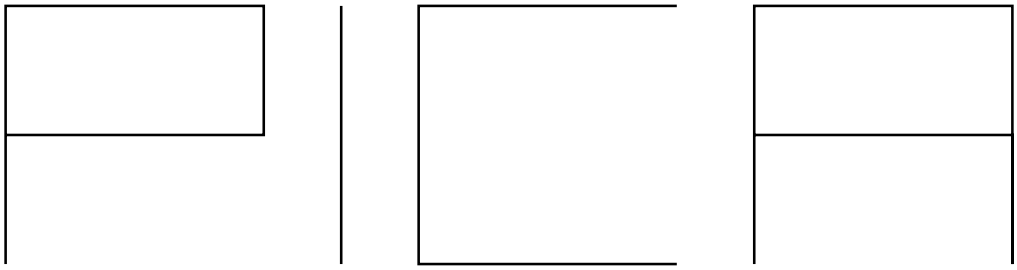
1

Of Robots and Men



In this chapter we describe the creation of robots, and the different types of movements that robots know. We propose you some simple experiments to practice what you learned in the previous chapters. We also present how robots may change direction in an absolute manner.

1 Creating Robots

In the previous chapter we mentioned that we created *a* robot, not *the* robot. Indeed, we can create as many robots as we want. Here is a new script 1.1 that creates two robots pica and daly.

Script 1.1

```
| pica daly |
pica := Bot new.
daly := Bot new.
pica color: Color yellow.
daly jump: 100.
```

The second line creates a robot named pica as in Script **??**. The third line creates a new robot that we refer to using the variable daly. Both robots are created at the same location. In line four, we ask pica to change its color so that we can distinguish the two robots.

Smalltalk is an object-oriented programming language. This means that we create objects and interact with them, or that objects can create other objects and communicate with them. Moreover, in Smalltalk, there are special objects, called *classes*, which are used to create objects. Sending the message new to classes creates an object described by its class. Sending the message new to the Bot class creates a robot.

To understand what classes are imagine that a class is a factory. A factory of tin boxes creates many boxes of the same size and shape. After being created, some boxes can be filled, others can be crushed. When one box is crushed, the other boxes are not crushed. The same thing happens with

object created inside Squeak. In our case, pica did not move, but daly did. You can think of a class as a factory able to produce unlimited supplies of objects of the same type. Once produced, each object exist independently of the other.

Class names always begin with an uppercase letter. That's why the name of the robot class is Bot with an uppercase "B". Color is also a class producing color objects but we should use a color name to specify the color we want to obtain. For example, Color yellow creates the color yellow.

> A class is a factory of objects. Sending the message new to a class creates an object of this class. Class names always start with an uppercase letter. Here Bot is the factory creating new robots and Color the one for colors.
>
> ```
> Bot new color: Color blue
> ```

2 Drawing Line Segments

Asking a robot to draw a line is rather simple as we already saw in the previous chapter. The message go: 100 asks a robot to move ahead 100 pixels and it leaves a trace during its move. However, even if you were an expert Chinese or Japanese caligrapher, you need to lift the brush from time to time when you drawing. For this purpose a robot knows how to jump, *i.e.*, moving forward without leaving a trace. A robot understands the message jump:, the argument is the same as with go:, it is a distance in number of pixels. Here is a script which draws two segments. Note that we are hiding the robots from the illustration using the message beInvisible so that you can get clearer pictures.

Script 1.2 (*Two lines)*

```
| pica |
pica := Bot new.
pica go: 30.
pica jump: 30.
pica go: 30.
```


Experiment 1.1
Experiment by changing the values in the previous script.

Experiment 1.2
Generate a script able to draw an SOS message using Morse code. In Morse, an "S" is represented by 3 short lines and an "O" is represented by three long lines as shown by figure on the right.

3 Changing Directions

A robot can orient itself along the eight directions as shown by Figure 1.1. These directions are like those of a map: east is pointing to the right, west to the left, north up, and south down. To make a

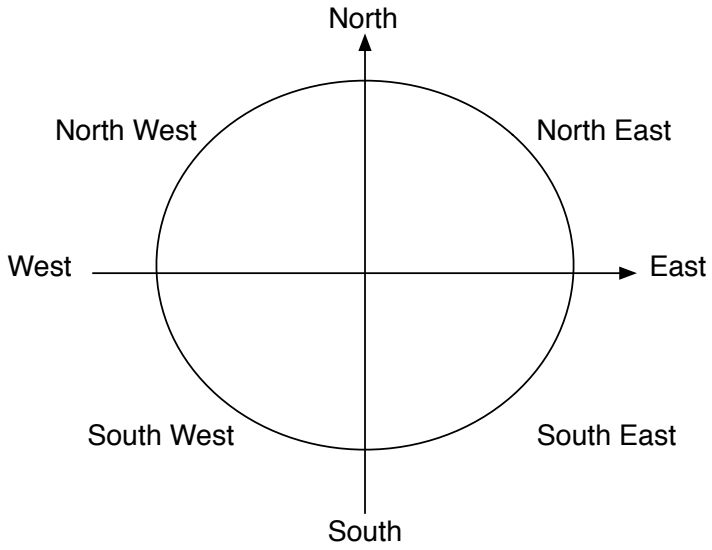


Figure 1.1: The default absolute directions to which a robot can point to.

robot pointing in a given direction just send it a message with the name of the direction. So, to ask pica to face south, one simply types pica south.

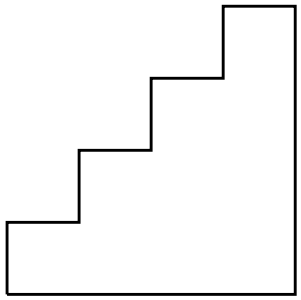
Robots understand the following messages: east, north, northEast, northWest, south, southEast, southWest and west. Note that in the next chapter we shall show you how to make a robot turns in a relative manner and from any angle.

The script 1.3 illustrates simply the basic four directions with four different robots. In this script, we use four different robots. Each of them, except pica, is oriented toward a different direction before moving. Now we can use the orientation methods to make more complex drawings. As a first exercise draw a square of 50 pixels size. Then draw another square of 250 pixels.

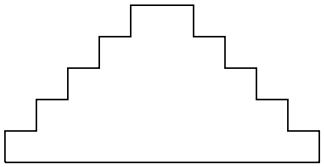
Script 1.3 (*A gaggle of robots)*

```
| pica daly klee sisl |
pica := Bot new.
pica color: Color green.
pica go: 100.
daly := Bot new.
daly north.
daly color: Color yellow.
daly go: 100.
klee := Bot new.
klee west.
klee color: Color red.
klee go: 100.
sisl := Bot new.
sisl south.
sisl go: 100.
```

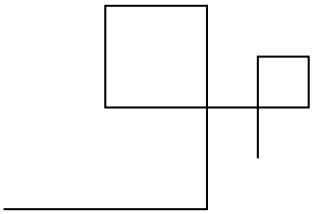
Experiment 1.3
Hopefully we are not limited to squares. One can do a wide range of geometrical figures. For example, here is a drawing of a small stairs. Define the script that reproduces it.



Experiment 1.4
Now, we are ready to draw a schematic side view of the pyramid of Saqqarah, built in 2500BC by Imhotep, the first known architect. Generate a script able to draw a side view of the pyramid of Saqqarah. This pyramid has four terrasses and its top is twice as large than the terrasses as shown in the drawing on the right.



Experiment 1.5 (*Pattern)*
Draw the picture shown on the right.

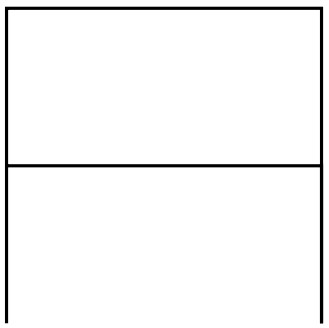


4 The ABC of Drawing

Even though the control of the direction of the line segments is somewhat limited, we can even start to program pica to write letters. The script 1.4 draws an "A".

Script 1.4 (*The letter A)*

```
| pica |
pica := Bot new.
pica north.
pica go: 100.
pica east.
pica go: 100.
pica south.
pica go: 100.
pica north.
pica go: 50.
pica west.
pica go: 100
```



Drawing a letter "C" is no more difficult. We can write a script to write "pica".

Experiment 1.6
Draw the name of pica. Note that as we do not want to have a line between the "C" and the "A" you should use jump:.

A Remark. Some purists may think that the script 1.4 could be improved. Indeed, the bottom half of the left bar of the "A" is drawn twice since the robot is going twice over this segment — once going south, once going north. Knowing how to write the best program is a difficult question, lot of issues like the speed, the readibility of the code have to be answered and depending of the language, the methods used,..., so there is no simple answer. However, if you ask yourself such a question chose the simplest one. Then if you have problem after because you program is too slow you can always speed it up.

5 Controlling Robot Visibility

We can control whether the robot should be displayed or not using the message beInvisible and beVisible. beInvisible hides the receiver. A hidden robot acts exactly as a normal one but it just does not show where it is. Be careful attention because the method hide already exists but it is defined by Squeak and can damage the robot environment. beVisible shows the receiving robot. A newly created robot is showing itself by default.

Summary

Expressions Messages	Description	Example
Bot new	Create a computer robot	pica := Bot new
| x y |	Declare variables used in the script	| pica |
jump: anInteger	Ask a robot to move forward by a given number of pixels without leaving a trace	pica jump: 10
go: anInteger	Ask a robot to move forward by a given number of pixels while leaving a trace	pica go: 10
beInvisible	Ask a robot to be invisible	pica beInvisible
beVisible	Ask a robot to be visible	pica beVisilble
east, north, northEast, northWest, south, southEast, southWest and west	Ask a robot to point to the corresponding direction	pica north
Color *x*xx	Create a color	Color blue
color: aColor	Ask a robot to change its color.	pica color: Color red