--perhaps you don't want "Fly" available unless the player has swallowed the magic potion--use a left parenthesis in front of the item:

MENU{(Fly/F}

You can add a dividing line to a menu by using the hyphen; always precede the hyphen with a left parenthesis so the player can't select the dividing line in the menu:

MENU{(-}

You might want to use the dividing line to separate the direction commands (North, South) from action commands (Jump, Fly).

You can create fancier menus--and perhaps give subtle clues to the player at the same time--by using different font styles in the menu. There are five type styles available, and you use them by typing the "<" sign and the style initial:

<B Bold
<I Italic
<U Underline
<O Outline
<S Shadow

The style definition follows the command word in the MENU statement:

MENU{Up<I}

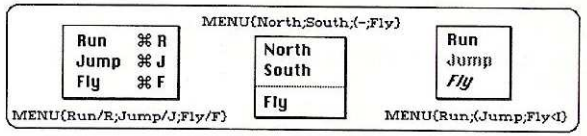


Fig. 5-1. The MENU Command

Using PRINT

The PRINT command is the simplest and most obvious of World Builder's code statements. Anything you enter in the braces following PRINT will be printed in the text window when the PRINT statement is executed.

PRINT{It's getting colder}
PRINT{You'll live to regret that...if you're lucky}

The text you enter in the text window when you design a scene is usually what appears as soon as the player enters that location in the game. Using the PRINT statement as a conditional command in the Code window (by using the IF-THEN construction described later), you can print additional information in the text window based on what has happened so far; or, you can print something instead of the original text window information, reserving that text for when the player uses the LOOK command.

Regardless of how you type the information in PRINT's braces, the text will automatically wrap to fit inside the text window of the scene. Each time you use a new PRINT statement, a new paragraph is started in the TEXT window.

Using MOVE

The MOVE command is used for two distinct activities: to move a character or object from one scene to another, and to put an object into someone's possession.

MOVE for Location Change

Move a character or object simply by naming it and the destination:

MOVE{porpoise}TO{tank room}
MOVE{magic ring}TO{cellar}

When a character or object is moved to a location, it will appear in that scene when the player reaches that location.

The player character is automatically moved to a new location when she types a direction command--as long as the direction is not blocked, of course. There are times, however, when you'll want to move the player character directly with the MOVE statement.

You might have your character meet a magician who transports her bodily to a different place:

MOVE{PLAYER@}TO{forest clearing}

Or, the character might press a button in a malfunctioning transporter and send herself to some random settting:

MOVE{PLAYER@}TO{RANDOMSCN@}

Another time you'll want to MOVE the player yourself is when you want the Up and Down direction commands to work; these are the only commands in the standard menu that the game doesn't handle automatically.

To build a three-dimensional map, such as floors in a castle with many rooms on each floor, create your scene map so that there is no access from one group of scenes to another by blocking the appropriate directions. Then, if the player types the Up or Down command in the right scene--a stairwell, for instance--you can move her up or down the stairs, and right into the otherwise inaccessible section of the game map:

MOVE{PLAYER@}TO{2nd Stair}

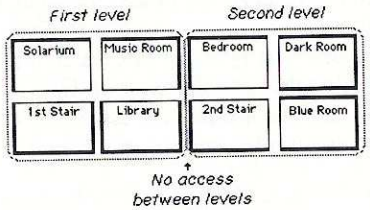


Fig. 5-2. Establishing Levels

MOVE for Possession

To "move" an object into a character's possession, name the object and the character to whom it should belong:

MOVE{magic ring}TO{PLAYER@}
MOVE{mushroom}TO{Alice}

When you move an object into the player's possession, it becomes part of her inventory--when the Inventory command is issued, the object will be part of the list.

An object that is moved into any character's possession no longer appears in a scene; it has, in effect, been moved from the scene to the character.

World Builder performs automatic "moves" when an offer has been accepted by a character. If the player offers an apple to a gremlin and she accepts it, the apple is no longer in the player's inventory, but is in the gremlin's possession. You don't have to write the code for this transfer.

Variables

A variable is a word that stands for something--although that something varies from one time to another. So, the special World Builder variable PLAYER@ may be a fledgling necromancer in one game, while it might stand for a valiant, if inexperienced, warrior in another. Another World Builder variable, LOOP#, stands for the number of commands given in a scene--that number varies every time the player does something or moves someplace.

World Builder variables can be divided into three general groups: world variables, command variables, and numeric variables. Variables are always printed in capital letters, with a special character at the end to denote the type of variable that it is.

World Variables

World variables use the trailing character @; they stand for objects, scenes, or characters. There are seven world variables.

1) PLAYER@

PLAYER@ denotes the player character.

2) STORAGE@

STORAGE@ is the only variable whose meaning never varies: it is a special place to store game elements when you don't want them to be found at any scene, or in anybody's possession.

When you create your world, you can specify STORAGE@ as the "initial scene" for any object or character that won't come into play until later in the game. Then, during the game, you can use MOVE to put the the stored object or character where you need it.

During a game, you can also move objects or characters into STORAGE@ to take them out of active play; moving PLAYER@ into STORAGE@ ends the game.

3 & 4) SCENE@ & MONSTER@

SCENE@ and MONSTER@ are variables that refer to the current scene in the game, and the character (other than the player) at the current scene. So, you can move an object into the current the scene without having to know what that scene is:

MOVE{emerald}TO{SCENE@}.

SCENE@ is obviously not a variable you would need to use in scene code--you always know what scene you're in when you're writing code for it. It is, however, an extremely useful variable when you're writing global code--instructions for the overall game, as described in the next chapter.

You can use MONSTER@ in scene code; you won't always know which character is in a given location. Some characters move around randomly, and some just follow the player around. Referring to MONSTER@ in your scene code means any old monster that's currently present. If the player casts the right spell, you might want to move the current monster to a cage, even though you don't know which monster that is when you're writing the code:

MOVE{MONSTER@}TO{cage}

5, 6 & 7) RANDOMSCN@, RANDOMOBJ@, & RANDOMCHR@

To keep the structure of a game loose, you have to be able to introduce some random elements--like having an object disappear into an unknown location, or having some random object show up in the current scene, or having any one of your many monsters show up in the attic.

You can use the random world variables to accomplish these purposes:

RANDOMSCN@, RANDOMCHR@, and RANDOMOBJ@

RANDOMSCN@ refers to a random scene. If the player casts a less-than-perfect spell, you might want to move a monster to some random location rather than kill him off or capture him:

MOVE{ogre}TO{RANDOMSCN@}

Or, you can move whatever the current monster is:

MOVE{MONSTER@}TO{RANDOMSCN@}

You could even have the player transported to a surprise location:

MOVE{PLAYER@}TO{RANDOMSCN@}

If the fearless adventurer has pressed a button in the transmutation room, you might want to make a random object or monster character appear:

MOVE{RANDOMCHR@}TO{transmutation room}

MOVE{RANDOMOBJ@}TO{SCENE@}

You can even put a random object in the player's pack, or give one of her possessions away to a random character:

MOVE{RANDOMOBJ@}TO{PLAYER@}

MOVE{canteen}TO{RANDOMCHR@}

Command Variables

If you went through the tutorial in Chapter 2, you're already familiar with the two "string" variables TEXT$ and CLICK$. A "string" is just a bunch of letters strung together to make words and phrases: *book of oaths* is a string, as is *apprentice*.

CLICK$ stands for the name of the object that the player has clicked on in the scene's graphics window. By using code that says:

IF{CLICK$="old hat"}...

you can set up rules for what happens when the old hat is clicked on.

TEXT$ stands for what is typed in the text window, or what is selected from the Commands menu.

IF{TEXT$="north"}...

works whether the command was typed in, selected from the menu, or issued by the Command-N equivalent.

TEXT$ is usually a command, but since it stands for anything typed in the Text window, you can let the player use questions like "Why?" or "Who are you?" in some scenes.

Numeric Variables

Numeric variables represent numbers; World Builder's numeric variables all end with the symbol #. There are three basic numeric variables.

LOOP# keeps track of how many commands the player has issued in a scene. If the player goes back to a previously-visited location, LOOP# is reset to 0, so it can keep track of the commands issued in the current visit.

You can use LOOP# to limit the player's moves in a scene, or to "time" her visits in a location:

IF{LOOP#=5}...

checks to see if five commands have been given at the scene.

VISITS# keeps track of how many scenes the player has visited--not how many different scenes, but how many scene changes all together. You can use this variable to keep track of a time element, too:

IF{VISITS#=42}...

checks to see if the player has entered scenes 42 times during the game.

VICTORY# is the number of characters killed so far in the game. You might let the player take a certain action only if she's bagged enough meanies:

IF{VICTORY#>5}...

Other numeric variables available in World Builder are discussed in the Chapter 6.

Comparison Operators

Comparison operators are the symbols used to compare two items. The comparison operators available in World Builder are:

=, >, <.

The operators have slightly different meanings depending on whether you are using them with numeric items (numbers and numeric variables), strings, or other items.

Numeric Comparisons

For numeric comparisons, the operators have their usual mathematical meanings:

{VISITS#<5} *means* "VISITS# is less than 5"
{LOOP#>3} *means* "LOOP# is greater than 3"
{LOOP#=2} *means* "LOOP# is equal to 2"

There is no operator for "does not equal"; if the variable you're working with might be higher or lower than the comparison number, you'll have to set it up like this:

IF{VISITS#<4}OR{VISITS>6}...

This construction boils down to the same thing as "VISITS# does not equal 5." (There's more information about IF and OR later in this chapter).

String Comparisons

The comparison operators have different meanings when used with the TEXT$ variable; you get lots of freedom in interpreting a player's commands.

When you use the equals sign with TEXT$, it means "is a substring of."

IF{TEXT$=eat sandwich}...

is true if the player's typed command included the string "eat sandwich." Being able to check for substrings can be handy. In a scene where there is a tree you can use:

IF{TEXT$=climb}...

This is true if the player typed "climb tree," "climb," "climb the tree" or "climb up it."

Sometimes you'll want the player's command to be compared to an exact string--no substrings allowed. In that case, use a double equals sign.

IF{TEXT$==ball of twine}...

means that TEXT$ must exactly match "ball of twine" for the comparison to be true. You can use the < symbol, and a double version, <<, to mean "is not a substring of" and "does not exactly match."

IF{TEXT$>open strongbox}...

is true as long as "open strongbox" is not included in the player's command.

IF{TEXT$>>open strongbox}...

is true as long as the player did not type that phrase exactly.

Item Comparisons

Comparison operators are also used to check on the location of objects and characters, and on the possession of objects.

If you say "object=character," you're checking if the object belongs to the character; if you say "object=scene" or "character=scene," you're checking if the object or character is in that location.

IF{diamond=dwarf}... *means* "if the diamond belongs to the dwarf"
IF{rapier=dungeon}... *means* "if the rapier is in the dungeon"
IF{hobbit=hole}... *means* "if the hobbit is in the hole"

To check if an object is *not* owned by a specific character, or if an object or character is *not* in a specific location, use the > or the < sign:

IF{diamond>dwarf}... *means* "if the diamond does not belong to the dwarf"

IF{rapier>dungeon}... *means* "if the rapier is not in the dungeon"

IF{hobbit>hole}... *means* "if the hobbit is not in the hole"

Comparison Operators

	=	<	>	==	>>
Numeric Items	equals	is less than	is greater than	X	X
Strings	is a substring of	is not a substring of	<same	exactly matches	does not match exactly
Other Items	belongs to or is located in	does not belong to or is not located in	<same	X	X

Fig. 5-3. Chart of Comparison Operators

Using IF-THEN

If you went through the tutorial in Chapter 1, you already know the basics of IF-THEN programming. In fact, if you've used English, you already know the basics of IF-THEN.

In writing scene code, you can give explicit commands, as with MENU statements, or perhaps by moving a random character to the scene every time the player visits it. However, most of your code will be conditional--it will be executed only if certain conditions exist. The conditions will have something to do with the player's action so far--what she possesses, the shape she's in, the places she's been--and what she does in the current scene. You can set up your code to respond to a myriad of situations by using IF-THEN. That, of course, is what variables and comparison operators are all about: they are used to test if certain conditions exist.

When you use IF-THEN, you type one or more conditions in braces between the IF and THEN:

```
IF{ring=PLAYER@)THEN
```

Beneath that statement, you list the commands you want carried out if the condition is true (in this case, if the player has the ring):

```
IF{ring=PLAYER@}THEN
  MOVE{MONSTER@}TO{STORAGE@}
  PRINT{Your magic ring has banished the
  creature}
```

When you type commands under an IF statement, they are automatically indented after you press Return.

You have to mark the end of the block of conditional commands. There are two words that can do this--EXIT and END--and each has a slightly different effect on what happens next.

First of all, if the condition in the IF statement is not true, the conditional commands are not executed, and it makes no difference whether you use EXIT or END. The game goes right past the whole IF-THEN block and on to the other code you put in the Code window. If, however, the IF condition is true, the commands are executed, and the use of EXIT or END controls what the game does next.

EXIT means to exit the scene code. If you have other commands in the scene code after the EXIT statement, the game will not execute them. END means "this is the end of the conditional commands." The game goes on to the other code you wrote in the window.

(Each time the player issues a command in a scene, the game reads through the scene code again.)

Multiple Conditions

You can make the game check more than one condition and execute a block of statements based on the result of that multiple check. Conditions in an IF statement are linked by AND or OR.

When you link conditions with AND, they both must be true for the conditional statements to be executed:

```
IF{ring=player}AND{VISITS#>5}THEN
```

means the conditional statements are executed if the player has the ring and has been in more than 5 scenes.

When you link conditions with OR, if either one is true, the conditional statements are executed:

```
IF{ring=player}OR{VISITS#>5}THEN
```

means if the player owns the ring or if he has been to more than five places the statements are carried out. (With OR, *at least* one of the conditions must be true--not *only* one. So if both conditions are true when you use OR, the conditional statements are executed.)

You can link more than two conditions in an IF statement by using multiple ANDs or ORs, but you can't mix ANDs and ORs in the same IF statement.

Nested IF-THENs

You can "nest" one IF statement inside another; the game won't even see the inner one unless the condition in the outer loop is true.

```
IF{VISITS#>50}THEN
        PRINT{You've been wandering around a long
        time now.}
        IF{LOOP#>12}THEN
            PRINT{You seem to be wasting time here.}
        END
END
```

In this nested construction, the game will print "You've been wandering..." if the player has been to more than 50 locations. If the player has been to more than 50 locations and has also issued more than 12 commands in this scene, the additional comment about wasting time is also printed. If the "wasting time" statement had been set up in a separate IF-THEN, it would be printed when LOOP# was greater than 12 regardless of what VISITS# had reached.

Make sure that *every* IF statement in a nested structure has a matching END or EXIT statement.

Chapter 6
Advanced Adventure Code

World Builder has three main categories of variables: string variables, world variables, and numeric variables. The first two categories were covered in the last chapter, as were the basic numeric variables.

There are, however, many other numeric variables available. Numeric variables can be divided into three categories: basic, user-defined and player attributes.

User Variables

User variables stand for numbers, just like other numeric variables. With user variables, however, you get to decide just what number is stored in the variable, and what it stands for.

A user variable is a letter/number combination of a single uppercase letter (A-Z) followed by a single digit (1-9); the usual numeric variable sign (#) is also part of the variable name. The following are examples of allowed user variable names:

A1# S5# Y7# G3#

To make a variable stand for a specific number, use the LET statement:

```
LET{A1#=10}
```

assigns the value 10 to the variable A1#. Numbers assigned to variables must be whole numbers in the range -32768 to 32767; you can't use commas in the number.

You can make a variable equal to a "constant," an actual number as in the above example, or you can make it equal to another variable:

```
LET{Z2#=LOOP#}
LET{K4#=VISITS#}
```

You can also use the basic mathematical operators of addition, subtraction, multiplication (*), and division (/) when you assign a value to a variable. So, you can use:

```
LET{V1#=VISITS#+LOOP#}
LET{J2#=A2#+B2#}
```

When you want the value of a variable to change without referring to any other variable--to increase it by 1, for instance, you do it this way:

```
LET{A1#=A1#+1}
```

Referring to the variable itself when you are changing its value means you want the new value to be calculated from the old value. The last example adds 1 to the variable A1# because

it says "the value of A1# is now 1 more than what it used to be." Cutting a variable value in half would be done like this:

LET{B2#=B2#/2}

Using User Variables

Now you know how to assign variable values...but why would you want to? Any time you want to keep track of something in your game, you can do it with numbers. When you create the player character, you define how many objects she is allowed to carry. But, what if you want another character to interact with the player based on how much she owns? Maybe the monster's greed is triggered by the sight of a full back pack. In that case, you'd want to know how many items the player is carrying.

Planning ahead, you set up a variable to be worth 0 in the beginning of the game, in the initial scene code:

LET{P1#=0}

All during the game, any time the player visits a particular scene, you can keep track of this with:

LET{P1#=P1#+1}

If you put this in a scene, every time the player enters that scene, P1# is increased by one. On the eleventh visit, you can set up something special with this condition:

IF{P1>10}...

In programmer jargon, this is called a counter. Increasing a counter like this is also referred to as "incrementing a variable."

Random Numbers

Using RANDOM# to set up randomly-occurring events adds the element of chance to your world, beyond that provided by using the random world variables of RANDOMCHR@, RANDOMOBJ@, and RANDOMSCN@.

World Builder can generate random numbers for you when you use the RANDOM# variable. RANDOM# gives a number from 1 to 100. The statement:

LET{B1#=RANDOM#}

assigns a random number to the variable B1#. If you then use the statement:

IF{B1#=1}....

the statements in the IF block will be executed only 1% of the time.

When you want to generate a random event with chances higher or lower than 1%, there are two basic approaches. You can assign a random number that you know is lower or higher than 100 to a variable :

LET{Z3#=RANDOM#/2}

assigns a number from 0 to 50 to Z3#--the random number is divided by 2, and if it is not a whole number, it's rounded down to a whole number.

LET{Q4#=RANDOM#+RANDOM#}

assigns a number from 2 to 200 to Q4#. (If you use RANDOM#*2, you'll get only even numbers from 2 to 200.)

Or, you can use the RANDOM# variable directly in your IF statement:

IF{RANDOM#>50} has a 50% chance of being true
IF{RANDOM#>66} has a 30% chance of being true
IF{RANDOM#<10} has a 10% chance of being true

When you use a statement like

MOVE{RANDOMCHR@}TO{bedroom}

any old character or monster will appear in the bedroom. If you want to randomly choose from a smaller pool of characters--say, just one of two--you can use RANDOM# to do the choosing, like this:

```
LET{C1#=RANDOM#}
IF{C1#<51}THEN
  MOVE{goblin}TO{bedroom}
EXIT
IF{C1#>50}THEN
  MOVE{vampire}TO{bedroom}
EXIT
```

In this last example, note that you must first assign the random number to a variable. If you use the two statements:

IF{RANDOM#<51}
IF{RANDOM#>50}

a different random number will be generated for each condition, and the statements will no longer be mutually exclusive.

Player Attributes Variables

The third type of numeric variable in World Builder is the attribute variable that keeps track of the strength of the player's attributes. Using these attribute variables, you can change the physical and spiritual characteristics of the player character during the course of the game, and you can use her current condition as

a test in an IF statement. (Note that the physical and spiritual attributes are also changed automatically during the course of a game, when the player is engaged in a battle with another character.)

A player's attributes are rated from 0 to 255. You may want to increase or decrease a specific attribute in response to something the player does:

```
IF{TEXT$="take magic ring"}THEN
  LET{SPIR.STR.CUR#=SPIR.STR.CUR#+25}
END
```

Picking up the magic ring WOULD increase the player's current spiritual strength by 25 points.

You might want to make the player's attributes a condition for something else to happen. If you only want a particularly difficult spell to work when her spiritual accuracy rating is at a peak, you can start with:

```
IF{SPIR.ACC.CUR#>220}THEN...
```

Player attribute variables refer to the player's current ratings as well as the base ratings set when you created the character. So, if you want something done only when the player has improved herself a certain degree, you can use something like:

```
IF{SPIR.ACC.CUR#>SPIR.ACC.BAS#+75}THEN...
```

A complete list of player attribute variables is in Appendix B.

Global Code

World Builder is set up so it can run a game for which you've designed scenes, objects, and characters, even if you haven't written any scene code at all; this works because there are certain default responses and activities that occur without your having to do any programming.

You know that you can alter the automatic responses by writing scene code, but there is still another level of coding between the scene code and the default code--the global code.

There are some statements that you might want to have in every scene code window because you want the game to constantly check something. If you always want the player warned when his strength is waning, you would need a something like:

```
IF{PHYS.STR.CUR#<50}THEN
  PRINT{You are very weary}
END
```


Putting this in every scene code window, however, would take up a lot of memory, so World Builder provides a Global Code option.

In order to utilize the Global code effectively, you have to understand how a game works. After scene code is executed, the game goes on to check the global code that you wrote; only after that are the default commands (e.g. LOOK) used.

An EXIT statement executed in a scene code, however, not only stops the game from reading the rest of the scene code, it prevents the game from going on to the global code. The game returns to the stage where it waits for the player to do something. Then, the process starts again after the player gives a command or clicks on an object--the scene code is scanned again.

If the game gets through the scene code without an EXIT being executed, it moves on to the global code. The same thing happens here--if an EXIT is executed, the game loops back and waits for the player to do something; it does not go on to the default commands. Once the player does something, the code scanning starts again at the scene level.

Coding on the global level is the same as on the scene level. To write Global code, use the **Open Global Code** command from the Window menu and enter the code you want to use.

LOOP Zero

Some scenes need to be set up before the player sees them. You may want to make sure that certain characters or objects are in place before the scene is drawn. You might also want to set up a special Commands menu with the MENU command discussed in the last chapter.

When the player arrives at a new scene the following things happen:

1. The game sets TEXT$ equal to "look" so that the scene will be described with the text you entered in the Text window when you created the scene.
2. The game sets the numeric variable LOOP# to zero because it has to count the number of commands given in a scene, and must start at zero.
3. The game scans the scene and global code, then the default commands.
4. The scene and text windows are drawn and the game waits for user input.

If you move characters or monsters into a scene during the "zero loop"--when LOOP# is equal to 0--they will be there by the time the scene is drawn on the screen, because the code is

scanned before the windows are drawn. You can put zero loop commands in the scene or global code.

If you want a random monster moved to the current scene if the player is carrying a special lamp, you can use this IF-THEN in the global code window:

```
IF{LOOP#=0}AND{lamp=PLAYER@}THEN
  MOVE{RANDOMCHR@}TO{SCENE@}
END
```

The SOUND Statement

Using the SOUND command in your World Builder code is simple: use the command followed by the name of the sound:

```
SOUND{scream}
```

Names of sounds are also used when you create scenes, objects, and characters; DATA windows for characters, for instance, ask you to name the sound that is used when a character first appears, when it dies, when it is losing a fight, and so on. All you have to do is put the name of the sound in the data box.

That's easy --but how do you get the sounds into the file that you're creating? Well, read on, that's what the next chapter is all about.

Chapter 7
Working With Sound

The Sound List Window

The Sound List window displays all the sounds that are in the open world. Sounds can be selected (one at a time) by clicking on their names within the Sound List. When a sound is selected the three buttons along the bottom of the Sound List Window become active.

Clicking on the Play button will play the selected sound at the current volume (set using the Control Panel desk accessory).

Fig. 7-1. The Sound List Window

Clicking on the Name button will bring up a dialog allowing you to change the selected sound's name. Edit the name, then click OK to record the name change, or click Cancel to retain the old name.

Important Note: Sounds are accessed by name within a given world. This means that when a sound's name is changed all references to the sound within the game (for example in the Character and Scene Data screens) must be manually changed to the new name.

Clicking on the Info button will display the size of the sound in bytes. This size is the amount of disk space used by the sound.

Clipboard Operations with Sound

A selected sound in the Sound List Window can be cut or copied using the standard editing commands found in the Edit menu. When a sound is on the clipboard and the Sound List window is selected the Edit menu's Paste command becomes active. Pasting will cause the sound on the clipboard to be added

to the end of the sound list. The newly pasted sound will be named 'Untitled' and will be selected. You can then click the play button to hear and identify the sound and then click the name button and give it a new name.

The Scrapbook desk accessory can be used to transfer several sounds at once between games. Sounds in the Scrapbook will be of type 'BSND' (see the lower right corner of Fig. 7-2). Since sounds aren't text or pictures, nothing will be displayed in the Scrapbook window. The sounds can be identified by pasting them into the Sound List window and then playing them.

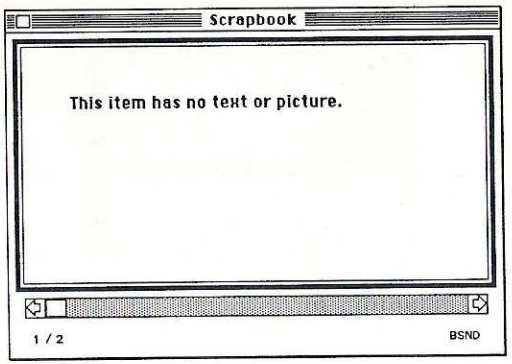


Fig. 7-2 The Scrapbook with a sound file

Sound Library Files

Since sounds are typically the largest objects in a World Builder game, there is a facility provided for storing sounds in files separate from games. These files are called Sound Libraries. A Sound Library, like a game, can contain one or more named sounds.

Fig. 7-3. Sound Library icon

Working With Sound Libraries

Sound Libraries are opened by World Builder in the same way that games are: by choosing Open from the File menu. All

available Sound Libraries (as well as all available games) will be included within the list of files displayed in response to the Open command. After selecting a sound, a dialog will be displayed informing you that the file is a Sound Library. Clicking OK here completes the process of opening the file.

When a Sound Library is open a Sound List window displaying the names of all the sounds in the Library appears on the screen. The Scene Map, Character List, and Object List will be absent and all menu commands applying to them will be dimmed, since these elements are not present in a Sound Library.

The Sound List window for a Sound Library works exactly the same as a Sound List Window for a game. You can Play, Name, or get Info about a sound just as explained in "The Sound List Window" section above. All clipboard operations with sounds in a Sound Library also work the same as those for sounds in games (see the "Clipboard Operations with Sound" section above).

Sound Libraries can be opened within World Builder and their sounds easily transferred to a game through the clipboard or Scrapbook. But sounds in a Library don't have to be pasted into a game to be used within that game, as explained below.

Using Sound Libraries From Games

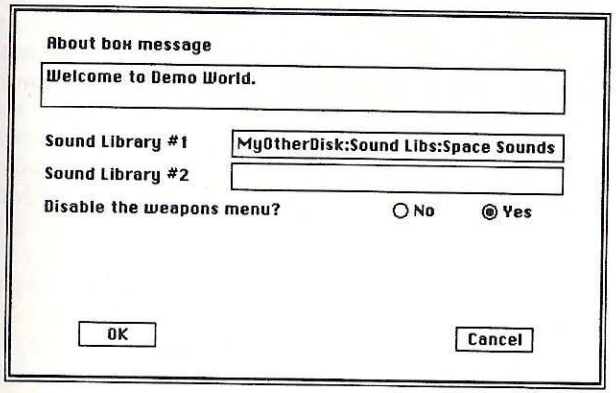


Fig. 7-4. The World Data dialog box

A World Builder game can optionally designate one or two Sound Libraries to be used by the game when it runs. These Sound Libraries will be automatically opened by the game when it is started, and the sounds within them will be used as though

they were in the game itself. Figure 7-4 shows the World Data dialog where the Sound Libraries can be specified. This dialog is brought up by choosing 'Open World Data...' from World Builder's Window menu.

To have a game automatically use a Sound Library simply enter the Library's name in the box labeled 'Sound Library #1,' or for a second Library, the box labeled 'Sound Library #2.' A Sound Library can be on a separate disk and/or in a separate folder from a game that uses it. If this is the case then a full pathname to the Sound Library file must be specified. A full pathname is the list of disk and folder names, separated by colons, that must be traversed to find a file. For example: in Fig. 7-4, a Sound Library named "Space Sounds" is on a disk named "MyOtherDisk" in a folder named "Sound Libs," so its full pathname is "MyOtherDisk:Sound Libs:Space Sounds."

Important Notes:

(1) The spelling of the file names in the World Data dialog is critical. If a Sound Library's name is misspelled or its disk is not online at the time the game starts then the game will not find and open the Sound Library. When this happens there is no error message displayed by the game. The sounds are treated as if they aren't available: the game will run normally, but where a sound would occur nothing will be played.

(2) Sounds are accessed by name within World Builder games. This means that a sound in a Sound Library must not have the same name as any other sound in the game that uses it, or the results will be unpredictable.

Sounds in Scenes

You can link a sound to a particular scene by specifying a "scene sound" in the DATA window for the scene.

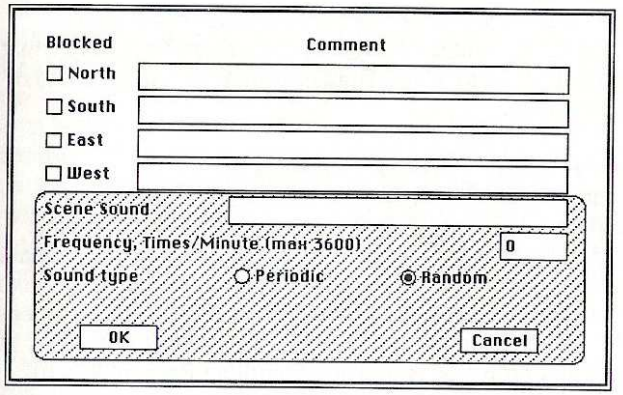


Fig. 7-5. Sound in a Scene

Once you fill in the name of the scene sound, you can decide whether it will occur at specific or random intervals while the player remains in the scene. Use the radio buttons to select Periodic (specific) or Random time intervals between sounds. If you choose Periodic, you should also specify the exact time interval by filling in the Frequency edit field; the number you enter is how many times per minute the sound occurs, so entering 4 means it will happen every 15 seconds.

There's a sample sound library on your World Builder disk called (what else?) "Example Sound Library" that you can use to experiment.

Sources of Sounds

There are many ways to obtain new digitized sounds for use in your own World Builder games. Your World Builder master disk contains a Sound Library called "Example Sound Library" that contains several sounds.

Sounds can be copied from games created by others. This includes the game "Enchanted Scepters," published by Silicon Beach Software, which contains over 30 sounds.

Silicon Beach Software also publishes disks containing nothing but sounds. These Sound Library Disks each contain one or more Sound Library files organized around a particular

theme. A flier which describes the contents of these disks and gives ordering information is included with World Builder.

Owners of the MacNifty™ Audio Digitizer can use sounds digitized with their MacNiftys. A utility program called **Sound Converter** is included with World Builder. The Sound Converter takes sounds stored in the MacNifty Digitizer format and converts them to World Builder format for use in games or Sound Libraries.

The MacNifty™ Audio Digitizer

Sounds digitized with the MacNifty Audio Digitizer are captured and stored using the SoundCap™ application that is included with the MacNifty hardware. The SoundCap application is capable of storing sounds in many different formats. In order for World Builder's Sound Converter to access them, the sounds must be stored in a particular format. The sounds to be used by World Builder must be stored at a Sampling Ratio of 2, with SoundCap's Data Compression and Studio Session Instrument settings turned off (refer to the SoundCap manual for explanations of the SoundCap terms and settings).

Important Note: Sounds for World Builder cannot be longer than 65,535 bytes (about six seconds playing time). The length of a sound can be checked within SoundCap using the Buffer Size command found in the Special menu.

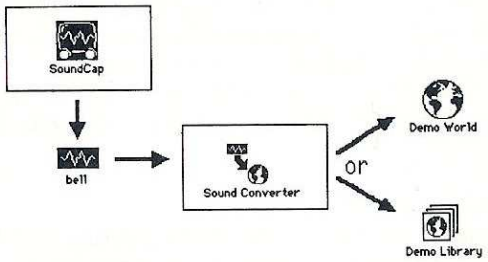


Fig. 7-6. Sound Conversion Process

Fig. 7-6 illustrates the sound conversion process. Once a sound has been stored in the proper format using SoundCap, the Sound Converter is run to convert the sound to World Builder format and add it to a game or Sound Library.

Using the Sound Converter

To run the Sound Converter, double-click on its icon. It will start up and display an initially empty desktop. Apple, File, and Edit menus are available.

Before a MacNifty sound file can be opened and converted, a destination Game or Sound Library must be selected. To select an existing game, choose "Open Game..." from the File menu, select a game from the list of files displayed and then click the Open button. To select an existing Sound Library choose "Open Library..." from the File menu, select a Sound Library from the list of files displayed, and then click the open button. To create a new, empty Sound Library choose "New Library..." from the File menu. A dialog will appear allowing you to name the new Library and select a disk and/or folder for it to reside in. When the file name and disk/folder are as you want them, click the New button and a Sound Library will be created and selected.

When a game or Sound Library has been selected, a window containing the list of sounds already in the file will be displayed, as shown in Fig. 7-7. A new Sound Library will have an empty window, since it contains no sounds yet. The window title will show the name of the file, and whether it is a game (Game) or a Sound Library (Lib).

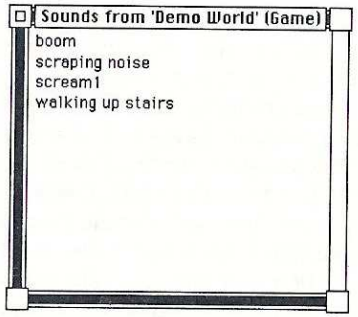


Fig. 7-7. The Sound Converter Window

As sounds are converted and added to the game or library, their names will be added to the end of the list of sounds. This list of sounds is for reference only, it may be scrolled forward or backward as appropriate to view all of the names, but sounds in the list cannot be selected.

When a game or Sound Library is selected the "Convert Sound..." command in the File menu becomes active. To convert a sound, choose this command. A dialog will appear allowing you to select a MacNifty sound file to be converted.

When you have selected the file you want to convert, click on the Open button. The Sound Converter will then open the sound file and verify that it is the proper format and size. If the sound is compressed or too long, an appropriate error message will be displayed and the file will be closed.

Important Note: Since SoundCap doesn't store the sampling ratio when it saves uncompressed sounds, Sound Converter has no way to verify that a sound that it converts was really at the required sampling ratio of two. So you can accidently put a sound with the wrong sampling ratio into a game or library. If you ever have a sound which sounds too slooooowwww when played in World Builder then it was recorded at a sampling ratio of 1. Sounds that sound too fast (like munchken voices) were recorded at sampling ratios of 3 or 4. When you find a sound like this in a game or library just delete it using World Builder's Cut command found in the Edit menu.

If the file is in the correct format, a dialog will appear allowing you to enter a name and repeat count for the sound.

The name will default to the name of the MacNifty file that contains the sound. Remember that names must be unique within a game or Sound Library.

The repeat count, which defaults to 1, is the number of successive times that the sound will be played each time it is requested by World Builder. For example: to get a drum beat sound in a game you might record a single hit of a drum and then use a repeat count of four to play four beats each time the drum is requested. To preview how a sound will work when repeated use the Set Repeat Count command in SoundCap, and then click on SoundCap's speaker icon to play the sound with the repeat count you have just set.

Important Note: Once a sound's repeat count has been set by the Sound Converter it cannot be changed. You must re-convert the sound, specifying the new repeat count you want.

When you are happy with the name and repeat count you have entered for a sound, click the OK button. If you change your mind about converting a sound, click the Cancel button to stop the conversion and close the MacNifty file.

If you clicked OK, a progress dialog will appear. It will first say "Reading MacNifty Sound...". When the sound has been read it will display "Converting Sound..." and a graphical progress indicator. When the sound has been converted it will be added to the game or library while the message "Writing World Builder Sound..." is displayed. After the sound has been successfully converted and written to disk, its name will appear at the end of the list of sounds in the window.

There is no Save command in the Sound Converter since the Game or Sound Library is automatically saved each time a new sound is added to it. When you are finished adding sounds to a game/library, choose Close or Quit from Sound Converter's File menu. Since the file has already been saved, closing and quitting are very fast operations.

Sound Quality

Sounds are stored by Sound Converter and World Builder in a compressed form that can significantly reduce the disk and memory space required for many sounds. There is one drawback to the compression scheme used in World Builder: some high frequency (high pitched) sounds can be distorted by the compression/decompression process.

If a sound sounds much worse when played by World Builder than it did when played by SoundCap there are three possible solutions. First, try slowing down the sound using SoundCap's ScratchBar command found in the Sound Effects menu. This will usually correct the problem, but may require slowing down the sound to the point where it no longer sounds like what was intended. Second, try reducing the sound's amplitude: choose Set Amplify from SoundCap's Settings menu and enter 0.5 for the Amplification Factor; now choose Amplify from the Sound Effects menu to lower the sound's amplitude. If this fails to produce better quality in World Builder, or becomes too faint, there is a third option. Try to record or produce a similar sound without all of the high frequencies found in the problem sound.

Appendix A
World Builder Specifications

Switcher Configuration

Preferred Memory Size:	512K
Minimum Memory Size:	250K
Save Screen:	Off

Important Note: Do **NOT** run a game in Switcher at the same time that you are editing it!

Compatibility

World Builder runs on Macintosh 512K, Macintosh Plus, Macintosh 512 Enhanced, Macintosh XL (sounds can be manipulated, but not played), and is compatible with all memory upgrades.

Games created by World Builder run on Macintosh 128K (except very large games, greater than 400K in size), Macintosh 512K, Macintosh Plus, Macintosh 512 Enhanced, Macintosh XL (sounds not played), and are compatible with all memory upgrades.

Limits

Game Size:	16 Megabytes
Sound Library Size:	16 Megabytes
Number of Scenes per game:	2,500
(Scene Map is limited to 50 x 50 scenes)	
Number of Characters per game:	32,767
Number of Objects per game:	32,767
Number of Sounds per game:	32,767
Item Name:	255 characters
(Scene, Character, Object, or Sound)	
Design Size:	20,480 bytes of graphics
(per Scene, Character, or Object Design)	
Text Size:	10,240 characters
(per Scene Text, per Scene Code, or Global Code per Game)	
Text in Data Dialogs:	200 characters per field
(per field of Scene, Character, Object, or World Data)	
Sound Size:	65,536 bytes uncompressed
(Played at 11,100 samples/second = 5.90 seconds playing time)	

Appendix B
Adventure Code Summary

Text Variables

CLICK$	=	The name of the object or character the player clicked on
TEXT$	=	The command the player typed in or selected from the Command menu

Numeric Variables

LOOP#	= The number of commands the player has given at the current scene
RANDOM#	= A random, uniformly distributed integer between 1 and 100
VICTORY#	= The number of monsters killed
VISITS#	= The number of scenes the player character has visited
PHYS.ACC.BAS#	= The base physical accuracy of player
PHYS.ACC.CUR#	= The current physical accuracy of player
PHYS.ARM.BAS#	= The base physical armor of player
PHYS.ARM.CUR#	= The current physical armor of player
PHYS.HIT.BAS#	= The base physical hit points of player
PHYS.HIT.CUR#	= The current physical hit points of player
PHYS.SPE.BAS#	= The base physical speed of player
PHYS.SPE.CUR#	= The current physical speed of player
PHYS.STR.BAS#	= The base physical strength of player
PHYS.STR.CUR#	= The current physical strength of player
SPIR.ACC.BAS#	= The base spiritual accuracy of player
SPIR.ACC.CUR#	= The current spiritual accuracy of player
SPIR.ARM.BAS#	= The base spiritual armor of player
SPIR.ARM.CUR#	= The current spiritual armor of player
SPIR.HIT.BAS#	= The base spiritual hit points of player
SPIR.HIT.CUR#	= The current spiritual hit points of player
SPIR.STR.BAS#	= The base spiritual strength of player
SPIR.STR.CUR#	= The current spiritual strength of player
A1#...Z9#	= 234 user variables, signed integers

World Variables

PLAYER@	= The player character
MONSTER@	= The current monster, if any
RANDOMCHR@	= A randomly selected character
RANDOMOBJ@	= A randomly selected object
RANDOMSCN@	= A randomly selected scene
STORAGE@	= A scene that can't be visited

IF-THEN Conditional Statement

IF{condition 1}**THEN**
...
EXIT or **END**
IF{condition 1}**OR**{condition 2}...**OR**{condition n}**THEN**
...
EXIT or **END**
IF{condition 1}**AND**{condition 2}...**AND**{condition n}
THEN
...
EXIT or **END**

PRINT Statement

PRINT{any text}

SOUND Statement

SOUND{sound name}

MOVE Statement

MOVE{charactername or variable}**TO**{scenename or variable}
MOVE{objectname or variable}**TO**{scenename or variable}
MOVE{objectname or variable}**TO**{charactername or variable}

LET Statement

LET{numeric variable=num variable, constant}
LET{numeric variable=num variable, constant+num variable, constant}
LET{numeric variable=num variable, constant-num variable, constant}
LET{numeric variable=num variable, constant*num variable, constant}
LET{numeric variable=num variable, constant/num variable, constant}

MENU Statement

MENU{item 1;item 2;...item n}

Index