1

A Tour of eToy

@@For the person responsible of the layout. I did not use figure with number because latex was putting them everywhere except where I wanted. So I want to have number and cross ref for figure but they should be well placed. I also would like to have the handle representation in the text when we talk about it. handles = round stuff we can get around an object) @@

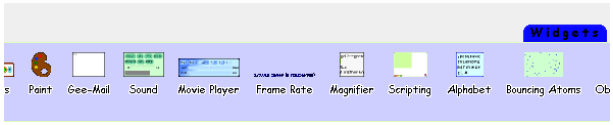
This chapter presents a small tour of the eToy system. The eToy system provides an interface for manipulating objects, sending them messages, composing graphically scripts and executing them. It is used in school by kids between 9 and 12 years. There is also a new book named "Powerful Ideas in the Classroom" that presents in detail how to use eToy to teach mathematics and sciences. You shall find a lot of information related to eToys at http://www.squeakland.org.

We present how to steer a airplane with a joystick, then how we can create animation, how to steer a car, and finally how to program a car to automatically follow a road. For all these tasks we suggest you to open a morphic project using the World menu (open item) give it a name and click on it to enter.

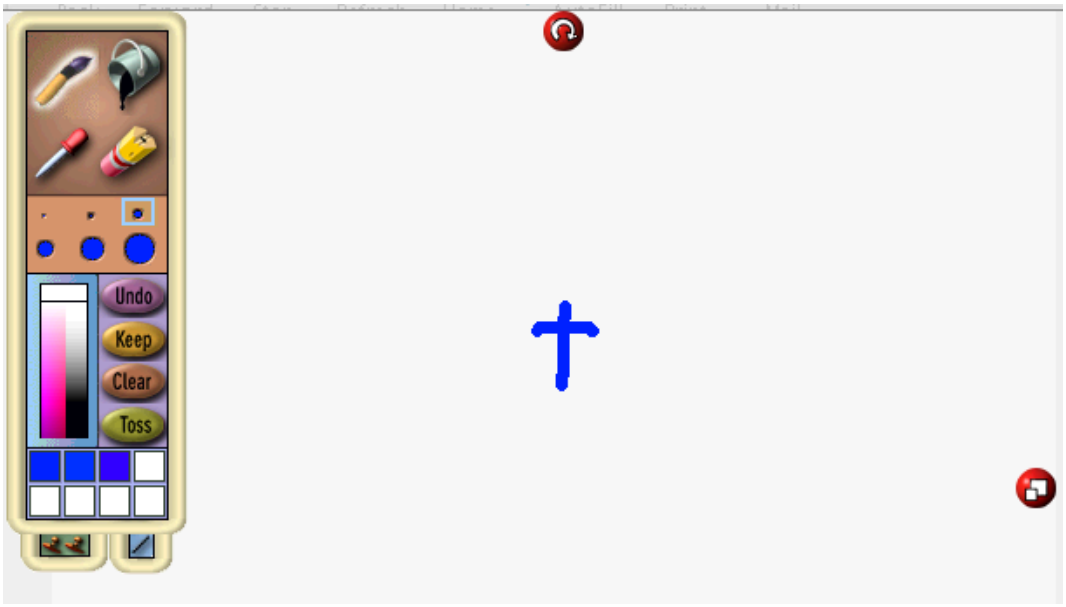
1 Steering an Airplane

To steer an airplane we first have to have an airplane then to find a joystick and create a script that connect them.

Step 1: Drawing an Airplane. To get an airplane the best thing to do is to draw it ourself. Open the blue flaps named widgets and drag the palette or paint editor from the flap to the desktop. This action opens a tools called Paint to paint drawing on the screen.

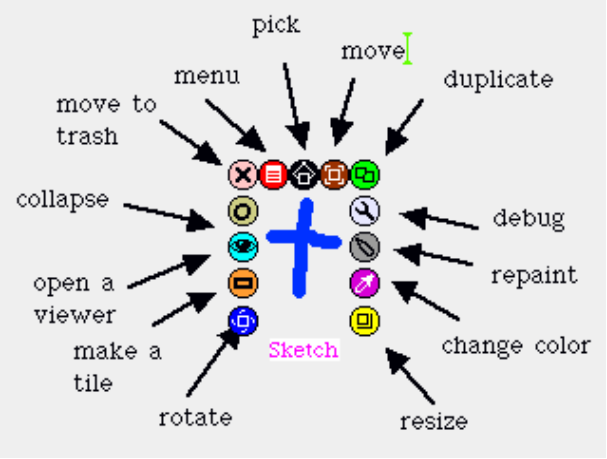


Using Paint draw a small airplane as shown in the following figure. Our airplane looks like a cross but you can draw a more sophisticated one. Once you are done with your drawing press the button Keep, the pain editor disappear and you get a sketch that looks like an airplane.



Vocabulary point. The airplane we just draw is called a Player in eToy jargon.

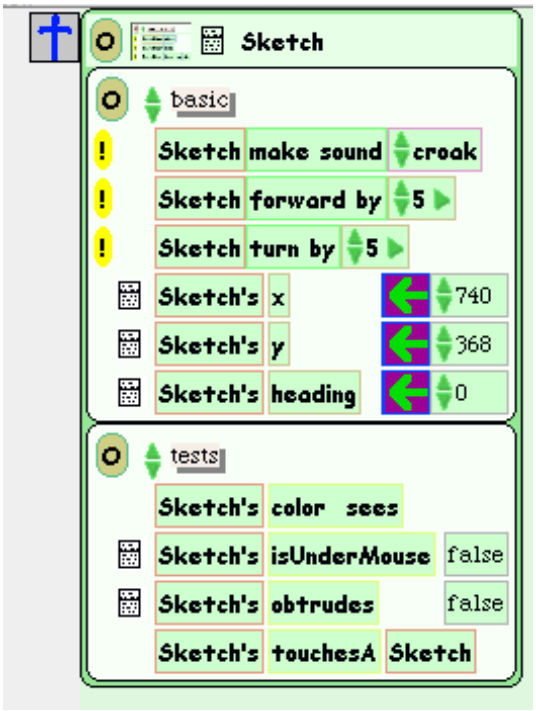
Step 2: Playing with halos. Now if you click on the sketch with the middle button you get halos around the sketch. Each halos has a color, a logo, and a function. Here we explain briefly each of them. Note that all the sketches do not show systematically the same halos some have less or different. You can always get a description of the halo by letting the mouse over it during a second.



- The pink halo with cross symbol destroys the sketch. Depending on your preferences, the sketch is simply put in the trash or really destroyed. When it is put in the trash you always have a chance to get it back and reuse it.
- The red halo small little rectangles brings the menu associated with the sketch. The menu depends on the sketch you are interacting with. It provides many actions to change the sketch.
- The black halo with pincer picks the sketch. Pay attention that picking a sketch changes its container. Use the brown halo to simply move the sketch inside its container. Use the red halo to embed into the sketch in the sketch on top of which it is.

- The brown halo with a square moves the sketch without changing its container.
- The green halo with two squares duplicates the sketch.
- The grey halo with a tool offers debug facilities which are normally used by expert squeakers.
- The dark gray halo with a pen repaints a sketch.
- The dark pink halo with a drop picker changes the color of the sketch. However, it does not work with sketches such as or airplane. To change the color of a sketch you have created you need to use the dark grey repaint halos. Once that is selected you will be taken back into the paint tools and can make any paint/color changes you wish. The dark pink handle is more appropriate to use to change font colors, rectangle colors, or border colors.
- The yellow halo with a square and bar changes the size of the sketch.
- The blue halo with a small whirling square rotates the sketch.
- The orange halo with a small rectangle produces a tile that represents the object for the tiling system of eToy.
- The cyan halo with an eye opens a viewer on the sketch. The viewer proposes a graphical representation of the methods and instance variables of the object.
- The pale green halo with an circle collapses a sketch.

To program in eToy we mainly use a viewer. Therefore open a viewer on the airplane by clicking on the cyan halo with an eye. You should obtain the tools shown by the following figure.



The following picture explains the main parts of a viewer. At the top the name of the sketch can be modified. By default your sketch is named Sketch. We suggest you to call it 'airplane' by editing the name in the viewer. Besides the name you can open a menu as shown in Figure 1.1. With this

menu you can add new variable to the object, you can get an empty script or a tile representing the object.

Then categories are displayed. The name of a category is displayed on the right of the two up/down green triangles. In Figure 1.1, the categories basics and tests are displayed. We can navigate them using the small double triangles. We can also click on the category name itself to get the list of the categories.

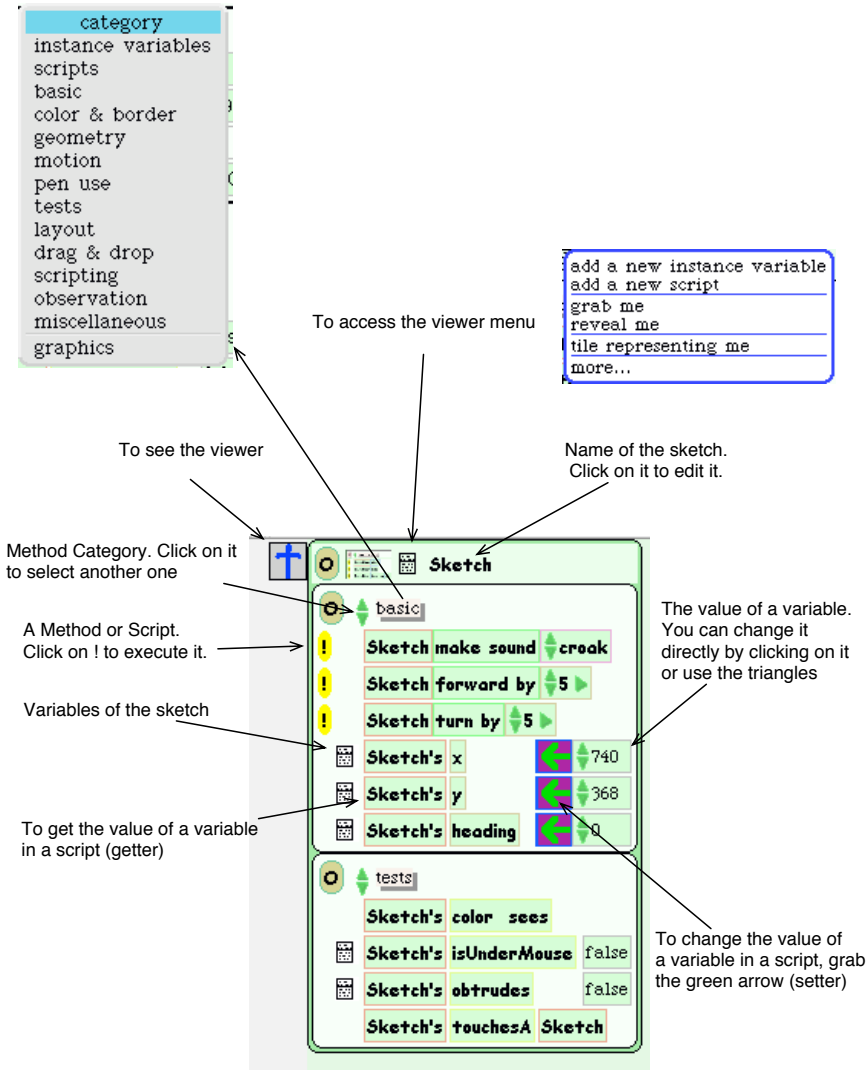


Figure 1.1: Understanding a viewer.

Below each category we have the list of methods. When a method can be executed, there is a yellow exclamation mark in front of it. Press the exclamation mark in front of the method forward by and you will see that the airplane is moving forward. Then you have methods that only access variable or change variable value. We call these methods accessors: getter methods and setter methods.

Note that you can modify directly the value of a variable by using the up/down green triangles or by directly typing the value in the variable box. Try for example to change the x variable to now be

800. In a script, as you cannot interactively change the value of a variable, you should use the getter and setter. You can get the getter for the variable x by dragging the x box in the desktop. To get the setter you should drag the big green arrow as shown by Figure 1.2.

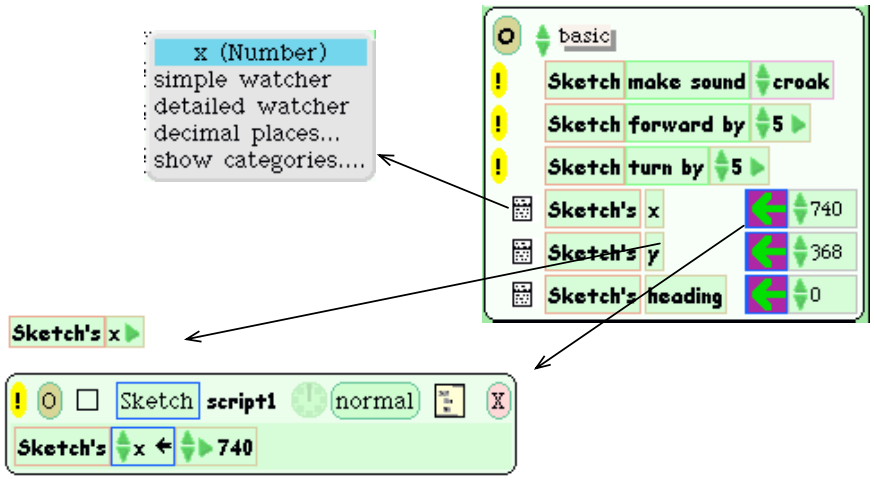


Figure 1.2: To get access to the variables within a script.

The number on the right of the big green arrow is the value of the variable. If you move the airplane using the brown or black halo you shall see the variables x and y changing their value.

You can also have watchers that is ways of spying the values of variables. To create a watcher, you should click on the small menu icon that is in front of a sketch variable. You should obtain the menu displayed in Figure 1.3. Then you can select to create a simple watcher which will only display the value of the variable or a detailed watcher that you can also use to change the value of the spied variable.

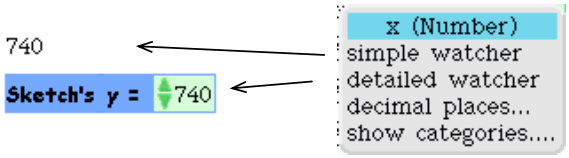


Figure 1.3: The watchers: way of spying variables.

Step 3: Dragging and Dropping a Method to Create New Scripts. Now if you take a method such as forward by from the viewer and drop it on the desktop, you create a script. You should get for example the following script as shown in Figure X3.

Figure X3

To execute a script you just have to click on the clock. The status of the clock is then set to ticking which means that your script is executed at regular time interval.

The following Figure shows the different parts of a script: there is the name of the sketch, the name of the script, a clock which lets us start and stop the script execution, the list of events the script can be linked to and the creation of conditional.

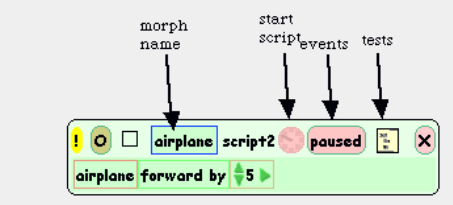


Figure XX2

Step 4: Adding Methods. Now the airplane is getting straight when we execute the script. What we would like is that the airplane turns. To do that we should add methods into the script. Drag the method turn by from the viewer and drop it into the script. The script should show you the place were you can drop the method by creating some green boxes. Drop the method into ones of the boxes. You should get a script that looks like the one in Figure x4.

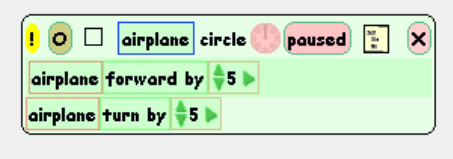


Figure x 4.

Now when you execute the script, the airplane should turn in circle. In fact a sketch is similar to a turtle and you can see what the airplane is doing by asking it to put its pen down. For that look for the variable penDown in the viewer and put its value to true as shown by the following script (see Figure X5) .

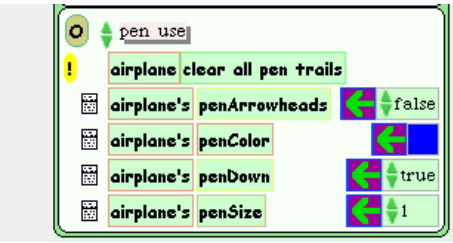


Figure X5

Once the pen is down the airplane draws on the desktop as shown by Figure x6.

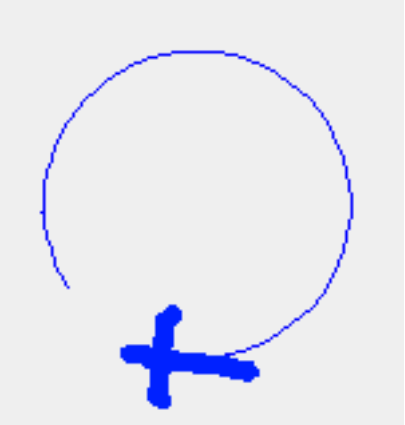
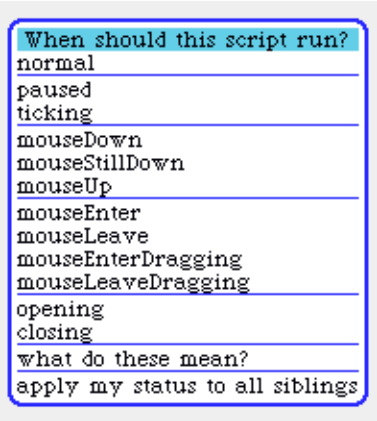


Figure X6

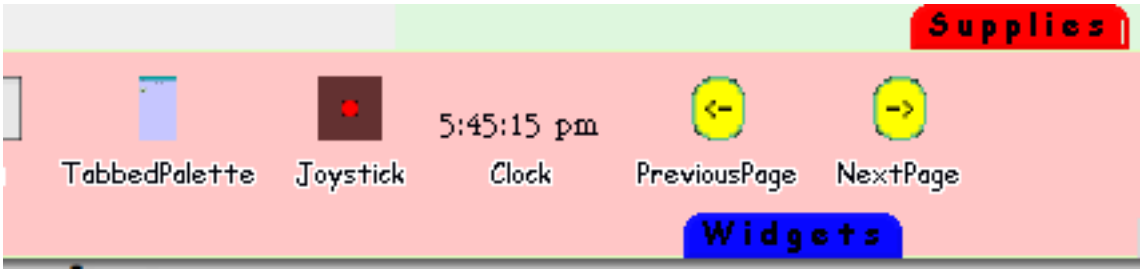
Note that you can select when the script is fired using the events button of the script. Figure X6 shows all the possible events that you can use. You can get this menu by clicking on the text right to the clock.



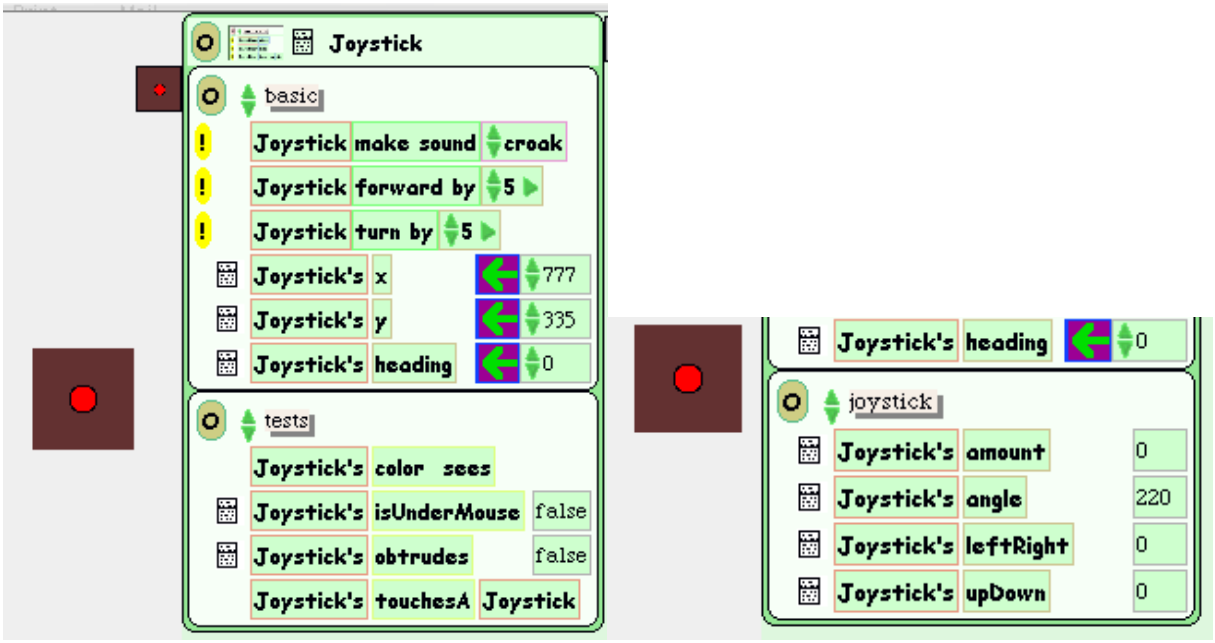
2 Joystick in Action

Now we would like to steer this airplane. The idea is that instead of turning always from the same amount we would like that the angle varies depending on the state of a joystick.

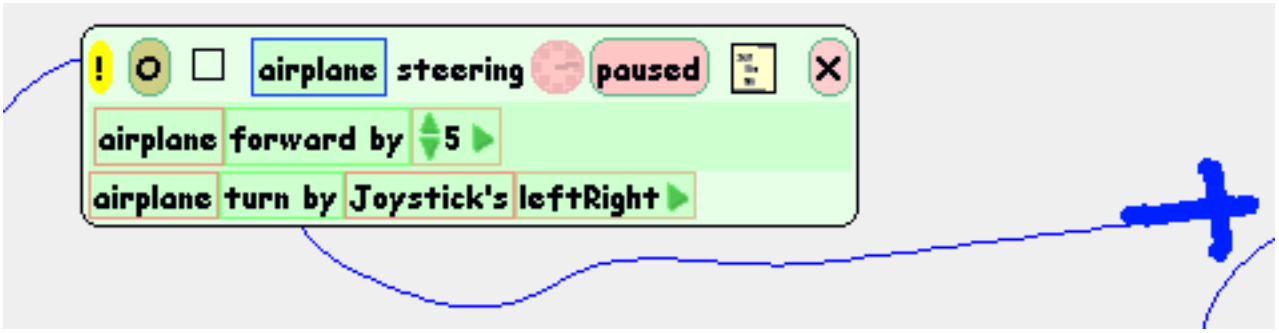
Step 1: Creating a Joystick. First we create a joystick by dragging one from the supplies flaps and dropping it into the desktop. The red circle represent the top of the joystick and with it you can indicate a direction and an amount of energy put in the movement.



Step 2: Experimenting with a Joystick. Second we open a viewer on the joystick using the turquoise halo. Then browsing the methods, you will find interesting methods under the category **joystick**. Move the joystick and look at the variables. The variable amount represents the amount of energy put in the movement, that is you can move the joystick in the same direction but with different strength. The variable angle represent the direction in which you are moving the joystick and we let you guess the roles of the two other variables.



Step 3: Linking the Joystick and the Script. Now to steer the airplane we have to change the value of the method turn by in the script by a value given by the joystick, for example the leftRight variable seems adequate. To do that drag the variable directly on the argument of the turn by method in the script. It may take some times for the script to accept the variable but you should obtain the following script.

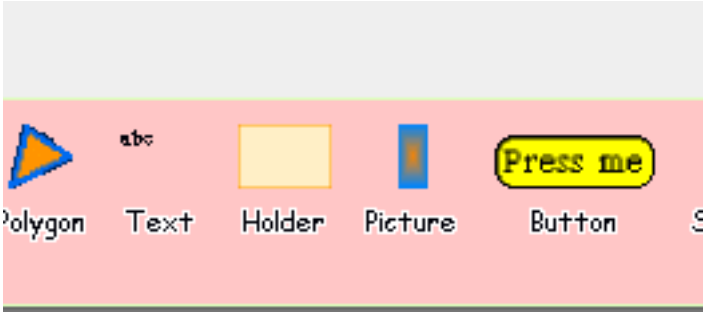


Now click on the clock and steer your airplane with the joystick. As you see steering the airplane could be improved by making a difference whether we turn fast or not. Try to find a solution, the amount variable can help you for that.

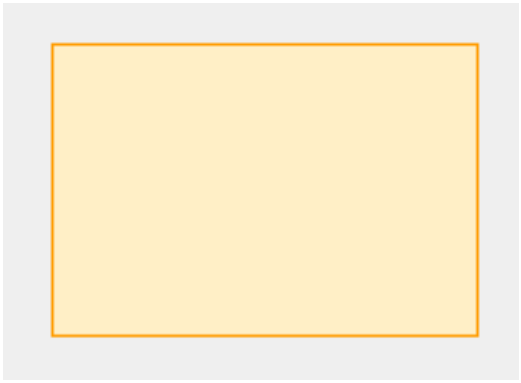
3 Creating an Animation

The idea for creating an animation is the following: first we draw the animations steps and put them into an animation holder. Then we create a simple sketch whose graphical representation will be replaced by the animation elements. For this purpose we write a script that makes the sketch looking like the different steps that are contained in the holder.

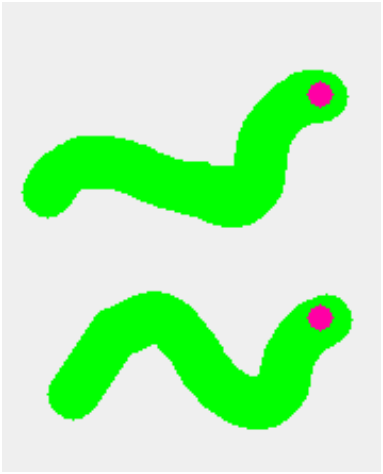
Step 1: Holder Creation. To create an animation we need to create an holder. An holder is a graphical object that can contain other graphical objects. It also knows what is the currently selected item among the ones it contains. To create an holder we drag it from the red flap named supplies, and drop it into the desktop.



An empty holder is then created.



Step 2: Drawing Animation Elements. For the second step we draw the picture we want to animate using the paint editor presented above. We suggest you to draw a first picture, then duplicate it using the green halo. Then select the dark grey halo to repaint the sketch. This way you can create several pictures by modifying one step by step. Here we paint a worm in two different positions.



Note that you can also use the red halo painting item (see Figure X7) but we suggest to use as much as possible the halos provided.

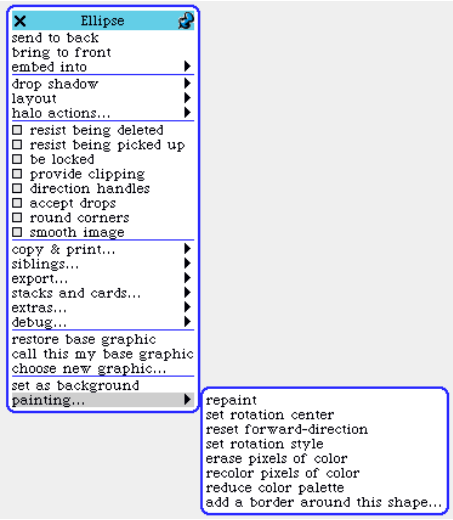
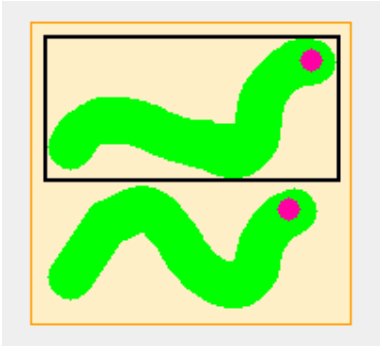
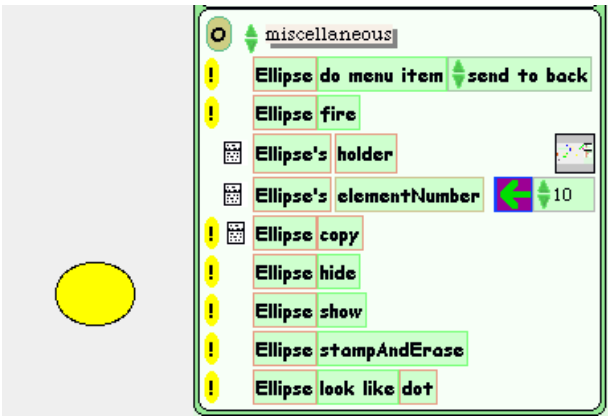


Figure X7

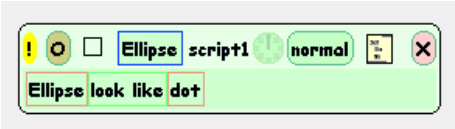
Step 3: Dropping the Pictures in the Holder. The next step is to simply drop the resulting pictures into the holder. A black rectangle represents the currently selected picture of the holder.



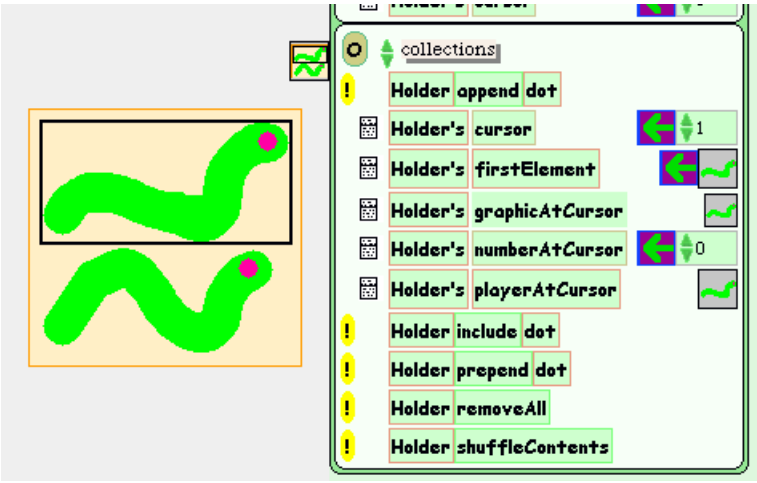
Step 4: Creating a Simple sketch Recipient of the Animation. Now we need a sketch whose graphical aspect will be used as a placeholder for the animation. Therefore we create a simple sketch for example an ellipse that we drag and drop from the supplies flap. Then we open a viewer on this sketch as shown by the next picture.



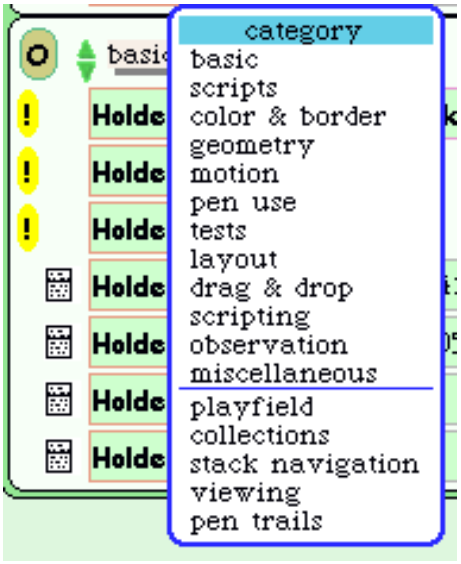
Step 5: Creating a Script with LookLike. From the simple sketch, here the ellipse, we create a new script by dragging the method lookLike dot (shown in he previous picture) from the viewer to the desktop. This action should create the script described by the following picture.



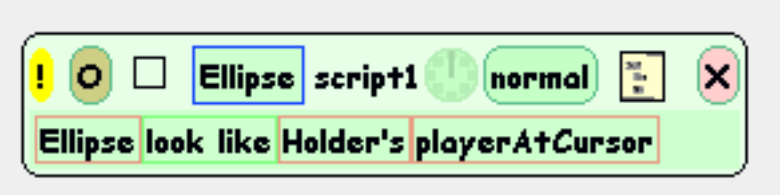
Step 6: Displaying the Selected Animation Element. Now we should indicate that the ellipse should look like the currently selected element of the holder. We select the holder and open it in a viewer as shown in the following picture.



We look for the category collection by pressing on the basic button as shown in the following picture.



From the list of methods we drag the method named playerAtCursor. This method returns the player, i.e, the graphical element contained in the holder that is at the current position, and we drop on the square with dot in the middle (this box represents an argument of the method lookLike). We obtain the following script.

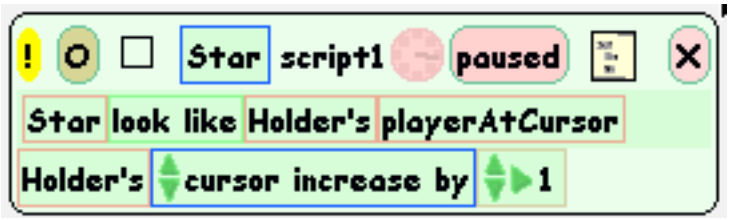


Step 7: Changing the Currently Selected Element of a Holder. If we execute the previous script using the ticking button, the script will just change the shape of the ellipse to be the currently selected graphical element. We do not have an animation yet. For that we need to find a way to change the currently selected item of an holder. In fact, an holder has an index, named cursor, that represents the rank of the currently selected item. It suffices to increment such an index to make the holder selecting another graphical element.

To change the value of the variable named cursor, we drag the arrow (shown on the right in the next figure), on the following line of the previous script.

We obtain the next script which say the the variable contains the value 1.

Now if you click on the double green triangles of the setter in the script you will see that there is the possibility to increase the cursor by a given amount as shown in the following script.



Now your animation should work.

Another Way. To achieve the same effect we can also reproduce the expression cursor := cursor +1 that will move the cursor to the next element. Therefore we drag the left part of the cursor from the viewer at the place of the 1 in the script and we get the following script.

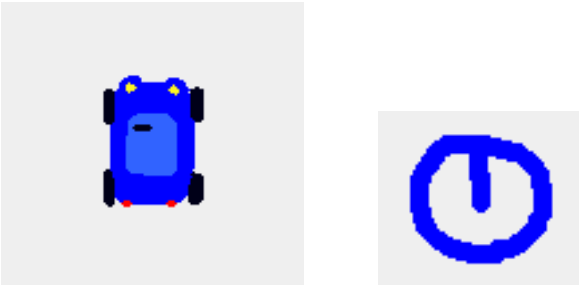
Now we just have to click on the small green triangle at the end of the cursor box in the script to make appear a plus 1 expression as shown in this final script. As you see the first solution is easier.

We suggest the reader interested to use eToy in his classroom to read the eToy book we mentioned in the start of this chapter, as it presents a lot of pedagogical aspects that can be used with this example.

4 Cars and Drivers

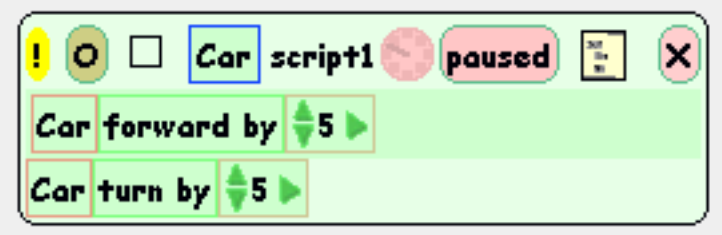
Now we would like to show you how we can build a script to drive a car with a wheel. This exercise is used for teachers to show the use of division to act as a demultiplior as in real cars, but this is also fun to do.

Step 1: Draw a Car and a Wheel. Using the painter editor draw a car and a wheel.



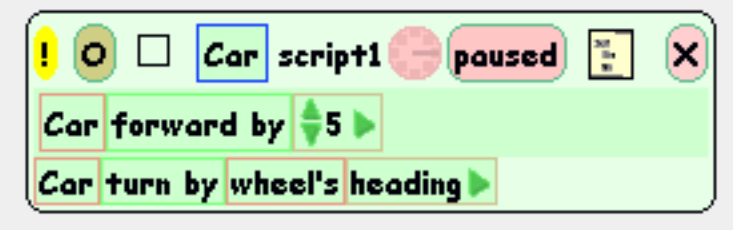
Step 2: Car Turning in Circle. Open a viewer on the car, rename the sketch to be named 'car', then drag and drop the method forward as we previously did. The following script should be created.

Then drag and drop the method turn below the previous method in the newly created script. You should obtain the following script that when run makes the car going in circle.



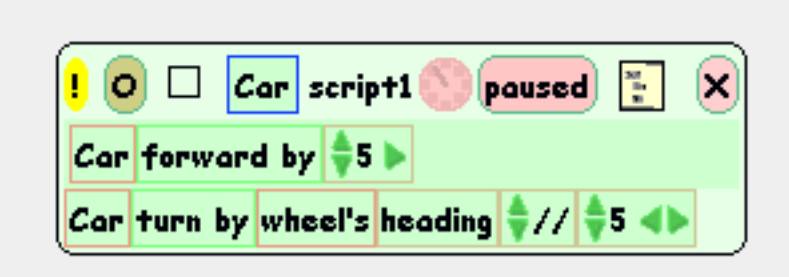
Step 3: Use Wheel Heading. Now we should link the angle of which the car turns with the wheel. We should find a way to rotate the wheel and know the amount of rotation. The first problem is easy to solve: If you bring the halos, click on the blue halo and turn. This rotates the wheel. Now for the second problem, open a viewer on the wheel, rename it wheel, look for the heading variable in the viewer (the expression should be wheel's heading). When you rotate the wheel using the halo, the value of the variable changes. This variable represents the rotation angle compared to the original picture.

Now we have all the pieces of the puzzle, we have to indicate that the car should turn not from a fix amount but from the wheel's heading. To do that drag the expression wheel's heading from the viewer onto the number 5 argument of the turn method in the previous script. You should have now the following script.



Now if you click on the small clock of the script, it runs and using the blue halo of the wheel you can control the car. However as you will notice this is not really perfect. Normally kids should emit hypothesis about the problem and try solution.

The problem is that we should divide the wheel's heading value to get a finer control when we are turning. To do that click on the rightmost small green triangle of the wheel's heading block. This adds extra boxes. Click on them to select the division //. You have now a script that looks like the following one.



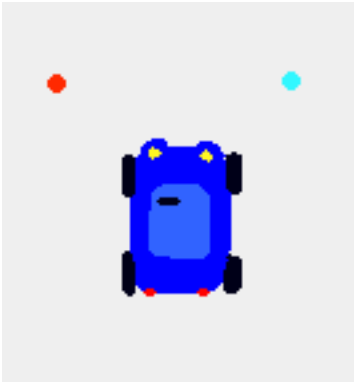
Have fun controlling your car.

5 Automatic Cars

Now we would like to program the car so that it finds its way on a road automatically. The idea is to equip the car with some sensors that tell it whether it is going off the road. Once the car is equipped with sensors and we program it, we want to put the car on the road and it should follow the road.

Note that here we will not show you a working solution because we believe that this is much better for you to experiment. Our solution in fact does not really work.

Step 1: Sensors. In eToy we can ask whether a certain color of a sketch sees another color under it. We can use this capability to have sensor. A sensor will then be a simple dot of color that will tell the car if it is passing above another color. To equip our car with sensors, we repaint the car and add two dots of colors as shown by the following figure.

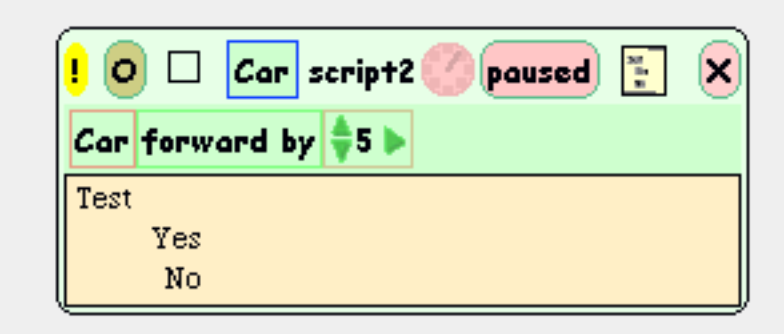


Step 2: The Road. Using the paint editor now we create a road with one single color. We suggest you to try later with different bands of colors to see how you have to change the behavior of the car.

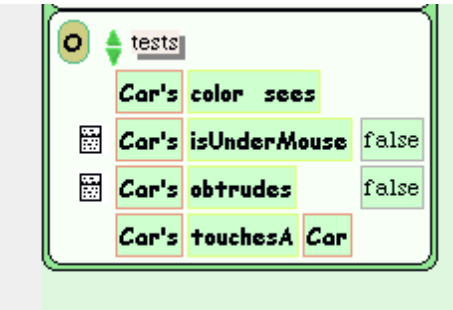
Step 3: Condition and Tests in Etoy. Open a viewer on the car then drag and drop the method forward by into the desktop. This creates a script that you normally get used.

Now we need to have a way to express different behavior depending on the sensor value. For that purpose we need to have the possibility to express a condition or a test. To get test you should drag the

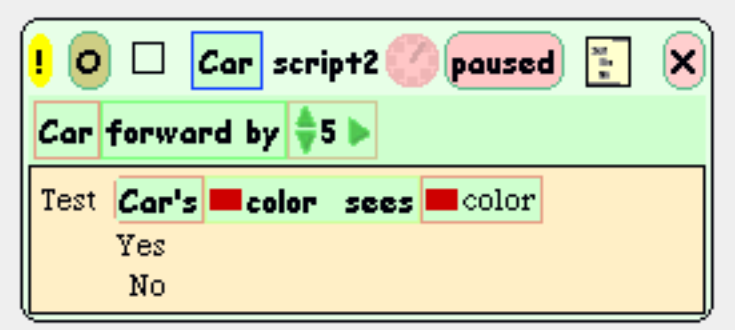
second from the right small pale square box at the top of the script near the cross button (see Figure XX2). So drag and drop such a tile test in the script, you should now get a script which looks like the following one.



Now we need to find a way to make our sensors active. Look and find the method color sees in the category test as shown below. This method returns true or false depending whether a colored part of the sketch of a given color passes over another color.

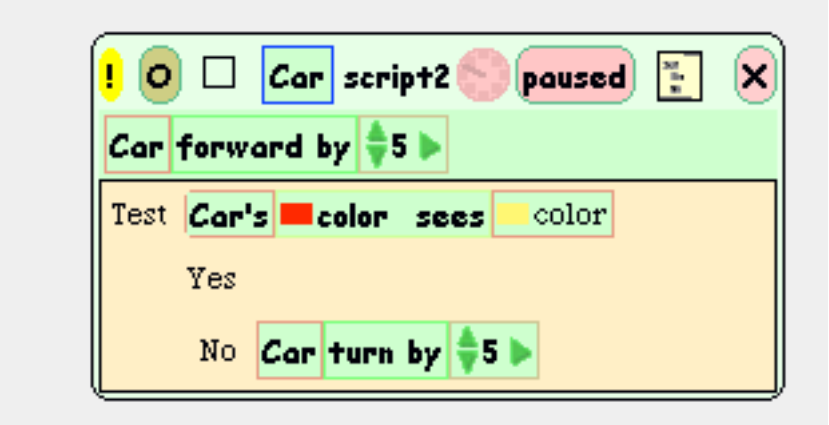


Drag such a method into the script besides the word Test. You should obtain a script similar to the following one.

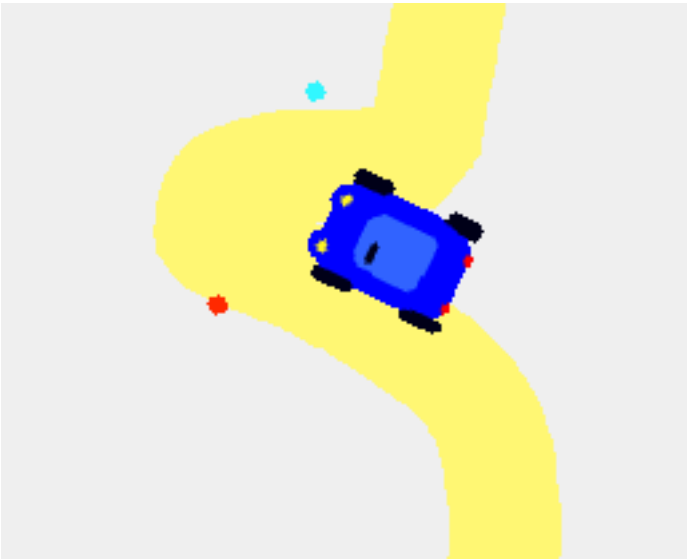


Step 4: Customizing Color-based Tests. Now we should have a way to define the color that the test should use. This is easy, we have to click on the colored square that is inside the method color sees itself. This automatically opens a color picker (see Figure YX). With the color picker picks the color of one sensor. The first color of the script should chane to reflect the color you selected. Do the same operation for the second colored box but this time pick the color of the road.

Figure YX



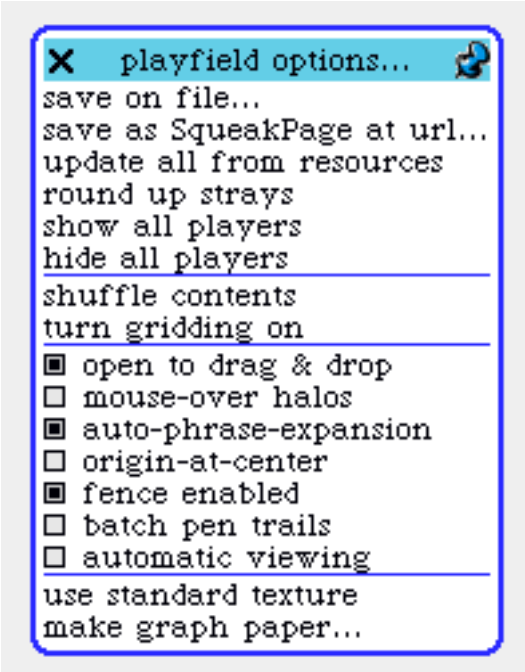
Step 5: Adding Actions. Now we can specify that the car should turn when the sensor does not see the color of the road. We just have to drag and drop the method turn by besides the word No in the script. Now you can pick the car, put it on the road and press the clock to run the script.



Obviously the behavior is not perfect and we let you change it or define your own solution.

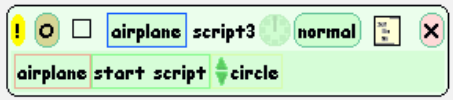
6 Some Tricks

We just want to show you some aspects that may help you. The eToy environment can be customized, we suggest you to open the menu **playfield options...** and let your mouse wander on the different items and read the balloon help.



Running Several Scripts. If you create several scripts they will run in parallel. To control all the scripts available on the desktop you can use the widget named All Scripts that is available in the **widget** flaps. Once you drop this widget in the desktop you get a panel that allow you to run and stop all the scripts currently on the desktop.

Note that you can also use the method start script and the related method contained in the category scripting to start, pause, or stop scripts.

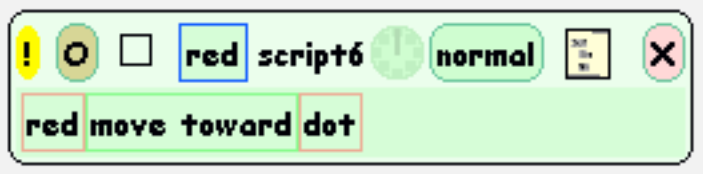


It is interesting to see that one script can invokes the exuction of other scripts. This let you create more complex scripts.

Cleaning. You can clean all the turtle trails using the method clear all pen trails that is available in the category pen use.

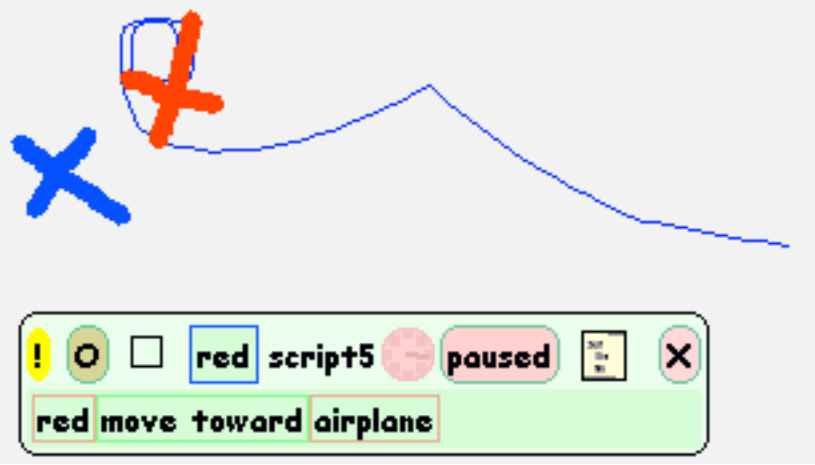
You can also clean the desktop from the traces made by the players. For that purpose use the last menu item of the menu **appearance** that you can get from the **World** menu.

Making A Tile. if you want to write a script which links two objects, you sometimes need to refer to this object. In such a case,you need to have a tile that represents the object to drop it in your script. Let us take an example, imagine we have two airplanes one blue and one red and we want the red to follow the blue one. We use then the method move toward as shown below.

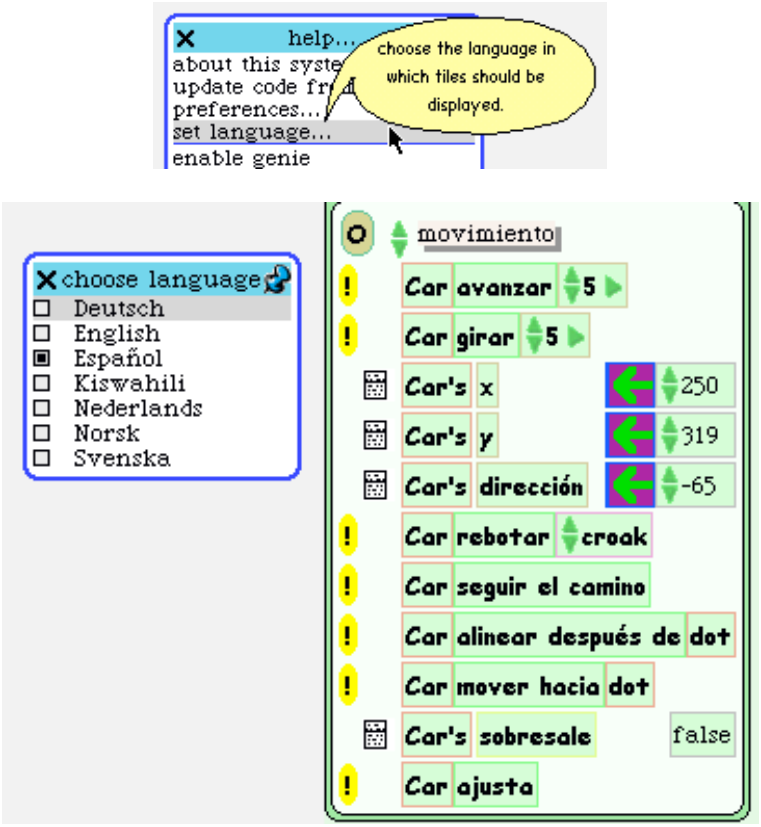


Now as we want that the red airplane move toward the blue airplane and not the dot as in the previosu script. We first use the orange halo of the blue airplane to get a tile representing it.

Then we drop this tile in the script. Now the red airplane is fllowing the blue one.



Internationalization. Now if you want to use eToy with small kids, you can choose the language you want. Sometimes the complete Squeak interface is translated. Bring the World menu, select the item **help...** followed by **the set language...**.



Again this is just a survey of the possibilities of Etoy and I suggest you to go to http://www.squeakland.org and check the material available there.