New Ways to Remember in MicroWorlds EX

Originally published in the October 2000 issue of Australia's *Hotsource* online newsletter

Most MicroWorlds users are familiar with the local and global variables built into the language. Logo, MicroWorld's underlying computer language, prefers the passing of values between procedures via local variable inputs. Global variables, created with MAKE or NAME, are containers that remember their contents across all procedures. Local variables are like short-term memory and global variables are useful for remembering things like the name of a player so that such data may be recalled later on in a program. MicroWorlds adds additional variable types, including the very powerful TURTLESOWN and project variables which save with the MicroWorlds project file.

What Do Turtles Own?
MicroWorlds turtles can now keep track of all sorts of properties. This enhances their intelligence and makes it easier to get turtles to do tricky things without the programmer needing to keep track of too many variables. Whenever you want one or more turtles to remember some information about its state or behavior, the command, TURTLESOWN will help you do so. Remembering a state or behavior is one thing, but in MicroWorlds you will also want a way of reporting that information to a procedure or other turtle.

Understanding Turtlesown

a) State variables
Turtles already own several state variables including SHAPE, COLOR, HEADING, POS, XCOR, YCOR and SIZE. Each one of these reporters has a corresponding one input command beginning with *SET*. In other words, you may change a state and report its value/condition.

Command	**Reporter**
SETSH	SHAPE
SETC	COLOR
SETH	HEADING
SETPOS	POs
SETX	XCOR
SETY	YCOR
SETBG	BG

Try typing the following and observe the results. Be sure you have at least one turtle on the page.

```
setsh “car
show shape
setpos [50 -75]
show POs
show xcor
show ycor
```

```
setc color + 10
seth 90
fd heading
repeat 5 [setsize size + 5]
```

TURTLESOWN is used to create other variables for every turtle in a current project. TURTLESOWN takes a word as input and that word is the name of the variable being created.
Turtlesown *“speed*

There are three actions associated with TURTLESOWN.
1. Telling MicroWorlds to give a state/variable to every turtle in the project
2. Assigning a variable to that state
3. Using that state variables in some way

b) Introduction to turtlesown

Let’s try a few simple examples.
1. Create a new project
2. Hatch two more turtles so that three turtles appear on the page
3. Change their colors so that we can tell them apart:

```
t1, setc “red
t2, setc “blue
t3, setc “yellow
```

Tell MicroWorlds to give every turtle the property, speed

```
turtlesown “speed
```

Set the speed value for each turtle and tell them to go:

```
t1, setspeed 1
t2, setspeed 5
t3, setspeed 10
everyone [forever [FD speed]]
```

In this case, everyone tells all of the turtles on this page to go FD by their speed value speed.
Let the turtles run wild and type the following to see what happens.

```
t2, show speed
```

There is another way to check a turtle’s property value by using the apostrophe as follows.

```
show t3’s “speed
```

You might want to use the apostrophe form in the following way.

```
t3, setspeed t2’s “speed
```

Now t3 and t2 are traveling at the same rate. What happens if you type: t3, setspeed -10 in the command center?

Turtlesown may also be used to restore a turtle to a previous state. Let’s say you want to give each turtle a random home on the screen, instead of [0 0], and after running around a bit want the turtles to go home. Try the following to see what I mean (Be sure there are at least three turtles on the page. The more turtles, the merrier!)

turtlesown “oldhome
everyone [setx random 2000 sety random 2000]
setoldhome POs *;this stores the turtles* current position in oldhome
everyone [rt random 360 FD random 3000]
everyone [setpos oldhome]

An important thing you need to understand is that TURTLESOWN is a command that will tell all of the turtles in a project to remember a particular variable during your work session. The best thing about turtlesown is that the state of each variable is stored with the project upon saving. This allows you to use these values at a later time.

Activities

You can use TURTLESOWN in all sorts of ways. It can be used to count laps around a track or keep track of a turtle's age. Here are a couple of simple ideas for storing information other than screen conditions. What kind of clever uses can you imagine for using TURTLESOWN?

Example 1 — Turtle Track and Field

In this exercise, we will keep track of how many times a turtle completes a circle.

- Create a textbox and name it COUNTER. Make the font size nice and big so you can read the contents of the textbox.
- Type the following in the command center:

```
CG
turtlesown "laps
setlaps 0
forever [FD 1 rt 1 if POs = [0 0] [setlaps laps + 1]
forever [setcounter laps]
```

Try typing before the above instructions and see what happens.

Example 2 — Dodge 'em Turtles

In this exercise, we will keep track of how often a turtle hits an object on the screen. If a turtle hits the red circle, it will make a random U-turn of between 150 and 209 degrees.

- Create a new project
- Hatch two more turtles to be sure that there are three on the page
- Make each turtle a different color so we can tell them apart
- Draw a red circle in the middle of the screen
- Double-click on the color red in the command center and type the following instruction:

```
setcollisions collisions + 1 rt 150 + random 60
```

- Click on the ONCE button and then click OK
- Click the eye tool or right mouse button on each turtle and give them the same instruction:

```
FD speed
```

Click MANY TIMES and OK

Type the following in the command center:

```
turtlesown "speed
turtlesown "collisions
everyone [setspeed random 10 setcollisions 0]
everyone [clickon]
```

Let the turtles run and bounce around the screen for a while and then check on their collision status.

```
t2, show collisions
show t3's "collisions
```

Example 3 — The Aging Turtle

We can ask turtles to keep track of their age as time elapses. The command, RESETT, sets the clock to zero and the reporter, TIMER, reports the number of tenths of a second that have elapsed.

Program each turtle to run the instruction, FD 5 *MANY TIMES*

- Create a new project
- Remove any turtles on the page

- Type the following procedures on the procedures page

```
to startup
dolist [x turtles] [remove :x]
carefully [remove "age] [] ;remove the age property if it already exists
turtlesown “age
resett ;reset the MicroWorlds timer to 0
end

to hatch
newturtle word “t ((count turtles) + 1) ;hatch a new turtle and name it t1…
setage 0
setx -250 + (count turtles) * 30 ;line the turtles up along the x axis
st
when [0 = remainder timer 10] [everyone [setage age + 1 setc color + 1 Seth age * 10]]
end

to turtles
get first pagelist “turtles
end
```

The when statement in the hatch procedure checks for each second that elapses and then asks every turtle to adjust it’s age, color and heading to show the aging process. Turtles is a reporter that reports the list of turtles on the current page. It is a good tool procedure to have in your toolbox.

- Create a ONCE button with the instruction, HATCH.
- Type STARTUP in the command center
- Click the hatch button occasionally to hatch a new turtle and watch time fly

You can check the age of a turtle by typing an instruction like, SHOW t1’s “age.

Part II - Project Variables

MicroWorlds adds another new data structure called project variables. The data in these project variables are saved with the project. Therefore, you can remember a piece of information within a project, even after your quit MicroWorlds

Local and global variables have their particular strengths and weaknesses, but neither hold their value when the file is saved. Project variables do. Therefore you can set the value of a project variable and use that value at a later time.

Understanding Project Variables

The createprojectvar “container command creates a variable, named *container*. You can then assign a value to that variable by typing: setcontainer 57 or setcontainer [Joe Smith] or setfoo “tails. In this way, SET is used to change the contents of a project variable, just as it is with a text box.

You can find the value of that variable by using its name. For example, typing show container would report the value stored in the container variable.

You can report the current project variables in a project by typing, show projectvars.
Delete a project variable by typing remove *“variablename*.
Project variables may also be used to keep track of things like high scores in video games.

Activity — Collecting Probability Data

We already explored flipping a coin in the *Running Text in Text Boxes* section, but let’s build on that idea in a way that uses project variables so that you may continue collecting data the next time you open the project. This variation on the coin tossing theme does not use text boxes to display and record the data. All of the record keeping is done in invisible variables.

- Create a new project.
- Create three text boxes named, table, headscount and tailscount.
- Create a many times button with the instruction, toss, on it.
- Write the following procedures on your procedures page.
- Type the procedures (below) in the procedures page.
- Type reset in the command center the first time to initialize and create the various variables.
- Click on the button to start the experiment.
- Stop the procedures at some point and save the file.
- Close the project.
- Open the project again and click the button. You should continue counting the trials.

Procedures:

```
to reset
setheadscount 0
settailscount 0
carefully [createprojectvar "heads setheads 0] [] ;if the project variable already exists do nothing
carefully [createprojectvar "tails settails 0] [] ;if the project variable already exists do nothing
table, ct
end

to toss
ifelse 1 = random 2 [gotheads] [gottails]
end

to gotheads
table, print “heads
setheads heads + 1
```

```
setheadscount heads
end

to gottails
table, print "tails
settails tails + 1
settailscount tails
end

to clearvariables
setheads 0
settails 0
end
```