INTRODUCTION TO PROGRAMMING VIA ROBOT GAME (OFFLINE)

DIGITAL CONTENT CREATION > 3.4 PROGRAMMING

TARGET GROUP	AGE GROUP	PROFICIENCY LEVEL	FORMAT	COPYRIGHT	LANGUAGE
School drop outs, Students (primary school), Students (secondary school)	Children, Teenagers	Level 2	Activity sheet	Creative Commons (BY-SA)	English, French

This is a workshop introducing participants to programming. It is based on a game in which participants will play the role of robots executing particular instructions. The main purpose is to understand that a computer only does exactly what it is told to do.

General Objective	Knowledge acquisition
Preparation time for facilitator	less than 1 hour
Competence area	3 - Digital content creation
Time needed to complete activity (for learner)	0 - 1 hour
Name of author	Rosaline Faliph
Support material needed for training	A landscape with represented obstacles: it can be printed on a large sheet, made with a very large sheet, or even drawn on the ground with chalk - Small objects to put on the landscape: pebbles, flowers, leaves, objects (real or printed and plasticized) - Some examples of programs to print and plasticize (see files below) - Tablets (optional)

Resource originally created in	French

WORKSHOP DIRECTIONS

1 Introduction

The objective here is to develop the audience's knowledge of **programming** in a context of an outdoor activity with little or no internet connection. It is designed for children or parent-child groups who do not have experience with programming. The idea is to combine offline activities with online exploration later. This activity is largely inspired by a project **Marie Duflot-Kremer**, professor at the University of Lorraine in France. Since 2011, she has been working on introducing young or inexperienced audiences to concepts in IT without the use of computers.

You can find the original workshop here (in French): https://members.loria.fr/MDuflot/files/med/robot.html

Note to facilitator: An area representing a landscape with obstacles will be needed. This can be printed on a large piece of paper made on a large sheet (i.e. fabric), or even drawn on the floor with chalk (design suggestion below)
Also required are small objects to place on the landscape: stones, flowers, leaves, etc. (actual or printed or plastic) Some example of programs to prints and laminate (see information below)

2 Step 1: Offline workshop – the idiot robot

This activity will have participants making and executing computer programmes without any digital material. For this, the facilitator places a sheet on the ground (see the images below). Participants should be divided into pairs: one plays the computer memory (they read the program), and the other plays the computer's central processing unit (they execute the program). Alternatively, you can also say that one of the participants will be the robot-wanderer (they identify wild flowers for example, or they pick up rubbish – it's up to you to invent whatever stories you like!) and the other will be the computer that gives instructions so they move around the mountain landscape. The instructions have already been written, but be careful! Some contain errors. It is up to participants to find the right ones and correct the ones that mean danger for our little robot!

1. Introducing the landscape

Here we are in the mountains: there’s a river, a volcano, a very dense forest, a lake, bridges… The robot must move around the landscape but stay at certain places. The forest, volcano, lake and river are therefore off limits.

2. Learning the language

To start, all participants are **robots**. Everyone stands on the sheet/drawing – each person in their assigned spot – and follows the instructions given by the facilitator, with a choice between, right, left, straight on or backwards. As soon as a player moves into a landscape feature (river, lake, forest or volcano) or leaves the area, they exit the game and wait to the side. Once everyone is out, you can start again. Things to be specified:

- *One step means to move by point, not two*
- *No moving diagonally*
- *All players most always face in the same direction. They can move sideways but cannot turn their bodies.*
- *Players are robots, so they must execute instructions without changing them, even if they mean walking into obstacles.*
- *Review the meanings of left and right for those who may make a mistake. For children who don’t know left from right, you can use coloured ribbons for example: one step to the pink side, one side to the green side as proposed by Marie Duflot-Kremer*

They will now have used **algorithms**, like a computer.

An algorithm is a **series of instructions** which solves one or multiple problems.

3. Executing programs in pairs

Set the departure point as shown on the map (at the bottom, 7th point from the left). All programs will start there. For each program we will divide the task between one participant who places the robot and another who plays the computer.

Execute programs 1 and 4. *Note*: the robot/computer only knows how to execute tasks attributed to them. It is not a question of making one extra step or adding an arrow along the way. The computer does not do this! We could even blindfold the ‘robot’ is necessary.

Programs 2 and 4 each have a ‘bug’. This will show that if the program is wrong, the computer will execute it calmly, going ahead with its task indiscriminately. It will have done nothing incorrect, as it was the programmer’s mistake. In each of the two programs, adding one arrow (to the right in program 2 and up in 4) can correct their errors. This shows that while a tiny error can create big problems (drowned robot), it can also be easily fixed. Here you can mention that every programmer makes mistakes in their programs (which are in general very long and complex) but that a good programmer will test/verify their work to find a correct bugs.

4. Changing the Instruction Set

Programs 1 to 4 move the robot around the landscape, but our robot can do more, such as collect specimens: stones, leaves, garbage, etc. Such variants can be implemented according to the available time and age of participants:

- place objects on the area as shown on the map and add the instruction ‘**collect**‘ which will mean the robot can pick up the objects situated on their fixed points,
- group consecutive identical arrows, for example, write ← x 4 instead of ← ← ← ←. This helps to avoid errors in counting arrows and makes the programs more compact (cf. program 5),
- replace the instructions ‘side step’ with instructions to turn 90º to the left or right (cf. program 6).

5. Writing programs

Once participants have understood the programming language, they will need to **write their own** (use a black or whiteboard if possible – it’s easier to correct errors that way that on paper). **List of example programs:**

- Collect any three objects,
- Return to the departure point carrying two objects,
- Collect three stones and go to the forest,
- Collect one of each type of object,
- Finish on a bridge with a flower, a stone and a leaf,
- Collect (in whichever order) two flowers and two leaves,
- Collect in this order, a leaf then a flower then a leaf then a flower (and nothing else).

By doing this you will notice that there are several programs possible for doing the same things. If you have time you can **compare the programs** that execute the same task in terms of efficiency (in IT we tend to think about complexity).

Comparing the invented programs: A program can do the same job as another and be **shorter** to write, or require fewer steps or cycles. Sometimes the difference is clear, for example, a program that gets right to the point is **more efficient** than another that moves the robot around the lake, then the forest and finally the volcano before doing what we want it to. Other times it is more difficult to compare. Between two robots, perhaps one moves quickly but takes a long time to bend down to collect an object. We might prefer having it move to the lake to collect three objects at once rather than having it get to the destination quicker but then needing to bend down three times separately. In general, between two working programs, we will naturally prefer whichever is more efficient. Efficiency can be measured in terms of the time/number of steps to execute, of energy consumed, of the amount of memory used, etc.

3 Step 2: LightBot

Propose that participants apply their new knowledge of programming to an exploration of the application: LightBot. This **doesn't require an internet connection**. The application can be explored offline. There are often several solutions to clear each level. It can be interesting to compare the solutions found by each participant. You may want to divide the group into pairs for this step too. The programming notions you will encounter on LightBot are:

- **Control sequential**: orders (or instructions) executed by the program consecutively
- **Procedures**: blocks of preprogrammed code which can be inserted to save time and efficiency
- **Loops**: a code structure that repeats an instruction indefinitely (often as long as one or several conditions are respected)
- **Debugging**: executing and re-executing a program and testing solutions to correct errors in a computer program.

4 Variant

You can also use the 'Coding out Loud' activity as an introduction but it is less adapted to an outdoor group activity.

5 Documents to print/use

[The landscape](#)

[The programs](#)