
MicroWorlds™ 2.0

Vocabulary

Macintosh Version

Contents

MicroWorlds primitives	5	cos	13
Infix Math Reporters	5	count	13
+ (Addition)	5	createprojectvar	13
- (Subtraction)	5	cu	14
* (Multiplication)	5	cut	14
/ (Division)	5	D	
= (Equal to)	6	delete	14
> (Greater than)	6	difference	14
< (Less than)	6	directories	15
A		distance	15
abs	6	dolist	15
and	6	done?	15
announce	7	dotimes	16
answer	7	E	
arctan	7	empty?	16
ascii	7	eol	17
ask	7	eot?	17
B		equal?	17
back (bk)	8	erfile	17
bg	8	errormessage	17
bottom	8	everyone	18
butfirst (bf)	8	exp	18
butlast (bl)	9	export	18
C		F	
cancel	9	files	19
carefully	9	fill	19
cb	10	first	19
cc	10	fontsize	20
cd	10	forever	20
cf	10	forward (fd)	20
cg	10	found?	20
char	11	fput	21
clean	11	freeze	21
clearname	11	freezebg	21
clearnames	11	G	
cleartext (ct)	11	get	22
clickoff	12	getpage	23
clickon	12	getproject	23
clipboard	12	glide	23
color	12	greater?	23
colorunder	12		
copy	13		

H

heading	24
hidetext	24
home	24
ht	24

I

identical?	25
if	25
ifelse	25
import	26
insert	27
int	27
item	27

K

key?	27
------------	----

L

last	28
launch	28
left (lt)	28
less?	28
let	28
list	29
listen	29
list?	30
ln	30
loadpict	30
loadshape	31
loadtext	31
local	31
log	32
lput	32

M

make	32
member?	32
merge	32
minus	33
mousepos	33

N

name	33
name?	34
namepage (np)	34
names	34
newbutton name	34
newpage	34

newprojectsize	35
newslider	35
newtext	35
newturtle	35
not	36
note	36
number?	36

O

opaque	37
or	37
output (op)	37

P

pagelist	38
parse	38
paste	38
pd	38
pe	39
pensize	39
pi	39
pick	39
pictlist	39
placepict	40
pos	40
power	40
prefix	40
presentationmode	40
print (pr)	40
printtext	41
product	41
projectlist	41
projectsize	41
projectvars	41
pu	41

Q

question	42
quotient	42

R

random	42
readchar	42
recycle	43
remainder	43
remove	43
repeat	43
rerandom	44
resetquicktime	44
reset	44

resetvideo	44	stop	56
rest	45	stopall	57
restore	45	stopme	57
right (rt)	45	sum	57
round	45		
run	45		
		T	
S		talkto (tto)	58
savehtml	46	tan	58
savepict	46	tc	58
saveproject	47	textcount	58
saveshape	47	textitem	59
savetext	47	textlist	59
search	47	textpick	59
select	47	textwho	59
selected	48	thing	60
sentence (se)	48	timer	60
set	48	top	60
setbg	49	touching?	61
setc	49	towards	61
setfont	50	transparent	61
setfontsize	50	turtlesown	61
setfooter	50	turtletype	62
seth	50		
setinstruction	51	U	
setinstrument	51	unfreeze	62
setpenseize	51	unfreezebg	62
setpos	51	unselect	63
setprefix	51		
setshape (setsh)	52	W	
setsize	52	wait	63
setstyle	52	waituntil	63
settc	52	when	63
setx	52	who	64
sety	53	word	64
shape	53	word?	64
show	53		
shownames	53	X, Y	
showtext	54	xcor	64
sin	54	ycor	64
size	54		
snaparea	54		
snapshot	54		
snapshot	54		
sol	55		
soundlist	55		
space	55		
sqrt	55		
stamp	56		
stamptext	56		

MicroWorlds primitives

Following is a description of each MicroWorlds primitive.

Infix Math Reporters

number1 + *number2* (Addition)

Reports the sum of *number1* and *number2*. See **sum**.

```
show 3 + 3  
6
```

number1 - *number2* (Subtraction)

Reports the result of *number1* minus *number2*. See **difference** and **minus**.

```
show 3 - 3  
0
```

number1 * *number2* (Multiplication)

Reports the product of *number1* multiplied by *number2*. See **product**.

```
show 3 * 3  
9
```

number1 / *number2* (Division)

Reports the result of *number1* divided by *number2*. See **quotient** and **remainder**.

```
show 3 / 3  
1
```

word-or-list1 = *word-or-list2* (Equal to)

Reports **true** if *word-or-list1* is equal to *word-or-list2*. An empty word is not equal to an empty list. See **equal?** and **identical?**.

```
show 3 = 3
true
```

number1 > *number2* (Greater than)

Reports **true** if *number1* is greater than *number2*. See **greater?**.

```
show 3.1 > 3
true
```

number1 < *number2* (Less than)

Reports **true** if *number1* is less than *number2*. See **less?**.

```
show 2.9 < 3
true
```

A

abs *number*

Stands for **absolute**. Reports the absolute value of its input.

```
show abs -33
33
```

and *true-or-false1 true-or-false2*
(**and** *true-or-false1 true-or-false2 true-or-false3...*)

Reports **true** if all its inputs report **true**. If more than two inputs are used, **and** and its inputs must be enclosed in parentheses. See **or** and **not**.

```
show and 2 = 2 3 = 5
false
show (and pos = [0 0] heading = 0 shape = 0)
true
```

announce *word-or-list*

Displays the message in a message box. See **question**, and **answer**.

```
announce [You win!!!]
```

answer

Reports the contents of the most recent answer typed in the question dialog box. **Answer** reports the contents as a long word (a sequence of characters including spaces). See *Question and Answer* in Section 3.

```
question [How cold is it?]
```

Type an answer in the dialog box. In the Command Center, type:

```
show answer
```

```
Very cold           Your answer.
```

```
show count answer
```

```
9                   There are 9 characters.
```

arctan *number*

Stands for **arc tangent**. Reports the arc tangent of its input. See **tan** and **cos**.

```
show arctan 1
```

```
45
```

ascii *char*

Stands for **American Standard Code for Information Interchange**. Reports the ASCII number which represents the character. See **char**.

```
show ascii "a
```

```
97
```

ask *who instruction-list*

Temporarily tells each element in the first input to run the instruction list. The first input can be one or more turtles or text box names. **Ask** does not change the current turtle or text box. See **everyone** and **talkto**.

There are many turtles on the page.

```
ask [t1 t2 t3] [fd 50 rt 90 fd 50]
```

If you have two text boxes on the page and you want to keep Text1 current:

```
ask "text2 [print "hello]
```

B

back (bk) *number*

Moves the turtle backwards. The turtle leaves a trace if its pen is down. The maximum number is 9999. See **forward**.

```
t1,  
pd bk 20
```

bg

Stands for **background**. Reports a number representing the color of the background. The background color is 0 when MicroWorlds starts up. See **setbg**.

```
setbg 10  
repeat 9 [setbg bg + 1]
```

bottom

Puts the cursor, in the current text box, at the end of the text. See **top**, **sol**, **eol**, and **eot?**.

```
text1,  
repeat 5 [print "hello]  
top  
bottom
```

butfirst (bf) *word-or-list*

Reports all but the first component of a word or list. See **butlast**, **first**, and **last**.

```
show butfirst [0 1 2 3]  
1 2 3  
show butfirst "hello  
ello
```

butlast (bl) *word-or-list*

Reports all but the last component of a word or list. See **butfirst**, **first**, and **last**.

```
show butlast [0 1 2 3]
0 1 2
show butlast "welcome
welcom
```

C

cancel *instruction-list*

Stops the process given as input. The process must have been launched by **launch**, **when**, **forever**, buttons, and clickable turtles. Corresponds to choosing Cancel from the Edit menu. The input must be the exact same instruction list that started the process. See **launch**, **when**, and **forever**.

```
t1,
forever [fd 1]
forever [rt 1]
cancel [rt 1]
cancel [fd 1]
```

carefully *word-or-list-to-run1 word-or-list-to-run2*

Runs the first list of instructions. If the first list contains an error, **carefully** runs the second list of instructions and sets **errormessage** to the error that occurred. If there is no error in the first list, the second list is ignored. See **errormessage**.

```
carefully [fd 50] [announce [No turtle!]]
```

If you try this with a turtle on the page, it will go forward. If you try it without a turtle on the page, the message will be displayed in a message box.

```
carefully [print "hello] []
```

If you try this with a text box on the page, hello will be printed. If you try it without a text box, nothing will happen. The error message will not be displayed.

cb

Stands for **cursor back**. Moves the cursor, in the current text box, to the previous character. See **cf**, **cd**, and **cu**.

```
text1,  
insert "hello  
repeat 5 [cb cb cf cf]
```

cc

Stands for **clear the Command Center**. Clears the text in the Command Center.

```
cc show "hello
```

cd

Stands for **cursor down**. Moves the cursor, in the current text box, to the next line. See **cu**, **cf**, and **cb**.

```
text1,  
repeat 5 [print "hello]  
top  
repeat 5 [cd wait 5]
```

cf

Stands for **cursor forward**. Moves the cursor, in the current text box, to the next character. See **cb**, **cd**, and **cu**.

```
text1,  
insert "hello  
top  
repeat 5 [cf insert "x]
```

cg

Stands for **clear graphics**. Clears the graphics on the page and returns the current turtle to the position [0 0], facing up.

Graphics include drawings made using the turtle commands and the drawing tools, stamped turtles, stamped text, and stamped frames from movies, and text made using **turtletype**. See **clean** and **freezebg**.

```
fd 50 rt 90  
stamp fd 50  
cg
```

char *number*

Stands for **character**. Reports the character represented by the ASCII number given as input. The number must be between 0 and 255. See **ascii**.

```
show char 97
a
show char 65
A
```

clean

Clears the graphics without changing any turtle's position. **Clean** does not affect the background color set by **setbg**. You can use Undo from the Edit Menu to undo this effect. See **cg** and **freezebg**.

```
setsh 12
fd 50
clean
```

clearname *word*

Clears a global variable from memory. See **make** and **name**.

```
make "speed 5
show :speed
5
clearname "speed
show :speed
speed has no value
```

clearnames

Clears all global variables from memory. See **make** and **name**.

```
make "speed 5
clearnames
show :speed
speed has no value
```

cleartext (*ct*)

Clears the text from the current text box.

```
text1, print "hello
cleartext
```

clickoff

Simulates a mouse click on the current turtle, turning it off if it was on. This command will only have an effect if the turtle is programmed to react to a mouse click. See **clickon**.

Type an instruction in the turtle's dialog box, and set it in Many Times mode. Click on the turtle to make it run its instruction, then type this in the Command Center:

```
t1, clickoff
```

clickon

Simulates a mouse click on the current turtle, turning it on if it was off. This command will only have an effect if the turtle is programmed to react to a mouse click. See **clickoff**.

Type an instruction in the turtle's dialog box. Then type this in the Command Center:

```
t1, clickon
```

clipboard

Reports the contents of the text Clipboard. The Clipboard contains the last text that has been cut or copied using the **cut** or **copy** command, or the equivalent Edit menu items.

```
show clipboard  
My friend Kim
```

color

Reports the turtle's color as a number. **Color** reports a number, even if a name was used as input for **setc**. It can be a decimal if the project was created in the "thousands" color mode. See **setc**.

```
setc "red  
show color  
15  
setc color + 1
```

colorunder

Reports the color under the current turtle as a number. The portion of a turtle that recognizes a color is its center.

Colorunder reports the background color as well as all the drawings. If the project was created in the 256 color mode, the output is an integer. It can be a decimal if the project was created in the “thousands” color mode.

```
t1, show colorunder
3
```

In a stop rule, always use the color *number*, not its name, to check the color under the turtle:

```
if colorunder = 15 [announce [You win]]
```

copy

Puts a copy of the selected text in the Clipboard. See **select**, **cut**, and **paste**.

Type some text in a text box and type the following instructions in the Command Center.

```
top select cd
copy
ct
```

paste paste The selected text is pasted twice in the text box.

cos *number*

Stands for **cosine**. Reports the cosine of its input. See **sin** and **tan**.

```
show cos 60
0.5
```

count *word-or-list*

Reports the number of components in the word or the list. See **item** and **textcount**.

```
show count "hello
5
```

```
show count [this is a list]
```

```
4
```

show count text1 Text1 is the name of a text box.

```
31
```

createprojectvar *word*

Stands for **create project variable**. Creates a project variable represented by a command and a reporter. For example, if the

project variable “amount” is created, the command **setamount** sets its value, and **amount** reports its value. Project variables are saved with your project. See **projectvars**, **make** and **name**.

```
createprojectvar "amount
setamount 22
show amount
22
```

CU

Stands for **cursor up**. Moves the cursor in the current text box, to the previous line. See **cd**, **cb**, and **cf**.

```
text1,
repeat 5 [print "hello]
repeat 5 [cu cu cd cd]
```

cut

Deletes the text selection in the current text box and puts a copy in the Clipboard. See **select**, **copy**, and **paste**.

Select some text in a text box and click in the Command Center to type the following commands:

```
cut
paste paste    The selected text is pasted twice in the text box.
```

D

delete

Deletes the character to the right of the insertion point, in the current text box.

Type some text in a text box and put the cursor in the middle of the text:

```
delete
```

difference *number1 number2*

Reports the result of subtracting *number2* from *number1*. See - and **minus**.

```
show difference 7 3
4
```

directories

Reports a list of subdirectory (folder) names. To change directories through a command, use **setprefix**.

distance *turtle-name*

Reports the distance from the current turtle to another turtle. In this example, there are two turtles on the page.

```
t1,  
show distance "t2  
122.4           Your number will be different.  
towards "t2  
fd distance "t2           T1 meets t2.
```

dolist *range instruction-list*

Runs the instruction list for each item in a list. The first input, *range*, is a list with a temporary variable name and a list of items. The second input is a list of instructions that uses the variable name included in the first input. In the following example, the instruction **remove :i** is run for each item of the first list. “**T**” successively takes the value t1, t2, and t3. See **dotimes**.

If there are three turtles on the page, the next instruction removes all three turtles. The second instruction will remove all turtles, whatever the number.

```
dolist [i [t1 t2 t3]] [remove :i]  
dolist [i get "page1 "turtles] [remove :i]
```

done? *instruction-list*

Reports **true** if the process in the instruction list is completed. The process must have been launched using **launch** or **forever**. The input must be an exact copy of the instruction list that started the process. **Done?** can be used as an input to **waituntil** in order to synchronize events. See **launch** and **forever**. For other processes, such as melodies and movies, use the **get** primitive with **on?** as input. See **get**.

Circle and **square** are procedures, and there are two turtles on the page.

In the following procedure, t1 draws a circle while t2 is drawing a square. It takes longer to draw a circle, but MicroWorlds will wait for both shapes to be finished before telling the turtles to go elsewhere on the page to draw more circles and squares.

```
to sq-circ
t1, launch [circle]
t2, launch [square]
waituntil [done? [circle]]
waituntil [done? [square]]
t1, rt random 360 fd random 50
t2, rt random 360 fd random 50
sq-circ
end
```

dotimes *range instruction-list*

Runs the instruction list for each value specified in the range. The first input is a list with a temporary variable name and a maximum number. The second input is a list of instructions that uses the variable name included in the first input. See **dolist**.

In the following example, the instruction **setbg** sets the background color for each value of **i**, from 0 to 7.

```
dotimes [i 8] [setbg :i wait 5]
```

E

empty? *word-or-list*

Reports **true** if the input is an empty word or empty list.

```
show empty? []
true
show empty? text1
false
```

This procedure can be used to check whether a user has typed an answer in a question dialog box:

```
to insist
question [Your name please...]
if empty? answer [insist]
end
```

eol

Stands for **end of line**. Moves the cursor, in the current text box, to the end of the current line. The line ends where the next carriage return is. See **sol**.

```
pr "hello
top
eol
insert "!
```

eot?

Stands for **end of text**. Reports **true** if the cursor, in the current text box, is at the end of the text. See **bottom**.

```
text1,
bottom
show eot?
true
```

equal? *word-or-list1 word-or-list2*

Reports **true** if the two inputs are equal. The inputs may be words, numbers, or lists. See **identical?** and **=**.

```
show equal? "a "A
true
show equal? [ ] "
false
```

erfile *pathname*

Stands for **erase file**. Erases any type of file (except locked files). The input must be the name of a file in the current directory or a full pathname. Colons are used to separate the names of directories, subdirectories, and files. If there are spaces in the names, enclose the pathnames in vertical bars.

```
erfile "myfarm
erfile "hd:myfiles:myfarm
erfile "|macintosh hd:my files:my farm|
```

errormessage

Reports the last error message trapped by **carefully**. If **errormessage** reports an empty word, it means that the last operation run by **carefully** did not report an error. See **carefully**.

There is no turtle on the page.

```
carefully [fd 50] [show errormessage]  
No turtle found for forward
```

everyone *instruction-list*

Makes all the turtles on the current page run the instruction, one after the other. See **ask**.

Create many turtles on the page.

```
everyone [setsh "cat"]
```

exp *number*

Stands for **exponential**. Reports the *number* to the power of the constant **e** (2.718282).

```
show exp 1  
2.718282
```

export *filename*

Exports a MicroWorlds 2.0 Macintosh project into Windows format. *Filename* should be the name of the Macintosh project that is currently open.

MicroWorlds 2.0 can import Windows projects and export projects to Windows almost seamlessly. All MicroWorlds objects such as turtles, shapes, procedures, text boxes, and buttons are converted automatically. After exporting a Macintosh project, you can run it almost effortlessly on Windows. But there is an important exception. Although the MicroWorlds project can be exported to Windows, the media resources that are external to MicroWorlds are not automatically converted with the project. This means that sounds and movies will not be automatically converted even though the icons will appear in the new Windows project. The media resources included with MicroWorlds 2.0 Windows and MicroWorlds 2.0 Macintosh almost exactly correspond, so if you used these sounds or movies in the project, they will not have to be converted. You can directly copy these resources from your Windows CD ROM.

The project appears as a PC file on the Finder, with *.mw2* added to the *filename*. See the *Importing and Exporting Read Me* accompanying MicroWorlds 2.0 for further information

and the problems/differences once a Macintosh project has been converted. See also **import** to import a MicroWorlds 2.0 Windows project into Macintosh format.

Example:

Before exporting a Macintosh project called Emotions which has a Quicktime sound called Myvoice, first create a folder and then place the original Macintosh project and sound in this folder. Open the Emotions project (the project that you want to convert), then type in the Command Center:

```
export "emotions
```

The file appears as a PC file on the Finder, with the extension .mw2 added. Use an external disk or a network utility to copy this file to your Windows computer. Use a sound conversion utility to convert the Macintosh sound file to a .WAV file. See the *Importing and Exporting Read Me* accompanying MicroWorlds 2.0 for further information.

F

files *filetype*

Outputs a list of files of the given filetype. *Filetype* can be: aiff (a Quicktime sound), moov (a Quicktime movie), pict (a picture), prj3 (MicroWorlds 2.03 and 2.05 projects), prj2 (a MicroWorlds 2.01 project), proj (a MicroWorlds 1.0 project), or text (a text file).

```
show files "text
```

fill

Fills a closed shape or the whole screen with the turtle's color. **Fill** will work regardless of the turtle's pen state (up or down). See **setc**.

```
setc "blue  
fill
```

first *word-or-list*

Reports the first component of the word or list. See **butfirst**, **butlast**, and **last**.

```
show first "hello  
h
```

fontsize

Reports the font size of the text at the insertion point in the current text box. If there is a text selection containing text with more than one font style, color, or size, **fontsize** reports an empty list.

```
text1,  
show fontsize  
12
```

forever *instruction-list*

Runs the instruction list repeatedly as an independent parallel process. Use the Cancel menu item, the Stopall menu item, or **APPLE - .** to stop the processes. See **launch** and **cancel**.

```
forever [rt 1]  
forever [fd 1]
```

forward (fd) *number*

Moves the turtle forward. The turtle leaves a trace if its pen is down. The maximum number is 9999. See **back**.

```
pd fd 20 rt 90  
pu fd 50  
pd fd 10
```

found?

Reports **true** if the last **search** instruction was successful. See **search**.

The following procedure will replace all occurrences of a word in the current text box with another word. Make sure you place the cursor at the top of the text box before running the procedure:

```
to replaceall :this :withthat  
  search :this  
  if not found? [stop]  
  insert :withthat  
  replaceall :this :withthat  
end
```

fput *word-or-list list*

Reports the list created by adding the first input at the beginning of the second input. The second input must be a list. See **lput**.

```
show fput "a [b c d e f]
a b c d e f
show fput "a [bcdef]
a bcdef
```

freeze *word-or-list-of-object***freeze** *page-name*

Freezes a text box, button, turtle, slider, or any object so that it cannot be moved, changed in size, or removed with the mouse. The input is the name of any object on the page, a list containing many names, or the name of a page.

If you have “frozen” a text box, you can’t move it or change its size. You can open its dialog box, type text in it, or use primitives such as **get** and **set** to make changes. If a turtle is frozen, you cannot use the mouse to change its shape, its size, stamp it, or remove it. If a button or turtle is frozen and set to Once mode, clicking it will not make it stop. This feature is particularly useful in interactive projects where you do not want a process to be stopped before finishing. **Unfreeze** undoes the effect of **freeze**.

```
freeze "t1
freeze [text2 text3 text4]
freeze "page1      Everything on Page1 is “frozen.”
```

freezebg

Freezes the background graphics in their current state. You can still draw over the background and erase the new drawings, but the background that was present before freezing won’t be erased. See **unfreezebg**.

```
pd
rt 11 fd 500
freezebg
lt 22 fd 500
cg
```

G

get *object property*

Reports a property of an object in the current project. The first input is the name of an object, a color, or a page. The second input is a property name.

Following is a list of the properties each object can have:

Pages:	turtles, texts, buttons, sliders, melodies, records, videodisks, qtmovies, qtsounds, audiocds, colordemons
Turtles:	visible?, rule, on?, own
Buttons:	pos, size, rule, on?
Sliders:	pos, showname?, limits, value
Text:	visible?, pos, size, transparent?, showname?, text
Colors:	turtlerule, turtlemode, mouseclick, on?
Melodies:	visible?, pos, on?, showname? instrument, volume, tempo
Recordings:	visible?, pos, on?, showname?
Movies:	visible?, pos, on?, showname?
AudioCD clips:	visible?, pos, on?, showname?
Videodisk clips:	visible?, pos, on?, showname?
Announce:	pos, size
Question:	pos, size
Sounds:	quality

See *Creating and Modifying Objects Under Program Control* in Section 3 for more information.

The following examples assume that the objects are in the project.

```
show get "t1" "rule"
forever [fd 1]
show get "t1" "on?"
true
show get "button1" "rule"
launch [page1]
show get "text1" "size"
160 100
show get "t1" "own"
```

```
speed 12
show get "question "size
354 132
show get "sounds "quality
good
```

getpage *page-name*

Displays the page. The input must be the name of a page in the current project. Typing the page name alone has the same effect.

In this example, menu is a page in the current project and this page is not currently showing.

```
getpage "menu
```

getproject *project-name*

Opens the project indicated. The input must be the name of a project in the current directory.

```
getproject "sunset
```

glide *distance speed*

Makes the turtle glide over the distance indicated. The second input indicates how fast the turtle will cover the distance. The distance and speed are in turtle steps. The maximum distance is 9999 and the maximum speed is 99.

```
glide 100 1
glide 100 10
```

greater? *number1 number2*

Reports **true** if the first input is greater than the second input. See **less?** and **>**.

```
show greater? 4 3
true
```

H

heading

Reports the current turtle's heading in degrees. The degrees correspond to those of a compass; 0 degrees is due North, 90 is East, 180 is South, and 270 is West. See **seth**.

```
cg
show heading
0
seth 180
show heading
180
```

hidetext

Hides the current text box. You cannot type in a hidden text box, but primitives such as **print**, **insert**, and **ct** still work. See **showtext**.

If you have a text box on the page, this instruction flashes the text box.

```
repeat 10 [hidetext wait 5 showtext wait 5]
```

home

Moves the turtle to the page coordinates [0 0], pointing up. See **setpos**.

```
t1, pd
fd 50
rt 90
fd 50
home
```

ht

Stands for **hide turtle**. Hides the current turtle. See **st**.

```
t1,
repeat 10 [ht wait 5 st wait 5]
```

I

identical? *word-or-list1 word-or-list2*

Reports **true** if both inputs are identical. A word and a list containing the same word are not identical. Corresponding characters of each input must match in terms of uppercase and lowercase letters. (Style, font, and text color are ignored.) See **equal?**.

```
show identical? "a "A
false
show identical? "hello [hello]
false
```

if *true-or-false instruction-list*

Runs the instruction list only if the condition (first input) reports **true**. See **ifelse**.

```
if colorunder = 15 [fd 50]
```

This procedure can be used to check whether a user has typed an answer in a question dialog box:

```
to insist
question [Your name please...]
if empty? answer [insist]
end
```

ifelse *true-or-false instruction-list1 instruction-list2*

Runs the first instruction list if the condition is **true**. Runs the second instruction list if the condition is **false**.

```
ifelse colorunder = 15 [fd 50] [bk 50]
```

This instruction can be used to check whether a user has typed an answer in a question dialog box. If the answer is not empty, it is displayed in a text box. On the Procedures page, you can format an **ifelse** instruction on several lines so it's easier to read:

```
to insist
question [Your name please...]
ifelse empty? answer
    [insist]
    [settext1 answer]
end
```

import *filename.mw2*

Imports the designated MicroWorlds 2.0 Windows project into Macintosh format. The *filename* must be completely specified, including the extension (mw2). If the file is not a MicroWorlds 2.0 English Windows file, the error message *Invalid File Format* will appear.

All MicroWorlds objects such as turtles, shapes, procedures, text boxes, and buttons are converted automatically. After importing a Windows project, you can run it almost effortlessly on the Macintosh. But there is an important exception. The media resources that are external to MicroWorlds are not automatically converted with the project. This means that sound/midi/movie files will not be converted to Macintosh format when the Windows project is imported even though the icons will appear in the new Macintosh project. (The movie icon is particularly confusing since the movie appears as a poster in Macintosh project, but the movie is inaccessible.) The media resources included with MicroWorlds 2.0 Windows and MicroWorlds 2.0 Macintosh almost exactly correspond, so if you used these sounds/midis/movies in the project, they will not have to be converted. You can directly copy these resources from your Macintosh CD ROM.

Note: Using the application MoviePlayer, sound and midi files from Windows can be converted.

The project will appear as Untitled, so you can name it as you wish. See the *Importing and Exporting Read Me* accompanying MicroWorlds 2.0 for further information about the problems/differences once a Windows project has been converted. See also **export** to export a MicroWorlds 2.0 Macintosh project into Windows format.

Example:

Before importing a Windows project called *flower.mw2*, first create a folder and then place the original Windows project and media resources in this folder. Copy the Windows files into this folder from an external disk or using a network utility. Use Save as from the file menu to access this folder directly.

Then type in the Command Center:

```
import "flower.mw2
```

The project appears on the screen as Untitled. Name the project. See the *Importing and Exporting Read Me* accompanying MicroWorlds 2.0 for further information.

insert *word-or-list*

Prints the input at the cursor position, in the current text box. The text is not followed by a carriage return. See **print**.

```
insert "hi
insert char 32    Prints a space.
pr "there
```

int *number*

Stands for **integer**. Reports the integer portion of its input. See **round**.

```
show int 9.9999
9
show int 2.5
2
```

item *number word-or-list*

Reports the specified element of a word or a list. The first input must be between 1 and the number of elements in the word or the list.

```
show item 2 "hello
e
show item 3 [this is a list]
a
```

K

key?

Reports **true** if a key is being pressed on the keyboard. You must click on the background of the page (outside of a text box, the Command Center, or Procedures page) for **key?** to work. Use **readchar** to reset **key?** to **false**.

First run the instruction, and press a key after MicroWorlds starts displaying the word **false**.

```
repeat 20 [show key? wait 5]
```

Click on the page.

```
false
false
true    You pressed a key.
true
...
```

L

last *word-or-list*

Reports the last component of the word or list. See **first** and **butlast**.

```
show last "hello
o
show last text1
m
```

launch *instruction-list*

Runs the input as an independent process. If **launch** is used in the Command Center, the cursor reappears as soon as the process is launched. Use **cancel** (or the Cancel menu item), the Stopall menu item, or **APPLE - .** to stop the process. See **cancel** and **forever**.

```
t1,
launch [glide 1000 1]
```

The cursor reappears in the Command Center. You can type another command:

```
t2,
launch [glide 1000 1]
```

left (lt) *number*

Turns the turtle to the left. The maximum number is 9999. See **right**.

```
repeat 10 [fd 40 bk 20 lt 36]
```

less? *number1 number2*

Reports **true** if the first number is less than the second number. See **greater?** and **<**.

```
show less? 22 22.5
true
```

let *list-of-names-and-values*

Creates one or more temporary variables. The variables will exist only while the procedure containing the **let** instruction and procedures called by this procedure are running. The input

is a list of paired variable names and values. **Let** can only be used in a procedure. See **local**.

```
to move
let [dist 100 head 90 delay 300]
right :head
wait :delay
fd :dist
end
```

Try the procedure in the Command Center. The turtle will move. When the procedure is over, check the value of the variables — they won't exist.

```
move
show :dist
dist has no value
```

list *word-or-list1 word-or-list2*
(**list** *word-or-list1 word-or-list2 word-or-list3...*)

Reports one list made by combining the inputs (words or lists). If more than two inputs are used, **list** and its inputs must be enclosed in parentheses. See **sentence**.

```
show list 2 3
2 3
show (list "a "b "c "d)
a b c d
make "x 10 make "y 20
setpos list :x :y
```

listen

Sets the global “turtle who” (the turtle obeying instructions from the Command Center). This command allows you to change the global “turtle who” within a process that has been launched (e.g., from a button). See **talkto** and **ask**.

Note: If you used a list as input to **talkto** before running **listen**, **listen** will display an error message.

Example:

Create 2 turtles, t1 and t2. Click on t2 to make it active. Test that t2 is active:

```
fd 50
show who
t2
```

Create a button with **t1, bk 50** as its instruction.

Click on this button. The other turtle, t1, goes back 50 steps.
Repeat the instruction:

```
show who  
t2
```

T2 is still the turtle that obeys instructions from the Command Center. The turtles addressed inside buttons (or other processes) are “local turtles” and don’t affect the “global turtle.” To change the “global turtle” while running a process (e.g., inside a button), use **listen**.

Change the button instruction to **t1, listen**

Click on this button. Write the following instructions in the Command Center:

```
fd 50  
show who  
t1
```

Now t1 is the “global turtle.”

list? *word-or-list*

Reports **true** if the input is a list. See **word?**.

```
show list? [Hello there]  
true
```

ln *number*

Reports the natural logarithm (logarithm in base e) of its input. See **exp**.

```
show ln 10  
2.302585
```

loadpict *pathname*

Stands for **load picture**. Loads a picture file onto the background graphics of the current page. The input must be the name of a PICT file (created with **savepict** or a drawing application) in the current folder or a full pathname. See *Importing Pictures* in section 3.

```
loadpict "scene
```

loadshape *file shape-number*

Loads the specified picture file into that shape. The picture file name will become the name of the shape if there is no other shape with that name in the project. The *file* must be the name of a picture file (saved in PICT format by **savepict** or another drawing application) in the current folder or a full pathname. A pathname starts with the name of the hard disk. Colons are used to separate the names of directories, subdirectories, and files. The *shape-number* can be any number from 1 to 64. See **savepict**.

Example:

```
loadshape "volcano 20
```

Loads the volcano picture into shape 20. The shape will be named volcano.

loadtext *pathname*

Loads a text file into the current text box or the Procedures page. The input must be the name of a file in the current folder or a full pathname. See *Importing Text* in section 3.

```
loadtext "mystory
```

local *word-or-list*

Makes the specified variable local to the procedure where **local** is used. **Local** can only be used in a procedure. See **let**, **make**, and **name**.

```
to move
local "dist
make "dist 100
fd :dist
end
```

Try the procedure in the Command Center. The turtle will move. When the procedure is over, check the value of the variables — they won't even exist.

```
move
show :dist
dist has no value
```

log *number*

Stands for **log**arithm. Reports the logarithm of the *number* given as input, in base 10. See **ln** and **exp**.

```
show log 100
2
```

lput *word-or-list list*

Stands for **last put**. Reports the list created by adding the first input at the end of the second. See **fput**.

```
show lput "f [a b c d e]
a b c d e f
show lput "V [Saturn]
Saturn V
```

M

make *word word-or-list*

Creates a variable and gives it the value *word-or-list*. The variable keeps its value as long as it is not cleared. See **name**, **clearname**, **clearnames**, and **createprojectvar**.

```
make "class [Peter Anne Dennis Geni Eric]
show :class
Peter Anne Dennis Geni Eric
show first :class
Peter
```

member? *word-or-list1 word-or-list2*

Reports **true** if the first input is a component of the second.

```
show member? "a [a b c]
true
```

merge *project-name word-or-list*

Copies pages, shapes, or procedures from another project into the current project. The first unput must be the name of a project in the current folder. This is the project you want to copy from. To set the current folder, choose Save As... from the File menu, find the folder that contains the source project, and click on Cancel instead of saving.

The second input can be a list containing names of specific pages. To copy all the pages, the second input must be the word **pages**. To import all the procedures, use **procedures** as input. If there are names in the new project that are the same as those in the current project, the new names will be renamed with a number. See *Import* under *Menus*, in Section 1.

```
merge "myproject "procedures
merge "myproject [page1 page4]
```

minus *number*

Reports the additive inverse of its input. To report the additive inverse of a variable, you must use **minus** (**minus :num** instead of **-:num**). See **-** and **difference**.

```
show minus 5
-5
show minus ycor
-100
```

mousepos

Stands for **mouse position**. Reports the x y coordinates for the current mouse position. See **setpos**.

```
show mousepos
60 63
t1, pd
forever [setpos mousepos]
```

N

name *word-or-list word*

Creates a variable and gives it the value *word-or-list*. The variable keeps its value as long as it is not cleared. See **clearnames**, **clearname**, **make**, and **createprojectvar**.

```
name [Peter Anne Dennis Geni Eric] "class
show :class
Peter Anne Dennis Geni Eric
show first :class
Peter
```

name? *word*

Reports **true** if the input is the name of a variable. See **make** and **name**.

```
make "age 10
show name? "age
true
```

namepage (np) *pagename*

Names the page being displayed with the input. **Namepage** corresponds to the Name Page item in the Pages menu.

```
namepage "presentation
```

names

Reports the names of all the variables with their values. See **shownames**.

Examples:

```
make "friends [Joanne Lea]
make "age 12
show names
make "friends [Joanne Lea]
make "age 12
```

An interesting use of **names** in regard to files is the following:

```
make "filelist projectlist
pr names
```

will print all the file names with the vertical bars showing when a name is more than one word.

newbutton *name [x y] instruction-list*

Creates a new button with the name and instruction specified. The position [x y] is the top, left corner of the button. The button is created in the Once mode. See **set** to change the button's settings.

```
newbutton "button1 [50 50] [fd 1]
```

newpage

Opens a new page called Page x (x is the next available number).

```
newpage
```

newprojectsize *list-of-numbers*

Sets the size on the computer screen (the page size) for new projects. The input is a list of two numbers: the width and the height of the page in turtle steps. Before using **newprojectsize**, there must be an empty project on the screen.

The page size of a project is saved with the project. Projects with different page sizes can't be merged.

Choose New Project from the File menu so there is a perfectly empty project on the screen.

```
newprojectsize [200 200]
newprojectsize "standard    Restores the default size.
```

newslider *name [x y] [min max current]*

Creates a new slider, with the specified name at the position indicated. The position [x y] is the center of the slider. The last input is a list of three numbers representing the minimum, maximum, and current value of the slider. The minimum and maximum possible slider values are -9999 and 9999 respectively. See **set** to change the slider's settings.

```
newslider "temperature [100 100] [0 8 3]
settemperature [10 20 15]    Sets the slider to the minimum
                              10, maximum 20, and the
                              current value of 15.
```

newtext *name [x y] [xsize ysize]*

Creates a new text box, with the name and size specified, with the top, left corner at position [x y]. The box can't be larger than the size of the page. See **set** to change the text box's properties.

```
newtext "info [100 100] [50 50]
setinfo [Click on Next]
```

newturtle *name*

Creates a new turtle with the name indicated. The new turtle appears at the position [0 0] and is hidden. Use **st** to show the turtle.

```
newturtle "Shelly
st
```

not *true-or-false*

Reports the logical inverse of its input. See **and** and **or**.

```
show empty? []  
true  
show not empty? []  
false
```

note *number duration*

Plays a note using the current instrument. The first input must be a MIDI note number. The second input is the duration in tenths of a second. Middle C is 60. The maximum number for **note** is 127. The maximum duration is 255.

Following is a list of MIDI values. Each line corresponds to one interval.

A	A#	B	C	C#	D	D#	E	F	F#	G	G#
				1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64	65	66	67	68
69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92
93	94	95	96	97	98	99	100	101	102	103	104
105	106	107	108	109	110	111	112	113	114	115	116
117	118	119	120	121	122	123	124	125	126	127	

```
note 60 2      Plays middle C.  
note 64 5
```

number? *word-or-list*

Reports **true** if the input is a number. If a number is above 2147483647 or below -2147483648, MicroWorlds will not recognize it as a number.

```
show number? 3.1416  
true
```

O

opaque *text-box-name*

Makes the current text box opaque. This is equivalent to unchecking Transparent in the text box's dialog box. See **transparent**.

```
transparent "text1
opaque "text1
```

or *true-or-false1 true-or-false2*
(**or** *true-or-false1 true-or-false2 true-or-false3...*)

Reports **true** if any of its inputs report **true**. If more than two inputs are used, **or** and its inputs must be enclosed in parentheses. See **and** and **not**.

```
show or (2 = 2) (3 = 5)
true
cg
show (or heading = 1 pos = [0 0] shape = 1)
true
```

output (**op**) *word-or-list*

Stops the procedure and reports a word or list. **Output** can only be used within a procedure. Use **output** to make reporter procedures.

This procedure takes a word as input. It reports the first number appearing in the word.

```
to firstnumber :word
if empty? :word [output "none]
if number? first :word
  [output first :word]
output firstnumber bf :word
end
```

Try the procedure from the Command Center.

```
show firstnumber "abc6ef5g
6
```

If a text box Text1 contains the sentence "You got the 5th best score":

```
show firstnumber text1
5
```

P

pagelist

Reports a list containing the names of all the pages in the current project. The first name in **pagelist** is always the page that's open.

```
show pagelist
menu page2 practice
show member? "menu pagelist
true
```

parse *word*

Turns a character string into a plain list. Spaces and carriage returns contained in the character string become separators in the list. **Parse** can be used to turn the long words (character strings including spaces) reported by text boxes into lists of words.

```
show text1
Hi out there
show count text1
12  A word of 12 characters, including spaces.
show count parse text1
3   A list of three words.
```

paste

Pastes a copy of the Clipboard in the current text box. The Clipboard contains the last text that has been cut or copied using the **cut** or **copy** command, or the equivalent Edit menu items. See **cut**, **copy**, and **select**.

Select some text and choose Copy from the Edit menu. Then go to the Command Center and type:

```
paste
```

pd

Stands for **pen down**. Puts down the pen of the current turtle. The turtle will then leave a trace when it moves, but not when it is dragged. See **pu**.

```
repeat 10 [pu fd 10 pd fd 10]
```

pe

Stands for **pen erase**. Puts down the eraser of the current turtle. The turtle will erase as it draws, using the current pen size. See **pu** and **pd**.

```
cg
setc "red
fill
pe
fd 50 rt 90 fd 50
```

pensize

Reports a number representing the pen size of the current turtle. The original pen size is 1. The maximum is 100. See **setpensize**.

```
t1, setpensize 10
setc "pink
pd fd 50
show pensize
10
```

pi

Outputs the mathematical constant **pi**.

```
show pi
3.141593
```

pick *word-or-list*

Reports an element chosen randomly from the word or the list. Picking from a word reports a character, picking from a list reports an element of the list (a word or a list). See **textpick** to pick a random line of words from a text box.

```
show pick "hello
e
```

pictlist

Stands for **picture list**. Reports a list containing the names of picture files in the current folder. Only PICT type files (created using **savepict** or other applications) are shown. See **textlist** and **projectlist**.

```
show pictlist
autumn scene
```

placepict *pict-file-name position size*

Stands for **place picture**. Imports a picture file, places it at the designated position, and adjusts its size to fit the size indicated. The first input is the file name in the current folder or a full pathname; the second input is the position for the top, left corner of the picture, and the third input is the size of the image on the page, in *x* and *y* turtle steps.

```
placepict "balloon [0 0] [100 100]
```

pos

Stands for **position**. Reports the *x y* coordinates of the turtle. The result is a list of two numbers. [0 0] is the center of the page. See **setpos**.

```
show pos  
50 50
```

power *number number*

Reports the first number, to the power of the second number.

```
show power 5 3  
125
```

prefix

Reports the current prefix setting that was set by **setprefix**. See **setprefix**.

presentationmode

Hides the Command Center, Tool Palette, the project's title bar, and MicroWorlds menus. The project is centered on the screen and the background is filled in. **Presentationmode** is used to display completed projects. To return to MicroWorlds regular mode, use **presentationmode** again or click the mouse in the area where the menu bar would be located. This command corresponds to the Presentation mode item in the Gadgets menu.

print (pr) *word-or-list*

Prints a word or list in the current text box. The text is followed by a carriage return. See **insert**.

```
text1, repeat 5 [print "hello"]
```

printtext

Prints out, on the printer, the contents of the current text box or the Procedures page, depending on what is currently showing. All text is printed, even text that's not visible. Text format (i.e., font, size, style) is not used for printing.

product *number1 number2*
(**product** *number1 number2 number3...*)

Reports the result of multiplying its inputs. If more than two inputs are used, **product** and its inputs must be enclosed in parentheses. See *****.

```
show (product 3 3 3)
27
```

projectlist

Reports a list of the names of MicroWorlds project files in the current folder. See **pictlist** and **textlist**.

projectsize

Reports the current project size, in turtle steps. See **newprojectsize**.

```
show projectsize
599 374
```

projectvars

Stands for **project variables**. Reports the list of currently defined project variables. See **createprojectvar**.

```
show projectvars
amount speed inventory
```

pu

Stands for **pen up**. Lifts the pen of the current turtle so the turtle won't leave a trace when it moves. See **pd**.

```
repeat 10 [pu fd 10 pd fd 10]
```

Q

question *word-or-list*

Opens a dialog box displaying the question and an area to type the answer. **Answer** reports what was typed in the dialog box. If you write a very long question, only the part that fits will be displayed. See **answer**.

```
question [How old are you?]
```

Type the answer in the dialog box. In the Command Center, type:

```
show answer
I'm 10    Your answer.
```

This procedure can be used to verify that the user has actually answered a question dialog box:

```
to insist
question [Your name please...]
if empty? answer [insist]
end
```

quotient *number1 number2*

Reports the result of dividing number1 by number2. See **/**.

```
show quotient 6 3
2
```

R

random *number*

Reports a random non-negative integer less than *number*. The maximum number is 9999.

```
show random 100
22
show (random 6) + 1
3
```

readchar

Stands for **read character**. Waits until a character is typed on the keyboard. You must click on the background of the page

(outside of a text box, the Command Center, or Procedures page) for **readchar** to recognize the character typed. See **key?**.

```
to getchoice
show [Type your choice, A B or C]
make "answer readchar
show :answer
end
```

recycle

Frees up unused computer memory space. See **space**.

```
show space
57974 3741136 Your results will differ.
recycle show space
59049 3743056
```

remainder *number1 number2*

Reports the remainder after *number1* is divided by *number2*. The remainder of a negative number will be negative. If *number1* and *number2* are non-integers, they are rounded to the nearest integers.

```
show remainder 100 3
1
```

remove *name*

Deletes the object, page, or turtle variable created with **turtlesown**. If the named object is not on the current page, MicroWorlds will look for it on other pages in the project. If the input to **remove** is **procedures**, the Procedures page in the current project will be cleared. You cannot remove the only page of a project.

```
remove "text1
remove "t1
remove "page1
```

repeat *number word-or-list-to-run*

Runs the list of instructions the specified number of times.

```
repeat 36 [fd 10 rt 10]
repeat 10 [setsh "usa1 wait 2 setsh "usa2 wait 2]
repeat 4 [fd 10 note 60 2 rt 90]
```

rerandom

Reproduces the same sequence of numbers generated by **random**. After running **rerandom**, **random** with the same input generates the same sequence of numbers the next time.

```
rerandom
repeat 2 [show random 10]
8
5
rerandom
repeat 2 [show random 10]
8
5
```

resetquicktime *word*

Resets the QuickTime movie to the beginning.

```
resetquicktime "qtmovie1
```

resett

Stands for **reset timer**. Resets the timer to 0. The timer starts when you start up MicroWorlds.

```
resett show timer
0
```

The next procedure displays a question just after resetting the timer. If you got the right answer, it tells you how quickly you typed it. The time is in tenths of a second.

```
to reflex
resett
question [What is 12 times 12? Answer quickly!]
ifelse answer = 144
  [announce se timer / 10 "seconds]
  [announce [Wrong answer]]
end
```

resetvideo *word*

Resets the QuickTime movie to the beginning.

Example:

```
resetvideo "qtmovie1
```

rest *duration*

Inserts a rest in a sequence of notes. The duration is in tenths of a second, and has a maximum of 255.

```
note 60 5 note 62 5 note 64 5 rest 5 note 60 5
note 62 5
```

restore

Restores the background to the way it was the last time a snapshot command was issued. Everything else remains intact. The turtle's position is not reset. When using the drawing tools, use the undo tool or menu item to undo an unwanted drawing action.

```
fd 50 rt 90
snapshot
fd 50
restore
```

right (rt) *number*

Turns the turtle to the right. The maximum number is 9999. See **left**.

```
repeat 10 [fd 40 bk 20 rt 36]
```

round *number*

Reports the input number, rounded to the nearest integer.

```
show round 3.3333
3
show round 3.5
4
```

run *word-or-list-to-run*

Runs a word or an instruction list.

This instruction runs the contents of a text box:

```
run text1
```

S

savehtml *directory*

Saves the current project as a series of HTML pages in a new *directory* created in the current folder. The *directory* name (and all the object and pages in the project) should be lowercase without any special characters included. This primitive saves your project into a Web page (or pages) with limited functionality. You must use buttons or turtles to change pages in the project. Each page is a snapshot of the screen: animation, text, video, sound, and music icons are part of the background and do not react to clicks.

Example:

Suppose you have a project with two pages named Room and Lake:

```
savehtml "champlain
```

In the finder, go to the current folder. You should now have a directory named Champlain with twice the number of files as pages. In this example, you will have the following

```
files:Lake.html  
lake.jpg  
Room.html  
room.jpg
```

If you have a procedure called **links** on the Procedures page of your project, you can link to other Web pages outside of your project. You can make as many links as you like inside the **links** procedure. Here is an example of a **links** procedure:

```
to links  
make "lcsi "http://www.lcsi.ca  
end
```

One of the pages in your project should have a button or turtle running the instruction **lcsi** to link to the Web site.

savepict *pathname*

Stands for **save picture**. Saves the background of the current page as a PICT file. Turtles, buttons, text boxes, sliders, and other objects are not part of the background. Stamped images, stamped text, and text typed with **turtletype** are part of the background. See *Exporting Pictures* in Section 3.

```
savepict "garden
```

saveproject

Saves the current project without closing it. The project must have a name in order for this command to work. This is equivalent to the Save Project item in the File menu.

Name the current project first.

```
saveproject
```

saveshape *file-name number*

Saves the specified shape as a picture file (PICT format).

Example:

```
saveshape "moon 1
```

savetext *pathname*

Saves the text found in the current text box or on the Procedures page as a standard text file. Formatting (font, style, color, for example) will not be saved. The text can be loaded with **loadtext**. See *Exporting Text* in Section 3.

```
savetext "mytext
```

search *word*

Tells MicroWorlds to search and select (highlight) the word in the current text box. Nothing happens if the word is not found. **Search** starts searching at the cursor position. Use **unselect** to undo the highlighting effect of search. See **found?** and **unselect**.

The following procedure replaces all occurrences of a word in the current text box with another word. Make sure you place the cursor at the top of the text box before trying the procedure:

```
to replaceall :this :withthat
  search :this
  if not found? [stop]
  insert :withthat
  replaceall :this :withthat
end
```

select

Tells MicroWorlds to start selecting text in the current text box. Any cursor motion (**top**, **bottom**, **cu**, **cd**, **cf**, and **cb**) will

select text.

```
text1,  
top  
select  
cd  
cut
```

selected

Reports as a word a copy of the characters in the block of selected text in the active text box. If no block of characters is selected, the empty word is reported. Use **parse** to convert the block of characters in a list.

Example:

Type “hello there” into a text box.

Select the word “there.”

```
show selected  
there
```

sentence (**se**) *word-or-list1 word-or-list2*
(**sentence** *word-or-list1 word-or-list2 word-or-list3...*)

Reports a list made up of its inputs (words or lists). If more than two inputs are used, **sentence** and its inputs must be enclosed in parentheses.

```
show sentence "a "b  
a b  
show (se "hello "there "everyone)  
hello there everyone
```

set *object property value*

Sets a property for an object to the specified value. The first input is the name of an object. The second input is a property name and the last one is the value. See *Creating and Modifying Objects Under Program Control* in Section 3 of the *User’s Guide*.

Following is a list of the properties each object can have. Special words for a property’s value are in parentheses. The properties ending in a question mark (?) always have the word “**true**” or “**false**” as its value.

Turtle: rule, on?
Button: pos, size, rule, on?
Slider: pos, showname?, limits, value
Text: visible?, pos, size, transparent?, showname?, text
Color: turtlerule, turtlemode (once, eachtime), mouseclick, on?
Melody: visible?, pos, on?, showname?
Recording: visible? pos, on?, showname?
QTMovie: visible? pos, on?, showname?
AudioCD: visible? pos, on?, showname?
Videodisk: visible? pos, on?, showname?
Announce: pos, size
Question: pos, size
Sounds: quality (good, better, best)

Examples:

```
set "text1 "visible? "false
set "t1 "rule [launch [seth random 360 fd 50]]
set "button1 "rule [forever [fd 1]]
set "slider1 "showname? "true
set "red "turtlerule [silly-sound]
set "red "mouseclick [announce [You win!]]
set "announce "pos [300 300]
set "sounds "quality "better
```

setbg *name-or-number*

Stands for **set background**. Sets the background color for the page. The input can be the name of a color or a number. If the input is a name, a quotation mark must precede it. **Setbg 0** or **setbg "white** are the only ways to reset the original background color.

Note: If you set the background color to the same color as portions of your drawings, the drawings will be integrated with the background (and lost).

```
setbg "red
setbg 22
setbg "white
```

setc *name-or-number*

Stands for **set color**. Sets the color of the turtle's pen. If the turtle has its original shape, it will change color to show the pen color. The input can be the name of a color or a number. If

the input is a name, a quotation mark must precede it. The original pen color is black, or color number 9. The input can be an integer or a one place decimal. Decimal numbers give exact colors when the project is in “thousands” color mode. See *256 Colors vs. Thousands of Colors* in Section 3, and **color**.

```
setc 15
setc "red"
```

setfont *word*

Sets the font for the selected text in the current text box. If no text is selected, **setfont** sets the font for the next characters typed. The font named must be present in your system.

```
setfont "chicago"
print [Hello there]
```

setfontsize *number*

Sets the font size for the current text box. If no text is selected, **setfontsize** sets the font size for the next characters typed.

```
ct
setfontsize 20
pr [To be or not to be]
```

setfooter *word-or-list*

Sets the text that will appear in the footer of printed pages. **Setfooter** [] makes an empty footer.

```
setfooter [Logo Lovers, Grade 4]
```

seth *number*

Stands for **set heading**. Sets the turtle’s heading to the specified direction (in degrees). The degrees correspond to those of a compass: 0 degrees is due North, 90 is East, 180 is South, and 270 is West. **Right** and **left** turn a turtle a number of degrees from its current heading. **Seth** makes a turtle point to a specific direction, regardless of its previous heading.

```
seth 0
seth 90
rt 90
seth 90
```

setinstruction *instruction-list*

Sets the instruction for the current turtle. This is equivalent to typing the instructions in the turtle's dialog box. If the list includes the word **forever** or **launch**, the instruction's mode is set to Many Times or Once accordingly. If these words are not included, the mode is unchanged.

```
setinstruction [fd 50]
clickon
clickoff
setinstruction [forever [fd 1]]
clickon
```

setinstrument *name-or-number*

Sets the instrument for the next **note** commands. The instrument names are **violin**, **clarinet**, **lute**, and **orchestra**, respectively numbered 1, 2, 3, and 4.

```
setinstrument "orchestra
note 45 10
```

setpensize *number*

Sets the turtle's pen size, which determines the thickness of the lines it will draw. The pen size can also be set by picking a pen size and the pencil in the Drawing Center and clicking on a turtle. The original pen size is 1. The maximum pen size is 100.

```
setpensize 20 fd 50
```

setpos [*x y*]

Stands for **set position**. Moves the turtle to the designated x y coordinates. The center of the page is [0 0]. The maximum number for x and y is 9999. Note that if a turtle lands on a programmed color after a **setpos** command, the color won't react: only **fd** and **bk** activate programmed colors. See **pos**.

```
setpos [50 50]
```

setprefix *pathname*

Changes the current volume and/or subdirectory (folder) name to *pathname*. Colons are used to separate the names of directories. See **prefix** and **directories**.

setshape (setsh) *name-or-number*

setshape (setsh) *list-of-names-or-numbers*

Gives a shape to the turtle. If the input is a name, a quotation mark must precede it. When a list of shape names or numbers is given as input, each **fd** and **bk** command makes the turtle cycle through the list of shapes. The shape can also be set by picking a shape from the Shapes Center and clicking on a turtle. See **shape**, and *Animation Techniques* in Section 1.

```
setshape "moon
```

setsize *number*

Sets a turtle's size. The original size is 40. The magnifiers in the Shapes Center also change a turtle's size. The turtle cannot be bigger than 150 or smaller than 5.

```
setsize 20 setsize 40
```

setstyle *word-or-list*

Sets the font style in the current text box. The input must be the name of a style (plain, bold, italic, underline). If no text is selected, **setstyle** sets the font style for the next characters typed. Combination styles can be created by naming styles in a list.

```
setstyle "bold
insert "tic
setstyle [italic underline]
insert "tac
```

settc *name-or-number*

Stands for **set** text color. Sets the color of the text in the current text box. The input can be the name of a color or a number. If the input is a name, a quotation mark must precede it. The original text color is black, or color number 9. If no text is selected, **settc** sets the text color for the next characters typed. You can also set the text color by choosing Color from the Text menu. See *256 Colors vs. Thousands of Colors* in Section 3.

```
settc "red
pr [I'm red]
settc 22
```

setx *number*

Sets the x coordinate of the current turtle. The y coordinate remains unchanged. See **sety**.

```
setx 100
setx -100
setx xcor - 10
```

sety *number*

Sets the y coordinate of the current turtle. The x coordinate remains unchanged. See **setx**.

```
sety 100
sety -100
sety ycor - 10
```

shape

Reports the shape name or number of the current turtle. A name will be reported if the last **setsh** command used a name as input. See **setsh**.

```
t1,
setsh "heart
show shape
heart
setsh 6
setsh shape + 1
```

show *word-or-list*

Prints a word or a list in the Command Center. Lists are displayed without their outer brackets.

```
show "hello
hello
show [hello there]
hello there
show pos
125 45
show heading
90
```

shownames

Displays the names of all the variables with their values in the Command Center.

```
make "friends [Joanne Lea]
make "age 12
shownames
make "friends [Joanne Lea]
make "age 12
```

showtext

Makes the current text box visible. See **hidetext**.

If there is already a text box on the page, this instruction flashes the text box.

```
repeat 10 [hidetext wait 5 showtext wait 5]
```

sin *number*

Stands for **sine**. Reports the sine of *number* degrees. See **cos**.

```
show sin 90  
1
```

size

Reports the size of the current turtle. See **setsize**.

```
show size  
40
```

snaparea *shape-number* [*x y*] [*xsize ysize*]

Copies the graphics in the defined area and pastes it in a turtle shape. [*x y*] is the top, left starting point, and [*xsize ysize*] determines the size of the rectangle that is copied into the shape. See **snapshape**.

```
pd rt 11 fd 5000  
snaparea 1 [20 20] [60 60]  
setsh 1
```

snapshape

Copies the background section underneath a turtle into the turtle's current shape. You cannot use **snapshape** if the turtle has its original turtle shape. The turtle must be completely visible on the page. If you want to copy the shape into an empty shape number in the Shapes Center, set the turtle to this number before using **snapshape**. Place the turtle on the background that you want to copy. See **snaparea**.

```
setsh 43  
snapshape
```

snapshot

Takes a snapshot of the background. The next time a **restore** command is used, the background will be restored to what it

was at the moment the snapshot was taken. Use **snapshot** before running a command (like **fill** or **turtletype**) that may produce unexpected results. See **restore**.

```
pd
fd 50 rt 90
snapshot
fd 50
restore
```

sol

Stands for **start of line**. Brings the cursor, in the current text box, to the beginning of the current line. The line starts where the previous carriage return is, possibly not on the same physical line. See **eol**.

```
text1,
repeat 5 [print "hello]
top
make "n 1
repeat 5 [insert :n sol cd make "n :n + 1]
```

soundlist

Reports a list containing the names of system sounds. Any of these sounds can be played by typing its name in the Command Center.

```
show soundlist
violin clarinet gluglu aerosol stamper oops Simple
Beep Quack Droplet Indigo Wild Eep Sosumi
```

space

Reports the amount of free memory space in bytes. See **recycle**.

```
show space
38877 3671872      Your results will differ.
```

sqrt *number*

Stands for **square root**. Reports the square root of its input.

```
show sqrt 9
3
```

st

Stands for **show turtle**. Shows the current turtle. See **ht**.

```
repeat 10 [ht wait 5 st wait 5]
```

stamp

Stamps a copy of the turtle's shape on the background. You can also stamp shapes using the stamper in the Shapes Center. Stamp works whether the pen is down or not.

```
t1,  
setsh "tree  
repeat 10 [stamp fd 40]
```

stamptext *text-box-name*

Leaves a copy of the designated transparent text box on the background. This is equivalent to clicking with the stamper on a transparent text box. See **transparent**.

Example:

```
stamptext "text1
```

Drag the text box elsewhere.

stop

Stops the procedure that is running. **Stop** can only be used in a procedure. See **stopall** and **stopme**.

The second line of this procedure is called a stop rule.

```
to countup :number  
if :number > 100 [stop]  
print :number  
countup :number + 5  
end
```

Try:

```
text1,  
countup 0
```

Other examples of stop rules.

```
if colorunder = 15 [stop]  
t1, if (distance "t2) > 100 [stop]  
if slider1 < 1 [stop]  
if empty? answer [stop]  
if empty? text1 [stop]
```

stopall

Stops all running procedures and processes including turtle and button instructions. **Stopall** can be used as a button, from the Command Center, or in a stop rule in a procedure. See **stop** and **stopme**.

```
forever [setsh "bird1 fd 2 setsh "bird2 fd 2]
stopall
```

As a stop rule, **stopall** stops not only the procedure that contains it, but also all running procedures. It can be used, for example, when you reach the final goal of a game. Use this command as a stop rule in a procedure or for programming colors.

```
if colorunder = 15 [announce [You win!] stopall]
if touching? "t1 "t2 [stopall]
when [touching? "t1 "t2] [stopall]
```

stopme

Stops the process in which the command is run. See **stop** and **stopall**.

The action will stop when turtle 1 is more than 100 steps away from turtle 2.

```
forever [go]
to go
t1, fd 1
if (distance "t2) > 100 [stopme]
end
```

sum *number1 number2*
(**sum** *number1 number2 number3...*)

Reports the sum of its inputs. If more than two inputs are used, **sum** and its inputs must be enclosed in parentheses. See +.

```
show sum 3 3
6
show (sum 3 3 3)
9
```

T

talkto (**tto**) *turtle-or-list-of-turtles*

talkto *text-box*

Makes the turtle(s) or text box current. This command has the same effect as typing the name of a turtle or text box followed by a comma.

```
talkto "t1
fd 50
talkto [t1 t2]
bk 50
talkto "text1
print "hello
```

tan *number*

Stands for **tangent**. Reports the tangent of its input. See **sin** and **cos**.

```
show tan 45
1
```

tc

Stands for **text color**. Reports the number of the text color at the cursor position in the current text box. If there is currently a text selection containing more than one color, **tc** reports an empty list. See **settc**.

```
ct
settc "red
print "Eureka!
show tc
15
```

textcount *text-box-name*

Reports the number of lines in the current text box. Lines are delimited by carriage returns (they are logical lines, not physical lines). Empty lines are counted, but not carriage returns at the end of the text. The number that **textcount** reports is the maximum number that can be used with **textitem**. See **textitem**, and *Text Boxes as Variables* in Section 2.

```
show textcount "text1
3                Text1 contains 3 lines.
```

textitem *line-number text-box-name*

Reports the contents of a line in the named text box. Lines are delimited by carriage returns (they are logical lines, not physical lines). Empty lines are counted, but not carriage returns at the end of the text. The first input must be between 1 and the number of lines in the text box. The line reported by **textitem** is a long word (a character string including spaces). Use **parse** to turn a long word into a list. See **textcount**.

```
show textitem 2 "text1
Kim
show textitem 4 "text1
Plato's cat
```

textlist

Reports a list of the names of text files in the current folder. Only text files (created by **savetext** or other applications) are reported.

```
show textlist
friends mytext
```

textpick *text-box-name*

Reports the text in a randomly-chosen line from the named text box. Lines are delimited by carriage returns (they are logical lines, not physical lines). **Textpick** can pick empty lines if they're in the middle of the text, but not if they're at the end of the text. Carriage returns at the end of the text are not taken into account. The line reported by **textpick** is a long word (a character string including spaces). Use **parse** to turn a long word into a list.

```
show textpick "text1
Kim
show textpick "text1
Plato's cat
```

textwho

Reports the name of the current text box.

```
text1,
show textwho
text1
if textwho = "text1 [ct]
```

thing *name*

Reports the value of the named variable. Corresponds to the use of a colon (:) preceding a word. See **make** and **name**.

```
make "age 10
show thing "age
10
show :age
10
```

timer

Reports a number representing the time elapsed since the program started, or since the last **resett** command was run. The number is in tenths of a second. See **resett**.

```
resett
```

Wait a little.

```
show timer
22
```

The next procedure displays a question just after resetting the timer. If you got the right answer, it tells you how quickly you typed it. The value of the timer is divided by ten in order to get the number in seconds.

```
to reflex
resett
question [What is 12 times 12? Answer quickly!]
ifelse answer = 144
  [announce se timer / 10 "seconds]
  [announce [Wrong answer]]
end
```

top

Moves the cursor to the beginning of the text in the current text box. See **bottom**.

```
pr "hello
top
pr "there
```

touching? *turtlename turtlename*

Reports **true** if the two turtles are touching each other. Reports **false** if they are not touching; always reports **false** if one of them is invisible.

```
show touching? "t1 "t2
true
waituntil [touching? "t1 "t2]
when [touching? "t1 "t2] [do-this]
```

towards *turtlename*

Sets the heading of the current turtle to point at the named turtle. See **distance**.

```
t1,
towards "t2           T1 faces t2.
fd distance "t2      T1 meets t2.
```

transparent *text-box-name*

Makes the current text box transparent. This is equivalent to checking Transparent in the text box's dialog box. See **opaque**.

```
transparent "text1
opaque "text1
```

turtlesown *word*

Assigns a variable to all the turtles in the current project. This variable can then be set to a specific value for each turtle. This command also creates a new primitive made of the word **set** followed by the name of the variable (e.g., **turtlesown "speed** creates a **setspeed** command as in **t1, setspeed 12**).

There are two ways to find the value of a given turtle variable: you can talk to a turtle and use the variable name to report the value (e.g., **t1, show speed** displays 12 in this example) or you can use the turtle name followed by 's (e.g., **show t1's "speed** displays 12).

Use **remove** to remove a turtle variable. This removes the named variable for all the turtles in the project.

```
turtlesown "speed
t1, setspeed 10
t2, setspeed 20
t3, setspeed 5
```

```
t1, show speed
10
show t2's "speed
20
everyone [fd speed]
everyone [forever [fd speed]]
```

Choose Stopall from the Edit menu.

```
remove "speed
```

turtletype *word-or-list*

Sets the turtle to type the given word, one character at a time. Following the **turtletype** command, the turtle will drop one letter at each **fd** or **bk** command until there are no more letters to drop.

```
turtletype "school
repeat 6 [fd 10]
turtletype "school
repeat 6 [fd 20]
```

U

unfreeze *word-or-list-of-page-elements*

Unfreezes the button, text box, turtle, slider, or all the objects on a page, so they can be changed using the mouse. Use the Help menu to find out the name of the element. See **freeze**.

```
unfreeze "button1
unfreeze "page1      Everything on Page1 is "unfrozen."
```

unfreezebg

Unfreezes the background that was frozen by **freezebg**. See **freezebg**.

```
rt 11 fd 500
freezebg
lt 22 fd 500
cg
unfreezebg
cg
```

unselect

Undoes the highlighting effect of **select** or **search**. See **select**.

```
text1, print "hello
top
select
cf cf
unselect
cf cf
```

W

wait *number*

Causes a pause in the execution of a program. The time is measured in 10ths of a second.

```
repeat 5 [ht wait 10 st wait 10]
repeat 5 [setsh "flag1 wait 2 setsh "flag2 wait 2]
repeat 5 [fd 50 rt 72 wait 10]
```

waituntil *true-or-false-list-to-run*

Tells MicroWorlds to wait until the instruction list reports **true** before going on to the next instruction. The input must be an instruction list that reports either **true** or **false**.

This procedure sets t1 to run forever. If it gets too far from t2, it will bounce back.

```
to bounce
t1,
forever [fd 1]
waituntil [(distance "t2) > 100]
cancel [fd 1]
rt 180
bounce
end
```

when *true-or-false-instruction-list instruction-list*

Starts a parallel process that repeatedly tests whether the first instruction list reports **true** or **false**. If it reports **true**, the second instruction list is run. To stop a **when**, use **cancel**, choose Stopall or Cancel from the Edit menu, or press **APPLE - . (period)**.

```
when [ycor < 50] [bk 20]
repeat 1000 [fd 1]
```

who

Reports the name of the current turtle.

```
t1,  
show who  
t1  
if who = "t1 [remove "t2]
```

word *word-or-list1 word-or-list2*

(**word** *word-or-list1 word-or-list2 word-or-list3...*)

Combines its inputs into one word and reports the word. **Word** can be used to turn a list into a word. If more than two inputs are used, **word** and its inputs must be enclosed in parentheses.

```
show word "hello "there  
hellothere  
show (word "hello char 32 "there)  
hello there
```

word? *word-or-list*

Reports **true** if the input is a word.

```
show word? "hello  
true  
show word? 3.5  
true
```

X, Y

xcor

Stands for **x** coordinate. Reports the x coordinate of the current turtle.

```
show xcor  
50  
setx xcor + 10
```

ycor

Stands for **y** coordinate. Reports the y coordinate of the current turtle.

```
show ycor  
50  
sety ycor + 10
```