1, 2, 3, code ! - Cycle 3 activities - Step 2.1. Introduction to the Scratch programming environment

Summary	The students learn about *Scratch*, a programming environment suited to elementary school students. They learn to launch the program and combine a few simple instructions.
Key ideas *(see Conceptual scenario)*	"Machines" • The machines around us merely execute orders (instructions). • By combining basic instructions, we can make them execute complex tasks. "Algorithms" • An algorithm is a method used to resolve a problem. • A loop allows the same action to be repeated multiple times. • Certain loops, known as "infinite loops," never stop. • Certain loops, known as "iterative loops," are repeated a predefined number of times. "Languages" • To give machines instructions, we use a programming language, which can be understood by both machines and people. • *Scratch* is a graphical programming environment that uses a simple language. • A program is the expression of an algorithm in a programming language. • Certain instructions are only executed when an event is triggered. This is known as event-driven programming. • Certain instructions are executed one after the other. This is known as sequential programming. • The execution of a program is reproducible (if neither the instructions nor the data to manipulate change, the program always gives the same result).
Equipment	For the class • Projector • Enlarged version (A3 or A4) of Handout 1 For each pair of students • A computer with Internet access or, if there is no good connection available, a computer with *Scratch* preinstalled. For each student • Handout 32
Glossary	Program, script, sprite, instruction, event
Duration	1 hour

Reminder:

It is essential for the teacher to complete the exercise before proposing it to the students! All they have to do is follow the steps in the sequence. Teachers with no *Scratch* experience will need about 3 hours to do the whole project, including black activities.

Teaching notes:

- You learn to program by programming, not by watching someone else program. It is interesting to consider the same problem in pairs (probably more so than to program alone), but it is important to be active. We therefore recommend placing the students in small groups in front of computers (ideally two students per machine) and asking them to "switch over" (pass the keyboard and mouse to their neighbor) every 10-15 minutes.
- If possible, we recommend organizing the class in half-groups, so as to avoid having too many pairs to manage at once. While half the class works on *Scratch*, the other should do something else independently.
- If possible, two *Scratch* sessions should be organized weekly, at least at the beginning.
- This step of getting to know *Scratch* is deliberately very directive (activities 0 and 1 are actually a demonstration by the teacher!). It is the only step presented in this form. All pairs will have to carry out a series of basic tasks. At the end of each activity, a group discussion ensures that everyone has understood and knows what to do. The other steps will be less directive, as students become more independent and progress at their own pace.
- To save time, switch on the computers before the session begins.

● Activity 0: Demonstration of the final game by the teacher (5 minutes)

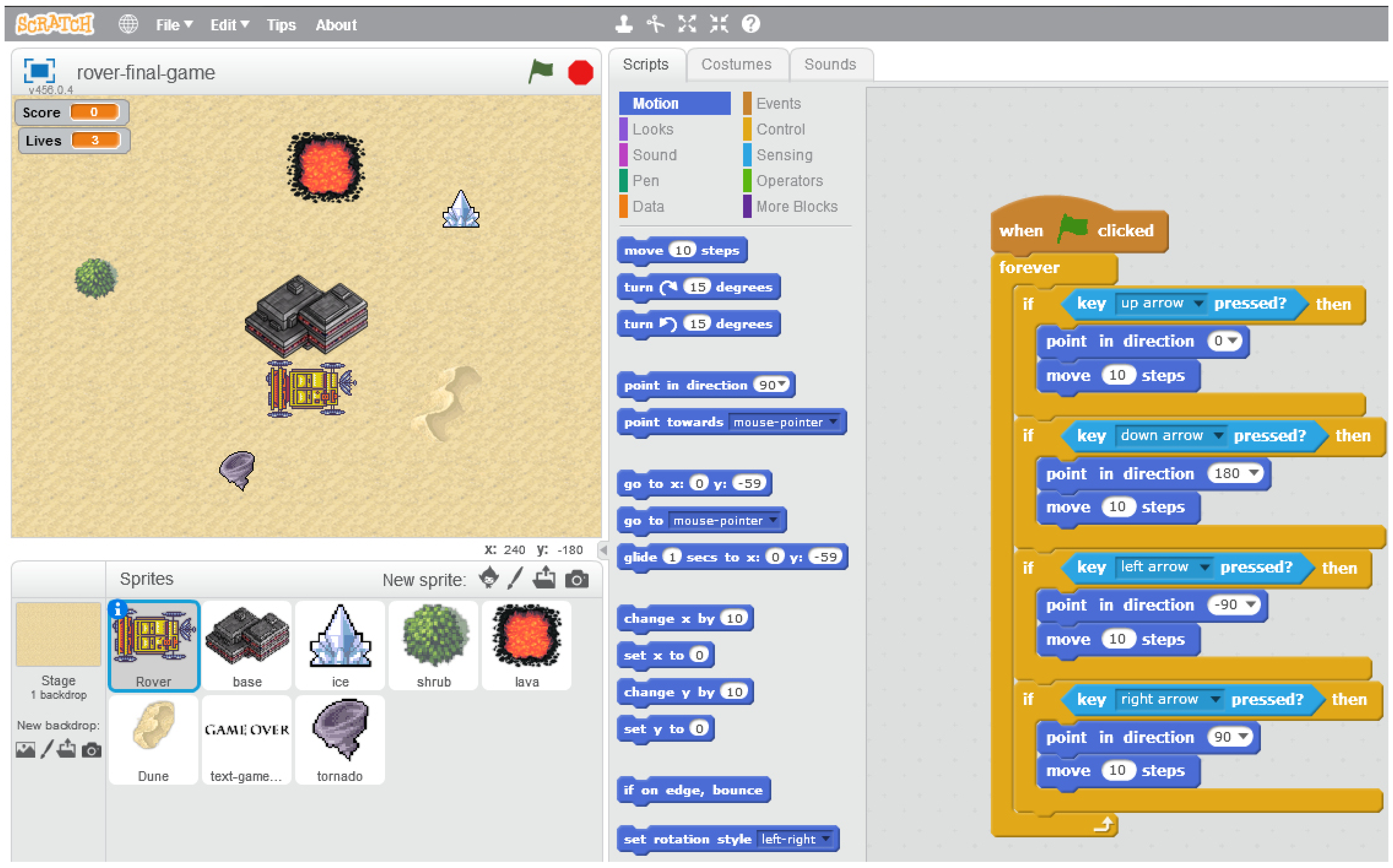
The teacher explains that the aim is to simulate an exploration mission (seeing as we can't go for real) through a video game we are going to program ourselves.

From their computer, the teacher opens *Scratch* and shows the "final" video game (that they have produced in advance), simply demonstrating the game without explaining how the program works.

Teaching notes:

- This demonstration is very important, as it is highly motivating and reassuring for students: they are actually going to program a "real" video game! It also helps tie in this programming activity with the unplugged activities in Sequence I.

  "*Here is our rover, which we are going to learn to move. It will collect resources to allow humans to live in the base, such as water and food (for each resource collected, the score increases). But watch out for traps! If the rover falls into a trap, it loses a life. Once it has no lives left, the game is over."*
- It is important to explain to students that such a game cannot be programmed in a single session but will require several (typically six or seven sessions, depending on their level and ambition).



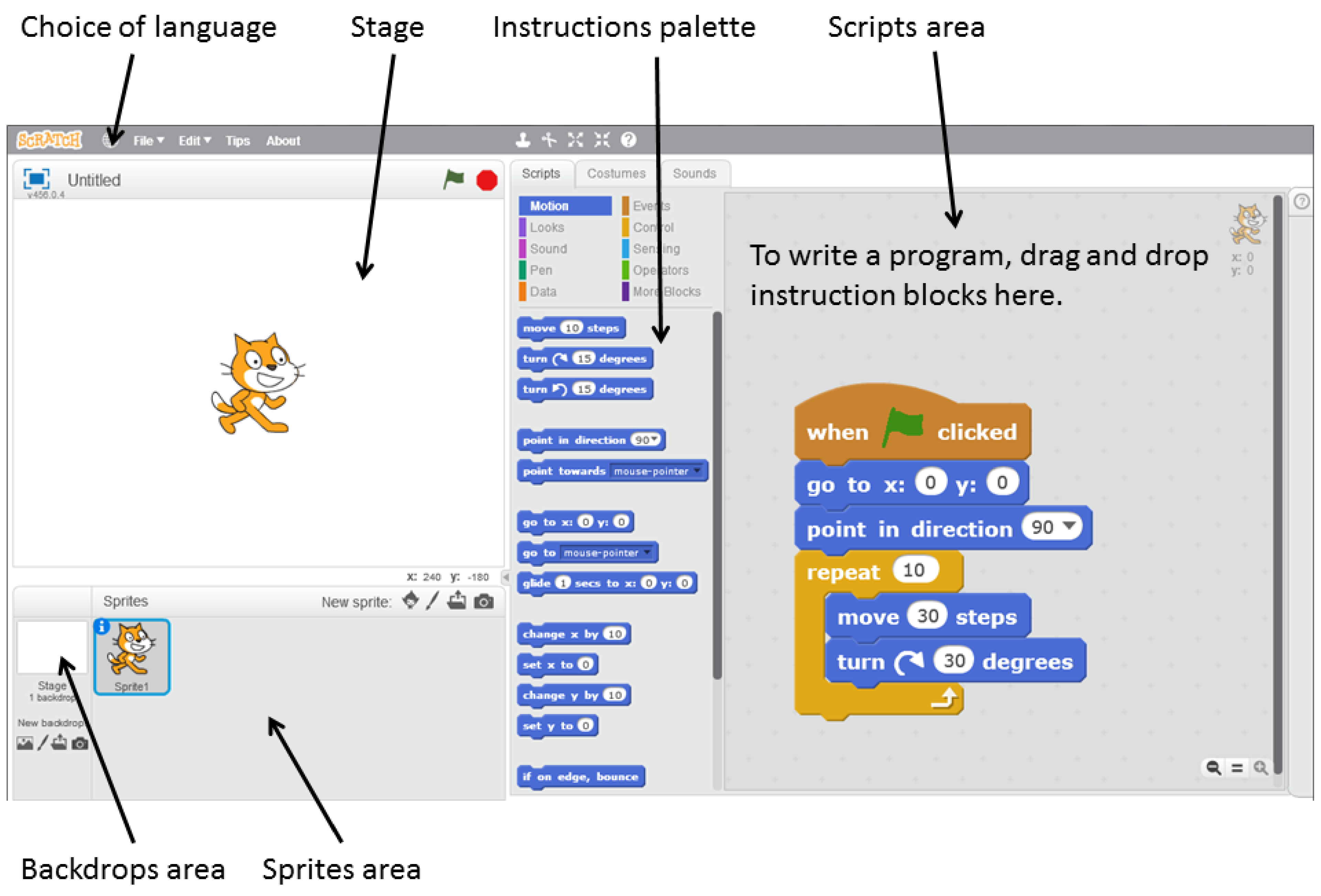
```
when [green flag] clicked
forever
    if key [up arrow] pressed? then
        point in direction (0)
        move (10) steps
    if key [down arrow] pressed? then
        point in direction (180)
        move (10) steps
    if key [left arrow] pressed? then
        point in direction (-90)
        move (10) steps
    if key [right arrow] pressed? then
        point in direction (90)
        move (10) steps
```

Screenshot: final version of the video game produced by the teacher
(Please note: a model program and an export in Flash format are available on the project website, page XXX).

Activity 1: Opening Scratch and getting to know its interface (10 minutes)

Each pair of students should **open *Scratch*** by double-clicking on the icon. They should cancel any update windows.

Scratch is usually configured in English the first time it is used on a machine. If necessary, students can change the language very easily by clicking on the globe icon (top left, next to the *Scratch* logo).



To write a program, drag and drop instruction blocks here.

```
when [green flag] clicked
go to x: (0) y: (0)
point in direction (90)
repeat (10)
    move (30) steps
    turn (30) degrees
```

The teacher should explain to the students that *Scratch* is a programming language designed specifically for learners. When you open the application, there is a sprite (a cat) on the screen. You can give it simple instructions.

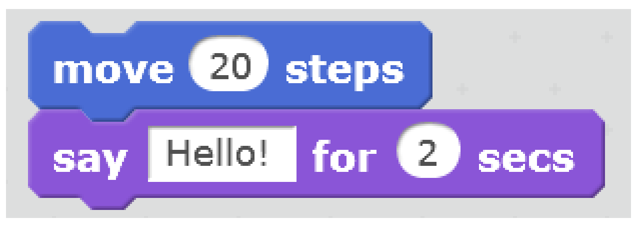
The teacher carries out a short demonstration (at the end of the session, the students will repeat these exercises).

For example, to ask the cat to move 10 steps, you can simply drag the instruction block "move 10 steps" from the scripts tab into the scripts area. If you then click on the block, you will see that the cat does move 10 steps (1 step = 1 pixel on the screen).

move (10) steps

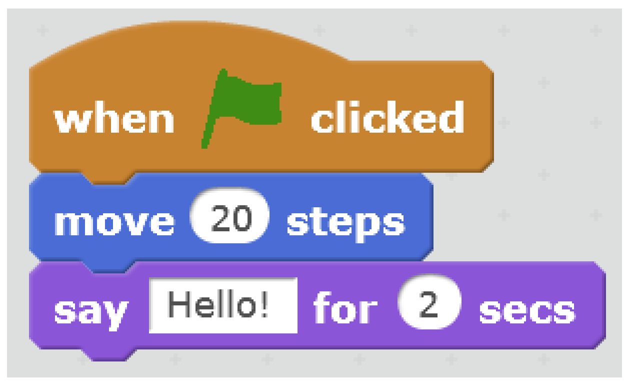
If you want to move 20 steps, you can simply replace "10" with "20" by clicking on the number.

If you now want the cat to move 20 steps and then say "Hello!," simply add the new instruction at the end of the program. The instruction "Say Hello!" is located under the "Looks" category of the scripts tab. You can replace the text "Hello!" with any words you like by clicking on it. You can write a program simply by snapping together the instructions in order.



```
move 20 steps
say Hello! for 2 secs
```

If you want the cat to do that each time you click on the green flag (top right of the stage; the green flag launches the program), then add the instruction "when green flag clicked," which is to be found under the "Events" category on the scripts menu. The result is:



```
when 🏴 clicked
move 20 steps
say Hello! for 2 secs
```

Finally, the teacher should show students how to delete an instruction (or a whole stack of instructions): simply drag the instruction (or stack) from the scripts area towards the scripts tab.

The teacher should very briefly introduce the *Scratch* interface, which includes:

- A **"stage"**: this is where the "game" (or, more generally, the program – *Scratch* isn't only for games!) is executed.
- A **"sprites" area**: sprites are the characters or objects that are controlled in the program (they can move, change shape, speak, interact with other sprites, etc.). When you open *Scratch*, only one sprite is displayed on the screen: a cat (other sprites will be added later, and the cat will be deleted).
- A **"backdrops" area**, right next to the sprites: the backdrop is static, unlike the sprites that can move. Be default, the backdrop in *Scratch* is a plain white screen (this will be changed later).
- A **"scripts" tab**, which offers:
    - A **scripts tab** (central column, to the right of the stage). This is where the instructions (or "blocks") we will use to build our program are to be found. There is a wide variety of instructions, grouped by color (e.g. anything to do with the motion of the sprite is in the deep blue category, anything to do with its looks is in the purple category, etc.).
    - A **"scripts" area**, to the right of the scripts tab. This is where the program is written, simply by dragging and dropping instructions from the tab into this area.
- The other tabs (Costumes, Sounds) are not useful for now.

🟢 Activity 2: Exploring Scratch independently (15 minutes)

The students have 15 minutes to explore *Scratch* by themselves. For now, they shouldn't try to change the stage or the sprite (that's for the next step, on page XX). They should simply try out simple instructions and order them to see what happens. The teacher should encourage them to explore the various categories of instructions, including:

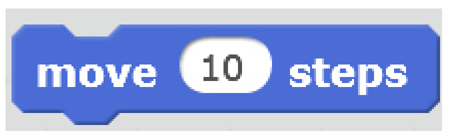
- "Motion" (deep blue)
- "Looks" (purple)
- "Events" (brown)
- "Control" (gold)

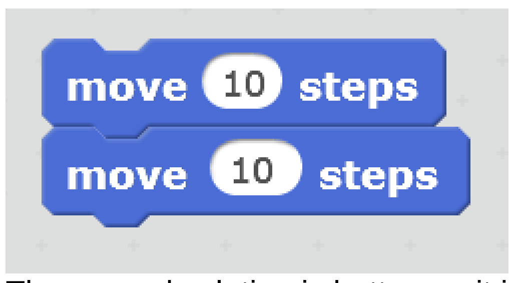
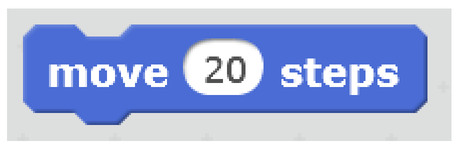
Third-grade class of Emmanuelle Wilgenbus (Antony, France)

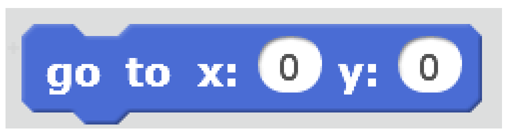
Teaching note:
Establish a rule of taking turns from the outset, so that the same student does not always control the keyboard and mouse.

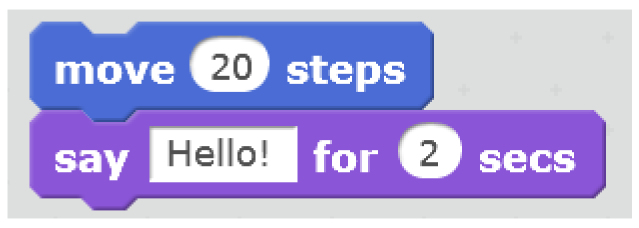
🟢 Activity 3: Short exercises (20 minutes)

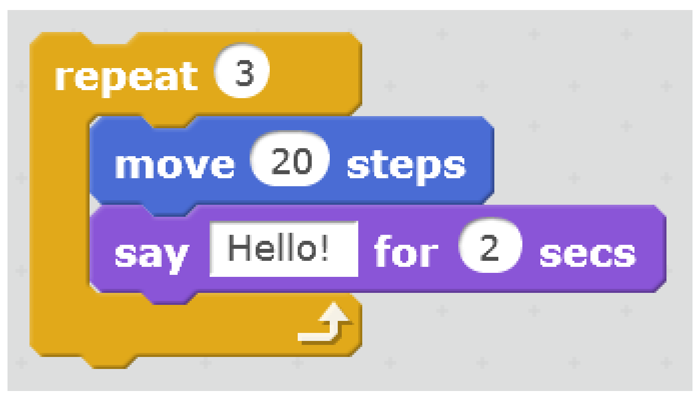
The teacher should give a series of short exercises (mostly going over what was done previously in the demonstration) for the students to carry out. After each exercise, a quick group discussion will ensure everyone knows how to do the exercise.

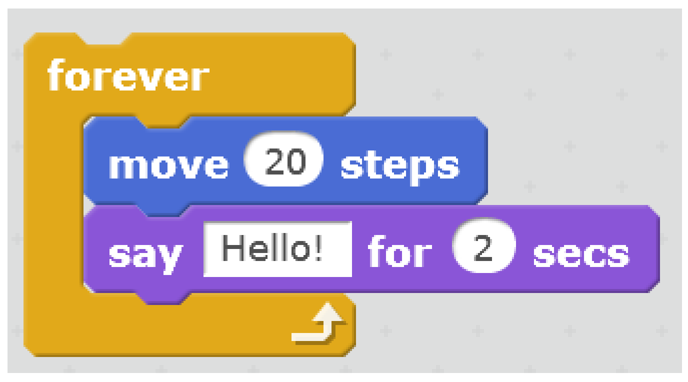
🟢	**Exercise 1 – Make the cat move 10 steps**  `move 10 steps`

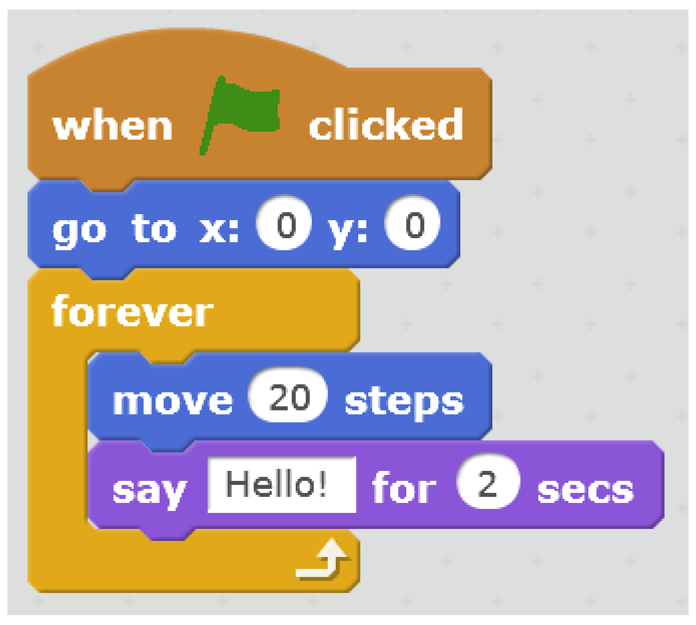
🟢	**Exercise 2 – Make the cat move 20 steps**
	Two possible solutions:
	
	`move 10 steps`
	`move 10 steps`
	or
	
	`move 20 steps`
	The second solution is better, as it is more elegant and easier to read.

🔴	**Exercise 3 – Return the cat to the center of the stage**  `go to x: 0 y: 0` Some students will no doubt get the trick, but most will need to be shown. However, it is essential for them to see this instruction now, as their movements will eventually send the cat off the screen and they won't know how to get it back.

🟢	**Exercise 4 – Make the cat move 20 steps and say "Hello!"**
	
	`move (20) steps`
	`say [Hello!] for (2) secs`
	Stress that "say" hello means "write" hello here: a speech bubble should appear on the screen with the text "Hello!" inside – we don't want to make the cat talk!
	Please note: It is possible to make the cat "talk" (play musical notes or a sound file imported into or recorded in *Scratch*), but it is highly unadvisable in class.

🔵	**Exercise 5 – Repeat 3 times: Make the cat move 20 steps and say "Hello!"**
	
	`repeat (3)`
	`    move (20) steps`
	`    say [Hello!] for (2) secs`
	Encourage students who do not find what they need to look in the "control" category (gold). They will find a similar instruction "repeat 10" that can easily be changed by replacing "10" with "3." That loop fits around the other instructions. Everything within the loop is executed three times.
	The cat stops for 2 seconds between each movement. To reduce this pause, simply shorten the period during which it says "Hello!" (if you write 0.5 instead of 2, the cat will only stop for half a second each time).

🟢	**Exercise 6 – Repeat forever: Make the cat move 20 steps and say "Hello!"**
	
	`forever`
	`    move (20) steps`
	`    say [Hello!] for (2) secs`
	This is very simple to do, using the same principle as the previous exercise but with a different sort of loop.

🔵	**Exercise 7: Same thing when you click on the green flag**
	
	`when ⚑ clicked`
	`go to x: (0) y: (0)`
	`forever`
	`    move (20) steps`
	`    say [Hello!] for (2) secs`
	All you really have to do is add the instruction "when green flag clicked" (from the "events" category), but it is even better if you ask the cat to start again from the center of the stage.
	This is when to explain the purpose of the **red button** (next to the green flag). Clicking on the red button stops the execution of the program (which would otherwise never stop in this case).

Review and conclusion

The class summarizes together what they learned in this lesson, including by listing the *Scratch* instructions they now all know.

- *move 10 steps*
- *go to x: _ y: _*
- *say Hello! For 2 secs*
- *repeat 10*
- *forever*
- *when green flag clicked*

It can be very useful for students to color the *Scratch* instructions one by one as they discover and understand them. After each new step, you can therefore see the progress of each pair and the whole class.

Handout 32 "Some useful *Scratch* instructions" can be photocopied for each student and enlarged for the class. This handout will be enriched later, once students have used tests, sensors, variables and operators.

Sequence II Step 2.2 >>