DEMETER PROJECT

13/03/2021

LEAR2PROGRAM UNIT

DemeteR

“The Web as I envisaged it, we have not seen it yet. The future is still so much bigger than the past.”

Tim Berners-Lee, Inventor of the World Wide Web.

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

INDEX

UNIT: LEARN2PROGRAM.

Subjects: Web programming, programming, Assembly and maintenance of microcomputer equipment

Classes / Levels: Vocational school (High and Medium degree)

Product I. Invention of a pseudo-language and bases of the game (problem)

Country: ESPAÑA
Class: 1º SMR
IT Teacher (Montaje y mantenimiento de equipos).

Product II. Creation of a web game interpreter

Country: ESPAÑA
Class: 1 DAW
IT Teacher: (Entornos de desarrollo), second IT Teacher (programación) third IT Teacher (Lenguajes de marcas).
Class: 2º DAW
IT Teacher: (front-end web development, development environments) and IT Teacher (back-end web development).

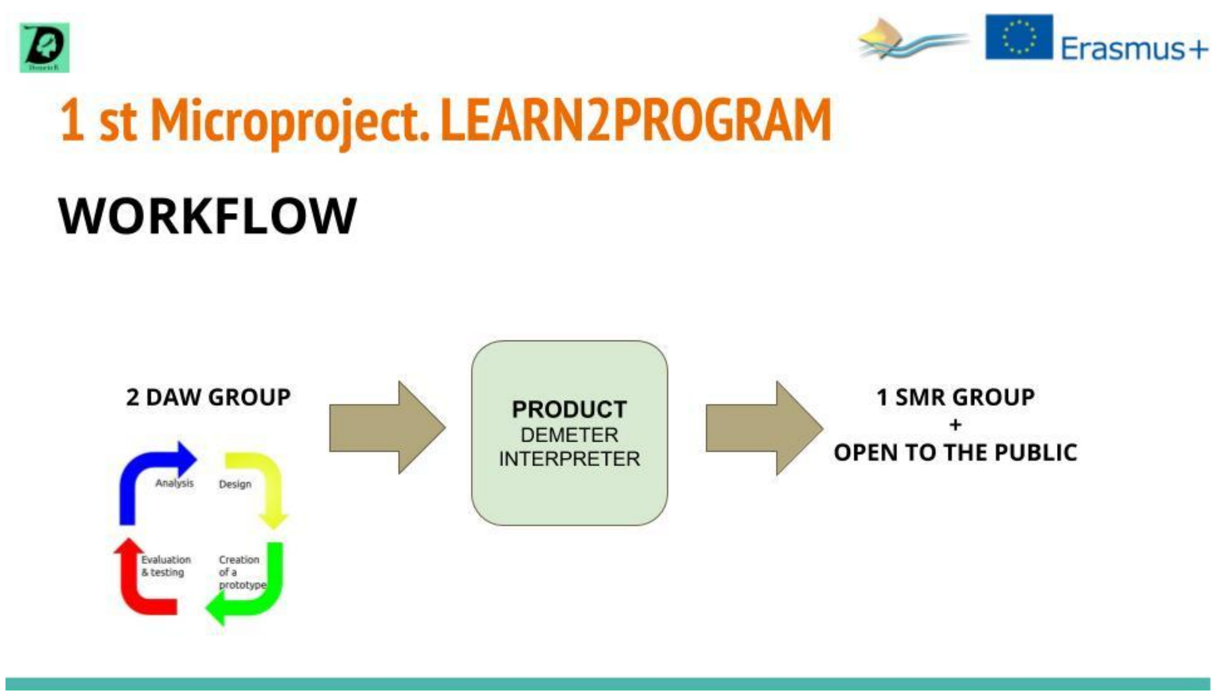


Figure 1.1 Learn2program workflow

Necessary resources and material

For the programming part of the online game, a team will be needed with suitable programming software (Atom, VisualStudio, etc.).

For the DEMETER programming part any device is valid.

For the analysis a program UML capable like DIA or other

The objective of this Unit is to create a game in which students learn to program in a simple and playful way. Gamification and thematic learning will be used as tools to improve motivation, student involvement and its commitment to the educational process.

First of all we have to find a game in which the expected learning is achieved. In our case, programming is the basis of several subjects. In Spain we have two vocational school studies, one with a lower level in which students learn to program and another with a higher level in which students will be full stack developers.

The school in Spain has an HVET specialized in Full Stack Web programming. We are experts in web development, then learning programming is one of our first phases in the learning process. The objective for this group of more advanced students is to develop a game that can interpret the orders that we are going to invent.

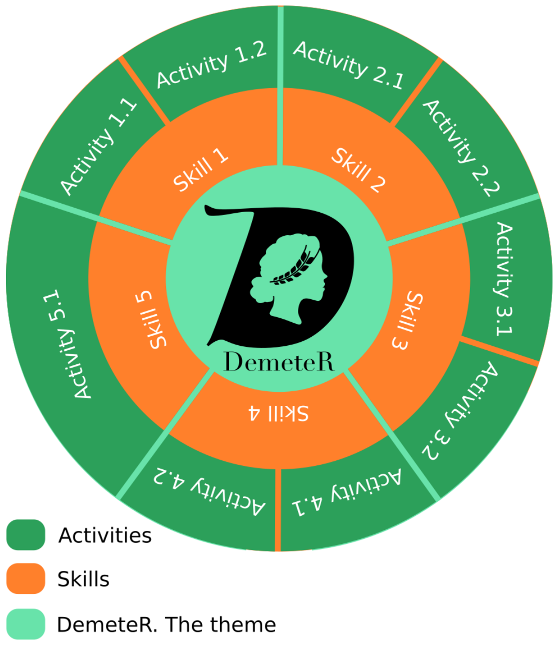


Figure 1.2. How we integrate skills and activities in thematic learning.

The unit is going to be divided into two distinct lessons (products):
Lesson 1. Invention of a pseudo-language and bases of the game (problem)
Lesson 2. Creation of a web game interpreter

Skills/ how is it monitored/ indicators

As we have different classes with different objectives you we have to keep track of them at different levels. Some objectives / skills that we try to promote among our students are:

Spain. Class 1º SMR

- Learning simple orders. Sequential orders (TR, TL or A).
- Understanding and use of loops. (While and For).
- Basic resolution of programming problems.
- Abstract thinking. The student will try to be able to create solutions to different scenarios (exercises).
- Cooperation and group work (even with foreign people).

Spain. Class DAW

- Abstract thinking. The student will try have be able to create solutions to different scenarios (exercises).
- Web programming.
- Originality and simplicity in problem solving.
- Working on complex projects.
- Cooperation and group work (even with foreign people - different language, different location, etc.).

How will it be measured or what indicators are we going to use?

- **For example through tests**. In the case of the class of the 1st SMR class, tests have been carried out to measure the degree of compression of the DEMETER programming language by proposing the student to solve different scenarios.

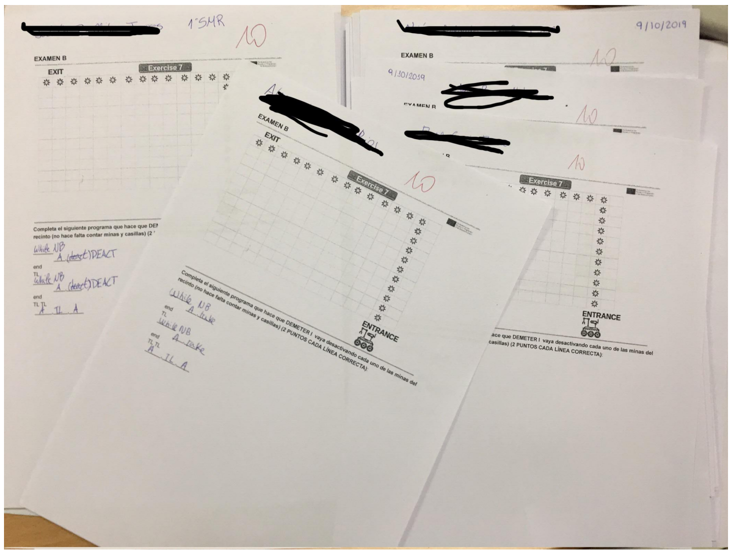


Figure 1.3. Results of the first DEMETER I programming test

The results have been extremely satisfactory. In the first test carried out on the students in which a scenario was explained and the students (27 students) had to solve the problem using the DEMETER I language, the average mark of the test was 8.66 points out of ten.

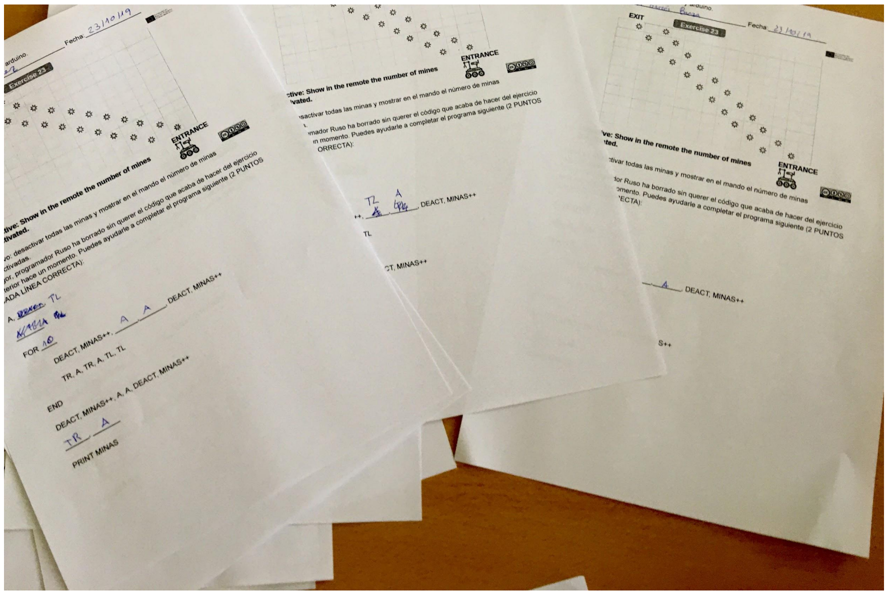


Figure 1.4. Results of the second DEMETER programming language test.

In the second test in the same class, the average mark of the class was 7,11 out of 10. These marks are quite good considering that we have some students with learning difficulties due to different factors.

- **Through the work done (a project)**. A rubric will be used.

Below are the points of the rubric (requirements) that will be used in the DEMETER web interpreter project:

- Aesthetic appearance of the final result.
- Easy to use. The website will have to be simple and intuitive.
- **HTML and JavaScript code will be in separate files.**
- **Listener are used to have the most independent code yet.**
- **The web application analyzes the code lexically and specifies whether there is a lexical error on any line** (for example, if I write whiel instead of while).
- **The web application analyzes the code syntactically and specifies whether there is a lexical error on any line** (for example, if there are while without end).
- **Sequential orders are correctly interpreted (TR, TL, A)**
- **Iterative orders are interpreted correctly (FOR)**
- **Iterative orders are correctly interpreted (WHILE)**
- The interpreter works when it finds nested loops.
- **The executor moves the robot correctly.**
- The interpreter generates the final code taking into account the scenario presented.

- **The interpreter works for different scenarios.**
- **The interpreter knows how to deal with blockages (black circles).**
- **The interpreter can deactivate mines.**
- When a mine is deactivated, the drawing of the mined area changes and shows the mine as deactivated.
- Code clarity
- Refactoring
- The interpreter works with the DEMETER II language.
- The rover moves smoothly (every xxx milliseconds one movement).
- Responsive design using Bootstrap 4.

The lines in bold means that the requirement is mandatory to consider the project as passed.

Lesson 1. Invention of a pseudo-language and bases of the game (problem)

Basis of the problem

The game consists of a bounded area called minefield which can have both mines and obstacles. This area is represented by a grid. The purpose of the rover (DEMETER) will be to enter into the grid to clear the field of mines avoiding obstacles and get out through the delimited exit. The student will have the solution visible in order to create a program in DEMETER pseudo-language that makes the rover fulfill its mission efficiently.

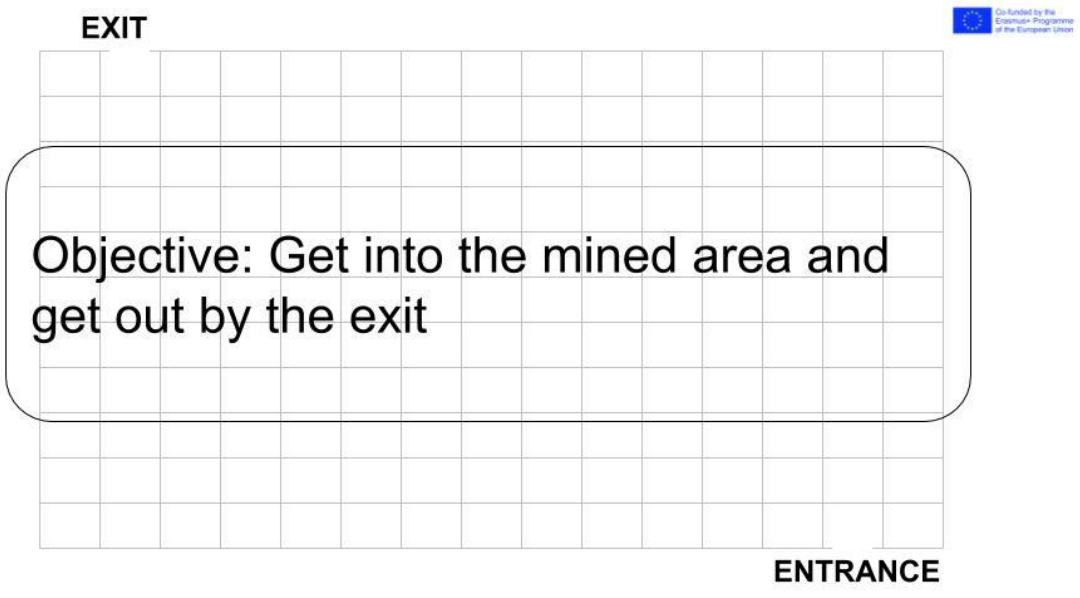


Figure 1.5. An example of the objective for the rover DEMETER.

Pseudolenguaje

Invention of an easy pseudo-language to be able to move a virtual rover which can deactivate the mines contained in a minefield and avoid possible obstacles (trees, stones, etc.).

The teacher will be responsible for giving students the problem to be addressed and together through correctly targeted brainstorming techniques they will have to generate a pseudo-language.

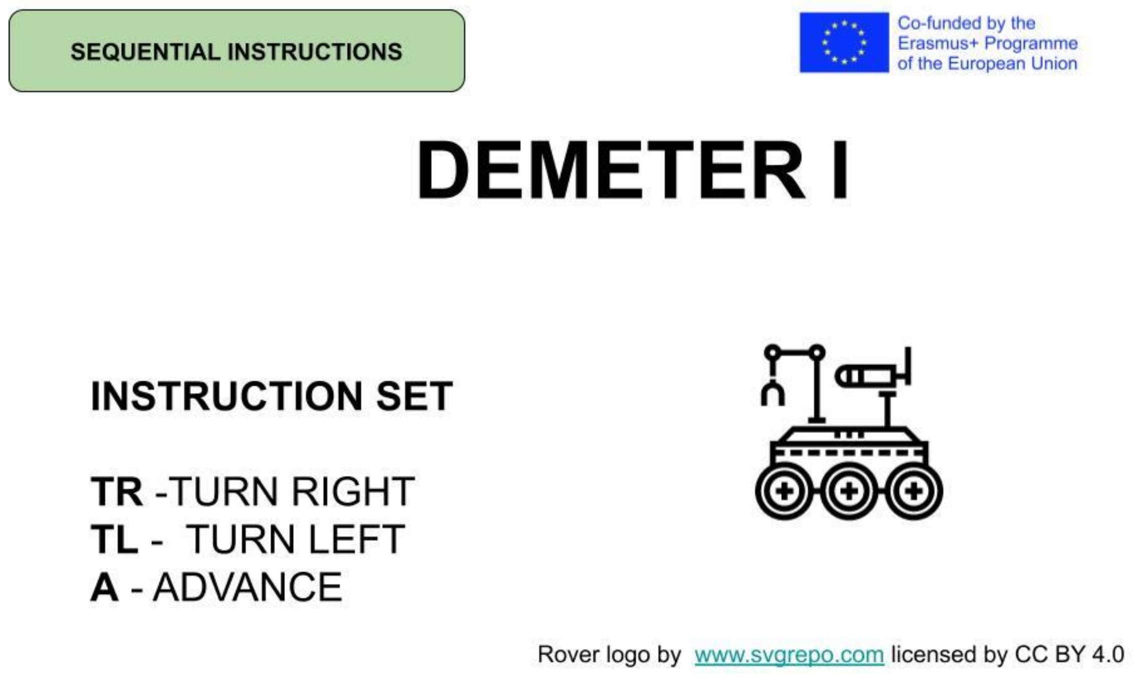
The rules to be used and the time in which a result should be obtained will be specified.

Outcomes:

- Pseudo-language.

The results of this pseudo-language will be shown below. The language has been divided into two parts, DEMETER I and DEMETER II. The DEMETER I language will have simple commands while the DEMETER II language will have more complex orders.

  - **Pseudo-language DEMETER I**


SEQUENTIAL INSTRUCTIONS

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

DEMETER I

INSTRUCTION SET

TR -TURN RIGHT
TL - TURN LEFT
A - ADVANCE

Rover logo by www.svgrepo.com licensed by CC BY 4.0

Figure 1.6. Example of DEMETER I programming language.

Instruction TR (sequential instruction)
TR -TURN RIGHT
The rover will turn right

Instruction TL (sequential instruction)
TL - TURN LEFT
The rover will turn left

Instruction A (sequential instruction)
A - ADVANCE
The rover will advance one square

Instruction WHILE (iterative instruction)

```
WHILE NB
        Y
END
```

DEMETER rover will do Y while there is nothing blocking it.

Instruction FOR (iterative instruction)
FOR X

```
        Y
END
```

DEMETER rover will do Y X times, where X is a number.

A black circle. Is an obstacle (tree, stone,...). DEMETER will have to avoid it.

Instruction DEACT (sequential instruction)
DEACT - The rover will deactivate a mine in the same square where the robot is.

- **Pseudolenguaje DEMETER II**

Use of variables. A variable is a register or memory area where values can be stored. In the case of DEMETER, only integer numbers will be stored.

Below is an example of the use of variables:

```
VAR = VALUE
VAR++
VAR -
PRINT VAR
```

Print X - displays in the remote the number X.

Other examples of use:

```
STEPS = 0
STEPS ++
STEPS ++
PRINT STEPS
```

The former code will display 2 in the remote control of the rover.

Conditional sentences. Conditional statements allow the programmer to have the rover do a thing or not depending on a condition previously evaluated. For this, the keyword IF will be used. Below is an example of that statement:

```
IF MINE THEN
        X
```

```
END IF
```

The forme code will ask if in the current square there is a mine.

```
IF MINE NEXT THEN
    X
END IF
```

The former code will ask if in the next square (in front of the rover) there is a mine.

```
IF BLOCK THEN
    X
END IF
```

Ask if in the next square there is something that will block the rover.

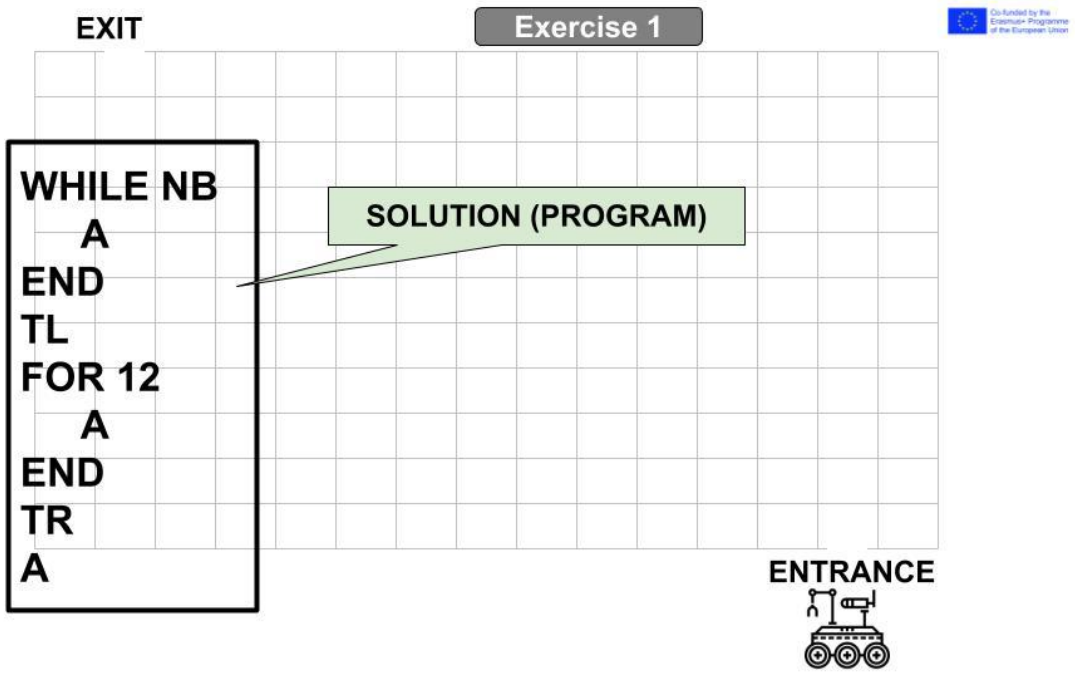
Other loop sentences:

```
WHILE NW
    X
END WHILE
```

While the next square is not a wall (border of the minefield), execute instruction/s x.

Lesson 2. Creation of exercises using our DEMETER Language.

Below are some examples of exercises created for students to practice the language DEMETER I and DEMETER II.



```
WHILE NB
    A
END
TL
FOR 12
    A
END
TR
A
```

Figure 1.7. First exercise

In the previous exercise a possible solution to it is shown. The objective of the exercise is to make the rover enter the mined area and exit through the exit. In this exercise we can see how the DEMETER I language is used to solve the problem.

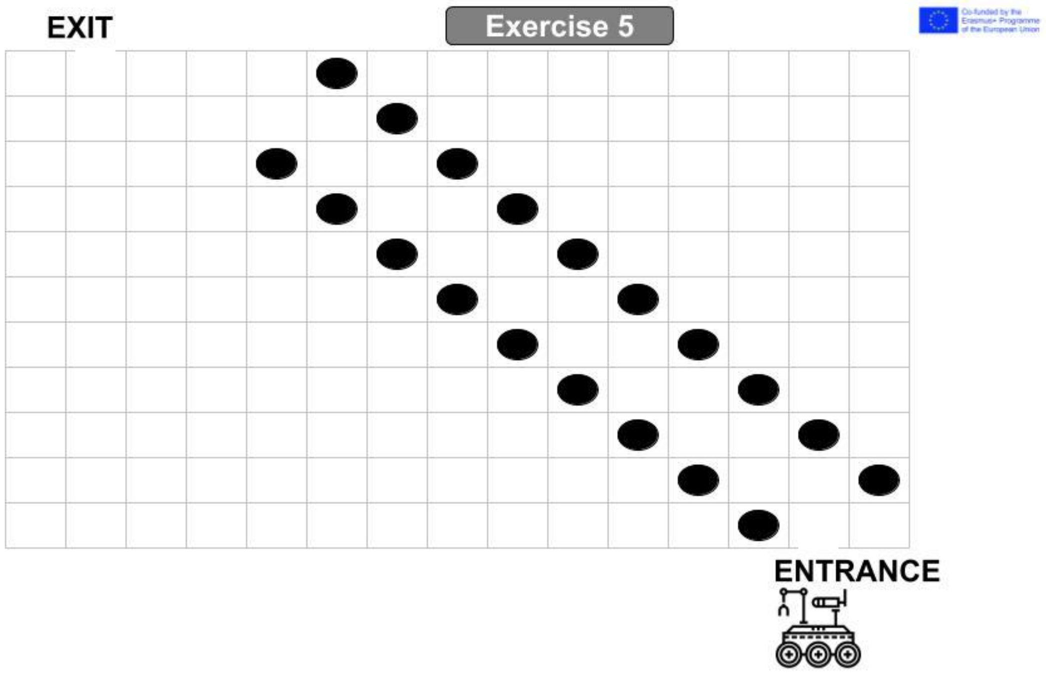


Figure 1.8. Second exercise

In this second exercise, the student will have to enter the minefield and avoid obstacles in order to find the exit.

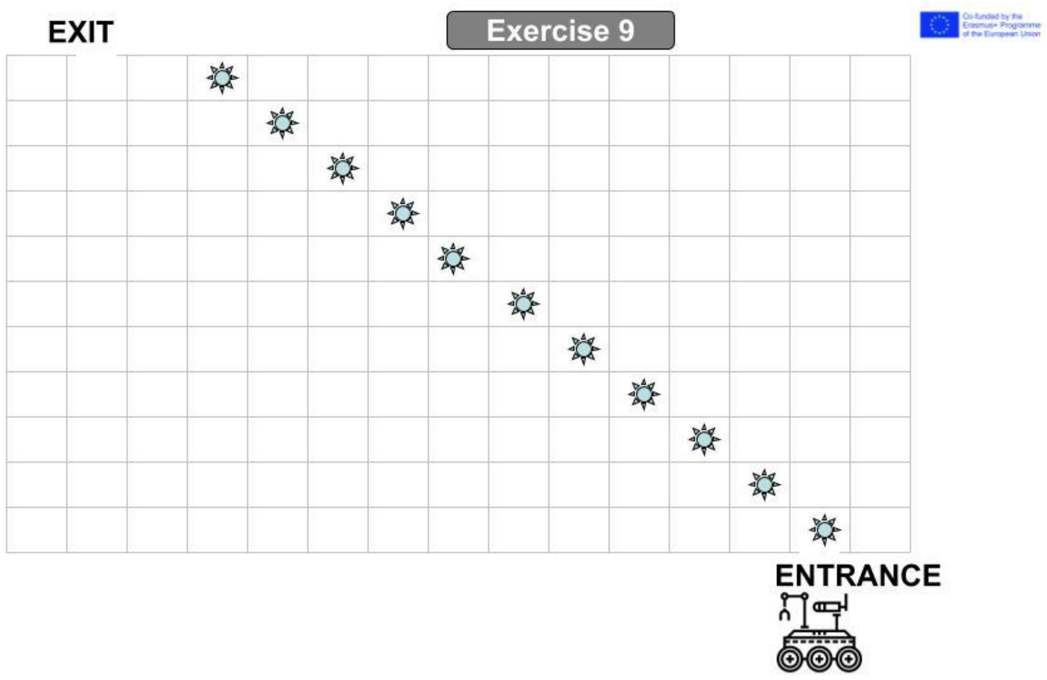
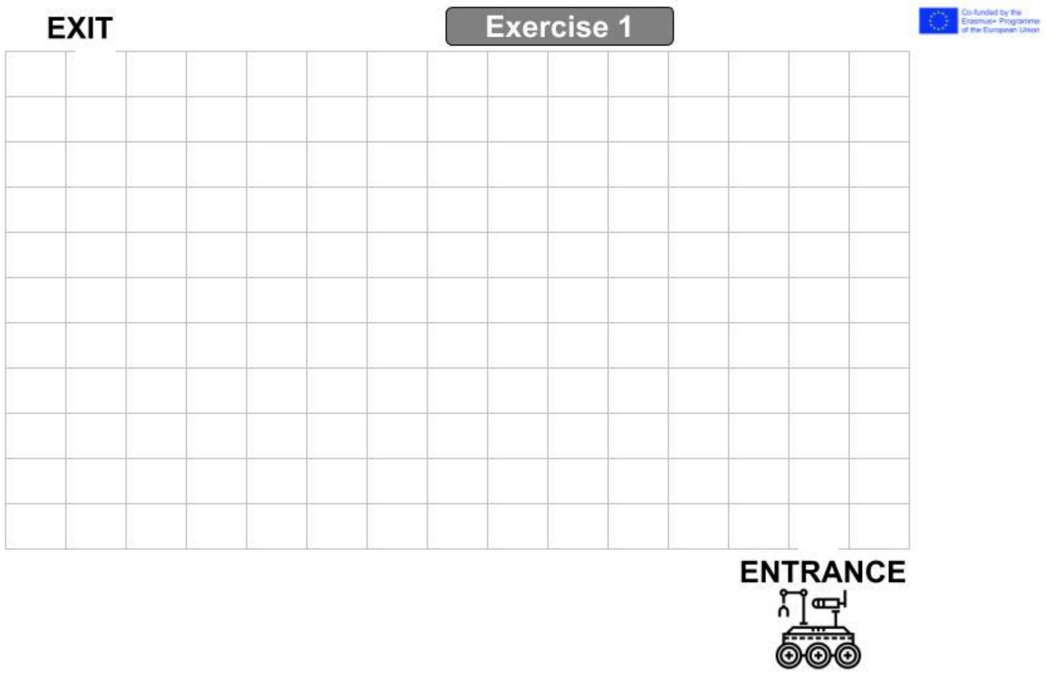


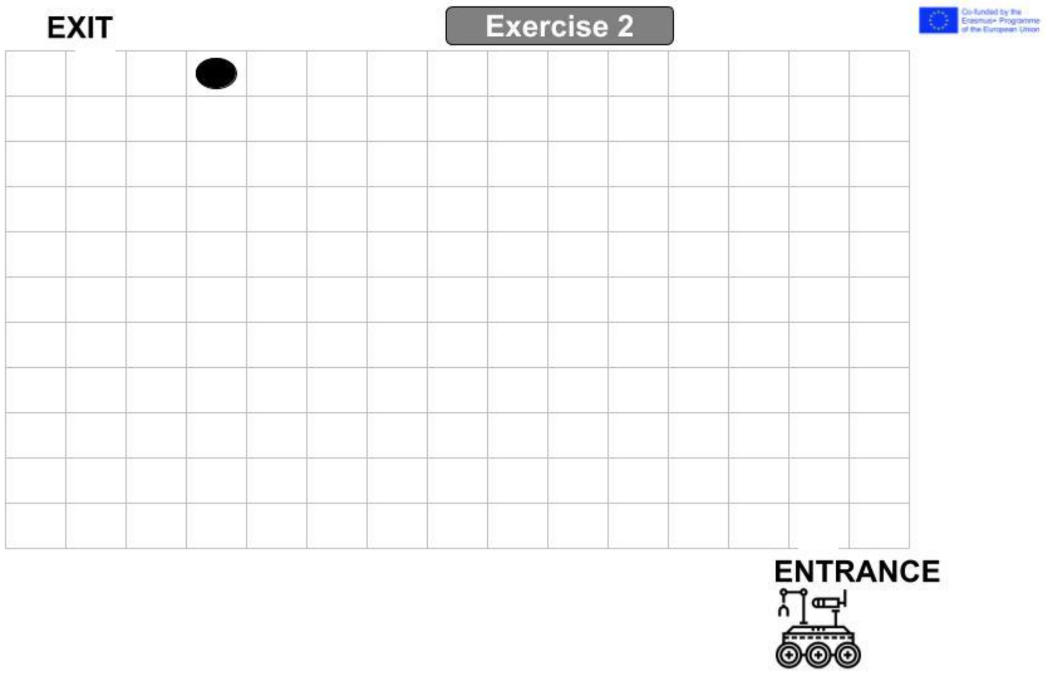
Figure 1.9. Third exercise

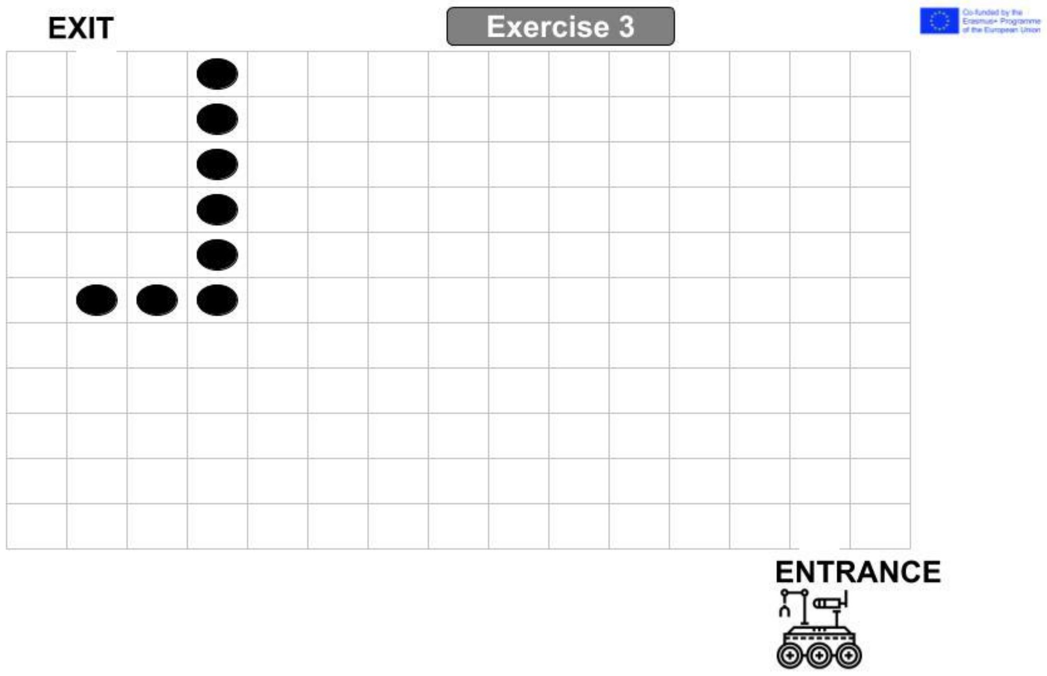
In this third exercise the rover will have to go through the precise squares and deactivating the mines that it finds in its path. All mines in the minefield must be deactivated to declare the area cleared of mines.

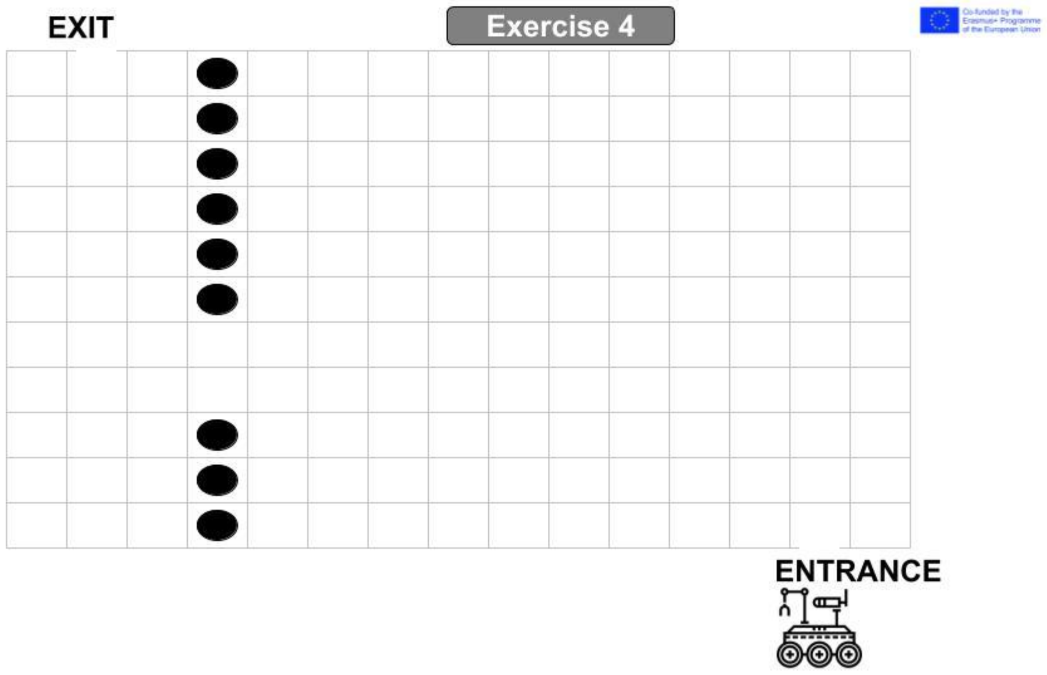
Result of this lesson. Exercises

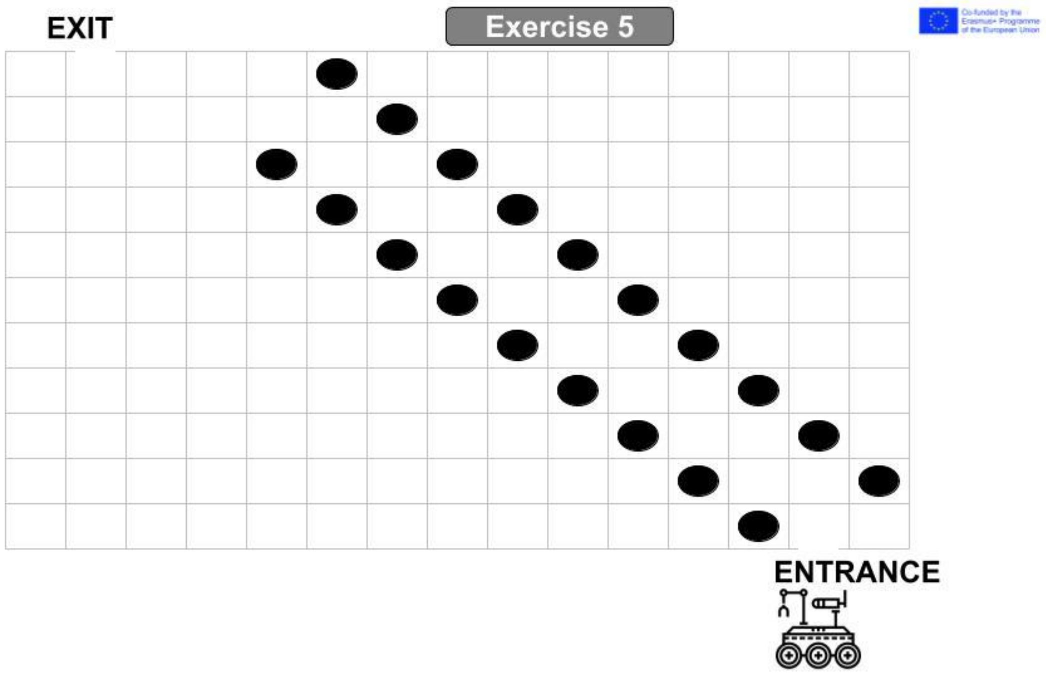
Below is a series of exercises proposed to be coded with the language DEMETER I and DEMETER II.
The exercises have been carried out collaboratively between the students and the teacher. The objective of the exercises is to use all the sentences of the DEMETER language in a gradual way. The first exercises are simpler while the final exercises are more complex.

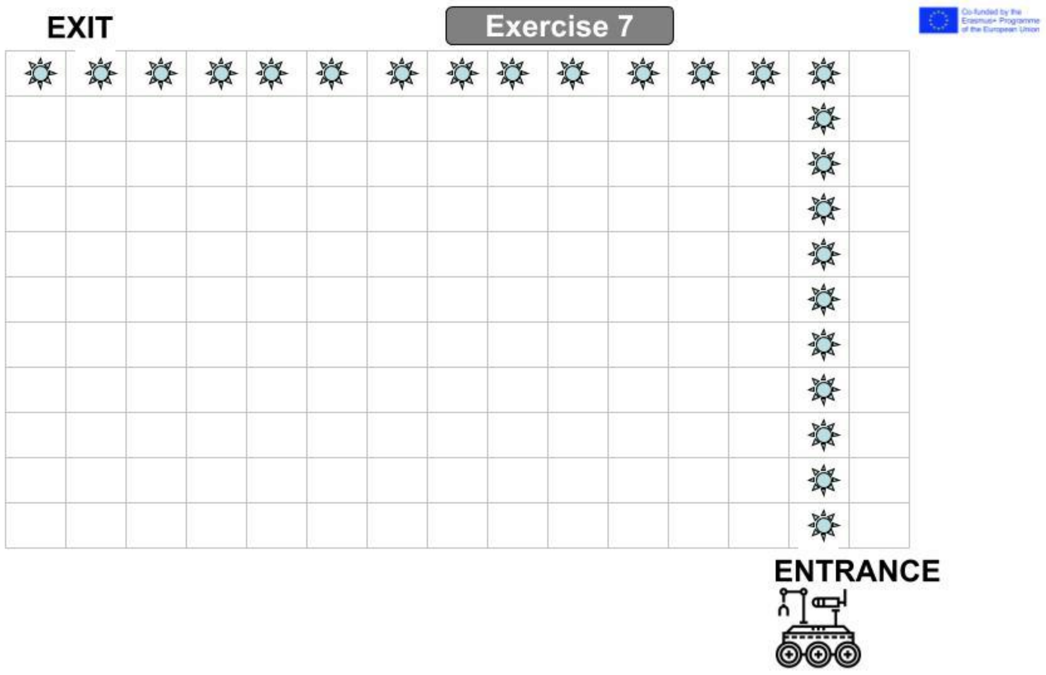

EXIT
Exercise 1
Co-funded by the Erasmus+ Programme of the European Union
ENTRANCE

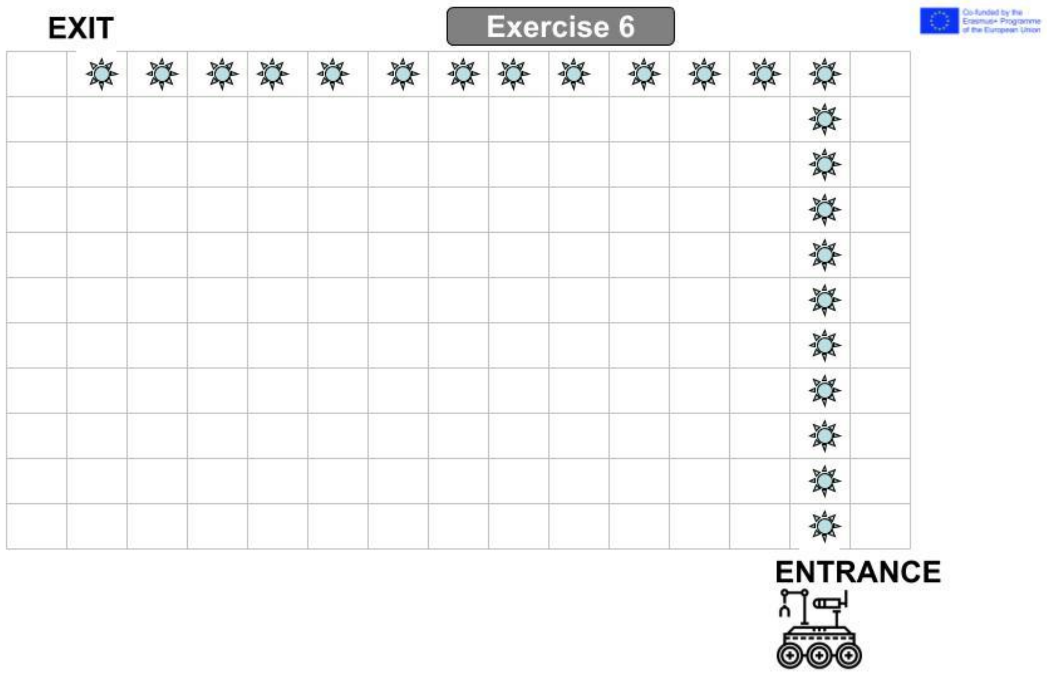

EXIT
Exercise 2
Co-funded by the Erasmus+ Programme of the European Union
ENTRANCE

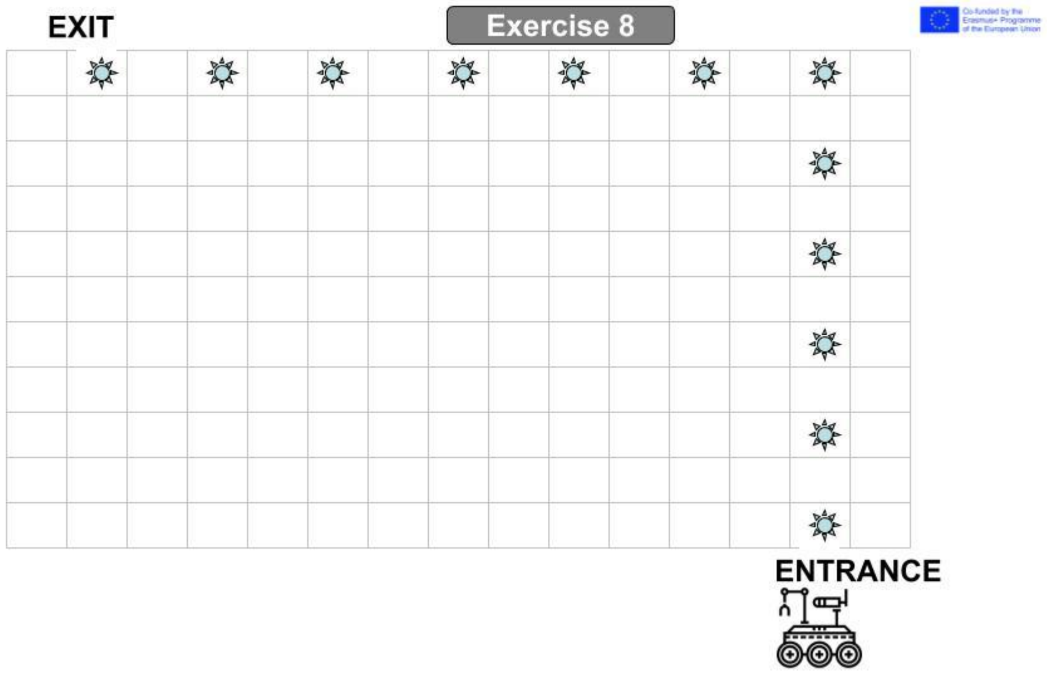

EXIT
Exercise 3
ENTRANCE

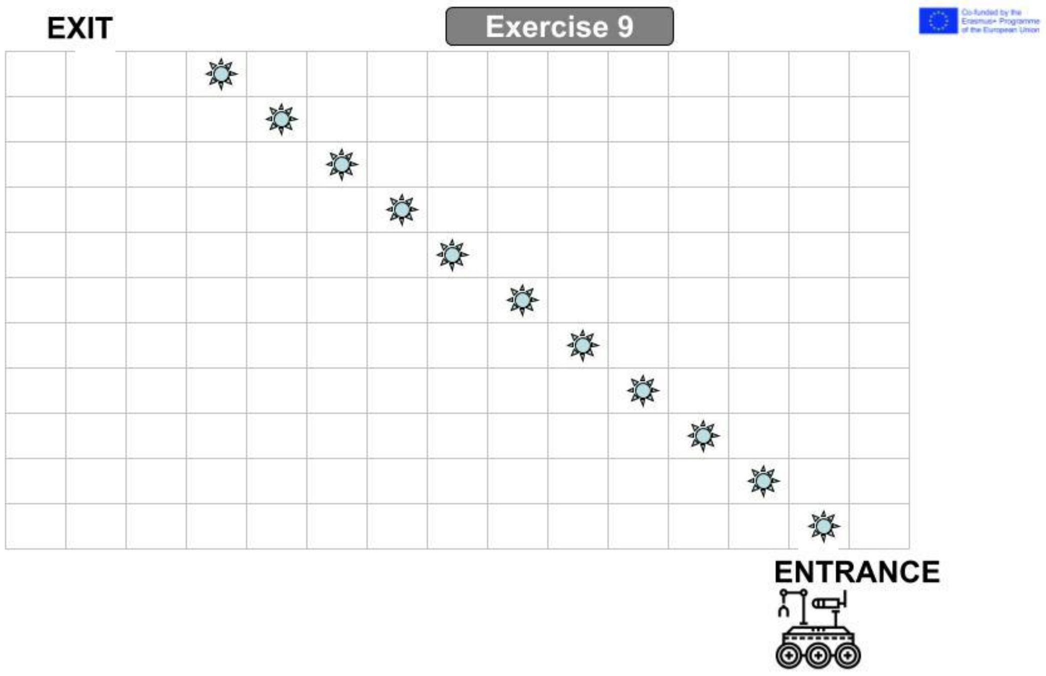

EXIT
Exercise 4
ENTRANCE

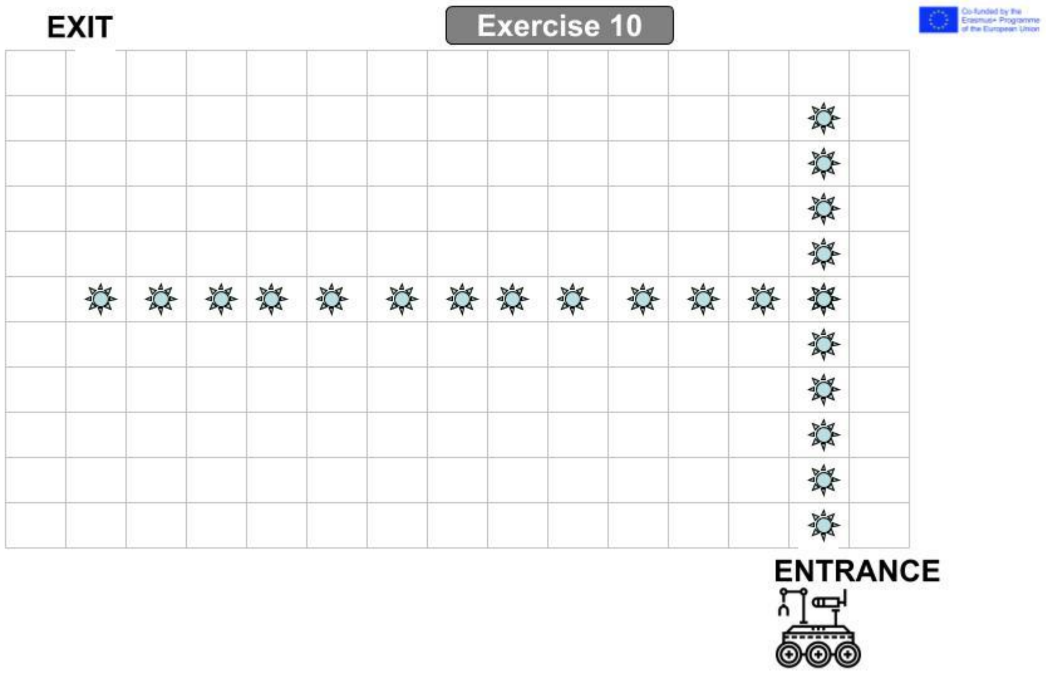

EXIT
Exercise 5
ENTRANCE

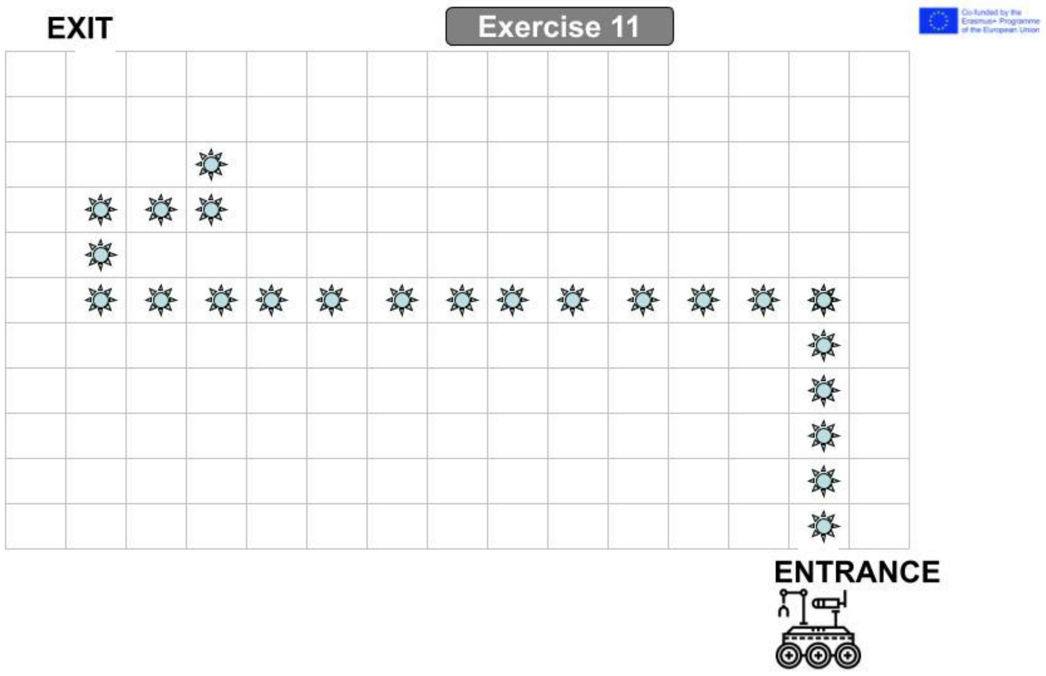

EXIT
Exercise 7
ENTRANCE

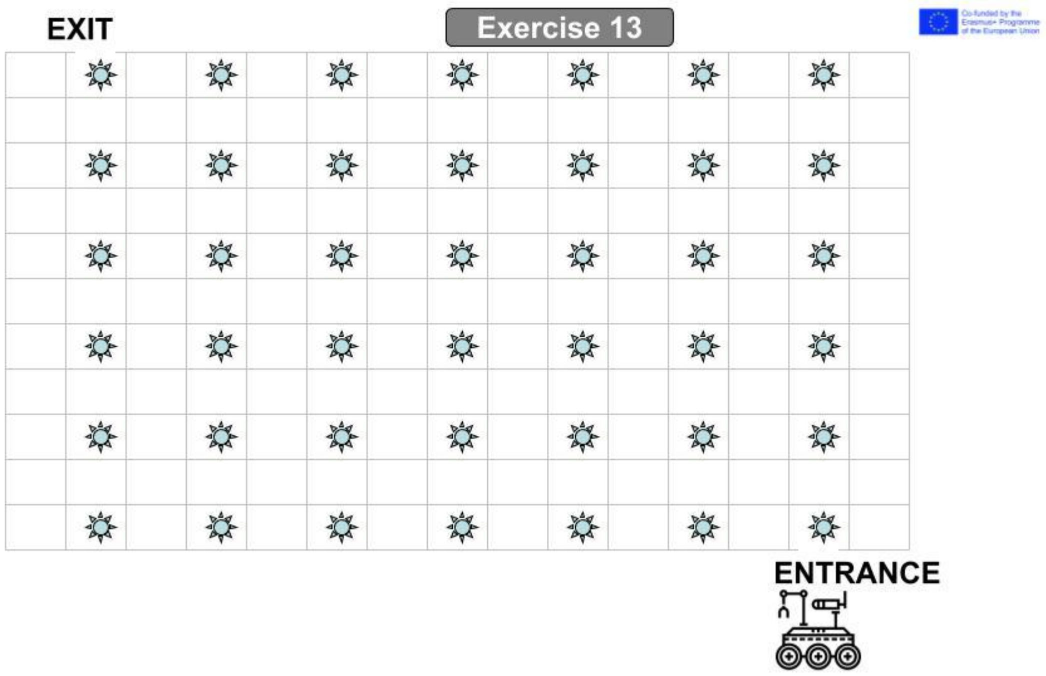

EXIT
Exercise 6
ENTRANCE

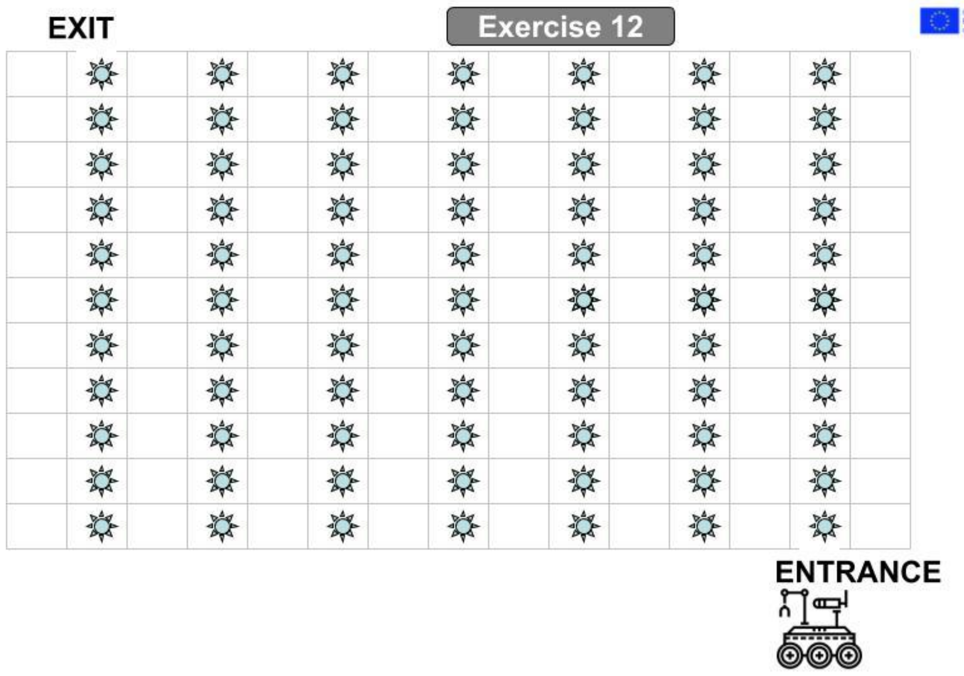

EXIT
Exercise 8
ENTRANCE

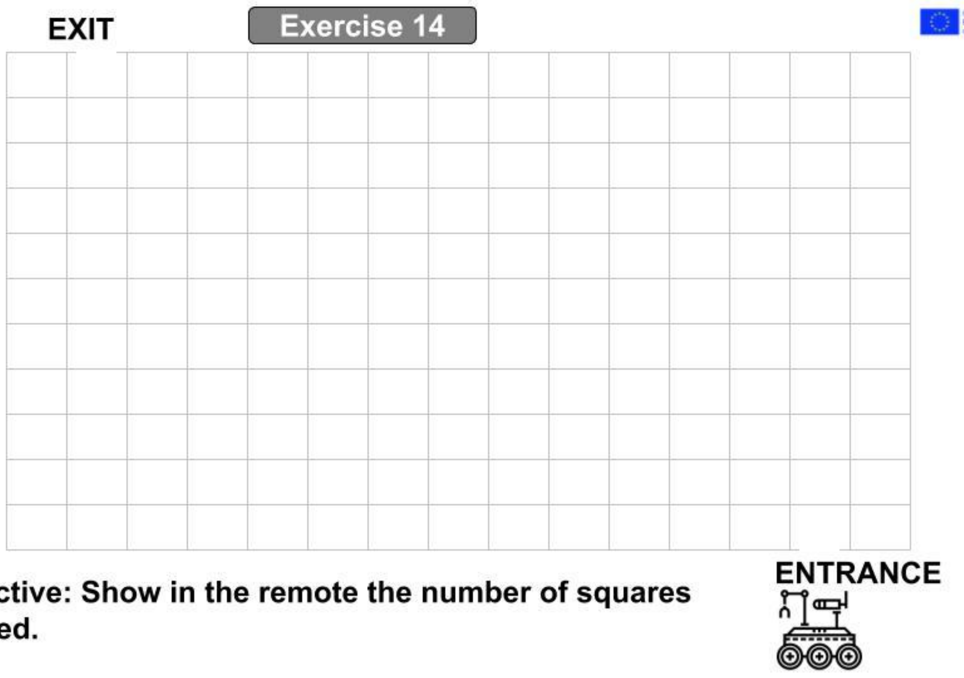

EXIT
Exercise 9
Co-funded by the Erasmus+ Programme of the European Union
ENTRANCE


EXIT
Exercise 10
Co-funded by the Erasmus+ Programme of the European Union
ENTRANCE


EXIT
Exercise 11
Co-funded by the Erasmus+ Programme of the European Union
ENTRANCE

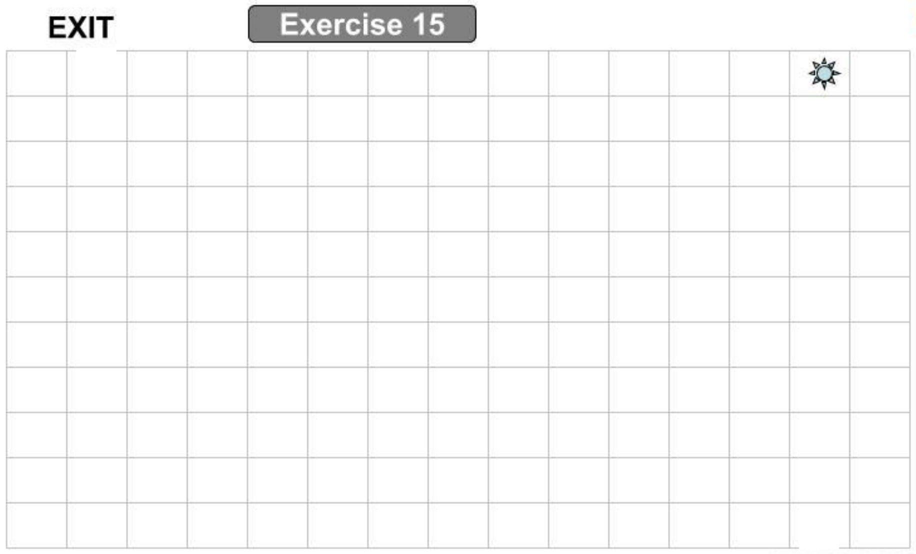

EXIT
Exercise 13
Co-funded by the Erasmus+ Programme of the European Union
ENTRANCE





Objective: Show in the remote the number of squares passed.

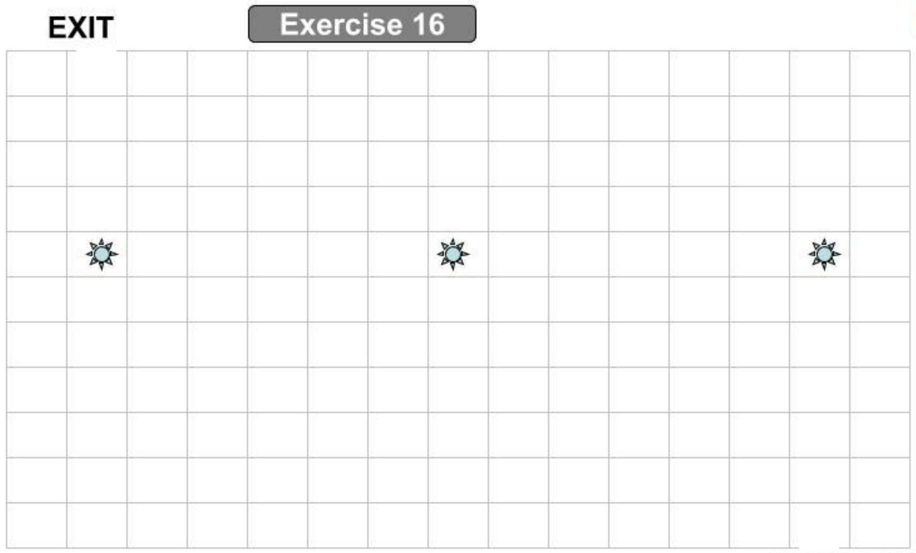
Exercise 15



ENTRANCE

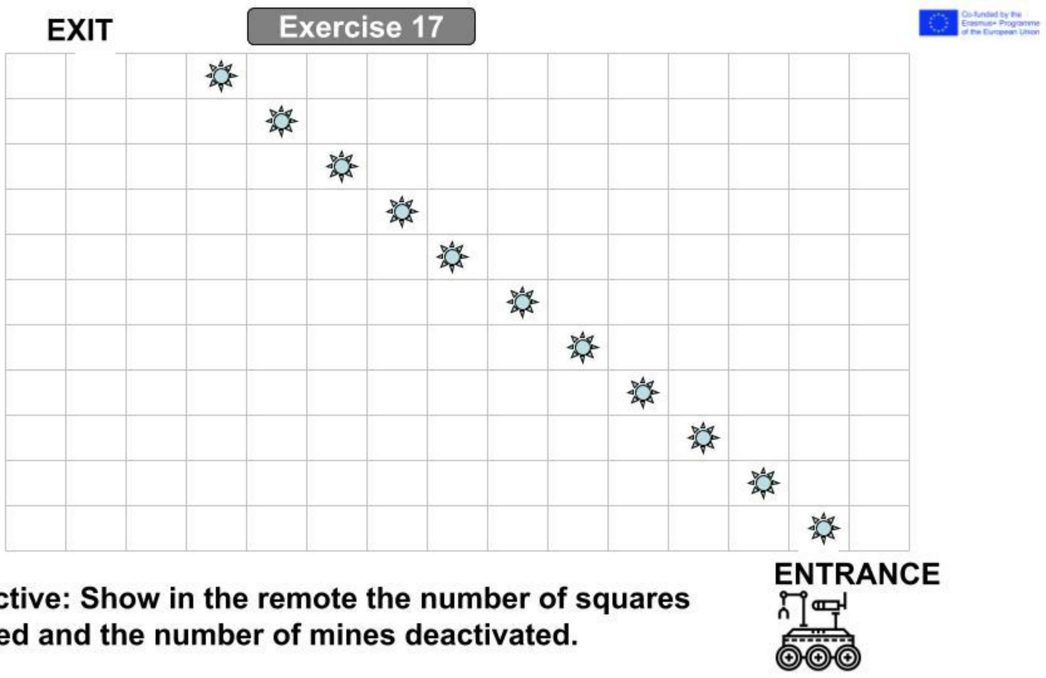
Objective: Show in the remote the number of squares passed.

Exercise 16

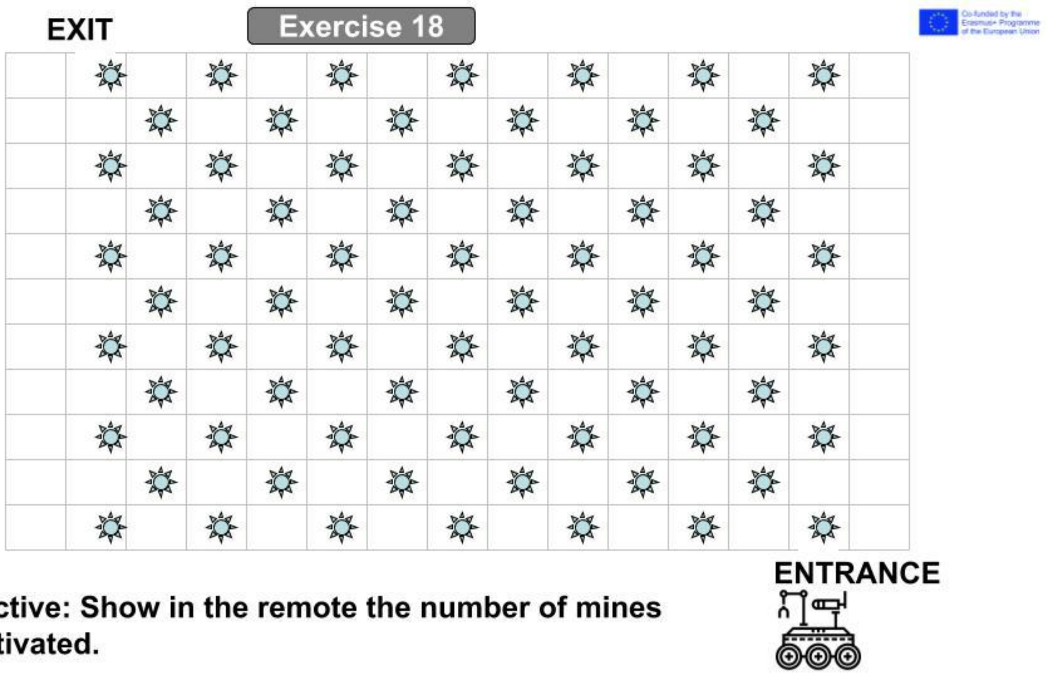


ENTRANCE

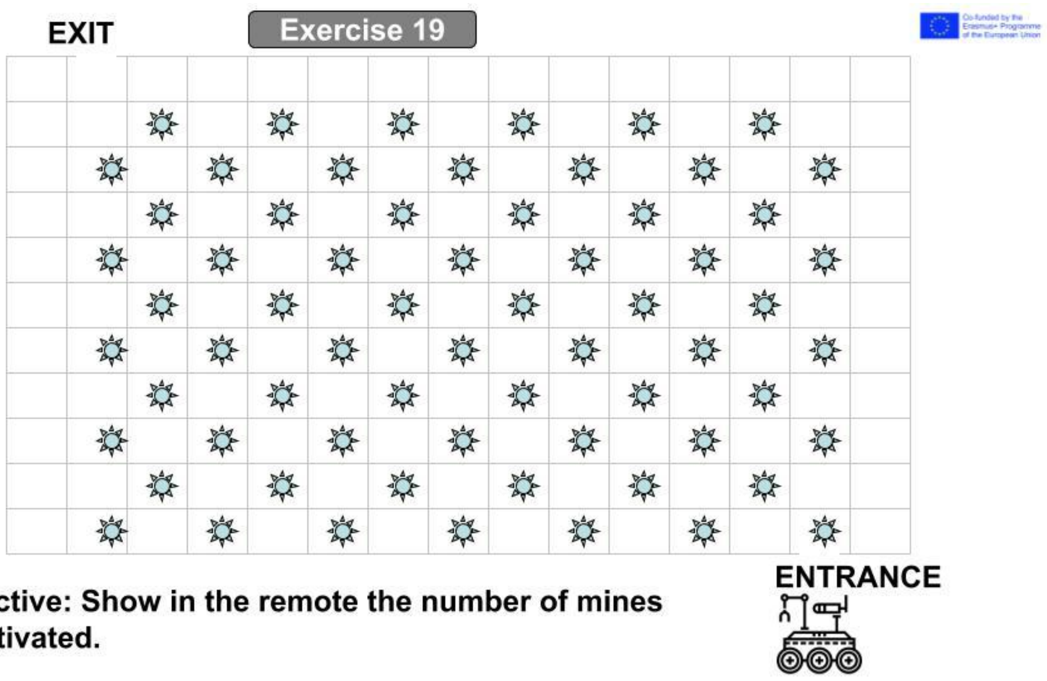
Objective: Show in the remote the number of squares passed.



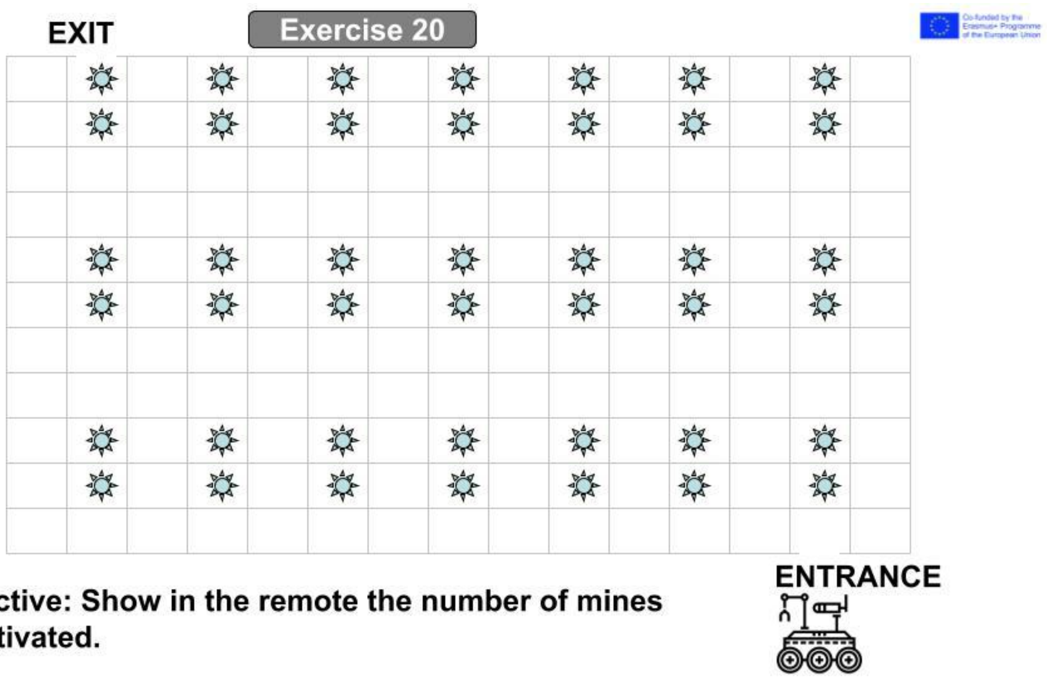
Objective: Show in the remote the number of squares passed and the number of mines deactivated.



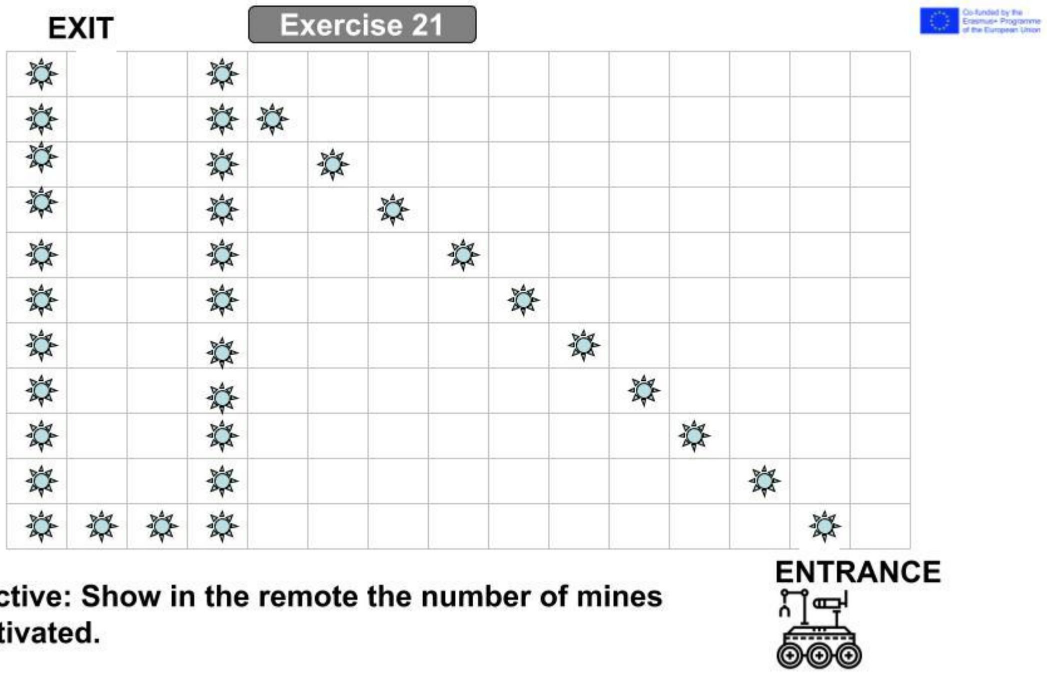
Objective: Show in the remote the number of mines deactivated.



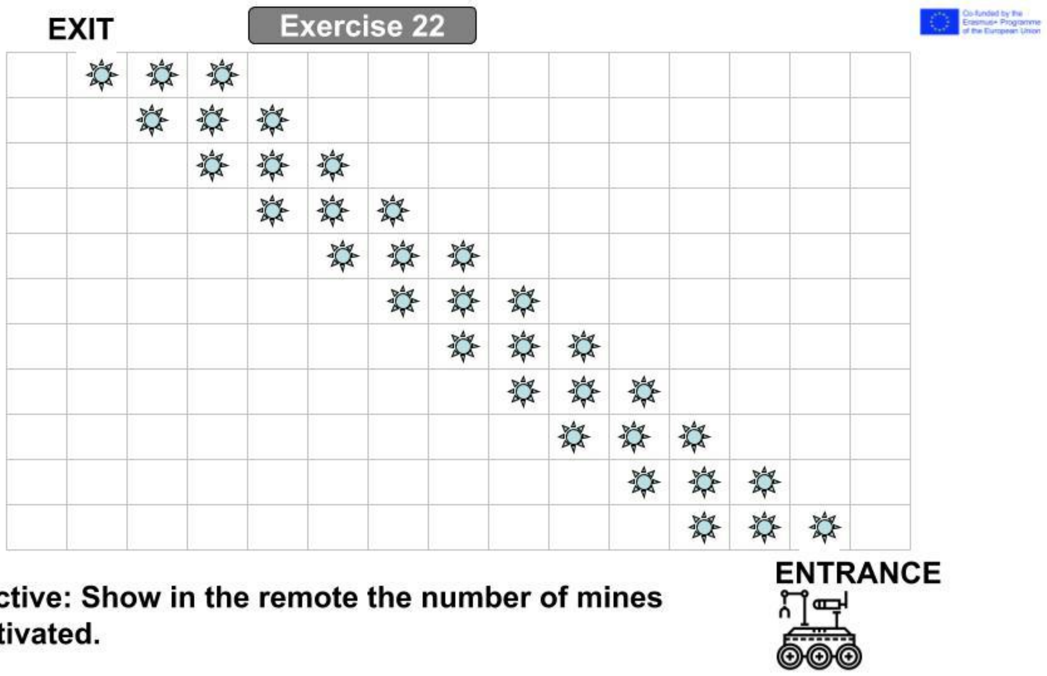
Objective: Show in the remote the number of mines deactivated.



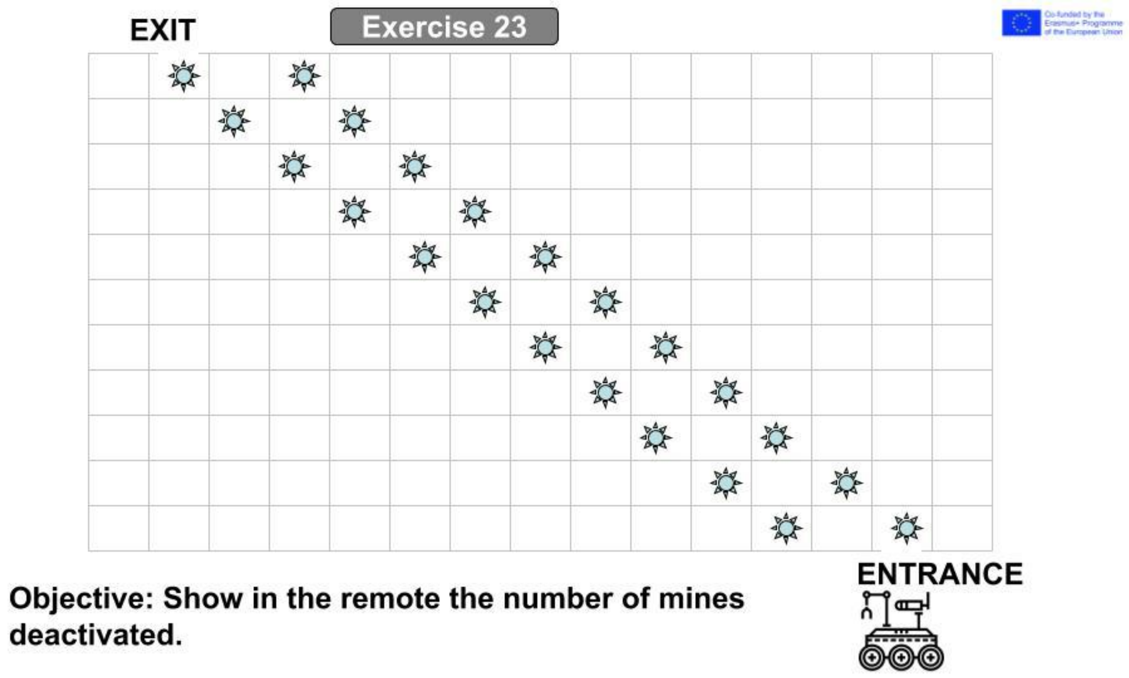
Objective: Show in the remote the number of mines deactivated.



Objective: Show in the remote the number of mines deactivated.



Objective: Show in the remote the number of mines deactivated.


EXIT
Exercise 23
ENTRANCE
Objective: Show in the remote the number of mines deactivated.

Lesson 3. Analysis of the Game using the UML

Students in the first course of web development are encouraged to do an analysis of a hypothetical DEMETER mine detector robot game. The objective of this approach is to create the analysis in an informal way and later on model it in UML (Unified Model Language).

The UML language is used in programming for the analysis and modeling of different computer systems. Due to its flexibility, this language is the most used today and learning it by programmers is essential.

This language is endorsed by the Object Management Group and there are numerous tools that make use of it to specify, build and document any computer system.

The documentation of a computer system is a fundamental part of the same since they will allow to carry out maintenance and production operations of the software. This language allows you to describe how the software is built and how it works.

UML is used to carry out technical documentation of projects. Similar tools and techniques are often used to document the user manual.

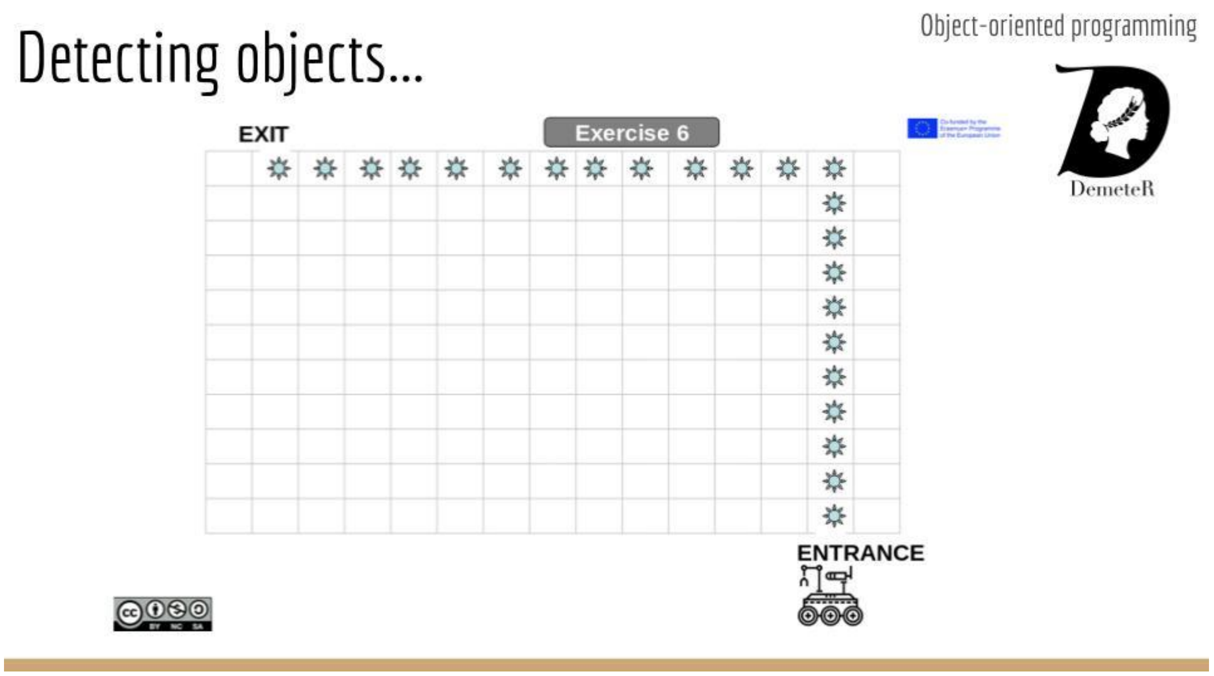


Figure 1.16. Basis of the problem

The bases of the system are explained so that the students, together with the teacher, make a first modeling of it (classes design). The objective of this collaborative exercise is:

1. **Extract the relevant classes from the problem**. Discarding those that are not part of the solution.
2. **State/attributes/properties and characteristics**. Detect the necessary attributes for these classes.
3. **Responsibilities of each class**. Establish a series of methods for each class that can complete the necessary functionality for the entire system.

The following classes have been detected:

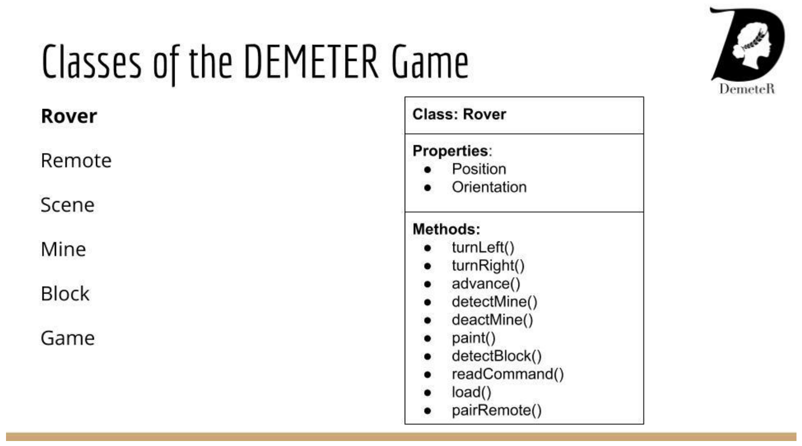
- Rover
- Remote
- Scene
- Mine
- Block
- Game

Attributes identify or rather differentiate one object from another. To detect the attributes we have followed the advice of collecting the nouns that are used to describe the system. Those nouns will be analyzed and we will see if they need to become attributes or do not belong to our domain / system.

In contrast, all the verbs of the system description were analyzed and collected to later analyze whether they correspond to valid methods or not.

The attributes and methods that have appeared in the analysis will be shown below for each class:

Class Rover:



Classes of the DEMETER Game

Rover

Remote

Scene

Mine

Block

Game

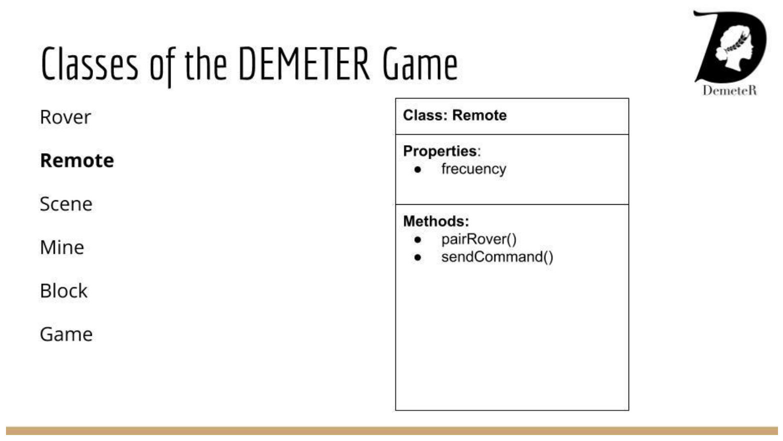
Class: Rover

Properties:
- Position
- Orientation

Methods:
- turnLeft()
- turnRight()
- advance()
- detectMine()
- deactMine()
- paint()
- detectBlock()
- readCommand()
- load()
- pairRemote()

Figure 1.17 Class rover

Class Remote:



Classes of the DEMETER Game

Rover

Remote

Scene

Mine

Block

Game

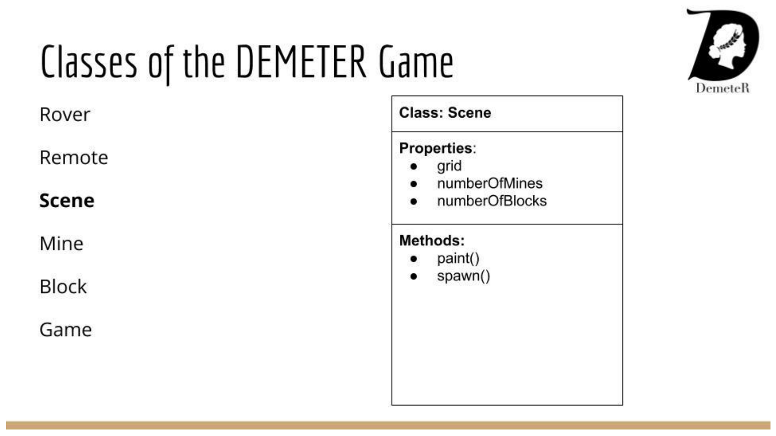
Class: Remote

Properties:
- frecuency

Methods:
- pairRover()
- sendCommand()

Figure 1.18. Class remote

Class Scene:



Classes of the DEMETER Game

Rover

Remote

Scene

Mine

Block

Game

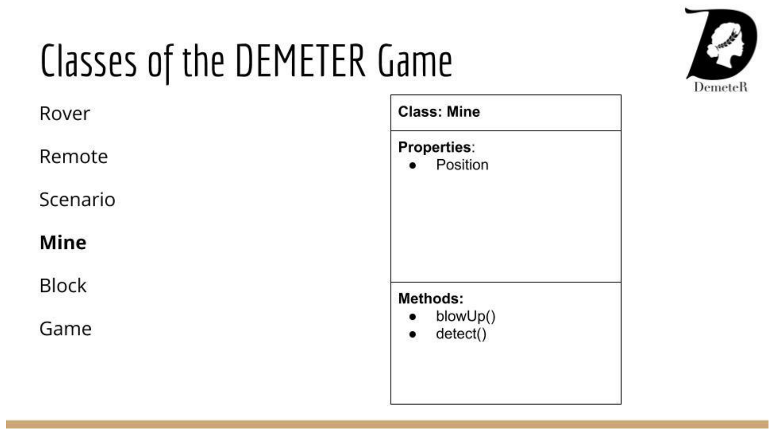
Class: Scene

Properties:
- grid
- numberOfMines
- numberOfBlocks

Methods:
- paint()
- spawn()

Figure 1.19. Class scene

Class Mine:


Classes of the DEMETER Game

Rover

Remote

Scenario

Mine

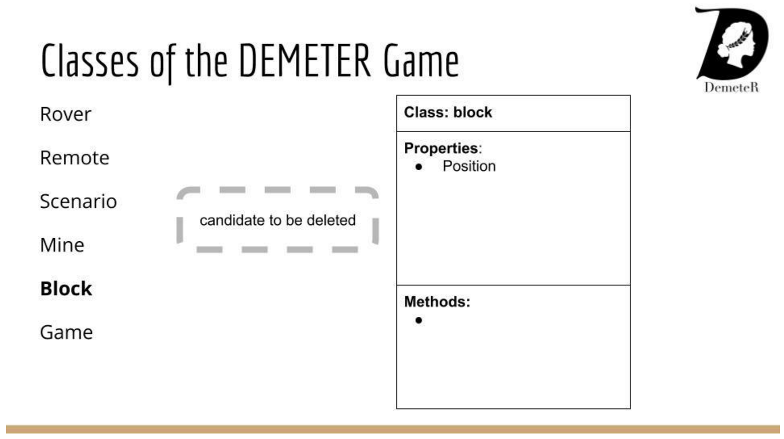
Block

Game

Class: Mine
Properties: • Position
Methods: • blowUp() • detect()

Figure 1.20. Class mine

Class Block:


Classes of the DEMETER Game

Rover

Remote

Scenario

Mine

Block

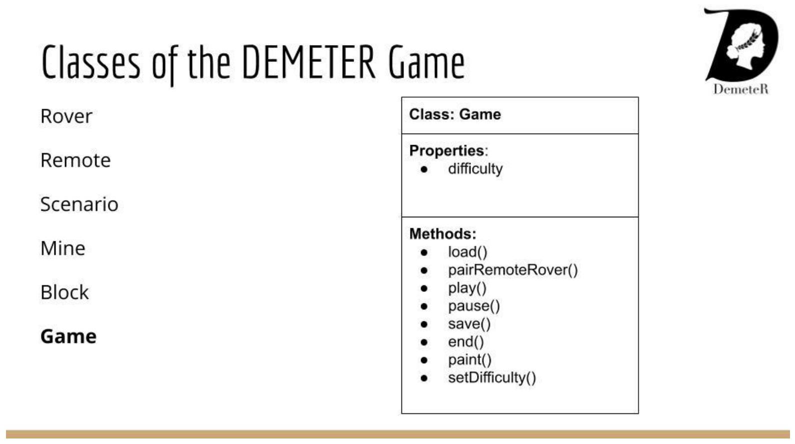
Game

candidate to be deleted

Class: block
Properties: • Position
Methods: •

Figure 1.21. Class block

Class Game:



Classes of the DEMETER Game

Rover

Remote

Scenario

Mine

Block

Game

Class: Game

Properties:
- difficulty

Methods:
- load()
- pairRemoteRover()
- play()
- pause()
- save()
- end()
- paint()
- setDifficulty()

Figure 1.22. Class game

Lesson 4. Creation of a web (DEMETER language) interpreter

Creation of a web game in which students can put this pseudo-language into practice. The system must translate and execute simple orders (sequential orders, loops, etc.).

The teacher must identify the most frequent misconceptions, programming strategy, etc. and redirect students in the analysis and design of the appropriate problem to achieve the necessary results.

The teacher will create prior to the development of the project a document for the development team that will detail the product to be obtained and the characteristics that it must have.

Development process

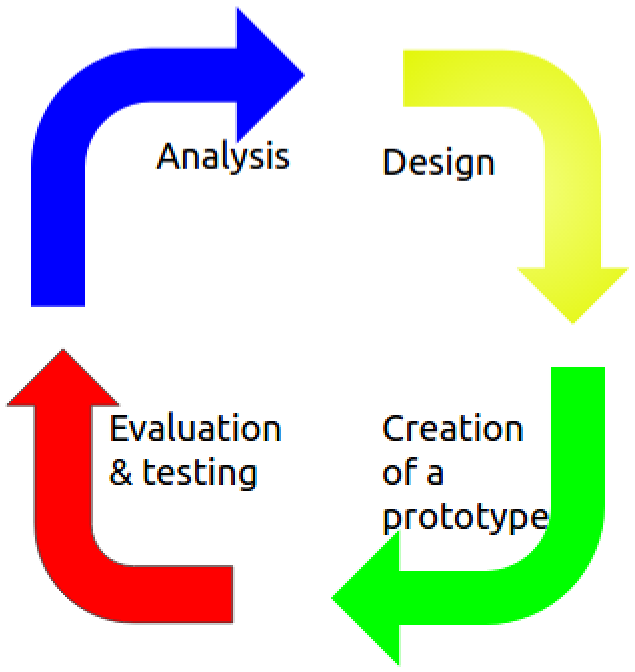


Figure 1.10 Phases of evolutive design process in DEMETER.

An evolutive development model will be followed with at least two prototypes for the creation of the web application. It has been raised in this way because a prototype can be made for the interpreter of the DEMETER I language and then a second prototype so that the interpreter can handle the languages DEMETER I and DEMETER II

The DEMETER II language is more complex and has more sentences with an added difficulty of implementation. That is the reason for using a prototype-based development model.

Phases in which the problem will be addressed.

- **Analysis**

In this phase, the requirements of the game to be created are collected, examined and formulated and any restrictions that may be applied will be analyzed. The outcome will be a document that explains roughly how the application should work.

The products that will be obtained in this phase are:

- **Diagram of use cases in UM**L (Unified Model Language). In this type of behavior diagrams we can model how the user interacts with our game. The interaction is not very intensive but this diagram will help the students to how implementing the web interface.

  - The **main classes** involved in this project will be studied. A first scheme will be created.

  - Analysis of the creation of a **lexical and syntactical analyzer**.

During the compilation two different phases are performed. One in which the DEMETER code is analyzed lexically where tokens or lexemes are detected (reserved words of the language) as well as assignment operations, etc.

During the second phase (syntactic analysis) the grammar correction of the problem will be checked.

  - Análisis de la generación de código intermedio e interpretación.

El código intermedio permitirá al intérprete interpretar y ejecutar cada una de las órdenes de una forma más sencilla, eficiente y rápida. Se podría

- **Design**.

In this phase the architecture of the web application will be defined and the technical details of each of the system modules will be described in depth.

Products that are going to be analyzed in this phase are:

  - **Class Diagram** In which the main classes of the system will be detailed.

  - **Design** of the creation of a lexical and syntactic analyzer. Where the system will be detailed in a way closer to coding.

- **Coding**

In this phase the application will be coded, taking into account the functionality defined during the design phase, in the following languages:

- JavaScript

- HTML5
- CSS3
- JSX
- BootStrap

The React framework will be used following the Flux design pattern. **Flux** is the name given to the observer design pattern which was modified slightly for use in React.

Responsive design using Bootstrap.
The Bootstrap library will allow us to make a responsive front-end which adapts to any device (tablet, smartphone, computer, etc.) and any resolution. This library is based on jQuery and is used by Twitter on its website. One of the reasons to choose this toolkit is basically because it is open source.

- **Testing**

In this phase, tests will be carried out to ensure that the application is programmed according to the specifications described. For these tests, correct and incorrect programs will be created using all the commands described in the DEMETER language.

The German team will be responsible for testing the application. In the case of finding an error, the German team must report it by slack to the Spanish development team which will correct it and notify when the modifications are uploaded to the server.

Web application structure.

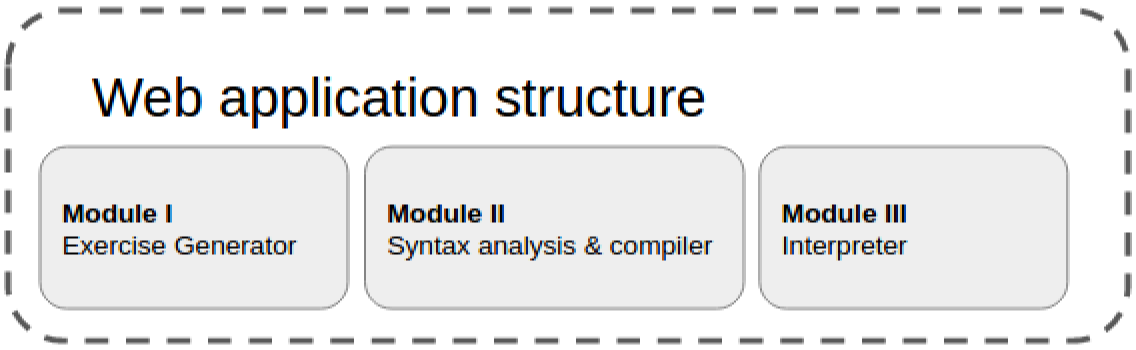


Figure 1.11. Web application structure.

The most efficient way to carry out the project of a game that allows interpreting the DEMETER language is to divide the web application into three modules:

Exercise generator

The first module that will be the exercise generator. It will select sequentially or randomly one of the scenarios or exercises to solve. For this we will need to have the data in a database or JSON server.

Syntactic and lexical analyzer

This second module will analyze the program introduced by the user and verify that it is correct from a syntactic and lexical point of view. In the case of finding an error, the program cannot be executed. The program will also warn of the error or errors found by offering some clues to the programmer in order to correct the error.

Once analyzed the DEMETER program both lexically and syntactically, the module will generate an intermediate language which will be more easily interpreted by the subsequent module "interpreter and executor".

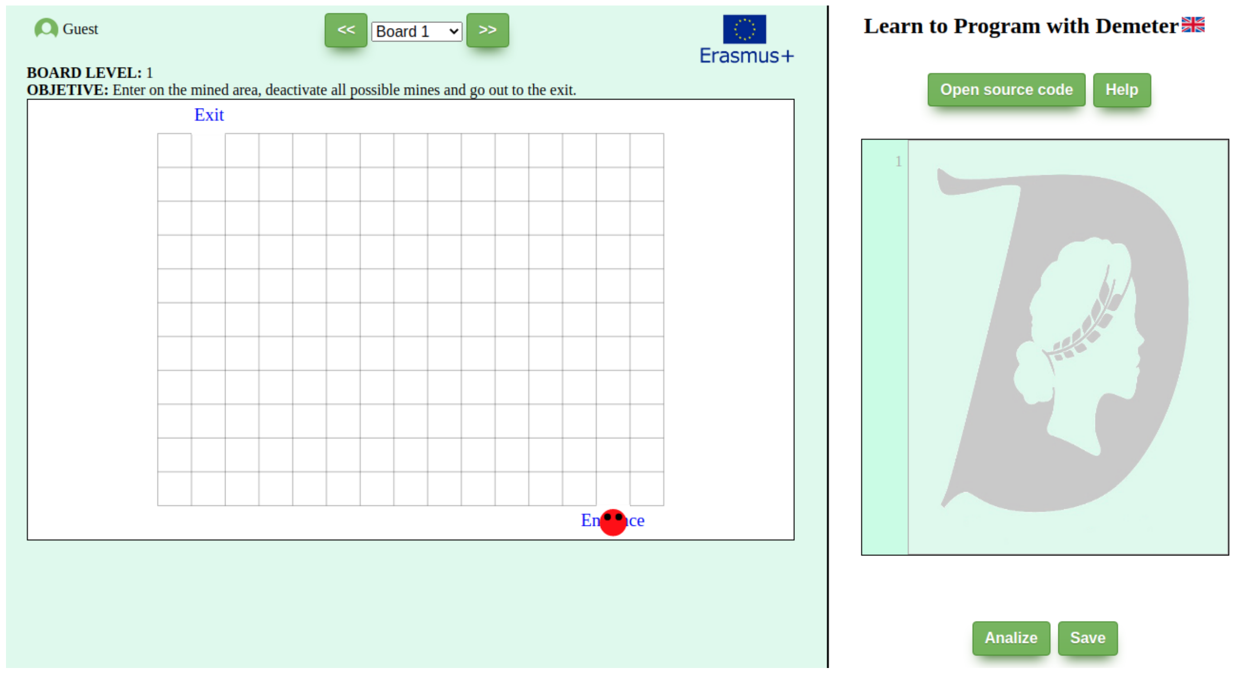
Interpreter and Executor.

Once verified that the program is lexically and syntactically correct and created the intermediate code more easily interpretable by the interpreter, each of the lines of the program will be interpreted, executing the corresponding actions with the robot in the minefield. The robot will basically receive the orders (right, left and forward).

The interpreter and executor must verify that each of the orders of the program can be executed without problems in the field of mines. In the case that the user enters an inappropriate order, the program must notify the user without executing it. For example, if the user generates the order to move forward and that means leaving the minefield.

URL and landing page

URL: http://thematic-learning.com/learn2program/



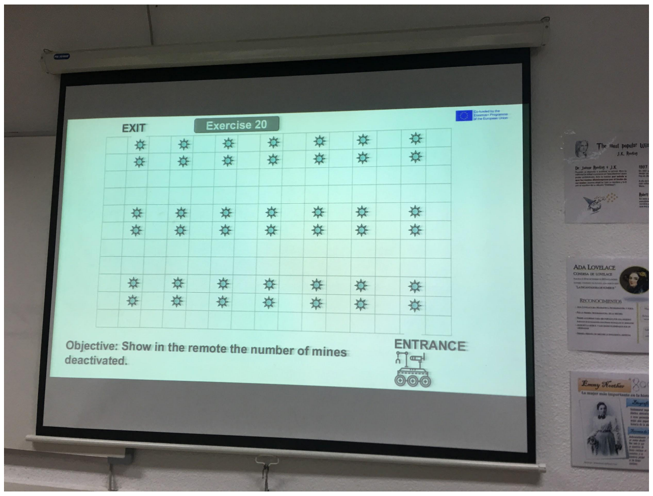
BOARD LEVEL: 1
OBJETIVE: Enter on the mined area, deactivate all possible mines and go out to the exit.

Figure 1.12 Learn2program landing page.

Unit 1. Preparation, resources and material

Material and tools to use:
Product I

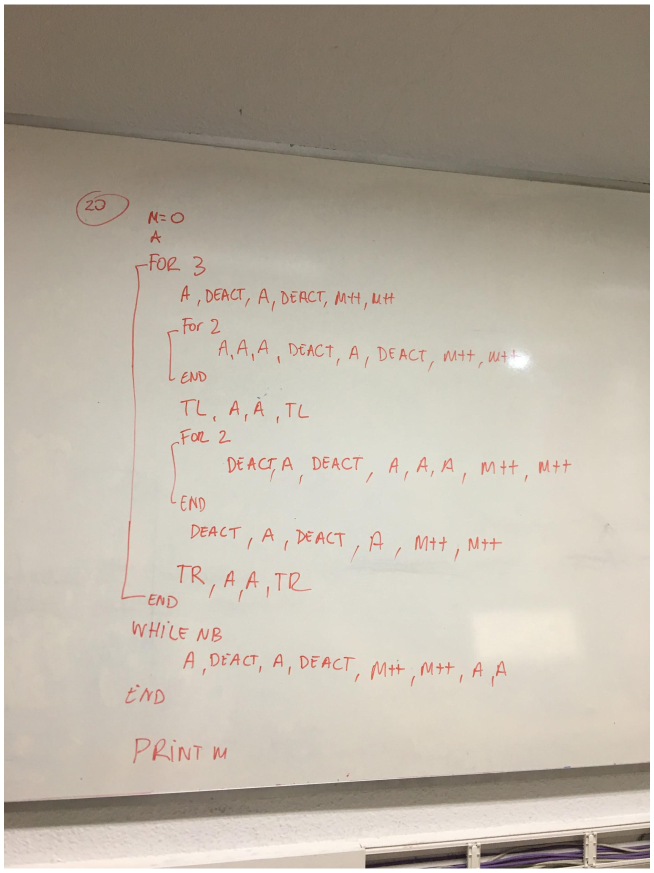
- **Presentation of exercises**. Each slide will show an exercise so that the student can solve it.



Objective: Show in the remote the number of mines deactivated.

Figure 1.14. Example of DEMETER II exercise.

- **Board**. To solve exercises manually.



```
20
N=0
A
FOR 3
    A , DEACT, A, DEACT, M++, M++
    For 2
        A, A, A , DEACT , A , DEACT , M++ , M++
    END
    TL , A, A , TL
    FOR 2
        DEACT, A , DEACT , A, A, A , M++ , M++
    END
    DEACT , A , DEACT , A , M++ , M++
    TR , A, A , TR
END
WHILE NB
    A , DEACT, A, DEACT, M++ , M++ , A , A
END

PRINT M
```

Figure 1.15. Example of DEMETER II program created by one student.

Product II.

- **Presentation of exercises**. Each slide will show an exercise so that the student can solve it.

- **Web server**. The products will be stored in thematic-learning.com, our DEMETER project website.

- **IDE (Integrated Development Environment)**. Atom. Atom will be used because it is a cross-platform tool. It can be used on OS X, Windows, or Linux. Atom is also a tool that complements well with Git.

- **CVS (Control Version System)**. GIT. A version control software will be used to manage the different releases of the program. GIT is chosen because it is free and open source and can maintain distributed versions. In addition GIT is easy to use and very efficient.

Bibliography

- Sánchez, J. (2003). Integración Curricular de TICs Concepto y Modelos. Revista Enfoques Educacionales.
- Ecured.cu. *Modelo de prototipos*. Retrieved from: https://www.ecured.cu/Modelo_de_prototipos. Revised on 20 june 2020.
- UML.org. *Unified Modeling Language*. Retrieved from: http://uml.org/. Revised on 15th of march 2020.
- Reactjs.org. *React A JavaScript library for building user interfaces*. Retrieved from: https://reactjs.org/. Revised on 7th of february 2020.
- Salcescu, C. 2019. *An introduction to the Flux architectural pattern*. Retrieved from: https://www.freecodecamp.org/news/an-introduction-to-the-flux-architectural-pattern-674ea74775c9/. Revised on 12th of april 2020.
- git-scm.com. *GIT--fast-version-control*. Retrieved from: https://git-scm.com/. Revised on 13th of april 2020.
- atom.io. *A hackable text editor for the 21st Century.* Retrieved from: https://atom.io/. . Revised on 18th of february 2020.
- getbootstrap.com. *Build fast, responsive sites with Bootstrap.* Retrieved from: https://getbootstrap.com/. Revised on 17th of january 2020.