

TOOLKIT COD1NC



**CODING FOR
INCLUSION**

www.codinc.fun



Co-funded by the
Erasmus+ Programme
of the European Union

INTRODUCTION	4
10 HOUR TRAINING FOR STUDENTS	5
OUTLINE PER ACTIVITY	5
SESSION OVERVIEW	6
SESSION 1: INTRODUCTION TO CODING	8
INTRODUCTION INTO CODING	8
ENTRY LEVEL CODING APPLICATIONS	10
COMPUTER SCIENCE UNPLUGGED	11
SESSION 2: INTRODUCTION TO SCRATCH AND ANIMATION SKILLS	15
SCRATCH: BASIC INTRODUCTION INTO VISUAL CODING	15
PEDAGOGICAL ANIMATION SKILLS I:	17
SESSION 3: CREATIVE USE OF SCRATCH	21
SCRATCH: CRAFTING A FIRST PROJECT	21
TANGIBLE COMPUTING WITH MAKEKEY MAKEKEY	22
SESSION 4: CREATING YOUR OWN SCRATCH PROJECT AND HOW TO TEACH IT TO OTHERS	25
SCRATCH: CREATING A NOVEL PROJECT	25
PEDAGOGICAL ANIMATION SKILLS II	26
SESSION 5: PRESENTATION AND REFLECTION	30
SCRATCH: FINISHING A NOVEL PROJECT & PRESENTATION	30
IMPLEMENTATION: EVALUATION & REFLECTION	30
EXTRA: INTRODUCTION INTO TEXT-BASED CODING	31
10 HOUR WORKSHOP PROGRAMME WITH PUPILS	33
SESSION OVERVIEW	33
SESSION 1: INTRODUCTION TO CODING	35
CODE ETIQUETTE : RULES OF CONDUCT	35
INTRODUCTION INTO CODING	36
ENTRY LEVEL CODING APPLICATION	38
SESSION 2: INTRODUCTION TO SCRATCH	40
COMPUTER SCIENCE UNPLUGGED	40
SCRATCH: BASIC INTRODUCTION INTO VISUAL CODING	43
SESSION 3: CREATIVE USE OF SCRATCH	45
SCRATCH: CRAFTING A FIRST PROJECT	45
TANGIBLE COMPUTING WITH MAKEKEY MAKEKEY	47
SESSION 4: DUO PROJECT 1	51
SESSION 5: DUO PROJECT 2 AND EVALUATION	53
EVALUATION: ACTIVE REVIEWING	53
THERESOLDS	55
WORKSHEETS	57
LINKS	57
MATERIALS	57
APPENDIX I: REFERENCES	58
APPENDIX II: WORKSHEETS	58
APPENDIX III : ENERGISER ACTIVITIES	58

APPENDIX IV: ACTIVE REVIEWING ACTIVITIES

60

INTRODUCTION

This training toolkit realised by the project CODINC is based on a adaption of the training materials developed by Capital Digital powered by Maks vzw.

During the CODINC project, we will organise a 10-hour training for students of a secondary school. The aim of the training is that the students are trained to give a training to pupils in a primary school. Every student should work with 5 pupils during 5 sessions from 2 hour each. We will work with the pupils of 2 primary schools and provide a training for the teachers from the primary schools. The teachers of the secondary school will be trained together with their students.

The CODINC project will adapt and scale up the Capital Digital methodology (and training Toolkit) into a different educational context (formal Education) and pilot it in 5 European countries, namely Belgium, Cyprus, Germany, Italy and Spain. CODINC specific objectives are the following:

- Increase and improve teachers' capacity to foster the STEAM education of disadvantaged youth in an inclusive educational approach based on peer learning
- Empower disadvantaged young people in the acquisition and development of IT and collaborative competences as well as problem solving, self-confidence and creativity through a peer-learning training programme on Coding
- Foster the development of a European "Coding for Inclusion" learning community among different actors and across different sectors (formal and non-formal education and training) able to sustain project results and amplify their impact.

Capital Digital is a programme that engages a peer-to-peer learning method by training young people from disadvantaged backgrounds to become animators to their younger peers. In practice Capital Digital successfully trained 15-18-year-old students from disadvantaged backgrounds to teach coding and programming to their younger 10-12-year-old peers. The young animators learned to engage pupils in STEAM and coding activities in a playful way. This programme became the first work experience for the animators which enhanced their confidence when deciding on a fitting educational and career path. The programme allowed them to connect with each other in a constructive way and to enjoy the role of educating their younger peers. Moreover, the project effectively supported young people by developing their critical thinking, creativity, digital and collaborative skills, and science capital. Most importantly, the Capital Digital peer-learning pedagogical method has a strong inclusive potential and fosters the STEAM education and the inclusion of disadvantaged students both inside and outside the classroom.

In this CODINC toolkit, you find a description of the training.

10 HOUR TRAINING FOR STUDENTS

The project starts with a 10-hour training program for the students in last years of secondary education (16 till 18 years old). During this training, the students will participate in **five sessions of two hours**, in which they will learn about the fundamentals of (visual) coding, computational thinking and pedagogical animation with pupils.

The aim is that after the training they should be capable to train pupils from 8 till 12 years old in coding.

Afterward the training, they are ready to take on the challenge and pass on their knowledge to the selected groups of primary school pupils.

Each session of two hours encompasses several activities. While these activities can be interchangeable to a certain degree, we suggest respecting the general order of the digital coding activities. The introductory and unplugged activities provide great scaffolding, which will make a smoother transition to pick up the Scratch visual coding language later.

There is an extra activity provided, offering a first look into the world of real text-based coding languages. This activity can be implemented if and when the students have a strong grasp of the principles of visual coding and there is time left in the program.

OUTLINE PER ACTIVITY

- Intro:
 - What is the activity?
 - Why?
 - Timeframe
 - Group Layout
- Goal of the activity
- Carrying out the activity
- Tips & Tricks
- Materials needed

SESSION OVERVIEW

10h Training	Activity	Objectives
Session 1: Introduction to coding hour 1 and 2	Introduction Into Coding	Providing insights into what coding is and why it's important First acquaintance with algorithms and sequences
	Entry Level Coding Applications: * Lightbot * Hour of Code	Experimentation and exploration with the 'Lightbot' and 'Hour of Code' applications. Getting familiar with algorithms, sequences, loops and conditionals.
	Computer Science Unplugged: * Sandwich Robot * Drawing Pixels	Offline experience with algorithms and sequences through the 'Sandwich Robot' exercises First acquaintance with the visual representation and digital construction of pixels on screens
Session 2: Introduction to Scratch and Animation Skills hour 3 and 4	Scratch: basic introduction into visual coding	First acquaintance with the Scratch visual coding language and its possibilities Getting familiar with algorithms, sequences, loops and conditionals in the Scratch coding language
	Pedagogical Animation Skills I: * Setting the rules of conduct * Group discussion on non-formal teaching	Participatory strategy to draft a document for the rules of conduct during the coding activities Strategies for implementing an appropriate atmosphere for non-formal learning with pupils. Strategies for increasing the autonomy of pupils in their learning process
Session 3: Creative use of Scratch hour 5 and 6	Scratch: crafting a first project	Creating a first game in Scratch with the help of example projects Deeper insights into the possibilities of visual coding Insight and practice on the computational thinking concept goals
	Tangible computing with Makey Makey	Insight into tangible computing and the relation to pure digital computing Designing an alternative controller for a digital game
Session 4: Creating your own Scratch project and how to teach it to others hour 7 and 8	Scratch: creating a novel project	Ideation and creation of a new, personal project in Scratch Applying computational thinking concepts to a novel design Mastering the visual coding blocks in the Scratch environment
	Pedagogical Animation Skills II: * Motivation management and energizers * Practical organisation & preparation	Insights and strategies on how to work and play with a large group of pupils. Insights and strategies on individual and group motivation
Session 5: Presentation and reflection hour 9 and 10	Scratch: finishing a novel project & presentation	Mastering the visual coding blocks in the Scratch environment Presentation of a personal project to the group Increasing confidence through successful experiences
	Pedagogical Animation Skills III: * Evaluation and reflection	Evaluation of the 10 hour program Reflection on the learnt skills and concepts Preparation of the implementation in primary schools

Extra	Introduction into text-based coding	Transition from visual coding environments to a real-world text-based coding language
-------	-------------------------------------	---

SESSION 1: INTRODUCTION TO CODING

Introduction into Coding

Summary: In this group discussion and activity, you as a teacher will inquire into the knowledge about coding that is already present in the students and supply additional insights. You will also explore together what makes a (good) algorithm.

Timeframe: 30 minutes

Learning outcomes:

- 1.1. Algorithms
- 1.2 Sequences
- 2.4 Abstraction
- 3.1 Working together
- 3.3 Describing thought processes
- 3.4 Learning from vicarious experiences
- 5.1 Problem identification
- 5.3 Implementation
- 5.4 Evaluation & reflection
- 6.1 Group animation
- 6.2 Non-formal teaching

Implementation:

Explain briefly what the purpose of this training program is (learning to code and picking up pedagogical skills, so the students can lead similar activities with pupils) and what they will learn across the five sessions.

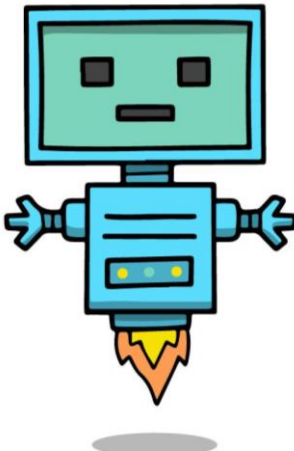
Next, you will continue the activity with a group discussion. You can use the following questions as a guideline, but feel free to add your own.

- What is code? Who knows what code does and who creates it?
- How are digital games, computer programs and smartphone apps made?
- Have you ever created a game, app or program yourself?
- Has anyone ever used the Scratch coding language before?
- What is an algorithm?

What is an algorithm?

An algorithm is a list of instructions you give to a computer or digital device, to solve a problem or get something done.

After inquiring into their knowledge into coding and algorithms, hand out pen and worksheet 1, along with the following assignment. They can complete their assignment together with another student, or alone.



Sally the Robot loves drinking tea, but she is not programmed by her creator to brew it herself.

Can you write down an algorithm for Sally on how to make and drink a nice cup of tea?

(Pssst: Sally needs very specific instructions, so she doesn't make any mistakes. Make sure she doesn't put too much water in her cup.)

Give your students a few minutes to write down their algorithms. Afterwards, you can have a short group discussion on what makes a good algorithm and where they might have gone wrong. Common mistakes when writing a first algorithm are:

- Not being specific enough, i.e. when you tell Sally to pour water into a cup, she will keep pouring until you tell her to stop.
- Putting steps in the wrong order, i.e. turning the cooker on before filling it with water.
- Forgetting about steps we humans think are obvious, but a robot doesn't know about.

Tips and tricks:

Are you looking for a fun way to brighten up the discussion or give a starting point to your students? The BBC made a great series of videos for pupils on computing and the concepts behind it. Take a look at these two together with your students for inspiration:

Algorithms: <http://bit.ly/bbc-algorithms>

Coding: <http://bit.ly/bbc-coding>

Materials:

For this activity you will need:

- Pens and paper
- A computer with a beamer and a screen to project on
- Worksheet 1 for every student (or per two)

Entry Level Coding Applications

Summary: In this activity, students will individually explore two different applications that teach the basic concepts of coding: 'LightBot' and 'Hour of Code'.



image by lightbot.com

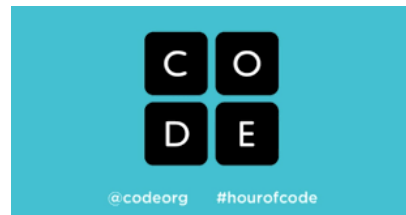


image by code.org

Timeframe: 45 minutes

Learning outcomes:

- 1.1 Algorithms
- 1.2 Sequences
- 1.3 Repetition and loops
- 1.4 Events and selection
- 2.1 Incremental & iterative work
- 2.2 Testing and debugging
- 2.4 Abstraction
- 5.1 Problem identification
- 5.2 Implementation
- 5.3 Evaluation & reflection

Implementation:

Prepare computers or tablets with one of the following applications: 'LightBot' and 'Hour of Code' (themed with Star Wars, MineCraft or Frozen).

Allow the students to test each application for about 20 minutes, to get familiar with the puzzles in them and how to solve them. Each of the apps is available in a wide range of European languages and provide comprehensive in-game instructions on how to use them. At the end of the activity, take 5 minutes to discuss what they learnt and how well it went.

Tips & Tricks:

- Use a rotation system between the two apps, so students can try both.
- LightBot has a more visual approach to coding, with symbols for movement and actions, while the Hour of Code apps use coding blocks with text on them, similar to the Scratch coding language.

Materials:

- Computer or tablet for every student
- LightBot Application:

- Install on tablet from [Google Play](#) (Android) or the [App Store](#) (iOS)
- Or use the online webapp on a computer: <http://lightbot.com/flash.html> (Flash)
- Hour of Code Application:
 - Star Wars themed: surf to <https://code.org/starwars>
 - Minecraft themed: surf to <https://code.org/minecraft>
 - Frozen themed: surf to <https://code.org/frozen>

Computer Science Unplugged

Summary: Unplugged activities help students understand the basic principles of computer science and computational thinking, without even touching a digital device. In the '*Sandwich Robot*' activity, students practice their abilities to construct an algorithm with a well thought through sequence. The '*Drawing Pixels*' activity is designed to comprehend how images are displayed on screens through pixels.

Separate your group into two segments and after 20 minutes, let the students switch between the two activities.

Timeframe: 45 minutes

Learning Outcomes:

- 1.1 Algorithms
- 1.2 Sequences
- 1.4 Events and selection
- 2.4 Abstraction
- 3.1 Working together
- 3.2 Negotiation practices
- 3.3 Describing thought processes
- 3.4 Learning from vicarious experiences
- 5.1 Problem identification
- 5.3 Implementation
- 5.4 Evaluation & reflection
- 5.5 Iteration

Implementation: Sandwich Robot

In this activity, you as a trainer are a robot and will be 'programmed' by the students. The goal of the programmers is to have the robot successfully making a delicious sandwich with butter and cheese.

Introduction:

Explain to the students that you are a robot and want to make a sandwich with butter and cheese. However, you are not programmed to do that. The students are the programmers or engineers and have to create an algorithm for you to follow.

Give them a few minutes to discuss and write down what they think is the correct algorithm to have a robot make a sandwich. After that, they get to tell you the code and you, the robot, execute their commands. Warning: a robot takes everything very literally. If the students to command you to pick up bread, pick up the whole loaf. When they tell you to put down the butter, put it on the floor. Make a lot of 'mistakes' so they will understand how precise programming really is.

Bonus: if you want to make your performance as a robot even better, make classic robot noises and repeat every command in a robot voice while you execute it (wrongly).

If you want, you can look at an example video here for inspiration: <http://bit.ly/CSUrobot>

This activity was translated from Codekinderen.nl

Implementation: Drawing Pixels

Information on Pixels:

Every day, we look at screens and see text, images, video... But how does a computer or a smartphone know how to display a tabby cat? Or a picture of a house?

All digital screens use the same technology to display visual information: with **pixels**. Those are tiny squares on the screen that can take any colour, from white to black and all the shades in between. All the pixels together make up the picture. Traditionally, pixel colours on a display are made by mixing amounts of red, green and blue (RGB). This combination is represented by a code numbers and letters, so the computer knows what to show. For example, the colour "hot pink" has the code `#ff69b4`.

Now, screens have a very high pixel density, so you can't see the little individual squares anymore. But if you look at old video games or so called 'pixel art', it becomes quite obvious to the eye. Even if you take a digital photograph and zoom in as much as you can, you can see all the separate pixels.



Space Invaders (Taito, 1978)



Super Mario Bros (Nintendo, 1985)

Class discussion:

Start the activity with a short class discussion. You can use the following questions, but can of course add your own.

- How are images displayed on a computer display?
- Can you send an image from one computer to another? How do you think that happens?
- How big is a pixel?
- What kinds of games do you know that have 'pixel art'?

Make your own pixel drawings:

Hand out pencils, erasers and printouts of Worksheet 2 for every student.

The students follow the instructions on the worksheet to draw their own pixel drawings, inside the provided grids. Every grid comes with number instructions on the side, which make up the code for the drawing. Be careful: the first number of every row is always white.

Drawing						Code			
						2	3		
						5	1		
						2	4		
						1	1	3	1
						1	1	3	1
						2	4		

On the right, you see the number code for each row. The first number is always white, the second number black, then white again, black again ... Rinse and repeat.

For example, on the first row, you have to leave 2 squares white and colour 3 squares black.

On the fourth row, you have more numbers.

The first number is 1 white, then 1 black, three white and lastly 1 white.

This activity was translated from Codekinderen.nl

Tips and Tricks:

- Do you and your students want to learn more or try some extra activities on how images are displayed on digital devices? Surf to <https://classic.csunplugged.org/image-representation> for more information.
- You can find a lot more cool and fun Computer Science Unplugged activities for free, in a variety of languages at <https://csunplugged.org>

Materials:

For the *Sandwich Robot* activity, you will need:

- A loaf of bread
- A pack of butter
- Several slices of cheese
- A plate
- A knife
- Cleaning supplies, because things will get messy

For the *Drawing Pixels* activity, you will need:

- Worksheet 2
- Pencils and erasers

SESSION 2: INTRODUCTION TO SCRATCH AND ANIMATION SKILLS

Scratch: Basic introduction into visual Coding



Summary: Scratch is a visual coding language and environment, to teach pupils and students to code in a fun, easy way. Your students can use Scratch to code their own interactive stories, animations, and games. In the process, they learn to think creatively, reason systematically, and work collaboratively — essential skills for everyone in today's society.

The Scratch coding language was created by Mitch Resnick and the MIT LifeLong Kindergarten Lab. Take a look at his TED talk on the importance of learning to code here: https://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code.

In the following Scratch activities, your students will learn the basics of coding with blocks: movement, colour changes, animations and so on; as well as create a first fun project and design their own digital game.

Timeframe: three times 60 minutes

Learning outcomes:

- | | |
|--|--|
| 1.1 Algorithms | 2.6 Information: collection & management |
| 1.2 Sequences | 3.1 Working together |
| 1.3 Repetition and loops | 3.2 Negotiation practices |
| 1.4 Events and selection | 3.3 Describing thought processes |
| 1.5 Parallelism | 3.4 Learning from vicarious experiences |
| 1.6 Conditionals and logical operators | 4.1 Combination |
| 1.7 Mathematical operators | 4.2 Exploration |
| 1.8 Variables and data management | 4.3 Transformation |
| 1.9 Functions | 5.1 Problem identification |
| 2.1 Incremental & iterative work | 5.2 ideation and brainstorming |
| 2.2 Testing and debugging | 5.3 Implementation |
| 2.3 Reusing and Remixing | 5.4 Evaluation & reflection |
| 2.4 Abstraction | 5.5 Iteration |
| 2.5 Modularization | |

Implementation: Basic Introduction

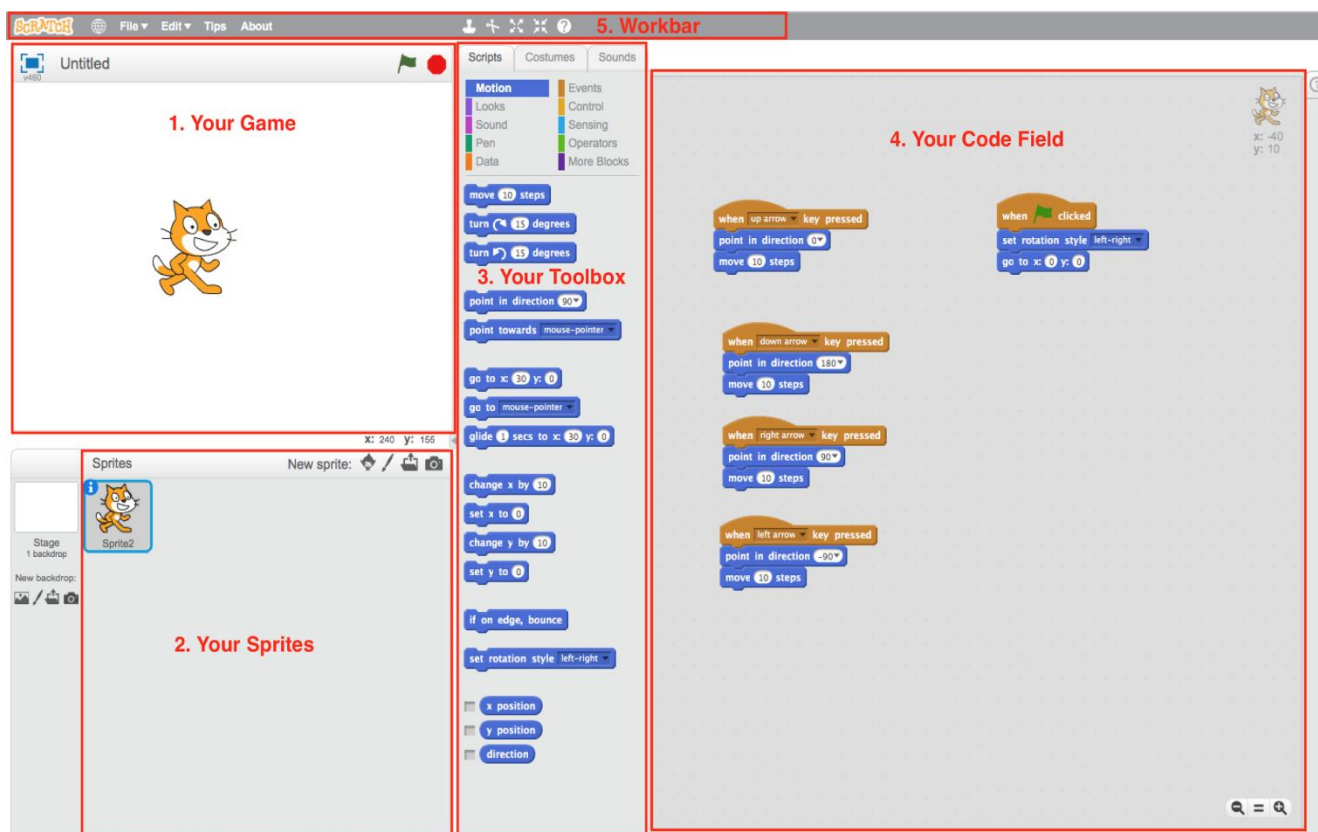
In this activity, your students will make a first acquaintance with the Scratch coding language and environment. In Scratch, you use building blocks to create a digital game, animation or app.

Getting started

You can choose to use the Scratch environment on computer (1) online at <https://scratch.mit.edu/> or (2) without internet by installing the offline editor (<https://scratch.mit.edu/download>). If you want your students to work online and save their work (recommended), have them create an account. Follow the instructions at <https://scratch.mit.edu/educators/faq> to help you create a teacher and individual student accounts.

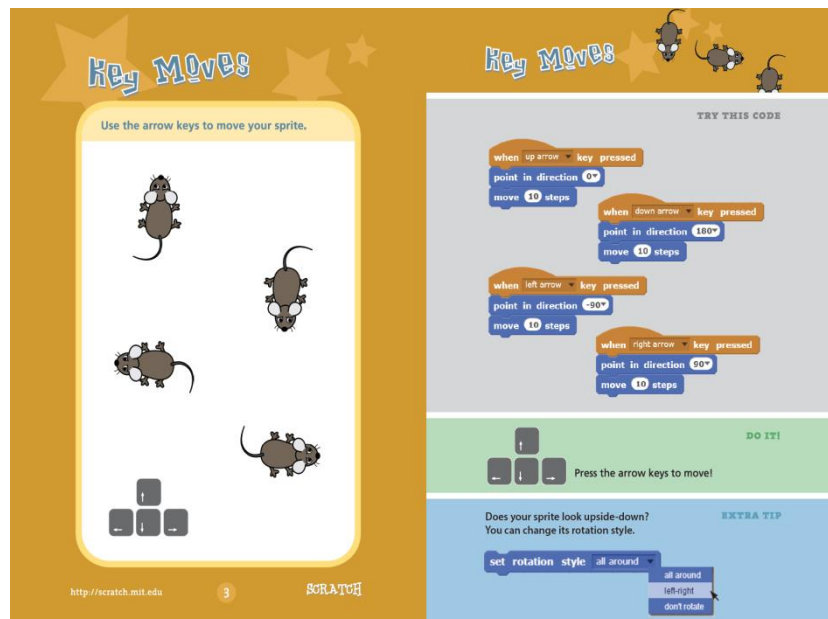
First view on the program

Of course, you first want to show your students what Scratch is before diving in. You can either do this by sharing your own screen on a beamer and making a few first coding moves. Or you can show an introductory YouTube video. There are plenty of them out there, but we suggest one made by a student called "Matt the Scratch Kid": <http://bit.ly/intro-scratch>.



Playing to discover

Now your students are ready to learn programming in the most fun way possible: by playing and experimenting. Hand out the sets of **starter cards**, which show a basic coding procedure each, such as moving with the keyboard arrows, using a mouse to move, changing a character's colour and so on. Challenge your students to try out all the cards in the set.



You can download the English set of starter cards for free at http://www.capitaldigital.org/nl/scratch_starter_cards/

Pedagogical Animation Skills I:

Summary: In these activities, the students will take a dive into pedagogical skills and how to become a group animator. These are all skills they will need and apply when they go teach coding activities to their younger peers. These are skills such as setting the rules of conduct together with the pupils, how to create a fun, non-formal teaching environment, how to manage motivation and so on.

Timeframe: For each of the three pedagogical parts 60 minutes

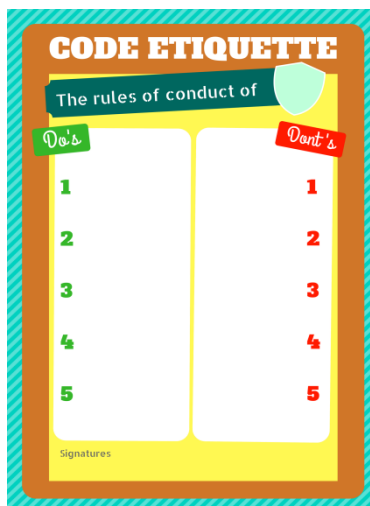
Learning outcomes:

- 3.1 Working together
- 3.2 Negotiation practices
- 3.3 Describing thought processes
- 3.4 Learning from vicarious experiences
- 5.1 Group animation
- 5.2 Non-formal teaching
- 5.3 Energizers and motivation management
- 5.4 Evaluation techniques
- 5.5 Rules of Conduct
- 5.6 Practical organisation skills & insights

Implementation: Setting the rules of conduct

While the code workshops aren't as strictly organised as a traditional school lesson would, there is a need for some rules of conduct. Especially when pupils are working with digital devices, the young students will need to have some agreements on behaviour and respect.

This activity is based on filling in a poster, called the Code Etiquette document. You can download this poster at <http://www.capitaldigital.org/nl/code-etiquette/>. Print it out on A3 size paper or larger.



The Code Etiquette is filled in through a group discussion, in which students will reflect on what rules are necessary, what pupils can and can't do, and what rules would be too strict. For example: while it is handy that pupils raise their hand before talking or asking a question, it is supposed to be a fun learning program. So it might be wise to loosen up on traditional school rules.

Start the activity by showing the poster. Next, you will hand post-its and writing material to all the students. Ask them to reflect on at least 5 things the pupils should do (positive rules), such as having respect, being careful with the material, using inside voices ... Next, they get to reflect on 5 things the pupils shouldn't do (negative rules), such as no running inside, no food or drinks near the computers, ...

Collect all the post-its and start grouping them together with the students. As a group, you will select the 5 most important Do's and Don'ts. Write them down on the poster and have everyone sign this.

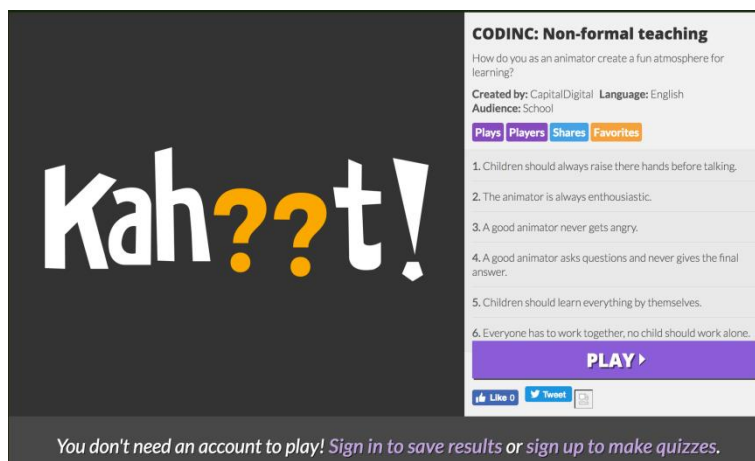
Using a collaborative approach to setting rules of conduct and having all the participants sign the poster, ensures that everyone is responsible for a good atmosphere. The students will later perform the same exercise with the pupils in their group.

Implementation: Group discussion on non-formal teaching

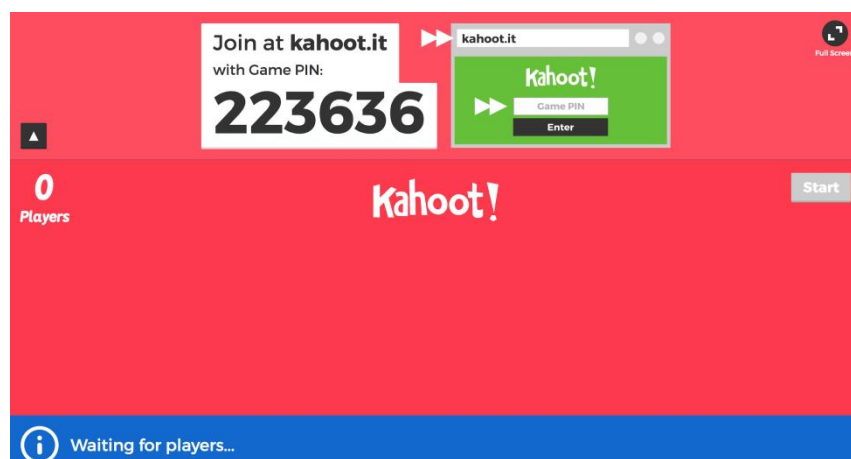
The second pedagogical activity deals with insights on non-formal teaching and creating a positive atmosphere for learning. The coding workshops are not supposed to be a very strict environment, in which no one can talk. The opposite is true: it should be a playful, social learning experience.

In this discussion, you will challenge the students to think about what makes a good learning environment and how they behave as an animator. For this goal, you can use the interactive Kahoot quiz: <http://bit.ly/cd-kahoot>. In this quiz, the students evaluate 10 statements to be either true or false. Of course, these are trick questions: there is no black and white when it comes to non-formal teaching. Everything depends on the situation, the group and the animators.

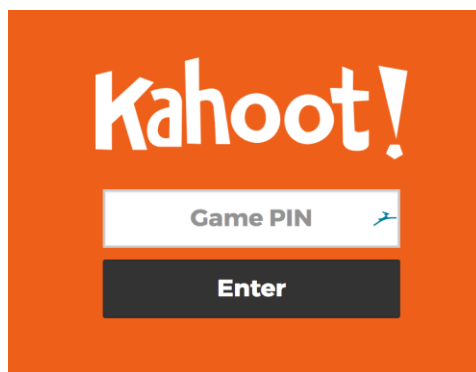
For this activity, you will have to set up your computer with a beamer, to show the quiz questions on the big screen. You as a teacher will surf to <http://bit.ly/cd-kahoot> and press play to start the quiz. Choose the classic 1 vs 1 mode.



The website will provide you with a pin-code for the quiz, that the students will need to participate.



The students each have a tablet, laptop or their own mobile device with internet. They surf to <https://kahoot.it/> and enter your pincode. After that, they can choose a nickname (this can be anonymous) and start answering question.



Every question has a 30 second time limit. After seeing the answer on the big screen, make sure you take enough time for the discussion on the answers. Why did they say true or false?

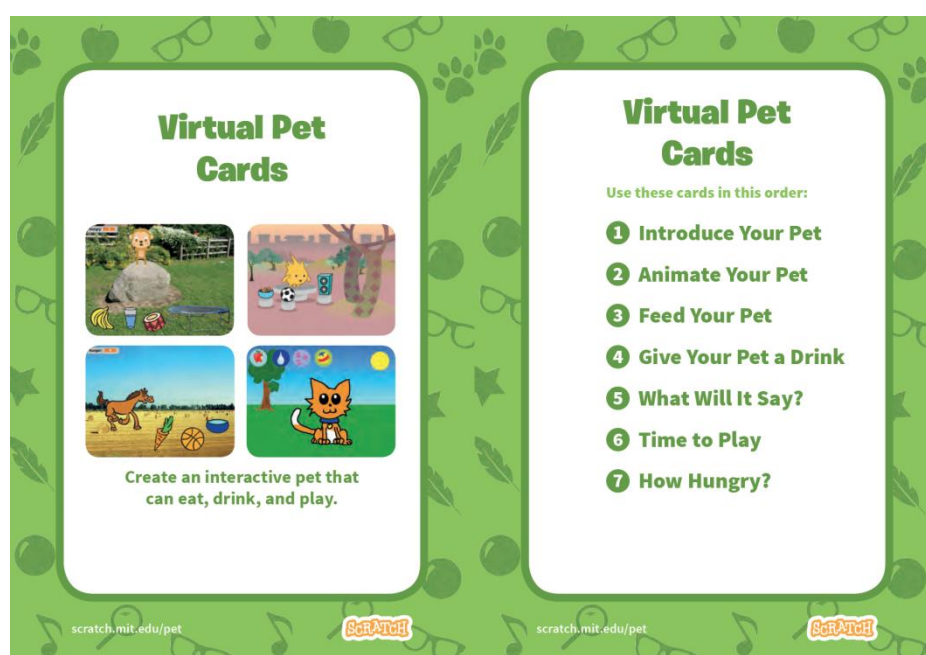
The questions in the quiz:

- | | |
|---|--------------|
| 1. Pupils should always raise their hands before talking. | True / False |
| 2. The animator is always enthusiastic. | True / False |
| 3. A good animator never gets angry. | True / False |
| 4. A good animator asks questions and never gives the final answer. | True / False |
| 5. Pupils should learn everything by themselves. | True / False |
| 6. Everyone has to work together, no child should work alone. | True / False |
| 7. Learning to code is serious, not playful. | True / False |
| 8. The ideal animator never touches the pupils' computer. | True / False |
| 9. The animator should always be a good example for pupils. | True / False |
| 10. Every activity needs to be prepared to the smallest detail. | True / False |

SESSION 3: CREATIVE USE OF SCRATCH

Scratch: Crafting a first project

The next time your students will be tackling Scratch, they already have a basic understanding of the language and what it's possibilities are. So now it is time to create a first project or game. Luckily, the Scratch website already provides a lot of cool example projects, that are quick to try out. Just like the basics, they come in fun, easy to follow project cards: <https://resources.scratch.mit.edu/www/cards/en/ScratchCardsAll.pdf> .



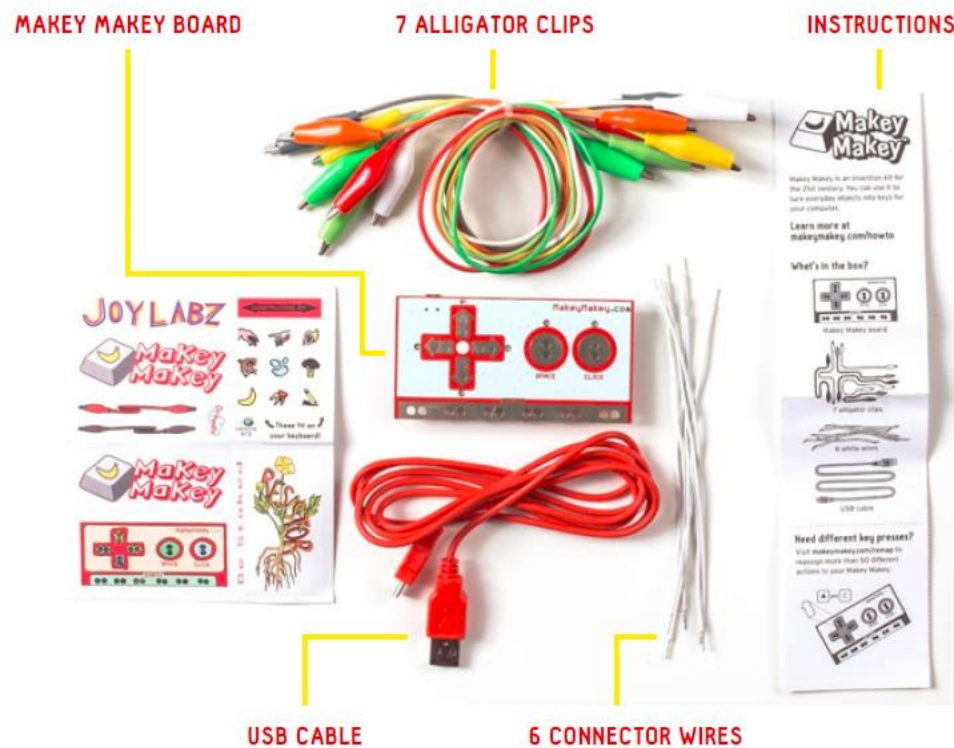
Download the cards, print them out around five times and put them in sorted stacks at the front of the classroom. Let every student pick a project that they would like to complete during this activity. From that point, it is just an easy ride following the instructions.

If a student finishes early, challenge them to either improve on the project they made (by adding a score, increasing the difficulty, changing the theme of the game ...) or to create a second, more difficult one.

At the end, allow for 10 minutes so that students can test each other's projects and have a quick discussion about what they learnt so far.

Tangible computing with Makey Makey

Summary: Makey Makey is an invention kit for the 21st century that ignites curiosity, challenges problem-solving ability, and fosters creativity. With Makey Makey everyday objects are transformed into touchpads empowering students to interact with computers as creative tools. The computer becomes an extension of their creativity, fostering imaginative play and discovery.



In this activity, you and your students will create innovative game controllers out of fruits, cutlery, tin foil and so on. Imagine playing the classic PacMan with a set of spoons or a piano made out of bananas and apples. This is the ultimate creative exercise, connecting the real, tangible world with digital coding.

Timeframe: 60 minutes

Learning outcomes:

- 1.1 Algorithms
- 1.2 Sequences
- 1.3 Repetition and loops
- 1.4 Events and selection
- 1.6 Conditionals and logical operators
- 1.7 Mathematical operators
- 2.1 Incremental & iterative work
- 2.2 Testing and debugging
- 2.3 Reusing and Remixing

- 3.2 Negotiation practices
- 3.3 Describing thought processes
- 3.4 Learning from vicarious experiences
- 4.1 Combination
- 4.2 Exploration
- 4.3 Transformation
- 5.1 Problem identification
- 5.2 ideation and brainstorming

2.4 Abstraction
 2.5 Modularization
 2.4 Information: collection & management
 3.1 Working together

5.3 Implementation
 5.4 Evaluation & reflection
 5.5 Iteration

Implementation:

Introduction:

You can introduce the MakeyMakey controller with a short video showing its possibilities, or with a small example performance. You can find a clear, well explained video on YouTube by following this link: <http://bit.ly/intro-makey>

The other way to introduce the MakeyMakey, is by connecting it to a computer through USB, running a Scratch algorithm that makes a sound whenever you press a button. Connect the ground cable to the Makey Makey (explained in Worksheet 4) and also a cable to the spacebar slot. Make sure the volume is turned up.

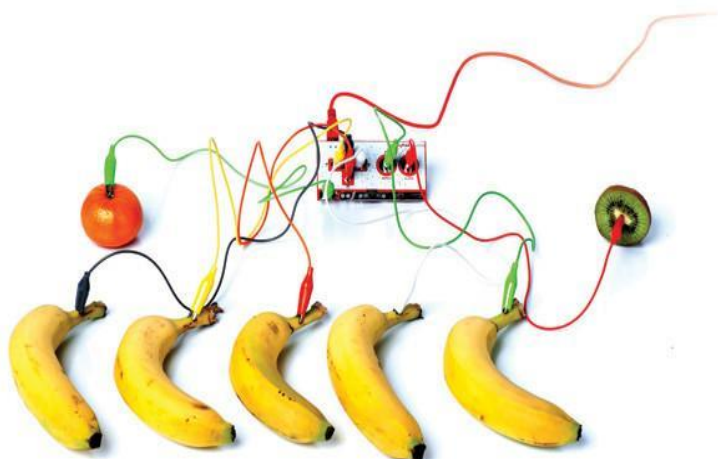
You, the teacher, hold the ground cable, while a student takes hold of the spacebar cable. All the students and you make a circle, holding hands except the last person (who is holding the cable). Have the two students -that are not holding hands- high five each other. A sound will ring on the computer. Have them try high fiving at different points in the circle.

Testing games with the Makey Makey

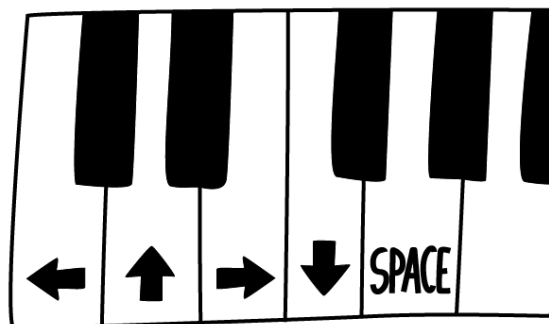
Set up your computers, each with a (different) app on it that is compatible with the Makey Makey. You can also use retro games, that you can find online, such as PacMan or Super Mario, or most Scratch games.

With each computer, there is a Makey Makey set (not yet connected), an instruction sheet on how to connect it and some objects that conduct electricity.

The students split up, spreading out across the different computers, so they can set up the Makey Makey devices and test the first game or application. After about 15 minutes, they rotate to the next game. From now on, every 10 minutes, they move up again one activity.



Here, you can see the classic “*Banana Piano*”, but there are many more ideas and existing games. Take a look at <https://labz.makeymakey.com/d/> for inspiration and choose the ones you like best.



Warning: the game will only work if the player is holding the grounding cable (and touching the metal end). This is because Makey Makey uses an ‘electrical’ circuit for transferring information and those circuits always need to be closed. Don’t tell your students in advance: let them figure it out while experimenting.

Tips & Tricks:

- Continuously holding the ground can be difficult sometimes, especially when playing a fast, more complicated game like PacMan. Try making a (tight) bracelet out of tinfoil, connecting it to the ground cable and let the student wear it on his or her skin.
- If you feel like you need a more extensive video, explaining how to use a Makey Makey, you can take a look here: https://www.youtube.com/watch?v=-X3hb_YynM
- Makey Makey offers a lot of tools and resources for teachers, such as their Educators Guide (https://makeymakey.com/education/Makey_Makey_Educators_Guide.pdf) and Lesson Plans (<https://makeymakey.com/lessons/lesson-plans.pdf>)
- Looking for more inspiration? Take a look at what other teachers and students are creating at <https://labz.makeymakey.com/>.

Materials:

- 10 or more computers / laptops
- 10 or more Makey Makey kits
- Assortment of materials that conduct electricity, for example
 - Bananas, apples, oranges ...
 - Tin foil, aluminium or copper tape
 - Plasticine or modeling clay
 - Metal spoons and forks

SESSION 4: CREATING YOUR OWN SCRATCH PROJECT AND HOW TO TEACH IT TO OTHERS

Scratch: Creating a novel project

Nearing the end of the training, your students are ready to make teams of two and to start creating their own game. Of course, they will start off with a brainstorming exercise, to figure out what kind of game they want to make as a team.

Class discussion

You can begin this activity with a discussion on video games. If you want, you can use the questions below, or add up your own to get the conversation started.

- What video games do you play? What is your favourite?
- What do you like about them?
- What makes your favourite game so good?
- What elements are needed in a game? (i.e. a protagonist, an enemy, a background, a story ...)

Brainstorm

After the discussion, the pairs can start thinking about what kind of game they would like to create: an action game, a platform jumper, a Tamagotchi-style game and so on. There are infinite possibilities. Remember that they do have to keep it simple, as they only will have around 30 minutes to 'finish' their game during the activity. Let every pair of students use Worksheet 3 to write down what kind of game they want to create.

Creation

The students will have a short time to create the game they have in mind. They will apply all the concepts and coding tricks they learnt in the previous activities, to make something of their own. Keep the Starter Cards close by, in case anyone gets stuck and needs to look up a certain type of code.

Tips & Tricks:

- Looking for more help on how to get started with Scratch? Use these resources on their website:
 - Interactive tutorial: https://scratch.mit.edu/projects/editor/?tip_bar=getStarted
 - PDF guide: <https://resources.scratch.mit.edu/www/guides/en/Getting-Started-Guide-Scratch2.pdf>
- Scratch also has a page with information specifically designed for teachers: <https://scratch.mit.edu/educators/>
- While Scratch itself is an MIT project, other universities also help teachers find and create resources to work with code. You can find a large library of inspiration, guides and curriculum help at <https://scratched.gse.harvard.edu>

- Having a teacher account can help you track your students' progress. Find out how to create one at <https://scratch.mit.edu/educators/faq>

Materials:

- A computer or laptop for every student
- At least 10 prints out sets of the Scratch starter cards: free to download from http://www.capitaldigital.org/nl/scratch_starter_cards/
- At least 5 print out sets of the Scratch project cards: free to download from <https://resources.scratch.mit.edu/www/cards/en/ScratchCardsAll.pdf>
- Worksheet 3 for the brainstorming exercise
- Pens and pencils

Pedagogical Animation Skills II

Implementation: Motivation management & energizers

In this activity, you will go over several techniques to manage motivation in a group and to keep pupils active. There are the so-called *Energizers*, differentiation between pupils and clear, but fun introductions.

Fun, clear introductions

Every activity should start with a good introduction. A good introduction contains:

- A clear explanation of the activity
- The goals of the activity
- A fun or motivating element

When writing an introduction for an activity, the students should keep the following questions in mind:

- How will I explain what the pupils have to do? Is this clear to them?
- How will I make the explanation interesting or fun?
- What goals do I want the pupils to achieve by the end of the activity?

Here are some fun ways to brighten an introduction, instead of just talking and listening:

- A short piece of theatre
- A video
- Show-and-tell

Energizers:

Traditionally, every activity in a coding program starts with a short energizer: a 10-15 minute game designed to boost the energy and motivation level of participants. In [Appendix III](#), you will find five fun energizer activities, that you can use at the start of an activity or whenever you feel morale is low. Some are meant to play outside (with a lot of space), while other activities can be played indoors. You can find plenty more different games and energizers online.

NOTE: these games should be spread out across the whole program. In this activity, you can play one as well, but mostly discuss the benefits with the students: why do you do it?

Differentiation between pupils

First, start off with a group discussion: why would pupils lose motivation or act out during an activity? What can be the reasons behind it?

There are plenty of causes for pupils to lose their motivation or attention, such as hunger, a bad night sleep, a fight with parents, ... But sometimes you can also find the cause in the activity itself: perhaps it is too difficult, the introduction was not clear, or even too easy.

During an activity with pupils, the students should pay attention to how they are doing. A child that finds it very difficult, might need more individual explanations. Or perhaps another one finds it too easy and is done too fast. This child may then be asked to help others.

Implementation: Practical organisation & preparation

The majority of a successful activity lies in the preparation. During this session, you will support your students in the practical preparation and the content.

Team creation:

Your students will be divided into teams of three. In this team, they will be organising activities for pupils in primary education.

You can choose to let the students make their own teams, but we do not recommend this. Try creating teams that are balanced in:

Skill differentiation	Some students are strong on a pedagogical level, while others are better with technology.
Gender	Female animators can be a great role model for young girls. Try not to have teams with only one gender.
Social relationship	We don't recommend teaming up best friends or 'enemies'. Try to make sure the dynamic between the three students will work well.

Tasks and responsibilities as an animator:

As animators, the students will have certain tasks and responsibilities to successfully carry out the coding activities with the pupils. The students will also decide which animator is responsible for which activity.

Animator Tasks and responsibilities	
Preparing the activities	Every animator has to fill out the preparation for their activities and learn the contents of the activity.
Setting up the materials	Before the start of every activity, the materials need to be set up. After the activity, the materials are checked and organised.
Introducing and accompanying activities	The animators create a good introduction and make sure all pupils are actively engaged in the activities.
Helping team members	Every animator helps his team during activities. No animator stays absent on the side.
Being a good example for pupils	Animators show good and respectful behavior, to be an example to the pupils.

Schedule

You and your students take a look at the activity schedule. Make sure all the activities are clear to them and they know what to do. You can find an empty activity schedule to print out in [Worksheet 5](#).

For every activity, the team needs to write down a date and time they will visit the group of pupils; as well as assign one or two main responsible animators for each activity. This student will take the lead on that activity, while the others will help and support him/her. Make sure this division is done fairly and not one student does all the work.

Preparations for individual activities

[Worksheet 6](#) is a template that your students can use for preparing each of the individual activities. It is important that they fill these in for all the activities, so they are well prepared and don't forget any details.

On the preparation sheet, they will have to fill out the content of the activity, the learning goals, the introduction, what materials are needed. Also they will make a detailed step-by-

step course of the activity, in which they fill out what they will do first, next, with a detailed description and timeframe.

Materials:

- Code Etiquette poster, printed on A3 size
- Worksheet 5 and 6 for each student
- Pens
- Post-it notes
- Paper

SESSION 5: PRESENTATION AND REFLECTION

Scratch: finishing a novel project & presentation

And finally, showing the games the students made can be a real success story. They can proudly show their creation to the other participants and even share it on their Facebook-wall or on other social media. Maybe not all the games are completely finished yet or some might still contain some bugs... but that's part of the process and because it's so easy to edit your code right away this part is actually a good kind of beta-testing phase.

The most fun way to approach the presentation of the games is to approach it like the arcades. The students are probably too young to remember the arcades but you can bet some know what an arcade cabinet is. Basically you give them the task to think about how to present their game in the best possible way. Is it with the adapted Makey Makey controls or just with the keyboard and/or mouse? Every laptop on a separate table, or do we put every laptop next to each other on one long table?

If the students worked in pairs you can do an Arcade carrousel: One student moves from laptop to laptop to try every game. He can move to another laptop when the trainer gives a certain sign like a hand-clap or an alarm. The other student stays with the laptop to give an explanation on how the game works and can change some of the code if the visiting student gives feedback or if a bug arises. When the carrousel is finished it's time for the other student to explore the other games and the roles switch.

If every student made their own project it's best to work in small groups. Put students by maximum four and let them show their game to each other. This way they have different input and can already try different games at the same time. It will be a bit more chaotic but it's a lot of fun!

And if the games are suitable for it you can also make it a bit competitive: try to get the highest score, be the fastest to finish a maze-game, etc.

It's important to let the students present their game, it gives them the feeling they finished an entire project on their own. Let them know this is just the beginning however, there is so much more to learn from Scratch and if they ever need more inspiration they can always play and remix hundreds of games on the Scratch online platform.

Implementation: Evaluation & reflection

The majority of learning experiences happen during the coding activities and games, but also while reflecting. This is why your students will learn how to perform fun evaluation activities with the pupils. This doesn't mean that they will be testing them or giving points. In this program, we use *Active Reviewing* techniques to evaluate the past activities in a fun

but meaningful way. This will ensure that pupils will retain much better what they learnt and go home with a more positive experience.

In Appendix IV, you can find five different activities to use as an evaluation. However, there are plenty more of them to be found on the internet.

At the end of the 10 hour training program, the trainer will perform a thorough evaluation with your students, through one of these techniques. Aside from simply doing it themselves, you will also discuss with your students on the benefits and reasons for an evaluation.

EXTRA: INTRODUCTION INTO TEXT-BASED CODING

Summary: Visual coding is amazing, but to become a real-world programmer, you also need to know real, text-based languages, such as Python, JavaScript, Java, Ruby ...

In this activity (when there is time left), students try a hand at coding like a pro through the online free game Code Combat.

Timeframe: 30 to 60 minutes

Learning outcomes:

- | | |
|--|--|
| 1.1 Algorithms | Testing and debugging |
| 1.2 Sequences | 2.3 Reusing and Remixing |
| 1.3 Repetition and loops | 2.4 Abstraction |
| 1.4 Events and selection | 2.5 Modularization |
| 1.5 Parallelism | 2.6 Information: collection & management |
| 1.6 Conditionals and logical operators | 5.1 Problem identification |
| 1.7 Mathematical operators | 5.3 Implementation |
| 1.8 Variables and data management | 5.4 Evaluation & reflection |
| 1.9 Functions | 5.5 Iteratio |
| 2.1 Incremental & iterative work | |

Implementation:

Your students will surf to the website <http://codecombat.org> .

CodeCombat is a fun game in which you use code to vanquish enemies, such as trolls and ogres. Students create a new (free) account and choose what language they want to learn. We suggest choosing Python (one of the most used open-source coding language to create apps) or JavaScript (a language that is used on websites).



Next, the students will individually follow the instructions and move through increasingly difficult levels.

Materials:

- A computer with internet for each student

10 HOUR WORKSHOP PROGRAMME WITH PUPILS

Session Overview

10h Program	Activity	Content
Session 1: Introduction to coding hour 1 and 2	Code Etiquette: Rules of Conduct	<ul style="list-style-type: none"> * Class discussion on rules of conduct * Filling in and signing the Code Etiquette poster
	Introduction into Coding	<ul style="list-style-type: none"> * What is coding? - explanation * What is an algorithm? - discussion and video * Writing an algorithm for the tea-making-robot
	Entry level coding applications: * Lightbot * Hour of Code	<ul style="list-style-type: none"> * Two stations to try out apps: Lightbot and Hour of Code
Session 2: Introduction into Scratch hour 3 and 4	Computer Science Unplugged: * Sandwich Robot * Drawing Pixels	<ul style="list-style-type: none"> * Two stations with unplugged activities: Sandwich Robot and Drawing Pixels
	Scratch: Basic Introduction into Visual Coding	<ul style="list-style-type: none"> * First acquaintance with Scratch: video * First coding exercises: Scratch Cards
Session 3: Creative use of Scratch hour 5 and 6	Scratch: Crafting a first Project	<ul style="list-style-type: none"> * Using the example sheets to create a game * Learning the ins and outs of Scratch
	Tangible Computing with Makey Makey	<ul style="list-style-type: none"> * Playing games with the Makey Makey controller * Creating a Makey Makey controller
Session 4: Duo project 1 hour 7 and 8	Duo Project: Ideation	<ul style="list-style-type: none"> * Brainstorming in pairs on a game idea * Writing down the game concept
	Duo Project: Creation	<ul style="list-style-type: none"> * Coding the original game * Solving code problems
Session 5: Duo project 2 and evaluation	Duo Project: Presentation	<ul style="list-style-type: none"> * Presenting all the games to each other

hour 9 and 10	Evaluation	* Active reviewing
---------------	------------	--------------------

SESSION 1: INTRODUCTION TO CODING

Code Etiquette : Rules of Conduct

Summary: In this first activity, the students will set the rules of conduct for the program, together with the pupils. While the code workshops aren't as strictly organised as a traditional school lesson would, there is a need for some rules of conduct. Especially when pupils are working with digital devices, the group will need to have some agreements on behavior and respect.

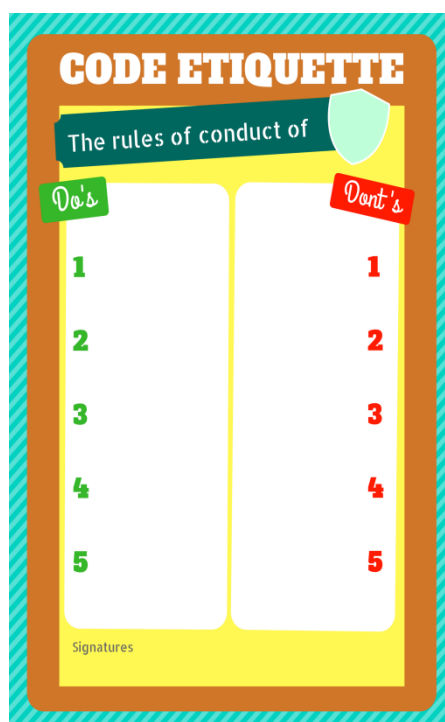
Timeframe: 30 minutes

Learning outcomes:


- 3.1 Working together
- 3.2 Negotiation practices
- 3.4 Describing thought processes
- 3.4 Learning from vicarious experiences

Implementation:

This activity is based on filling in a poster, called the Code Etiquette document. You can download this poster at <http://www.capitaldigital.org/nl/code-etiquette/>. Print it out on A3 size paper or larger.



CODE ETIQUETTE

The rules of conduct of 

Do's	Don't's
1	1
2	2
3	3
4	4
5	5

Signatures

Start the activity by showing the poster. Next, you will hand post-its and writing material to all the pupils. Ask them to reflect on at least 5 things the pupils should do (positive rules), such as having respect, being careful with the material, using inside voices ... Next, they get to reflect on 5 things the pupils shouldn't do (negative rules), such as no running inside, no food or drinks near the computers, ...

Collect all the post-its and start grouping them together with the pupils. As a group, you will select the 5 most important Do's and Don'ts. Write them down on the poster and have everyone sign this.

Tips and tricks: Using a collaborative approach to setting rules of conduct and having all the participants sign the poster, ensures that everyone is responsible for a good atmosphere.

Materials:

- Code Etiquette poster, printed on A3 size
- Pens
- Post-it notes

Introduction into Coding

Summary: In this group discussion and activity, you as an animator will inquire into the knowledge about coding that is already present in the pupils and supply additional insights. You will also explore together what makes a (good) algorithm.

Timeframe: 30 minutes

Learning outcomes:

- 1.1 Algorithms
- 1.2 Sequences
- 2.4 Abstraction
- 3.1 Working together
- 3.3 Describing thought processes
- 3.4 Learning from vicarious experiences
- 5.1 Problem identification
- 5.3 Implementation
- 5.4 Evaluation & reflection

Implementation:

Explain briefly what the purpose of this workshop program is (learning to code, to create cool digital games and apps) and what they will learn across the five sessions.

Next, you will continue the activity with a group discussion. You can use the following questions as a guideline, but feel free to add your own.

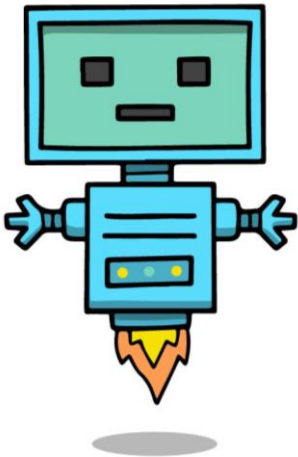
- What is code? Who knows what code does and who creates it?
- How are digital games, computer programs and smartphone apps made?

- Have you ever created a game, app or program yourself?
- Has anyone ever used the Scratch coding language before?
- What is an algorithm?

What is an algorithm?

An algorithm is a list of instructions you give to a computer or digital device, to solve a problem or get something done.

After inquiring into their knowledge into coding and algorithms, hand out pen and worksheet 1, along with the following assignment. They can complete their assignment together with another pupil, or alone.



Sally the Robot loves drinking tea, but she is not programmed by her creator to brew it herself.

Can you write down an algorithm for Sally on how to make and drink a nice cup of tea?

(Pssst: Sally needs very specific instructions, so she doesn't make any mistakes. Make sure she doesn't pour too much water in her cup.)

Give your pupils a few minutes to write down their algorithms. Afterwards, you can have a short group discussion on what makes a good algorithm and where they might have gone wrong. Common mistakes when writing a first algorithm are:

- Not being specific enough, i.e. when you tell Sally to pour water into a cup, she will keep pouring until you tell her to stop.
- Putting steps in the wrong order, i.e. turning the cooker on before filling it with water.
- Forgetting about steps we humans think are obvious, but a robot doesn't know about.

Tips and tricks:

Are you looking for a fun way to brighten up the discussion or give a starting point to your students? The BBC made a great series of videos for pupils on computing and the concepts behind it. Take a look at these two together with your pupils for inspiration:

Algorithms: <http://bit.ly/bbc-algorithms>

Coding: <http://bit.ly/bbc-coding>

Materials:

For this activity you will need:

- Pens and paper
- A computer with a beamer and a screen to project on
- A worksheet for every pupil (or two)

Entry Level Coding Application

Summary: In this activity, pupils will individually explore two different applications that teach the basic concepts of coding: 'LightBot' and 'Hour of Code'.



image by lightbot.com

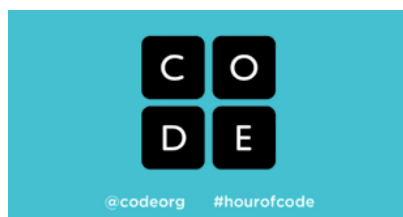


image by code.org

Timeframe: 45 minutes

Learning outcomes:

- 1.1 Algorithms
- 1.2 Sequences
- 1.3 Repetition and loops
- 1.4 Events and selection
- 2.1 Incremental & iterative work
- 2.2 Testing and debugging
- 2.4 Abstraction
- 5.1 Problem identification
- 5.2 Implementation
- 5.3 Evaluation & reflection

Implementation:

Prepare computers or tablets with one of the following applications: 'LightBot' and 'Hour of Code' (themed with Star Wars, Minecraft or Frozen).

Allow the students to test each application for about 20 minutes, to get familiar with the puzzles in them and how to solve them. Each of the apps is available in a wide range of European languages and provide comprehensive in-game instructions on how to use them.

At the end of the activity, take 5 minutes to discuss what they learnt and how well it went.

Tips & Tricks:

- Use a rotation system between the two apps, so students can try both.
- LightBot has a more visual approach to coding, with symbols for movement and actions, while the Hour of Code apps use coding blocks with text on them, similar to the Scratch coding language.

Materials:

- Computer or tablet for every student
- LightBot Application:
 - Install on tablet from [Google Play](#) (Android) or the [App Store](#) (iOS)
 - Or use the online webapp on a computer: <http://lightbot.com/flash.html> (Flash)
- Hour of Code Application:
 - Star Wars themed: surf to <https://code.org/starwars>
 - Minecraft themed: surf to <https://code.org/minecraft>
 - Frozen themed: surf to <https://code.org/frozen>

SESSION 2: INTRODUCTION TO SCRATCH

Computer Science Unplugged

Summary: Unplugged activities help pupils understand the basic principles of computer science and computational thinking, without even touching a digital device. In the '*Sandwich Robot*' activity, pupils practice their abilities to construct an algorithm with a well thought through sequence. The '*Drawing Pixels*' activity is designed to comprehend how images are displayed on screens through pixels.

Separate your group into two segments and after 20 minutes, let the pupils switch between the two activities.

Timeframe: 60 minutes

Learning outcomes:

- 1.1 Algorithms
- 1.2 Sequences
- 1.4 Events and selection
- 2.4 Abstraction
- 3.1 Working together
- 3.2 Negotiation practices
- 3.3 Describing thought processes
- 3.4 Learning from vicarious experiences
- 5.1 Problem identification
- 5.3 Implementation
- 5.4 Evaluation & reflection
- 5.5 Iteration

Implementation: Sandwich Robot

In this activity, you as an animator are a robot and will be '*programmed*' by your pupils, the programmers. The goal of the programmers is to have the robot successfully making a delicious sandwich with butter and cheese.

Introduction:

Explain to the pupils that you are a robot and want to make a sandwich with butter and cheese. However, you are not programmed to do that. The pupils are the programmers or engineers and have to create an algorithm for you to follow.

Give them a few minutes to discuss and write down what they think is the correct algorithm to have a robot make a sandwich. After that, they get to tell you the code and you, the robot, execute their commands. Warning: a robot takes everything very literally. If the students to command you to pick up bread, pick up the whole loaf. When they tell you to put

down the butter, put it on the floor. Make a lot of 'mistakes' so they will understand how precise programming really is.

Bonus: if you want to make your performance as a robot even better, make classic robot noises and repeat every command in a robot voice while you execute it (wrongly).

If you want, you can look at an example video here for inspiration: <http://bit.ly/CSUrobot>

This activity was translated from Codekinderen.nl

Implementation: Drawing Pixels

Information on Pixels:

Every day, we look at screens and see text, images, video... But how does a computer or a smartphone know how to display a tabby cat? Or a picture of a house?

All digital screens use the same technology to display visual information: with **pixels**. Those are tiny squares on the screen that can take any colour, from white to black and all the shades in between. All the pixels together make up the picture. Traditionally, pixel colours on a display are made by mixing amounts of red, green and blue (RGB). This combination is represented by a code numbers and letters, so the computer knows what to show. For example, the colour "hot pink" has the code `#ff69b4`.

Now, screens have a very high pixel density, so you can't see the little individual squares anymore. But if you look at old video games or so called 'pixel art', it becomes quite obvious to the eye. Even if you take a digital photograph and zoom in as much as you can, you can see all the separate pixels.



Space Invaders (Taito, 1978)



Super Mario Bros (Nintendo, 1985)

Class discussion:

Start the activity with a short class discussion. You can use the following questions, but can of course add your own.

- How are images displayed on a computer display?
- Can you send an image from one computer to another? How do you think that happens?
- How big is a pixel?
 - What kinds of games do you know that have 'pixel art'?

Make your own pixel drawings:

Hand out pencils, erasers and printouts of Worksheet 2 for every pupil.

The students follow the instructions on the worksheet to draw their own pixel drawings, inside the provided grids. Every grid comes with number instructions on the side, which make up the code for the drawing. Be careful: the first number of every row is always white.

Drawing						Code			
						2	3		
						5	1		
						2	4		
						1	1	3	1
						1	1	3	1
						2	4		

On the right, you see the number code for each row. The first number is always white, the second number black, then white again, black again ... Rince and repeat.

For example, on the first row, you have to leave 2 squares white and colour 3 squares black.

On the fourth row, you have more numbers. The first number is 1 white, then 1 black, three white and lastly 1 white.

This activity was translated from Codekinderen.nl

Tips and Tricks:

- Do you and your pupils want to learn more or try some extra activities on how images are displayed on digital devices? Surf to <https://classic.csunplugged.org/image-representation> for more information.
- You can find a lot more cool and fun Computer Science Unplugged activities for free, in a variety of languages at <https://csunplugged.org>

Materials:

For the *Sandwich Robot* activity, you will need:

- A loaf of bread
- A pack of butter
- Several slices of cheese
- A plate
- A knife
- Cleaning supplies, because things will get messy

For the *Drawing Pixels* activity, you will need:

- Worksheet 2

- Pencil and erasers

Scratch: Basic Introduction into Visual Coding

Summary: Scratch is a visual coding language and environment, to teach pupils and students to code in a fun, easy way. Your pupils can use Scratch to code their own interactive stories, animations, and games. In the process, they learn to think creatively, reason systematically, and work collaboratively — essential skills for everyone in today's society.



In the following Scratch activities, your pupils will learn the basics of coding with blocks: movement, colour changes, animations and so on; as well as create a first fun project and design their own digital game.

Timeframe: two times 60 minutes

Learning outcomes:

- | | |
|--|--|
| 1.1 Algorithms | 2.6 Information: collection & management |
| 1.2 Sequences | 3.1 Working together |
| 1.3 Repetition and loops | 3.2 Negotiation practices |
| 1.4 Events and selection | 3.3 Describing thought processes |
| 1.5 Parallelism | 3.4 Learning from vicarious experiences |
| 1.6 Conditionals and logical operators | 4.1 Combination |
| 1.7 Mathematical operators | 4.2 Exploration |
| 1.8 Variables and data management | 4.3 Transformation |
| 1.9 Functions | 5.1 Problem identification |
| 2.1 Incremental & iterative work | 5.2 ideation and brainstorming |
| 2.2 Testing and debugging | 5.3 Implementation |
| 2.3 Reusing and Remixing | 5.4 Evaluation & reflection |
| 2.4 Abstraction | 5.5 Iteration |
| 2.5 Modularization | |

In this activity, your pupils will make a first acquaintance with the Scratch coding language and environment. In Scratch, you use building blocks to create a digital game, animation or app.

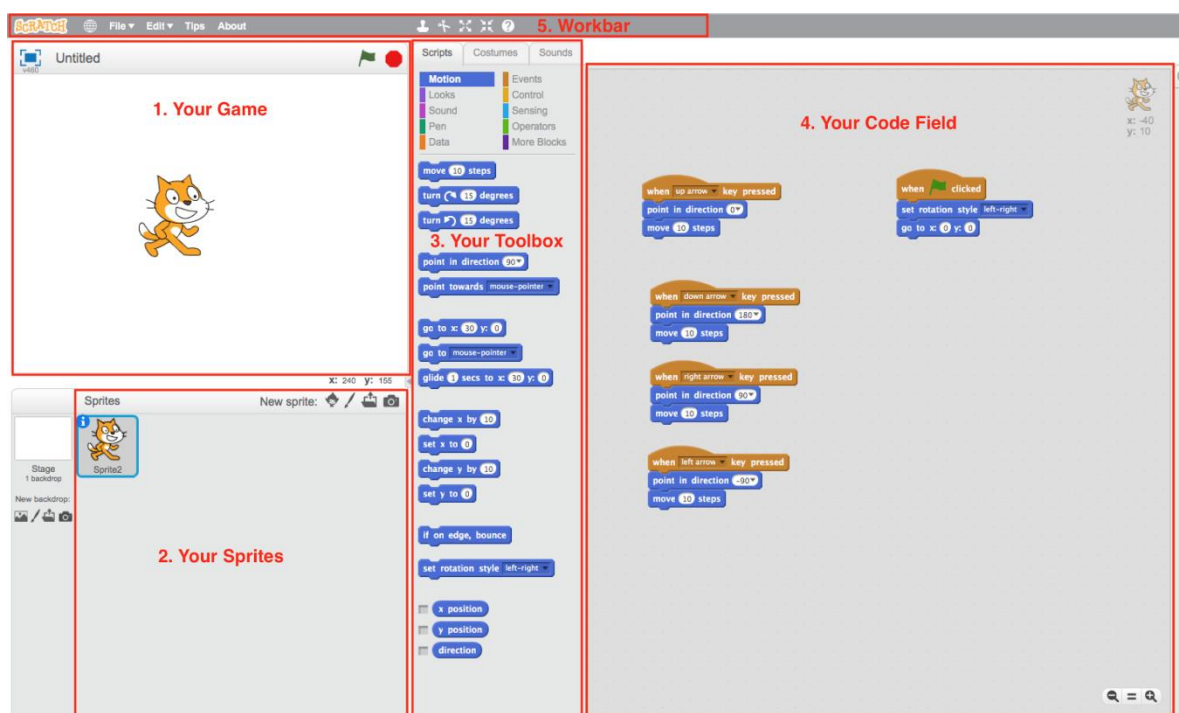
Getting started

You can choose to use the Scratch environment on computer (1) online at <https://scratch.mit.edu/> or (2) without internet by installing the offline editor (<https://scratch.mit.edu/download>). If you want the pupils to work online and save their work (recommended), have them create an account. Follow the instructions at

<https://scratch.mit.edu/educators/faq> to help you create a teacher and individual student accounts.

First view on the program

Of course, you first want to show your pupils what Scratch is before diving in. You can either do this by sharing your own screen on a beamer and making a few first coding moves. Or you can show an introductory YouTube video. There are plenty of them out there, but we suggest one made by a student called "Matt the Scratch Kid": <http://bit.ly/intro-scratch>.



SESSION 3: CREATIVE USE OF SCRATCH

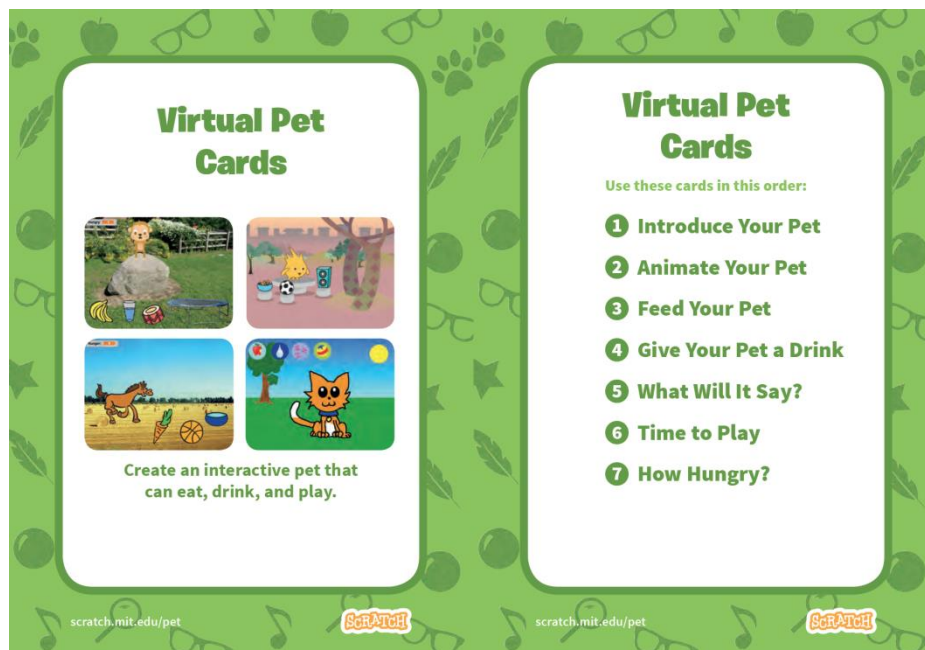
Scratch: Crafting a first project

Now your pupils are ready to learn in the most fun way possible: by playing and experimenting. Hand out the sets of **starter cards**, which show a basic coding procedure each, such as moving with the keyboard arrows, using a mouse to move, changing a character's colour and so on. Challenge your students to try out all the cards in the set.



You can download the English set of starter cards for free at http://www.capitaldigital.org/nl/scratch_starter_cards/

Now it is time to create a first project or game. Luckily, the Scratch website already provides a lot of cool example projects, that are quick to try out. Just like the basics, they come in fun, easy to follow project cards: <https://resources.scratch.mit.edu/www/cards/en/ScratchCardsAll.pdf> .



Download the cards, print them out around five times and put them in sorted stacks at the front of the classroom. Let every student pick a project that they would like to complete during this activity. From that point, it is just an easy ride following the instructions.

If a student finishes early, challenge them to either improve on the project they made (by adding a score, increasing the difficulty, changing the theme of the game ...) or to create a second, more difficult one.

At the end, allow for 10 minutes so that students can test each other's projects and have a quick discussion about what they learnt so far.

Tips & Tricks:

- Looking for more help on how to get started with Scratch? Use these resources on their website:
 - Interactive tutorial: https://scratch.mit.edu/projects/editor/?tip_bar=getStarted
 - PDF guide: <https://resources.scratch.mit.edu/www/guides/en/Getting-Started-Guide-Scratch2.pdf>
- Scratch also has a page with information specifically designed for teachers: <https://scratch.mit.edu/educators/>

Materials:

- A computer or laptop for every pupils (if not possible, work in groups of two)
- At least 10 print out sets of the Scratch starter cards: free to download from http://www.capitaldigital.org/nl/scratch_starter_cards/
- At least 5 print out sets of the Scratch project cards: free to download from <https://resources.scratch.mit.edu/www/cards/en/ScratchCardsAll.pdf>
- Worksheet 3 for the brainstorming exercise
- Pens and pencils

Tangible Computing with Makey Makey

Summary: Makey Makey is an invention kit for the 21st century that ignites curiosity, challenges problem-solving ability, and fosters creativity. With Makey Makey everyday objects are transformed into touchpads empowering students to interact with computers as creative tools. The computer becomes an extension of their creativity, fostering imaginative play and discovery.



In this activity, the pupils will create innovative game controllers out of fruits, cutlery, tin foil and so on. Imagine playing the classic PacMan with a set of spoons or a piano made out of bananas and apples. This is the ultimate creative exercise, connecting the real, tangible world with digital coding.

Timeframe: 60 minutes

Learning outcomes:

- | | |
|--|---|
| 1.1 Algorithms | 3.2 Negotiation practices |
| 1.2 Sequences | 3.3 Describing thought processes |
| 1.3 Repetition and loops | 3.4 Learning from vicarious experiences |
| 1.4 Events and selection | 4.1 Combination |
| 1.6 Conditionals and logical operators | 4.2 Exploration |
| 1.7 Mathematical operators | 4.3 Transformation |
| 2.1 Incremental & iterative work | 5.1 Problem identification |
| 2.2 Testing and debugging | 5.2 ideation and brainstorming |
| 2.3 Reusing and Remixing | |
| 2.4 Abstraction | 5.3 Implementation |
| 2.5 Modularization | 5.4 Evaluation & reflection |
| 2.4 Information: collection & management | 5.5 Iteration |
| 3.1 Working together | |

Implementation:

Introduction:

You can introduce the MakeyMakey controller with a short video showing its possibilities, or with a small example performance. You can find a clear, well explained video on YouTube by following this link: <http://bit.ly/intro-makey>

The other way to introduce the MakeyMakey, is by connecting it to a computer through USB, running a Scratch algorithm that makes a sound whenever you press a button. Connect the ground cable to the Makey Makey (explained in Worksheet 4) and also a cable to the spacebar slot. Make sure the volume is turned up.

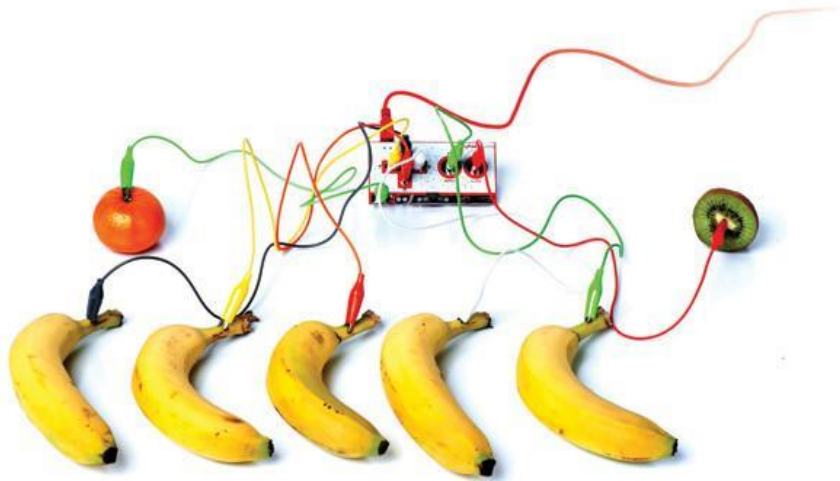
You, as trainer, hold the ground cable, while a pupil takes hold of the spacebar cable. All the pupils and you make a circle, holding hands except the last person (who is holding the cable). Have the two pupils -that are not holding hands- high five each other. A sound will ring on the computer. Have them try high fiving at different points in the circle.

Testing games with the Makey Makey

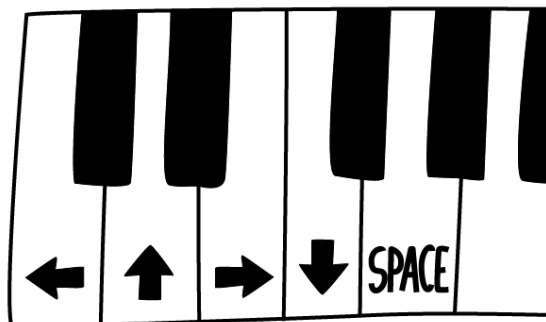
Set up your computers, each with a (different) app on it that is compatible with the Makey Makey. You can also use retro games, that you can find online, such as PacMan or Super Mario, or most Scratch games.

With each computer, there is a Makey Makey set (not yet connected), an instruction sheet on how to connect it and some objects that conduct electricity.

The pupils split up, spreading out across the different computers, so they can set up the Makey Makey devices and test the first game or application. After about 15 minutes, they rotate to the next game. From now on, every 10 minutes, they move up again one activity.



Here, you can see the classic “*Banana Piano*”, but there are many more ideas and existing games. Take a look at <https://labz.makeymakey.com/d/> for inspiration and choose the ones you like best.



Warning: the game will only work if the player is holding the grounding cable (and touching the metal end). This is because Makey Makey uses an ‘electrical’ circuit for transferring information and those circuits always need to be closed. Don’t tell your students in advance: let them figure it out while experimenting.

Tips & Tricks:

- Continuously holding the ground can be difficult sometimes, especially when playing a fast, more complicated game like PacMan. Try making a (tight) bracelet out of tinfoil, connecting it to the ground cable and let the student wear it on his or her skin.
- If you feel like you need a more extensive video, explaining how to use a Makey Makey, you can take a look here: https://www.youtube.com/watch?v=-X3hb_YynM
- Makey Makey offers a lot of tools and resources for teachers, such as their Educators Guide (https://makeymakey.com/education/Makey_Makey_Educators_Guide.pdf) and Lesson Plans (<https://makeymakey.com/lessons/lesson-plans.pdf>)
- Looking for more inspiration? Take a look at what other teachers and students are creating at <https://labz.makeymakey.com/>.

Materials:

- 10 or more computers / laptops
- 10 or more Makey Makey kits
- Assortment of materials that conduct electricity, for example
 - Bananas, apples, oranges ...
 - Tin foil, aluminium or copper tape
 - Plasticine or modeling clay
 - Metal spoons and forks

SESSION 4: DUO PROJECT 1

Summary: The main activity of the code program is for the pupils to design and code a game in pairs. Here, they will use all the skills they have learned throughout the other activities, to create something original. They will need to think like a designer, solve problems with code, work together, present their work and so on.

This activity is divided into three phases: ideation, creation and presentation. Every phase is just as important as the other two. In the ideation phase, pupils will brainstorm about what kind of game they want to make and what coding elements they will need. In the creation phase, they will use Scratch to code their game and solve many problems (bugs) they encounter on the way. Last but not least: they will show their game to their peers and give a short presentation. The activity as a whole is focused on creative work, positive feedback and experiencing success.

Timeframe: three times 60 minutes

Learning outcomes:

- | | |
|--|--|
| 1.1 Algorithms | 2.6 Information: collection & management |
| 1.2 Sequences | 3.1 Working together |
| 1.3 Repetition and loops | 3.2 Negotiation practices |
| 1.4 Events and selection | 3.3 Describing thought processes |
| 1.5 Parallelism | 3.4 Learning from vicarious experiences |
| 1.6 Conditionals and logical operators | 4.1 Combination |
| 1.7 Mathematical operators | 4.2 Exploration |
| 1.8 Variables and data management | 4.3 Transformation |
| 1.9 Functions | 5.1 Problem identification |
| 2.1 Incremental & iterative work | 5.2 ideation and brainstorming |
| 2.2 Testing and debugging | 5.3 Implementation |
| 2.3 Reusing and Remixing | 5.4 Evaluation & reflection |
| 2.4 Abstraction | 5.5 Iteration |
| 2.5 Modularization | |

Duo Project: Ideation

The ideation phase of this activity starts off with a class discussion on video games. Take a whiteboard and marker to write down the main points of the discussion, such as game genres and the different elements of a good game.

You can use the questions below, or create your own:

- What games do you like to play? Why are those fun?

- What makes your favourite game so amazing?
- What elements make or break a game?
- How many different kinds of games can you name? What genres are there?

After the general class discussion, it is time to divide the pupils into pairs. You can choose to (1) assign pairs yourself or (2) have them pair up according to interest.

One method to pair up the pupils by interest, is by hanging up sheets with on each a different genre or theme of games on them across the classroom. Pupils take position next to the genre or theme they want to work on. Then, they form pairs within their genre or theme.

Once the pupils are paired up, hand them each a copy of Worksheet 3: Brainstorming. They will fill in the different questions from 1 to 7. The last step is thinking of a name for the game. The pupils will brainstorm on the theme of their game, what kind of character will be the protagonist, how you can win the game, what actions can be taken and so on.

Encourage them to also make a rough drawing of what the game will look like on the back of the brainstorming sheet.

Duo Project: Creation

The pupils will then start to create the game they have mapped out. They will apply all the concepts and coding tricks they learnt in the previous activities, to make something of their own. Keep the Starter Cards closeby, in case anyone gets stuck and needs to look up a certain type of code.

SESSION 5: DUO PROJECT 2 AND EVALUATION

Duo Project: Presentation

One of the most fun and important parts of making your own games, is presenting your hard work to your peers. Make sure you take plenty of time for each pair to talk about the game they made, show it to the group and have other pupils test it. It can also be a good idea to invite other class groups or even the parents.

To keep the presentation from becoming a stressful performance, turn it into a light hearted event, with a drink and snacks, and perhaps even handing out a cool certificate.

Getting acknowledgement for you hard work and seeing the result of it is a key element in the empowering force of learning to code. Pupils will feel successful for completing their own game and showing it to the world. They have every right to be proud.

Tips & Tricks:

- Looking for more help on how to get started with Scratch? Use these resources on their website:
 - Interactive tutorial:
https://scratch.mit.edu/projects/editor/?tip_bar=getStarted
 - PDF guide: <https://resources.scratch.mit.edu/www/guides/en/Getting-Started-Guide-Scratch2.pdf>
- Scratch also has a page with information specifically designed for teachers: <https://scratch.mit.edu/educators/>

Materials:

- Whiteboard
- Markers
- Worksheet 5
- Pens and pencils
- Laptop with Scratch for every pair of pupils
- Beamer for presentation

Evaluation: Active Reviewing

Summary:

The majority of learning experiences happen during the coding activities and games, but also while reflecting. This is why your students will learn how to perform fun evaluation activities with the pupils. This doesn't mean that they will be testing them or giving points. In this program, we use *Active Reviewing* techniques to evaluate the past activities in a fun but

meaningful way. This will ensure that pupils will retain much better what they learnt and go home with a more positive experience.

Timeframe: 30-40 minutes

Learning outcomes:

- 3.1 Working together
- 3.2 Negotiation practices
- 3.3 Describing thought processes
- 3.4 Learning from vicarious experiences

Implementation:

In Appendix IV, you can find five different activities to use as an evaluation. However, there are plenty more of them to be found on the internet.

At the end of the 10 hour training program, you will perform a thorough evaluation with your pupils, through one of these techniques. Aside from simply doing it themselves, you will also discuss with your students on the benefits and reasons for an evaluation.

Tips and tricks:

Materials:

- Pens and paper
- Post its

PRACTICAL ORGANISATION

- One student for 5 pupils

THERESOLDS

In the survey, most of the partners mentioned the fact that they foresee problems with teachers and pupils.

First of all, Maks, during the Capital digital project, never had problems convincing the schools for free coding activities if we provide the materials and the hardware needed. But a good and structured preparation with the teachers is needed.

Teachers

Teachers are curious to see what the activities are and once they see that the pupils are enthusiastic and enjoying the activities, all the thresholds fall away.

Directions are proud to provide coding activities in the classroom: coding is "in" and parents like the fact that coding is introduced in the school. It gives a special "prestige" to the school.

Talk with the direction and try to get their support in a written way by making a convention.

Say no to chaos in the classroom, bring some structure

Teachers are afraid of chaos in the classroom, because in their eyes non-organization is an obstacle for the learning of the pupils. That's why the exercises need to be given in a very structured way. It is important to think about the way you organize the classroom and how the different exercises will be scheduled. In our training curriculum with the students we have to discuss this. Can the structure of the classroom be changed for this day or not?

How can we make group work with pupils without changing the structure of the classroom? Is there another room that can be used for the activities?

This is an important thing to discuss with the teachers before the start of the project.

Discuss the timetables before you start with the project

For the coding activities it is interesting to have some blocks of two-three working hours, instead of courses from 50 minutes like custom in secondary schools (in Belgium ?- is it also so in Spain, Cyprus and Germany).

5 blocks of 2 hours or 2 blocks of 3 and 2 of 2 are ideal to plan the activities with the students. In primary school, it is easier to schedule this than in secondary.

Pupils

Show a try out in the classroom, before you start the project

Introduce the coding activity in the classroom by yourself, telling the students what they can expect from this project and how the project will be developed.

Let them try out something practical, so they see that this is fun.

Most of the pupils are glad when there are projects in the classroom, it changes from the daily routine and dry courses they get.

Ask them about their fears for working with the pupils and discuss this with them.

If you have the possibility, try to look with them what kind of primary school they want to work with. If not explain them why the school you choose, need their help for learning the pupils to code.

Role models for little brothers and sisters

In most of the disadvantaged families, ties between siblings are very strong and “big” brothers and sisters like to help the little ones. Talk about being a role model for the little ones and the role they can play in the fact that pupils will discover coding.

WORKSHEETS

https://docs.google.com/document/d/1wEyC7jfkImP7ks15qr_cx7pFo7kqc5Y0X0lhqYq5U6c/edit#

LINKS

Lightbot (online en apps) - <http://lightbot.com/hour-of-code.html>

Hour of Code: Star Wars - <https://code.org/starwars>

Hour of Code: Minecraft - <https://code.org/minecraft>

MATERIALS

Item	Est. price / item	Number	Est price total €
Makey Makey set	around 55 euro for a set	10	550
Computers / laptops		10	
Tablets (optional)	100	10	1000
Crafting materials			
Writing materials			

APPENDIX I: REFERENCES

Ananiadou, K. (2009). *21st Century Skills and Competences for New Millenium Learners in OECD Countries*.

Brennan, K., & Resnick, M. (2012). *Using artifact-based interviews to study the development of computational thinking in interactive media design*. Paper presented at annual American Educational Research Association meeting, Vancouver, BC, Canada.

Raspberry Pi Foundation. (2017). *CodeClub Curriculum*. Retrieved March 6, 2018, from <https://codeclubprojects.org/en-GB/curriculum/>

<http://curriculumvandetoekomst.slo.nl/21e-eeuwse-vaardigheden>

APPENDIX II: WORKSHEETS

https://docs.google.com/document/d/1wEyC7jfkImP7ks15qr_cx7pFo7kqc5Y0X0lhqYg5U6c/e/dit#

APPENDIX III : ENERGISER ACTIVITIES

- Untangle Yourself

Untangle yourselves is a great energizer to get people moving. It has a very interesting message on finding your way out of a tangled situation.

Running the activity

1. Ask the group to form a circle
2. Ask everyone put their hands up
3. Give the tangling instructions
4. Ask the group to untangle themselves without letting the hands go, and try to form a circle

The tangling instructions

*With your right hand, grab someone's left hand
With your left hand, grab someone's right hand
You cannot grab the hands of people next to you.*

- Reverse Hide & Seek

Sardines is an active game that is played like hide and go seek — only in reverse! One person hides, and everyone else searches for the hidden person. Whenever a person finds the hidden person, they quietly join them in their hiding spot. Soon, the hidden group starts to look like a bunch of sardines!

- Big Fat Pony

Let's stand in a circle.

I'll start running around the circle and sing the song.

This is the story of my pony,
Story of my big fat pony,
This is the story of my pony,
This is what they told me.

At this moment I stop in front of a person in the circle and start dancing with her/him:

Front, front, front, my baby.
Side, side, side, my baby.
Back, back, back, my baby.
This is what they told me.

Then I take the person I danced with and start running around with her/him again.

- Woosh:

Have everyone form a standing circle. Tell students you have a ball of energy that you will send around the circle. This ball of energy moves around the circle with the sound, "Whoosh!" Encourage students to engage their whole body as they say "Whoosh!" and send it to the person next to them. Send the energy around the circle once. When it comes back to you, introduce the next way it moves: If you say "Whoa!" and put your hands up, that reverses the direction of the energy. Play for a bit with "Whoosh" and "Whoa." Then add in "Zap" which sends the energy to someone across the circle while making eye contact and pointing to that person. Play with these three commands.

- People Bingo

People bingo is a great ice breaker game for adults because it's fun, easy to organize and almost everyone knows how to play. In as little as 30 minutes, you can energize a classroom or a meeting and help your students or coworkers [get to know each other better](#) with just a handful of bingo cards and some clever questions.

Whether your event has three people or 300, it's easy to play people bingo. Here's how to get started.

[Create Your People Bingo Questions](#)

If you know your participants, make a list of 25 interesting traits that describe different aspects of them, things like, "plays the bongos," "once lived in Sweden," "has a karate trophy," "has twins" or "has a tattoo."

If you don't know your participants, make a list of more general traits like "drinks tea instead of coffee," "loves the color orange," "has two cats," "drives a hybrid" or "went on a cruise in the last year." You can make these easy or difficult depending on how much time you want the game to take.

Still stuck for ideas? Start with this list of [100 great people bingo questions](#).

Make Your People Bingo Cards

It's very easy to make your own bingo cards using regular printer paper. Check out these easy step-by-step instructions to learn [how to make a people bingo card](#).

There also are lots of places online where you can create customized people bingo cards. Some are free; some are not. One site, [Teachnology](#), has a card maker that allows you to shuffle the phrases on each card. Another site, [Print-Bingo.com](#), allows you to customize with your own words or use their suggestions.

APPENDIX IV: ACTIVE REVIEWING ACTIVITIES

Active Reviewing' describes how animators can bring together the worlds of talk and action in experience-based learning by making use of these active learning methods.

Active reviewing improves our ability to learn from experience. Most active reviewing is simple, basic and direct. Used wisely it can enliven and sharpen the process of reviewing experience.

The Hand

Everyone draws the outline of their left or right hand on a sheet of paper. In the fingers they write the following things:

- **Thumb:** what you liked best today
- **Index finger:** something you have to pay more attention to or keep in mind
- **Middle finger:** what you didn't like today
- **Ring finger:** something that is important to you, a value or something you have learned
- **Little finger:** something you feel insecure about, you want to practice more...

Finally, everybody reads aloud what she or he has written on the fingers

These sheets can be hang up afterwards in the workshop room.



Type of activity: Active reviewing – Evaluation

Duration: 30 min

What you need: blanc sheets of paper for all the participants, pens, crayons, markers...

Educator participates? Yes

▪ Fridge, Trash, Backpack

At the end of the training, ask participants which of the learned activities, they put in the trash, which one in the fridge and which one in the backpack.

You can draw a fridge, a trash can and a backpack and work with post-its.

Type of activity: Active reviewing – Evaluation

Duration: 30 min

▪ My Balloon

Imagine you are the pilot of your own balloon; to raise up you need to let go of some sandbags. Once over the side of the basket they are gone for good.

Please be careful where you dump your sandbags and consider those below

▪ Twitter Unplugged

Make an evaluation of the activity using 140 characters, hashtags

