

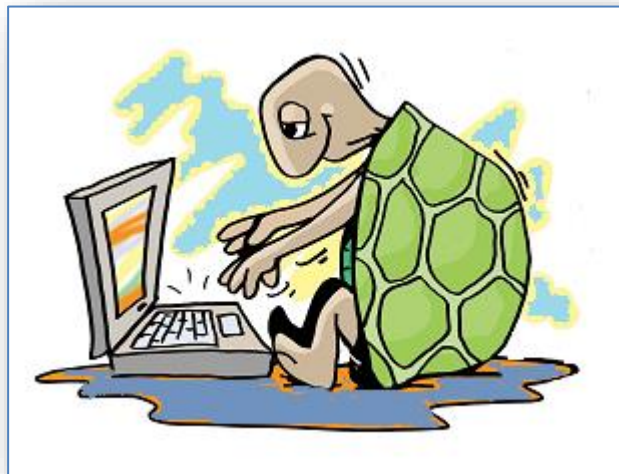
EVANGELOS C. ZIOULAS

IT Teacher



CHAPTER 3

LOGO TURTLES




TURTLES BASIC FEATURES

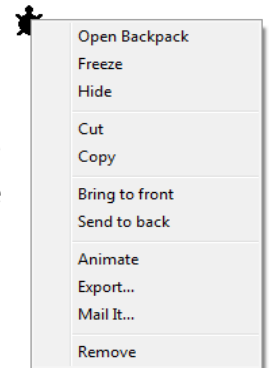
In MicroWorlds Pro programming environment **turtles** are prevalent. They can be used in order to design or decorate our page as well as be used as buttons or animation. Each turtle has a set of attributes that characterize it: *name, position, direction, pen width, pen color, shape*.



To make a turtle execute our commands, we should click on it.

CREATING TURTLES

We select the **Turtle tool**  in the Toolbar and then we click anywhere on the page to create the turtle. Then a new turtle appears. If we right-click on the turtle we can open its menu which contains a set of basic commands.



MOVING & TURNING TURTLES

To **move** a turtle, we should drag it anywhere on the page.




To **change** its **direction** we should change turtles heading by dragging its head. However, turning the turtle by dragging its head only works if the turtle has the original "turtle" shape. In case of other shape, when turning the turtle we should hold the **Shift** button (Although the turtle rotates, the shape on the turtle does not look any different).



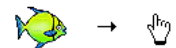
ANIMATED TURTLES

We can make a turtle start moving if we right-click on it and select **Animate** from its menu. An instruction is automatically inserted in the turtle's backpack to make the turtle move. Then the turtle starts moving to the direction it is oriented.

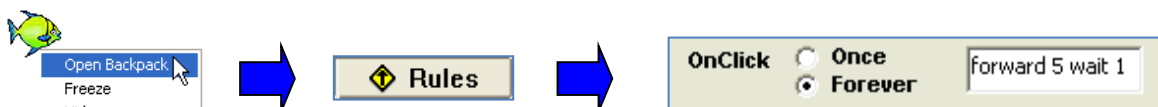


We can **click** again on the turtle to **stop** it or **restart** it (or we can click the **Stop ALL**  button in the Toolbar, or press the **Alt** button to pause it).

If the turtle is not facing in the right direction, we should hold down the **Shift** key, click on the turtle and drag it in the right direction. The turtle won't move, but its heading will follow the movement of the hand pointer.



To view the instruction that's causing the turtle to move, we **right-click** on the turtle and open its **backpack**. After that, click on the **Rules** tab and check the instruction in the **OnClick** field. If we change the values after **forward** or **wait** and click elsewhere in the page we can see the effect.



CHANGING THE SHAPE

To change the turtle's shape we click on the **Painting/Clipart palette** button in the Toolbar to open the Painting/Clipart palette.



After that, we choose one of the collections of shapes: **Singles**



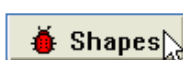
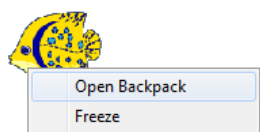
or **Sets**:



We **click** on the **shape** of our choice in the Painting/Clipart palette and **click** exactly on the **turtle** (if we miss the turtle, the clipart will be dropped on the page instead of becoming a turtle shape):



The turtle now wears that shape. We can open the turtle's backpack by right-clicking on the turtle and choose **Open Backpack** from its menu. If we click on the **Shapes** tab we can open the turtle's private shape collection. All the shapes that we give to this particular turtle are stored here.



It's sometimes easier to use the turtle shape while we are preparing our program, and then switch to the final shapes when our program is complete.

The original **turtle shape** turns to show its heading and changes color according to the pen color.



To set the turtle back to the original turtle shape, we click on the **Turtle tool** in the Toolbar and then we click on the turtle whose shape we want to change.



CHANGING THE SIZE

There are three ways to change the turtle's size.

1. Enlarge or shrink the turtle using the magnifiers in the Toolbar. Select the **magnifier** and click on the turtle.
2. Use the command **setsize** and a number e.g. **setsize 80** (original size is 40 when max is 160).
3. Change the value in the **Size field** in the **State** tab of the Turtle's **backpack**.



THE TURTLE'S STATE

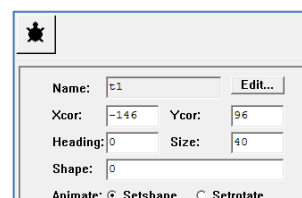
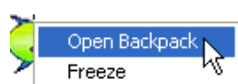
In the turtle's **backpack**, we can change its **size**, **position**, **shape**, **heading**, **pen state** and **visibility**.

To make all these changes we **right-click** on the turtle to open its backpack and we click on the **State** tab.

In case we have several turtles on the page, we can click on the large turtle icon in the top left corner of the backpack so that the turtle that owns that backpack flashes.



We can type the values we want in the different fields, we click elsewhere in the backpack and the turtle's state is immediately updated on the page.



PEN COLOR AND PEN SIZE

To change a turtle's pen color and pen size we need to use the Painting tools. However first, we need to make sure that the turtle is set to the original turtle shape.

We click on the Painting/Clipart palette button in the Toolbar to open the **Painting palette**.



Afterwards, we click on the **Painting tools** button



and we select the **pencil**.



We choose a **color** and select a **brush** (the size of the brush represents the pen size).



Finally we click on the turtle with the pencil and the color of the turtle automatically changes.



When we create a new turtle, its pen is up. We use the command **pd** to put the turtle's pen down, or we select **Pen Down** in the State tab of the Turtle's backpack.

Now, if we put the turtle's pen down and type **forward 50**, the turtle draws a line of 50 steps in the color and pen size that we have selected. We should also have in mind that **only the size of the brush** is used by the turtle. The turtle does not use pen patterns or the characteristics of the brush (fading etc.).

CHANGING SHAPE WHILE MOVING

We click on the **Painting/Clipart palette** button in the Toolbar.



Then we click on the **Sets shapes** button.



Some of the shapes come in sets of two, three, or four:



We can use them to make more attractive animations...

For this reason, in the **Clipart library**, we select all the shapes of the set we wish to copy (e.g. the horses). To do this, we click on the first shape of the set and while holding the **Shift** key we click on the last shape of the set to select the whole set.



While the shapes are selected in the Painting/Clipart palette, we **click on the turtle**. Copies of all shapes are placed in the backpack. The turtle looks like one of the shapes, although it has all the shapes in its shape list. If we open the turtle's **backpack** and click on the **State** tab we will see all the available shapes.



The **Shape field** also shows the shape

Shape: horse1 horse2 horse3 hor:

list:

Finally, if we **right-click** on the Turtle and choose **Animate** from the menu, the turtle starts moving by changing its shapes simultaneously.

If the turtle moves in the wrong direction, we should hold down the **Shift** key and drag the turtle in the right direction. We can also set the heading in the State tab in the backpack.



Each time the turtle moves forward, it changes shape because there is more than one shape in the turtle's shape list (we can check it in the **State** tab).

CHANGING SHAPE WITHOUT MOVEMENT

We can also use the previous technique to create an animation without movement.

First, we create a new Turtle and we click on the **Painting/Clipart palette** button in the Toolbar:

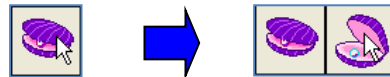


Then we click on the **Sets shapes** button:



We find two shapes that can be used to create an animation without movement (e.g. clams)

In the **Clipart library**, we select all the shapes of the set we wish to copy (the clams in this example). To do this, we follow the same procedure of the previous technique (shift + click).

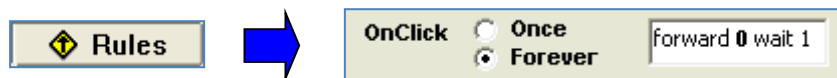


Then, we **click** on the **turtle**, so copies of all the shapes are placed in its backpack.

If we **right-click** on the turtle and choose **Animate** in its menu we can see that the turtle not only changes shape, but it moves too.

To have it change shapes without moving, we should click on the **Rules Tab** in its backpack and we change its commands by using **0** (zero) instead of **5** as input for **forward** in the **OnClick** field.

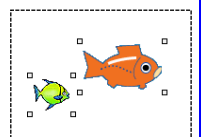
Now, each time the turtle moves forward, it switches shape and moves forward 0 steps.



REMOVING TURTLES

There are several ways to delete turtles if we have too many on the page:

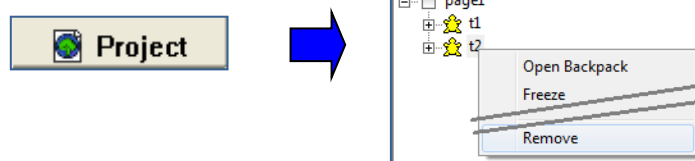
1st technique: We select the turtles by dragging around them, and then we press the **Delete** key or the **Backspace** key on the keyboard.



2nd technique: We right-click on the turtle that we want to delete and we choose **Cut** or **Remove** from the pop-up menu.




3rd technique: We open the **Project** tab, we right-click on the turtle that we want to remove, and select **Remove** from the menu.



NEW TURTLES & TURTLE NAMES

The turtles are numbered as they appear: **t1**, **t2**, and so on. However, when we have several turtles, it is a good idea to give them meaningful names.

We can use the **Turtle tool** to create several turtles. For each turtle, we click on the Turtle tool,  and then we click on the page.

Afterwards, we **right-click** on one of the turtles to open its **backpack**, we click on the **State** tab and then we click on **Edit**, beside its current name, we type a **new name** and click OK.



In the **Command Center**, we can type the following commands:

```
jenny,  
fd 50  
t2,  
fd 50
```

← We must not forget the comma

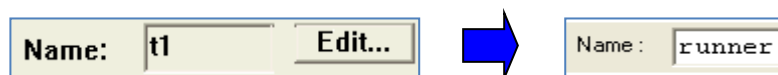
TALKING TO TURTLES

Turtles have names. If we have several turtles on our page, always **only one listens** to our commands.

- ▶ The **last turtle** that **we clicked on**, unless we created another turtle. Then it is ...
- ▶ The **last turtle** that **we created**, unless we talked to a turtle. Then it is ...
- ▶ The **last turtle** that **we talked to**.

Here's how we can talk to a Turtle:

First, we create several turtles on your page. The turtles are named **t1**, **t2**, **t3** and so on. It is often better to **use meaningful names** instead of these. We open the **backpack** of one of the turtles (e.g. t3), we click the **Edit** button beside its name and we enter a new name in the field. We should use a **one-word name** with no spaces.



We click **OK** to close this dialog box and then we check the name in the turtle's backpack.

Now we can go in the **Command Center** and type commands such as the following:

```
t1, fd 100  
t2, rt 90  
runner, fd 50  
t1 bk 50  
runner,  
fd 50
```

← First, we should use the name of one of the turtles and then we should give our commands.

← We must not forget the comma.
In case we forget the comma, the system gives back an error message.

Error Messages:

If we **forget the comma**, MicroWorlds thinks that the name is something to run (to execute) and will probably display an error message such as:

```
I don't know how to t1
```

If the **turtle name doesn't exist** (the turtle has not been created, it has been removed, or it has been renamed), MicroWorlds displays this error message:


```
I don't know how to t2,
```

COLOR DETECTION

A turtle can be programmed to react when it passes over a specific color. It can even do different things for different colors. Turtles react to any color visible on the current page, including those from the Wallpaper. Any turtle movement will trigger the color detection.

1st Step: Let's draw a color boundary on our page:

We click on the **Painting/Clipart palette**.  Then we click on the **Painting tools** button: 


We **pick one** of the painting tools (e.g. filled rectangle tool) and a large  brush:

We **choose one color** (we should remember from which column we took this color).

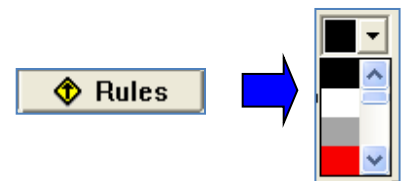


We **draw a solid line** on the page. ★ 

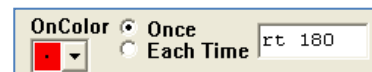
2nd Step: Now we can program the turtle:

We create a turtle if we don't have one on the page. If we have one, we select the **Normal pointer** in the Toolbar , and open its **backpack**.

We click on the **Rules** Tab and under the words **OnColor**; we choose the color to be detected. It must be the color representing the column we picked for the drawing on the page (this drop-down menu scrolls).



Now, in the field next to the color, we type an **instruction** such as:



The previous instruction makes the turtle turn right 180 degrees (bounce).

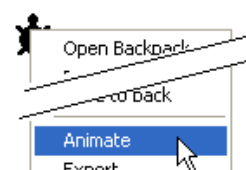
We leave the mode to **Once**. A small dot appears on the color to indicate that the turtle is programmed to detect it.

3rd Step: We try it out:

We **right-click** on the turtle and we choose **Animate** from the menu.

If the turtle is never going to hit the color, we stop it and change its direction.

When it detects the color it should bounce.



Possible Issues:

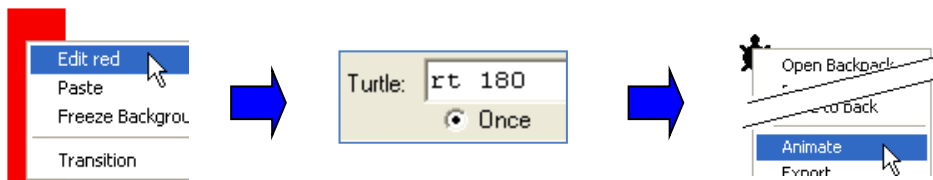
- A) If the turtle is **already on the color**, it does not bounce. We should drag it out and let it run over the color again.
- B) If the **line is thin enough**? The Turtle goes forward by jumping 5 steps at a time. If the line is narrower than 5, it may jump over the line.

Turtles react to any color on the **current page** and those available from the **Wallpaper**.

We can also **program a color** so that any turtle reacts when "stepping on" that color. We can even program different colors to trigger different actions.

We get the **normal pointer** and we right-click on the color on the page and choose **Edit colorname** from the menu (colorname depends on the color we chose e.g. red).

In the **Turtle field**, we type an instruction to make the turtle bounce; we leave the mode to **Once** and click **OK**. Finally, we **right-click** on the turtle and choose **Animate** from the menu:



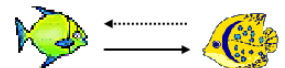
If the turtle is never going to hit the color, we stop it and change its direction. It should bounce when it passes over the programmed color.

Additional possible Issues:

- A) It **doesn't bounce**? We right-click on the color line and choose **Edit colorname** from the menu verifying that it is programmed.
- B) If we program the **same color** in a **Turtle's backpack** and in the **Painting/Clipart palette**, the instructions in the backpack are used and those in the Painting palette are ignored.

COLLISION DETECTION

Turtles can be programmed to **react when they touch (or are touched by)** any other turtle. Both the **"active"** turtle (the one that is touching) and the **"passive"** turtle (the one that is touched) will react to the collision and do whatever they are instructed to do.

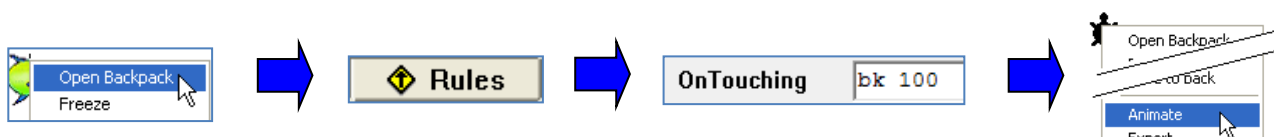


First, we **create** two turtles on the page and we open one turtle's **backpack**.

We click on the **Rules** Tab and in the **OnTouching** field, we type an instruction e.g. **bk 100**

Now, we can **right-click** on the same turtle and choose **Animate** from the menu.

Finally, we drag the other turtle somewhere in the path of the moving turtle. The moving Turtle steps back each time it touches the other one.



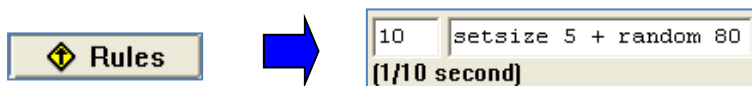
TIMER ACTIONS (OnTick)

Turtles can be programmed to repeatedly run an instruction after a preset interval, based on a **clock** built into MicroWorlds.

First we create a turtle; we open its backpack and click on the **Rules** Tab. There are 2 fields next to **OnTick**. The small one is the **delay**. The larger one is the **instruction**.

The number in the Delay field is in **tenths of a second**.
10 tenths equals one second.

In the Instruction field, we type an instruction that changes the turtle's size randomly 10 times in a second (the default size is 40):



As soon as we click out of this field, the turtle starts running the instruction. This goes on forever.

To stop the OnTick instruction temporarily, we click on the **StopAll** button in the Toolbar.



If we click on the StopAll button again, we restart the action.

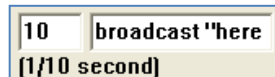
Possible Issue:

If we have created an **error in the OnTick** field, MicroWorlds displays **error messages** at each tick. We should stop everything using the **StopAll** button in the Toolbar and fix the instruction before restarting it.

BROADCASTING MESSAGES

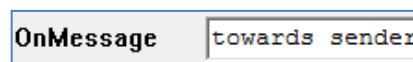
Turtles can be programmed to **broadcast messages** and **react to broadcasted messages**. A turtle can even tell from whom the message comes.

1st Step: We **create a turtle** who **broadcasts a message** and open its backpack. We click on the **Rules** Tab. We type the following instruction in the **OnTick** field so the turtle **broadcasts the message** "here" once every second.



We can give a different color to the turtle using the Painting tools or a command e.g. **setc "red"**.

2nd Step: We **create a turtle** who **listens for messages** and open its backpack. We click on the **Rules** Tab. We type the **towards sender** instruction in the **OnMessage** field.



This means that when the turtle hears the message, it points towards the sender of the message. (The turtle does not care what the message is; it just reacts if there is a message.)

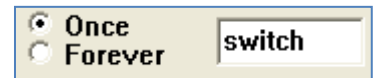
3rd Step: We can make **several copies** of the **listening turtle** by. To do it, we **right-click** on the turtle, choose **Copy** from the menu, **right-click** on the page and choose **Paste** from the menu. Now, we can scatter some turtles around the page.

4th Step: We animate the "leading" Turtle. To do it, we **right-click** on the leading turtle (the one who broadcasts the message) and choose **Animate** from the menu. All the other turtles keep pointing at it even as it moves.

BROADCASTING SPECIFIC MESSAGES

Turtles can also be programmed to **broadcast specific messages** or to **react to the text** of broadcasted messages.

1st Step: We create a turtle who **broadcasts a specific message**, and open its backpack. We click on the **Rules** Tab and we type the following instruction (**switch**) in the **OnClick** field.



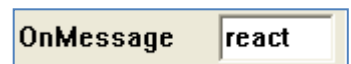
Then, we define the **switch procedure** in the **Procedures** tab of the turtle's backpack:



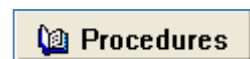
```
to switch
  ifelse color = 15
    [setc "green broadcast "go]
    [setc "red broadcast "nogo]
end
```

If we click on the turtle several times, it should alternate between being red and green. Besides changing color, the turtle also broadcasts the word "go" or "nogo".

2nd Step: We create a turtle who **listens for messages** and opens its backpack. We click on the **Rules** Tab and we type the **react** instruction in the **OnMessage** field.



Then, we define the **react procedure** in the **Procedures** tab in the turtle's backpack:



```
to react
  if message = "go [clickon]
  if message = "nogo [clickoff]
end
```

Right now, clickon and clickoff don't do anything because the turtle is not programmed to react to mouse clicks.

We should **right-click** on the listening turtle and choose **Animate** from the menu. We now have two ways to start its animation. We can click directly on the animated turtle, or we can click on the "red-green" turtle who acts as a traffic light.



3rd Step: We make several copies of the "listening" turtle. To do it, we **right-click** on the turtle, choose **Copy** from the menu, **right-click** on the page and choose **Paste** from the menu.

4th Step: We try the "red-green" turtle by clicking on the red-green turtle a few times. The other turtles should react to it as if it were a traffic light.

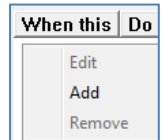
CHECKING EVENTS

A turtle can check **if an event happens** all by itself. In case this event happens, it will do what we programmed it to do.

In the next example, let's say that t1 turtle tries to run away from t2.

1st Step: We **create t1** and **t2** on the page (we may give an individual color to each one of them to distinguish them).  

2nd Step: We program t1 so it "runs away" from t2. To do it, we open **t1's backpack**, we click on its **Rules** tab and right-click in the area below "**When this**" and choose **Add** from the menu.



This opens the **When dialog box**. We then type the following instructions in the two fields:

When:	(distance "t2" < 100
What:	runaway

We must use the **parentheses** in the When instruction line for this to work; otherwise, MicroWorlds compares "t2" and 100 instead of comparing "distance "t2" and 100.

From t1's point of view, this instruction means: if the distance between myself (t1) and t2 is less than 100, run the procedure "runaway."

Define the **procedure runaway** in the turtle's **Procedures** tab:

```
to runaway  
  towards "t2  
  rt 180  
end
```

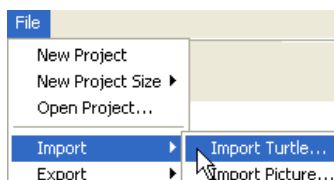
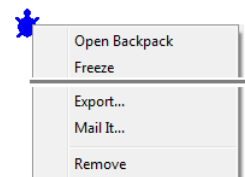
3rd Step: We animate t1. To do it, we **right-click** on t1 and choose **Animate** from its menu. Whenever it gets too close to t2 (less than 100 steps), it turns in the opposite direction. We can move t1 around and let it run towards t2.

EXPORTING TURTLES

If we have created a turtle with several properties (shapes, procedures, etc.) that we wish to use in a different project, we can easily **export** it (save it to our hard disk) and **import** it into another project.

To export a Turtle we right-click on the turtle and choose **Export** from the menu.

This opens our **Save dialog box**. Then, we type a **file name**, choose a **location**, and click **Save** (the file is saved in **.mwa** format).



To import the Turtle into a different project, we choose **Import** and **Import turtle** in the File menu.