Codewards

Hour of Code

Lesson for 40 minutes

Lesson Guide

Hour of code

40 minutes

Lesson aims:	1. Get acquainted with game based approach of the product. 2. Quick introduction to the subject. 3. Get familiar with basic concepts of programming and the ways they are applied to real life.
Terminology:	1. Computer command 2. Objects 3. Algorithm 4. Optimization
Computer activity:	Students will solve the tasks (challenges) on different topics. New tasks will appear automatically by the completion. Commands: `move, rotate, load, put` Arguments: `right, left` Objects: `robot, crane`
Materials required:	Computers (tablets) with access to the Codewards system.

Lesson plan *(Version 1)*

1. Introduction to the story.
2. Solving the tasks + individual explanation of the concepts.

Part 1. Lead - in

Estimated time – 5 minutes

Show the an intro clip of the story and introduce the storyline.

Teacher: *«Today we are on a very important mission! We have to restore the operating system of the underwater station and make it work again. While completing our challenges, we will learn how to use computer commands to control different objects».*

Part 2. Computer practice

Estimated time – 30 minutes

Complete the tasks on the map:
Task 1
Task 2
Task 3
Task 4
Task 5
Task 6
Task 7
Task 8
Task 9

While doing the tasks, give individual explanations to the learners, if needed.

Part 3. Close up

Estimated time – 5 minutes

Teacher: *«Today we have been using computer code to fix the dome and the pipeline of the underwater station. We have done a great job, but many things are still broken. I hope our unit will get together again to accomplish new exciting missions.»*

Lesson plan *(Version 2)*

Part 0. Getting aquatinted

Everyone gets acquainted and introduces themselves.

Part 1. Lead in

Estimated time – 10 minutes

Show the intro clip of the story and introduce the storyline.

Teacher: *«Today we are on a very important mission! We have to restore the operating system of the underwater station and make it work again. While completing our challenges we will learn how to use computer commands to control different objects».*

Part 2. Computer practice

Estimated time – 55 minutes

Teacher: *«Let's open our control panel and see how it works.»*

- **Go to Task 1**

Before doing the task, introduce the system of commands.

Concept №1:

WHO + WHAT + HOW

Explain that the simplest way to control objects is through command system:
WHO + WHAT (we should do) + HOW (how many steps, which way, etc.)

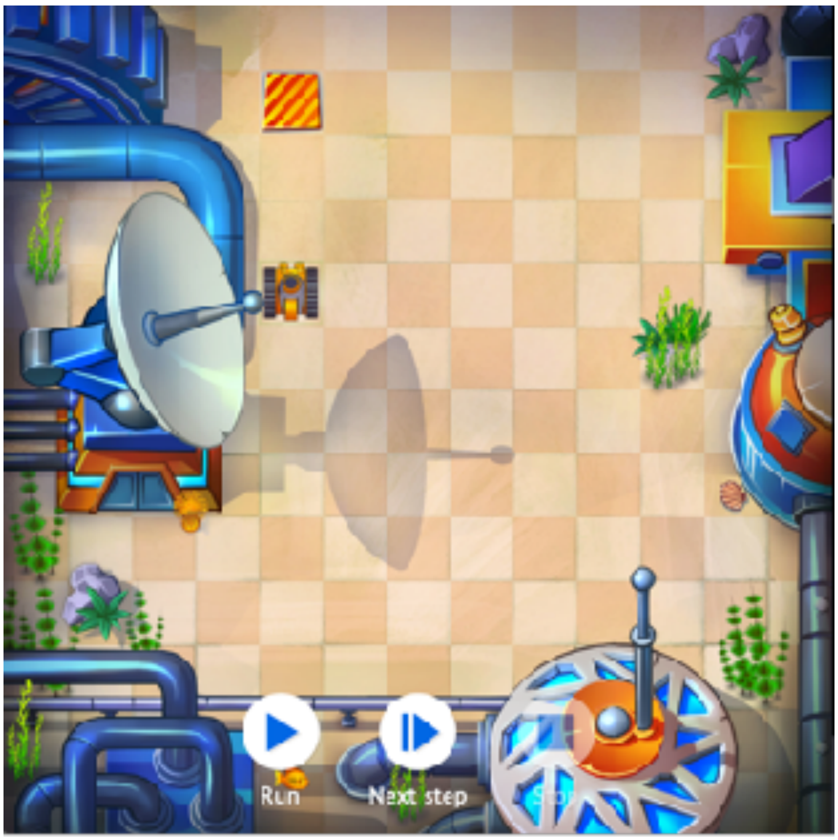
Practice with an action game **«I will program you»**

Version 1. Learners should program their teacher using «WHO + WHAT + HOW» approach. Children call out commands - the teacher executes them. *(E.g. Teacher turn right)*

Version 2. One by one students come to the board, holding a sign with a robot in their hands. Other children «program the robot», the teacher writes their commands on the board.

Teacher: *«Let's try doing the same thing on computers – do Task 1 and Task 2»*

Task №1



Task: Run the program. Let's see what happens!

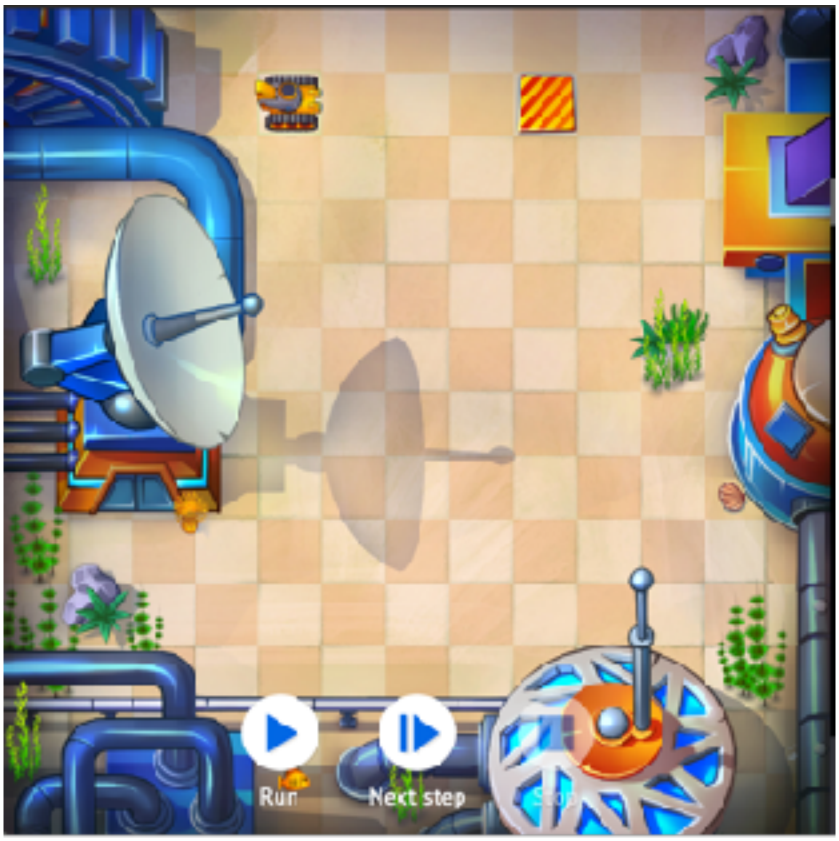
In this task the code is already written. Students should click «Run» and launch the program.

No source code

Final code:

```
robot.move 3
```

Task №2



Task: Get to the highlighted square. You know how to do it!

In this task the code is not written. Students should write the code themselves and run the program.

No source code

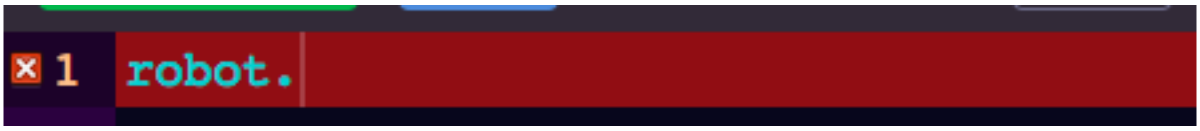
Final code:

```
robot.move 4
```

Tips&Tricks:

- If a learner can't move to the Task 2 — demonstrate the navigation button
- If a leaner can't calculate the distance — give an example with human steps or chess

When the tasks are done, show on the screen how to solve Task 4. While doing the task deliberately make a mistake in the code. The line with the mistake will get highlighted:



```
robot.
```

Draw children's attention to the fact that we are not robots and make mistakes quite often. In computer world mistakes are our friends. They show us what requires correction to achieve the goal. Demonstrate how Codewards system highlights the line with the mistake.

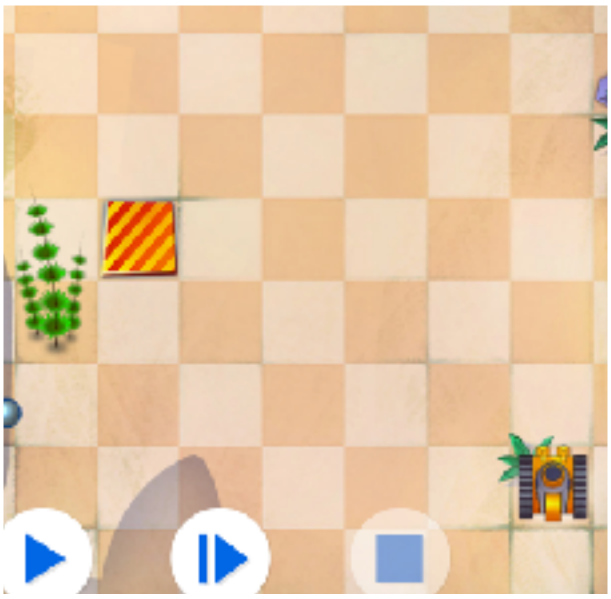
Teacher: *«It's simple when the object you control knows only one command. But some mechanisms are more complicated. E.g. a washing machine can wash wool using one program and wash jeans using the other. A microwave oven can heat the food up and can defrost it, etc.»*

If the mechanism can perform multiple actions, which «parts» of the command system would have more than one option: «who», «what» or «how»?

Most of the kids will give the right answer - «what». Praise them and pay attention to the fact that now we have two «variable zones»:

1. «how» (they saw it previous tasks when the number of robot's steps differed)
2. «what» (if the robot can perform several commands).

How many actions should the robot perform to reach the goal?
The robot should perform three actions - made of two different commands executed one by one. Before, we used only one command to reach the goal.



When we write the commands in a sequence leading the mechanism to the goal, this sequence is called **algorithm -** a sequence of commands being performed one by one.

- **Go to Task 3**

Let’s learn a new command – **`rotate`**

> `rotate` – turn. This command goes together with direction - turn where? Left or right.
>
> The complete command looks like this:

```
robot.rotate left
robot.rotate right
```

If you want to know where to direct a robot, first, define where the robot’s head is, then put yourself on its place and think where to turn.

Note that before writing a program, we need to imagine how the robot will get the point. Then we break this process into a clear sequence of commands, which the robot knows. And finally write those steps down in our code language.

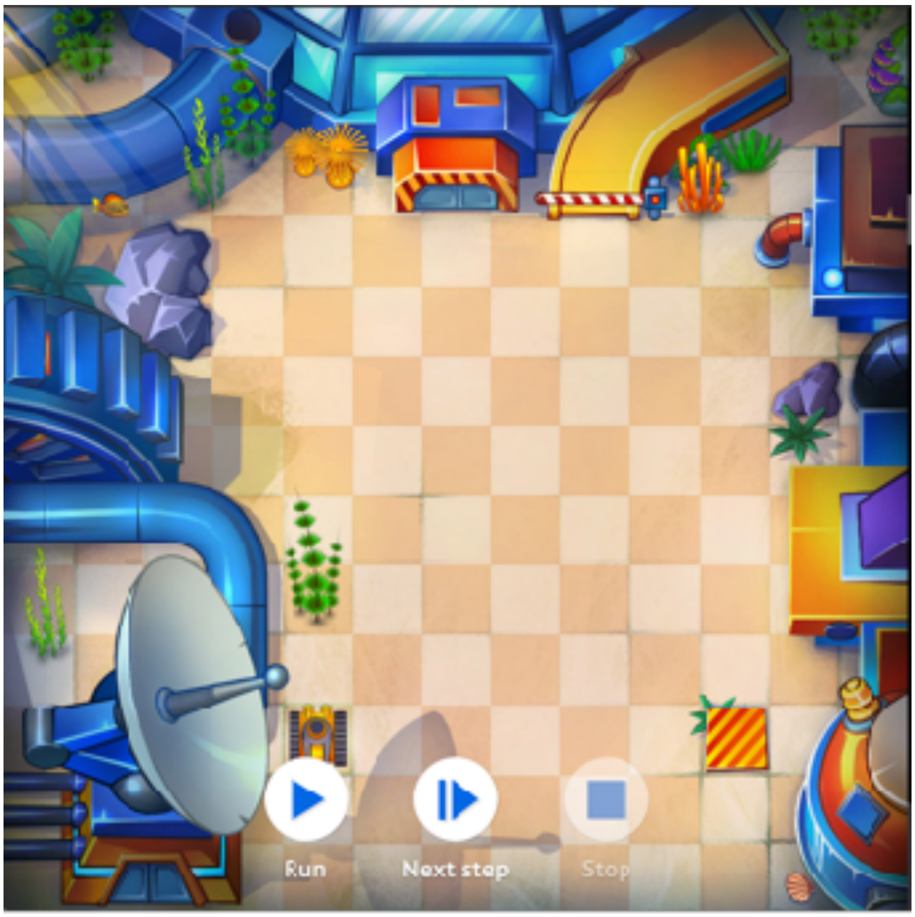
It is basically what all programmers do - writing special algorithms (sequence of commands) for the robots to do what they have planned.

Do Tasks №3, №4 and №5.

To help students get through the tasks quicker, keep asking:

- How many actions should the robot do?
- Which command matches the action?
- How should the command be modified?

Task №3



Task: Go to the highlighted square. Write the code yourself.

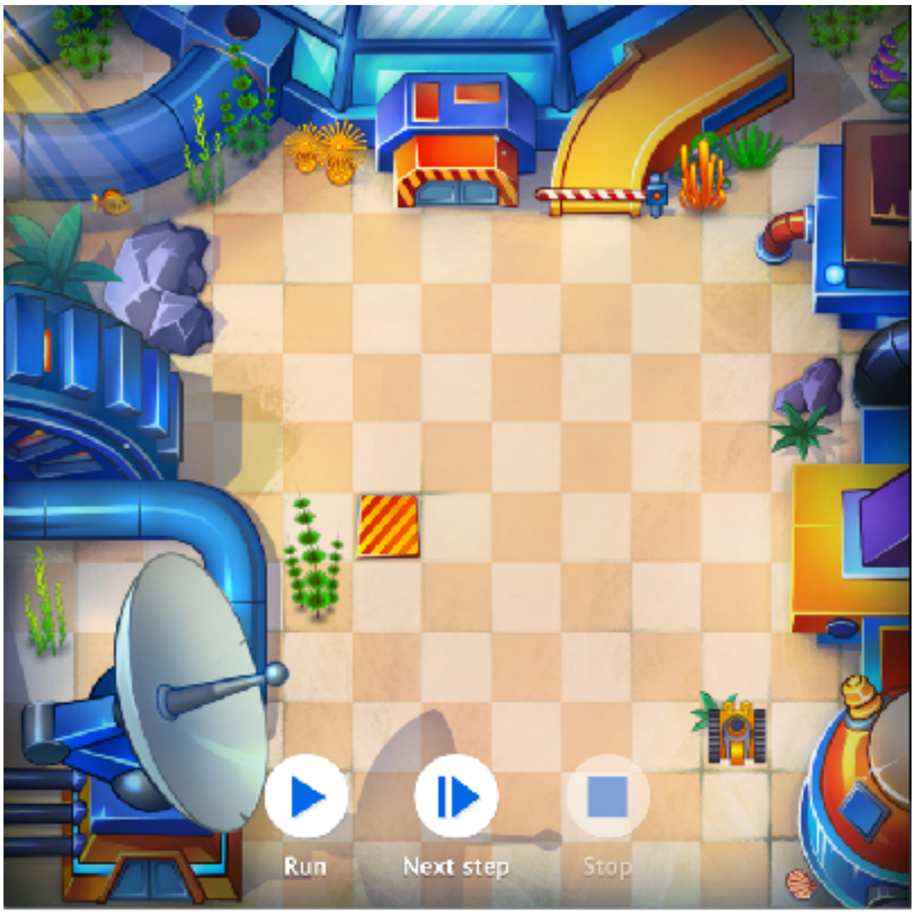
In this task the code is not written. Students should write the code themselves and run the program.

No source code

Final code:

```
robot.rotate right
robot.move 6
```

Task №4



Task: Go to the highlighted square. Write the code yourself.

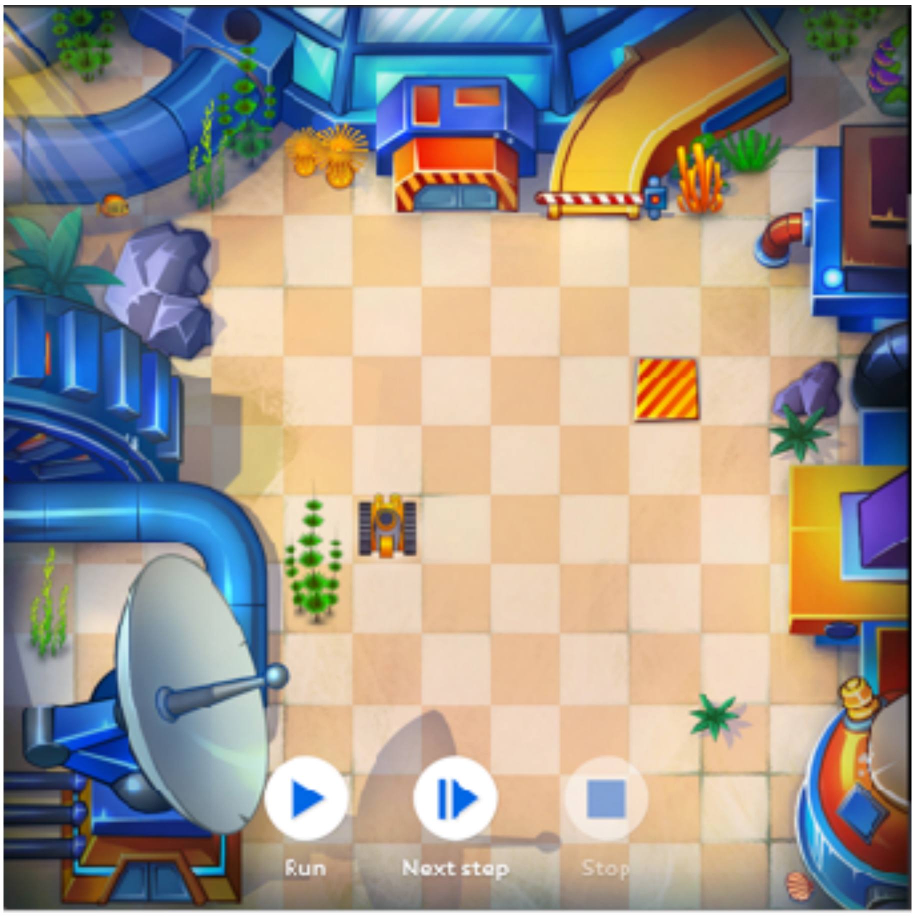
In this task the code is not written. Students should write the code themselves and run the program.

No source code

Final code:

```
robot.move 3
robot.rotate left
robot.move 5
```

Task №5



Task: Get to the highlighted square. Let's go!

In this task the code is not written. Students should write the code themselves and run the program.

No source code

Final code:

```
robot.move 2
robot.rotate right
robot.move 4
```

Tips&Tricks

- If a learner has written something odd and doesn’t know how to delete the text - show how to use Backspace with selecting the area or without.

- If a learner has written a different working algorithm and receive less than 3 stars for the task - tell the learner that everything is fine and you will get back to it bit later. It is more likely that the algorithm written contains more steps than required.

- If a learner runs the program after writing each step, Codewards system will indicate «Try again» - explain that before you run the program, you have to write the whole sequence of commands and only then run it. That is the key skill we are learning now.

- **Open Task 6**

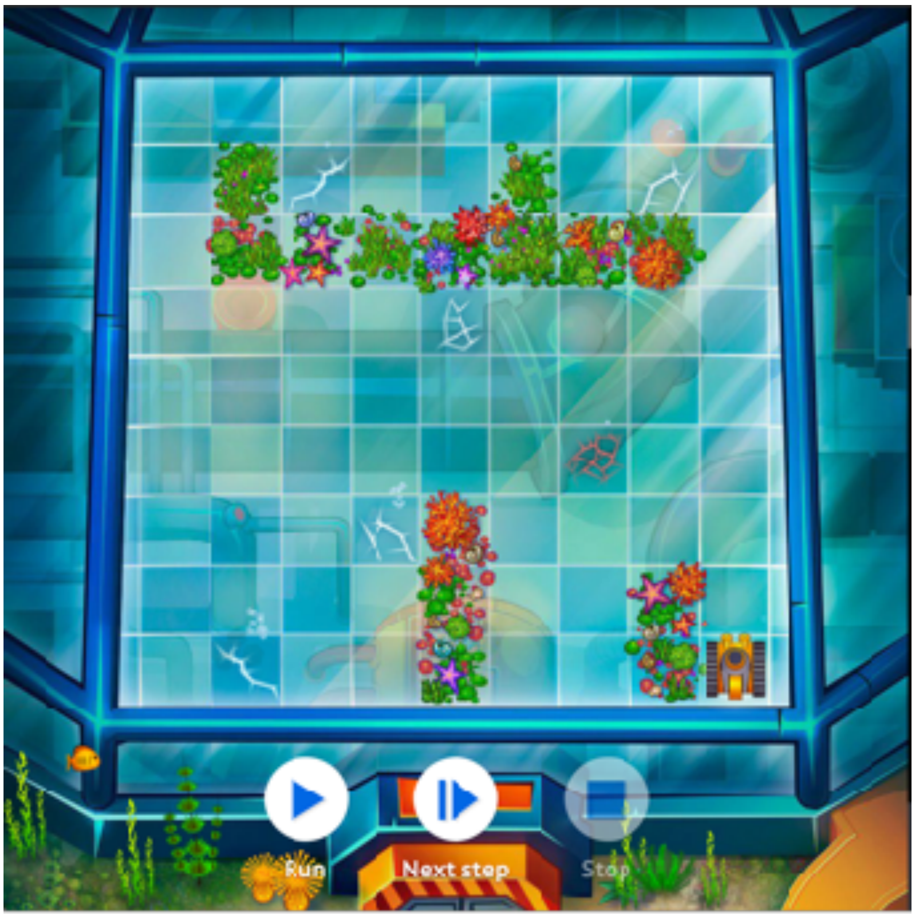
Ask the following questions:

- Does the robot reach the goal? - yes
- How many squares does the robot move? - 5
- How many commands are there in the algorithm? - 5
- Is that possible to rewrite the program to reduce the number of lines or the number of squares the robot moves? - yes

Explain that if we want our robots to work effectively, we should write ***optimal*** programs, which will save their time and other resources.

Do Tasks №6 and №7

Task 6



Task: Run the program to fix the crack in the dome!

In this task the code is already written. Students should click «Run» and launch the program. Point that to fix the crack the robot needs to go on it. The crack that needs to be fixed is highlighted.

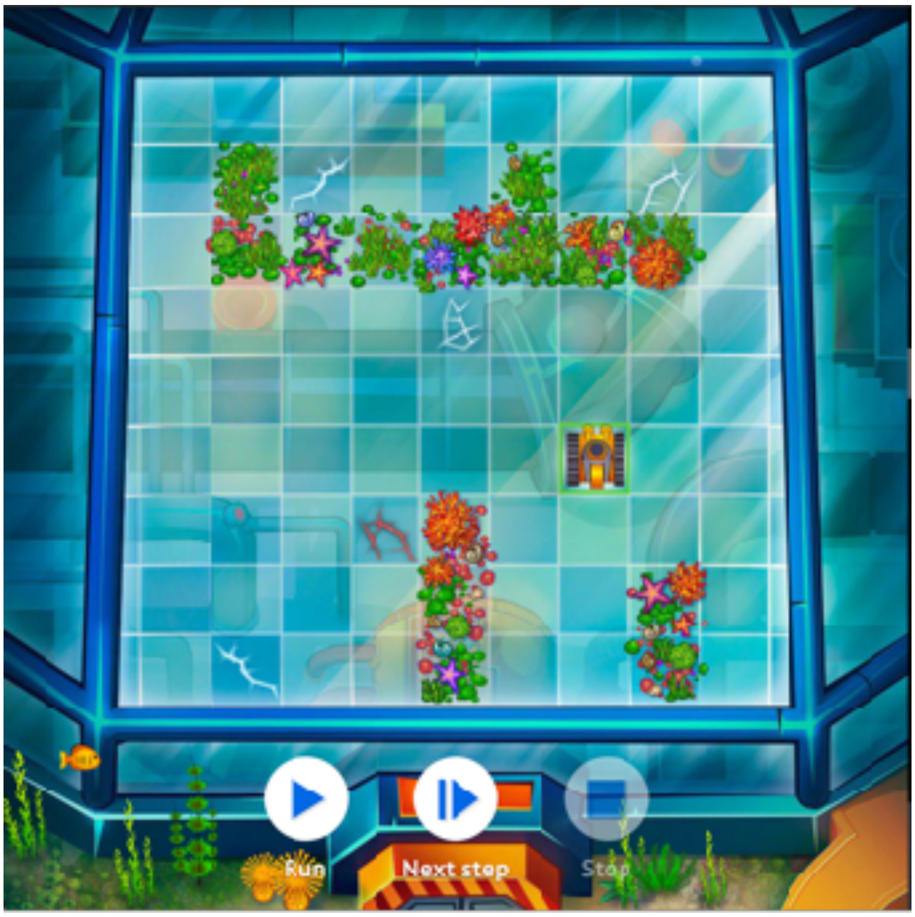
Source code:

```
robot.move 3
robot.rotate left
robot.move 2
```

Final code:

```
robot.move 3
robot.rotate left
robot.move 2
```

Task 7



Task: Fix the crack. Write the missing code.

In this task the code is not complete. Students should complete the lines and run the program. The crack that needs to be fixed is highlighted.

Source code:

```
___________
robot.move 3
___________
robot.move 1
```

Final code:

```
robot.rotate left
robot.move 3
robot.rotate left
robot.move 1
```

Teacher: *«Let's focus on what we've discovered today»:*

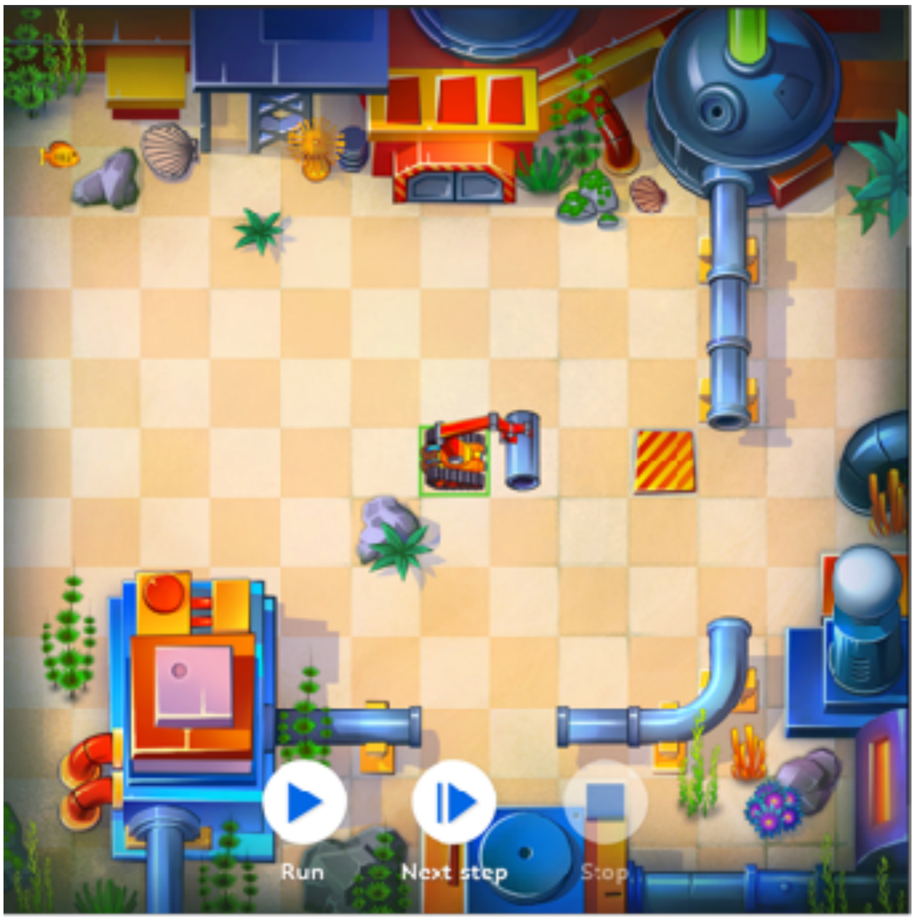
- Mechanisms and robots are controlled through commands with the following structure: WHO + WHAT + HOW
- Some mechanisms can perform only one command, but most of them - multiple.
- The sequence of commands to reach a certain goal is called an **algorithm.**
- An algorithm can be **optimal** and **non-optimal.** The non-optimal algorithm takes more resources. We should write only optimal algorithms to save our time and energy.
-

Teacher: *«The last important task is left - Let's go!»*

What is new:

1. This robot can perform several commands.
2. New commands: **`load`** - takes the pipe , **`put`** - lays the pipe down.
3. This task has a number of solutions - kids should find the optimal one.

Task 8



Task: Get the crane to pick up a part of the pipe and put it in the nearest break in the pipeline. Write the code yourself.

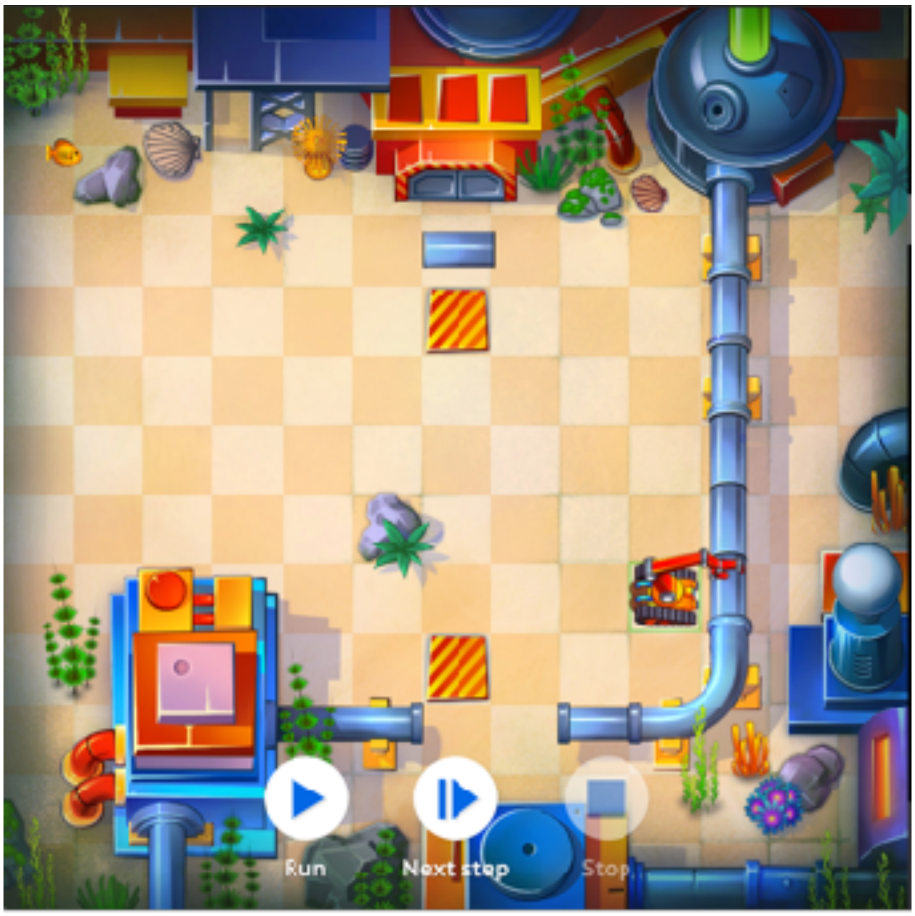
In this task the code is not written. Students should write the code themselves and run the program.

No source code

Final code:

```
crane.load
crane.move 3
crane.put
```

Task 9



The task: Take a piece of the pipe and install it in the shown break in the pipeline. To install a piece, pick it up with the crane and get the crane to the highlighted checkpoint. Write the code yourself.

In this task the code is not written. Students should write the code themselves and run the program.

No source code

Final code:

```
crane.rotate right
crane.rotate right
crane.move 3
crane.rotate right
crane.move 4
crane.load
crane.rotate right
crane.rotate right
crane.move 5
crane.put
```

Part 3. Close up

Estimated time – 5 minutes

Teacher: *«Today we have been using computer code to fix the dome and the pipeline of the underwater station. We have done a great job, but many things are still broken. I hope our unit will get together again to accomplish new exciting missions.»*