Discover or illustrate coding concepts with a dystopian puzzle game

Previous compulsory steps / Prior students' knowledge	Coding basics, otherwise good teacher supervision required
Learning objectives	Discover programming with the first levels. Supplement a class on coding concepts with the harder ones.
Subjects	Digital Literacy, Programming, Mathematics
Recommended Age	15-18
Material needed	Game: 7 billion humans (Windows, MacOs, Linux, AppStore, GooglePlay)
Sequence duration	60 minutes
Individual or group activity	Group Activity
Skills developed (after learning goals)	Problem solving, Planning
Game price range	<15 €, <6 € (mobile apps)
Similar games that you can use with the sequence	Human Resource Machine (https://tomorrowcorporation.com/humanresourcemachine)

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

Project code: 2020-1-FR01-KA201-080669

Step by step: how to implement the sequence

In this sequence you are going to play a programming puzzle game together with your class. The game uses dark humor and a cute yet dystopian setting, making it a light satire of modern office work. By progressing through levels of increasing difficulty, you will discover new programming concepts. The first levels are fairly simple, but you will need to supervise and provide more advice for the later ones.
It would be better to prepare the session by playing through the first 10 levels on your own. This will enable you to move on to more challenging levels and to avoid game explanations if your students' skills are sufficient.

- **Step 1 – Presenting the game and the sequence (5 minutes)**

Tell your students that you are going to play a puzzle game in order to approach programming principles playfully. If you already presented programming concepts to your class, use it as a way to illustrate them in a funny way.
Launch and start the game. It will ask you to select an employee ID (ie. a game save slot) and an avatar. You can go through this process quickly.
A cinematic sequence is played that presents the satiric universe of the game followed by the level selection screen.
Choose the next level ("Welcome new employees").

- **Step 2 – Demonstrating game mechanics (10 minutes)**

Explain the game interface and mechanics by going through the first 3 levels.
For this step, perform the actions yourself and comment them. Make sure that your students understand the mechanics.

Breakdown of the interface:

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

Project code: 2020-1-FR01-KA201-080669

In-game screenshot: basic mechanics step 1

The box with programming commands sits right next to it.

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

The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Project code: 2020-1-FR01-KA201-080669

In-game screenshot: basic mechanics step 2

Each step-command corresponds to one floor tile in the workers' room.

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

Project code: 2020-1-FR01-KA201-080669

Execute your program by clicking the play button in the bottom left corner:

In-game screenshot: executing the program

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

Project code: 2020-1-FR01-KA201-080669

Level 4 (“Long distance delivery”) presents the concept of loops:

In-game screenshot: introducing loops

Play with your class in an interactive manner from level 4 onwards (next step).

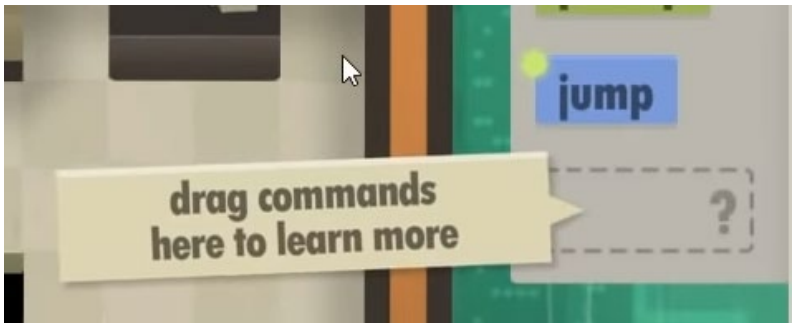
- **Step 3 – Go through levels with your class (35 minutes)**

For each level:

- Clearly state the objective of the level
- Explain the new commands that the game made available as you progress:
  - Explain how they work by using the help slot at the bottom of the command list

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

Project code: 2020-1-FR01-KA201-080669



- Combine them with other commands to demonstrate how they work by running an unfinished test code.

- Give your students a few minutes to think of a solution, then ask who wants to give it a try and implement a student's solution.
  - If it works, great! Make sure everyone understands before moving on.
  - If it doesn't, enter the debug mode and try to understand what went wrong.

The debugging process is explained in level 6 ("little exterminator 1"). You can run your code line by line using the arrow button:

If your students struggle, break down the steps for them and ask them which command must be used for each step.

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

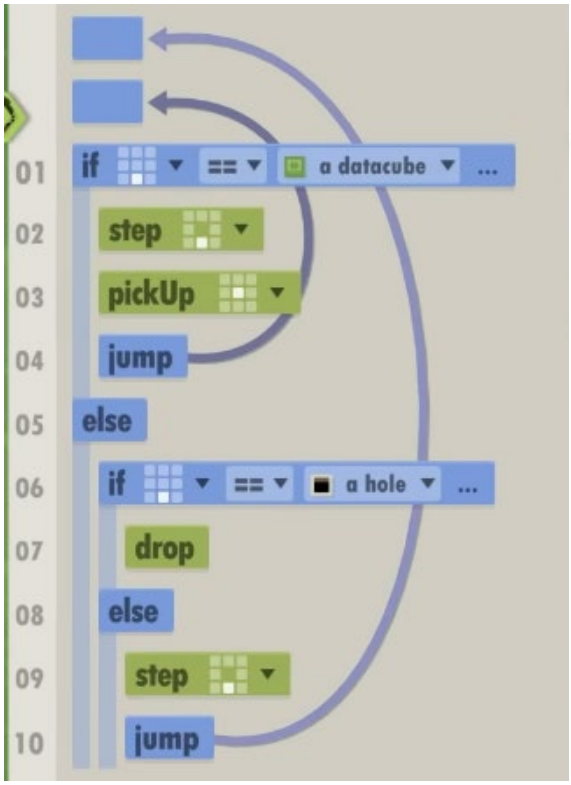
Project code: 2020-1-FR01-KA201-080669

If they are totally blocked, give away the answer. Waiting too long would kill the dynamism of the session. The pleasure of being given the solution to a problem can be as enjoyable as solving it yourself (think about mystery novels). However make sure students completely understand the solution.

- **Step 4 – Debrief and overture (10 minutes)**

Explain the black-box concept in programming: the inner workings of a function does not need to be known to use it.

Compare an algorithm in Python with an algorithm extract from the game and compare them.



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

The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Project code: 2020-1-FR01-KA201-080669

```
x = 0
y = 0
tile = workspace.get(x,y)
character = Character(tile, workspace)

while(True):
        if(isinstance(workspace.get(character.getCurrentTile().getBottom()), “Datacube”)):
                character.step(character.getCurrentTile().getBottom())
                character.pickup()

        else:
                if(isinstance(workspace.get(character.getCurrentTile().getBottom()), “Hole”)):
                        character.drop()
                else:
                        character.step(character.getCurrentTile().getBottom())
```

These are different representations (languages) of the same algorithm. We don’t know the code inside the functions but all we need to know is what action they perform. This way we can arrange them to produce an algorithm that suits our needs.

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

Project code: 2020-1-FR01-KA201-080669

Resources

Getting the game

https://store.steampowered.com/app/792100/7_Billion_Humans/

https://www.epicgames.com/store/fr/p/7-billion-humans

https://apps.apple.com/fr/app/7-billion-humans/id1393923918

https://play.google.com/store/apps/details?id=com.tomorrowcorporation.sevenbillionhumans

Information

Guevara, W. G. (2018, November 29). What is “black-box code” and why it’s important. thatsoftwaredude.com Retrieved from https://www.thatsoftwaredude.com/content/8881/what-is-black-box-code-and-why-its-important.

Bristol, J. B. (2019). 7 Billion Humans: Amusing puzzler challenges kids, teaches programming principles. commonsense.org Retrieved from https://www.commonsense.org/education/app/7-billion-humans.

All screenshots used in this lesson were taken from 7 Billion Human, Tomorrow Corporation (2018)

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

The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Project code: 2020-1-FR01-KA201-080669