INTRODUCTION TO BLOCK PROGRAMMING ON SCRATCH

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

Through Scratch, learn to block program in creative digital workshops.

General Objective	Skillset building
Preparation time for facilitator	less than 1 hour
Competence area	3 - Digital content creation
Time needed to complete activity (for learner)	0 - 1 hour
Support material needed for training	Computers with Scratch and internet connection, Beginner and intermediate cards (Ghostbuster, Minus, One-Armed Bandit) "Explain programming " and " Scratch presentation " explanatory notes, video projector recommended
Resource originally created in	French

WORKSHOP DIRECTIONS

1 Introduction to Scratch

***Specific elements related to a Scratch-based workshop**:* Scratch is a tool created to teach children how to code without first needing to learn a programming language. Users can program putting colour coded **blocks** together. Each block represents an instruction: move a sprite, draw a colour, change the value of a variable, etc. This is why we call it block programming. It is available in many languages and in two versions:

- Online. The advantage here is that users can keep their projects on their accounts (they need only an email address) but this version relies on an internet connection.
- Offline. This version needs to be installed in advance (for this workshop for example) but does not require an internet connection afterwards once installed.

It is up to you to choose which version to use depending on the group and what you have available.

Its **interface** is divided into several areas and has a number of functionalities. See the workshop ' Introduction to the Scratch Interface' first before continuing with this one. Scratch allows users to go quite far with their programming. It is possible to invent a wide variety of activities, games and animations using the tool. You could imagine an introductory workshop which would introduce some basic principles of programming through the creation of a project or the guided making of a game or animation. If you can divide the workshop into several sessions, you could work on a series of creations each taking on more complexity. Or, for a group already having some experience with whom you can work with for several sessions: a minimalist video game with one session for the designing of the game on paper, then the making of it in Scratch, test and correction and finally the presentation to the others.

***Facilitation tips**:* If you are using the offline version, create Scratch accounts to assign to participants! It often happens that group members close the application accidentally. Having accounts made ensures their progress can be saved and recovered. You can create several accounts using one email address.

2 Preparation

Familiarise yourself with Scratch by following some activities in advance. This is the best way to

understand how it works, the issues a beginner might run into as well as tips to be aware of. It will equip you to help learners during the workshop. Define prior the objective of the session: will it be on block programming for the first time, or will you have a group that already has some experience? According to the context, **choose** one or several of the 'Scratch
Tutorial' activities.

For more experienced groups, you can also prepare some extra project ideas or choose a **theme** for the session designed to give them ideas for games, animations, stories, or whatever else they may come up with. **Before the session**, check the material, install Scratch (if you choose to use the offline version) and run it. If you will be several facilitators, feel free to discuss and assign roles.

Optimise your time to help the participants as best you can. Think about spreading out the work stations so that you can move easily between groups. You may be giving a workshop in a space with its own equipment and a rigid layout. In this case, you will need to adapt to the number of participants.
We recommend you have no more than two participants working together per work station.

***Facilitation tip**:* If possible create two areas. One without computers for the beginning and end of the session for discussion, and a practical area with computers.

3 Facilitating the workshop

During the workshop, you should adopt the role of an **assistant** : you are there to help participants find answers, not to provide them yourself. When someone is having trouble with step or gets an unexpected result, take the initiative to ask questions to inspire the participant to come to their own response to the problem. For example : 'What kind of result do you want?' (Identify the context) 'What have you tried and how?' (Identify the method, the data) 'What is the problem?' (Identify the problem) 'What might have caused the problem?' (Identify the origin) 'Have you tried…?' (Guide to a solution using hints)

Encourage **assistance between participants** so the more experienced can help the beginners, for example.

4 Welcoming participants

Explain to the group that you are there to help them find the answers, to not give the answers yourself.

Communicate the objective clearly, underlining that there are no right or wrong answers, programs or designs. Everyone is here to learn and have fun. To reach the objective in the easiest way possible, feel free to define a few rules together, for example:

- Computers can only be used for Scratch (not for going on YouTube for example)
- While a facilitator is speaking to the group, no one may use the computers
- Groups are allowed and encouraged to help each other during the session
- They can move around freely as long as they are not disturbing the others, etc.

You can ask each participant to introduce themselves in turn, for example by giving the following information: their first name, their favourite game on computer/tablet/console/mobile, if they have already done some programming, etc.

Facilitation tips :
Allow participants to take liberties. Don't just impose restrictive rules which would risk creating a constraining atmosphere and losing the trust of your group.

5 Starting the workshop

If your group has already used Scratch, move to the next step.
If not, **introduce** the tool using the projector, before having participants sit at the computers, in order to avoid distraction. Explain what it does and show the different parts of the interface. Show how to drag and drop the blocks join them together. Show how to start and stop the game/animation.
Avoid giving too much information at this stage. If further specific components will be necessary (sound, costumes, block creation, speech, etc.) give out the corresponding coding cards (see link below). These can be used to help in specific circumstances.

Extra tools *:* You can use the coding cards available here.

6 Main part of the workshop

Introduce the chosen **project** or **theme**. Give the **time** available for completion. Make groups, assign computers if necessary and leave participants begin the activity you have set up. Everyone should go at their own **pace**

.

Note : depending on the number of computers and participants, you will need to **group** the participants. Try to not put more than two people at one computer (see below). You can allow the duos to form themselves, or if you are concerned about lapses in concentration, assign participants partners. In the latter case, mixing the more experienced with the lesser is a good idea! Move from group to group, making yourself available to those in difficulty.

Remember to assist them, not the resolve problems they might be having with Scratch (see this related workshop). However, it will of course be up to you to solve problems related to materials, group management, Following the chosen activity, participants will need to deal with various **concepts** of programming: loops, variables, conditions, etc. At the point where a group first arrives at this step, ask everyone to pause for a moment while you explain the concept simply and how to apply it in Scratch. Then allow work to continue. That way you will avoid needing to repeat the same explanation several times by – you will be applying suitable pacing to the workshop.

Facilitation tips : Don't put more than two participants per computer. A greater number will make concentration and understanding difficult.

7 Ending the workshop

When a pair or individual has finished their project, they can either help the others who might be stuck or look for ways to modify the design (change the colours, shapes, make a variant, etc.) If you have time, and some of the groups have been able to complete their project, keep a moment at **demonstration** time so everyone can present, using the projector, their design to the others and explain how the coding works. How can participants leave with their creations ?

- If you have been using the online version, participants will be able to access their games on the platform through an account you can create for them.
- If the offline version, it is possible to save the work in a file. Go to file then 'download'. You should receive a .sb2 file which you can put on a USB key or send via mail to then open on Scratch by going to file, then 'import'.

If possible, leave ten minutes at the end of the workshop to do a quick survey amongst participants to

allow them to give their impressions and what they thought of the workshop, what did they like, etc.

Facilitation tips *:* Feel free to ask each participant in turn if everything went well, if they had questions, etc. If someone came into difficulty, don't give the solution immediately, but encourage them to reflect to understand the logic and solve the issues in their code themselves. Never take the mouse or control of the control of the computer yourself. Guide the group members and leave them do it themselves. If some finish quickly, you can, depending on the time, either give them a new assignment or encourage them to come with something new themselves or improve the project they have already completed.

8 After the workshop

Discussion amongst facilitators **Regarding participants**:

- Were you able to keep their attention?
- Were you able to make yourself understood?
- Did you feel comfortable?

Amongst facilitators:

- Were each of you able to fulfil your role properly?
- Did you communicate enough amongst yourselves?
- What worked well?
- What didn't work well?
- What can you we do to improve?

To finish, put away the material.

What next ? It is important to think of the next sessions. You will surely ask yourself what to do next, particularly if you will be working with the same group. You can guide them to progressing gradually through the tutorials and suggest other games but also different levels. You can check the curriculum created on the Libraries without borders website and come up with your own progression. Why not create a game together involving several stages or levels.

9 Annex

Listed below are some basic concepts that might come in useful during the workshop depending on the

chosen activities. Feel free to do some quick research in advance, or possibly do a quick overview with your participants to respond to their questions collectively.

- Computers don't think. They are idiots (You can find out more from this workshop). They just do what we tell them. Telling a computer what to do is called giving them **instructions** that should be simple, precise, clear and complete. Scratch allows us to do this simply using blocks.
- Several instructions given together form a **program** which is designed to obtain a result (animate a game, make a website available, etc.)
- Computers don't speak human languages. Scratch 'translates' for us (we know how to read blocks in the language of our choice – they are then translated so the computer can read them). The computer 'speaks' binary (this subject should be dealt with in specific workshops).
- A **condition** is a special instruction that allows a result to be changed (the action taken by the computer) depending on a test, for example '*if* I have no lives left, *then* I lose and the game ends *else* [i.e. otherwise] the game continues'.
- A **loop** is a special instruction that means a series of instructions will repeat several times or infinitely. The loops will continue or stop depending on one or more conditions (see above).
- A **variable** is like a box in which the computer stores information. For example, the number of points won during the game.