

Logo Activity 10

Decision Statements

So far when you have written procedures with recursive calls to repeat an action, you needed to use the HALT button to interrupt the program. This is not very useful if you want the program to stop automatically or if you want to have the user ask it to stop. To stop an infinite loop, you need to learn about the IF statement. IF statements are Logo commands that allows you to make decisions based on some condition. Start by typing:

```
SETSCREENCOLOR 3
```

```
SHOW SCREENCOLOR
```

The number 3 appears in the Command Center. SHOW is a command used to display text in the Command Center. SCREENCOLOR is a reporter that returns the number of the current background color. Now define this procedure:

```
TO CHANGE
  SETSCREENCOLOR RANDOM 16
  SHOW SCREENCOLOR
  WAIT 50
  CHANGE
END
```

Type

```
CHANGE
```

and watch the background color change as the current SCREENCOLOR numbers appear in the Command Center.

Suppose you wanted to change the background color randomly until the background becomes black. Modify CHANGE as follows:

```
TO CHANGE
  SETSCREENCOLOR RANDOM 16
  IF (SCREENCOLOR > 0) [WAIT 50 CHANGE]
END
```

This new statement is called a decision statement. The condition follows the keyword "IF", in this case, "SCREENCOLOR > 0". You can then read the statement as follows: "If the screen color is greater than zero, WAIT 50, and then call the CHANGE procedure.

Logo checks to see if the condition (SCREENCOLOR > 0) is true. If it is true, it runs the list of commands found within the brackets([]). The list may contain several commands, however including too many commands may make the command difficult to read. In such a case, it may be better to call a procedure.

When you run CHANGE, you see that the background color changes continuously until it turns the background color to black (0). Sometimes the screen color will change color many times; sometimes not at all. The number of changes depends on the number reported by RANDOM. When a SCREENCOLOR of 0 is set the IF condition is false and the next statement is executed, in this case the END statement.

This procedure can be altered to allow the user to determine the SCREENCOLOR. This is done by allowing a user to enter a value from the keyboard while the program is executing.

```
TO CHANGE
  CT
  PRINT [WHAT BACKGROUND NUMBER WOULD YOU LIKE? ]
  SETSCREENCOLOR READWORD
  IF (SCREENCOLOR > 0) [WAIT 50 CHANGE]
END
```

In this version of the program, rather than getting the background color from a randomly generated value the user enters a value as the program is run. Let's take a closer look at the program execution.

1. CT is used to keep the text screen clear of excessive clutter.
2. The PRINT statement prompts the user to enter a value from the keyboard.
3. The SETSCREENCOLOR READWORD command accepts an input from the keyboard that is used to determine the screen color.
4. A screen color other than 0 will make the program continue.

The program has a weakness. If a user enters an **invalid** background value such as an "A", the program quits because of the error. However, placing appropriate user values at the end of the print statement can make user choices more clear. Change the PRINT statement to the following:

```
PRINT [WHAT BACKGROUND NUMBER WOULD YOU LIKE? (0-15) ]
```

You will notice that when a user enters 0 (a black screencolor) the program ends.

Some conditions read more clearly using "not equals". If you wanted to say "If the background is not equal to 0 then call the procedure "CHANGE" you would write:

```
IF NOT (SCREENCOLOR = 0 ) [WAIT 50 CHANGE]
```

Simply place the word NOT in front of the condition.

