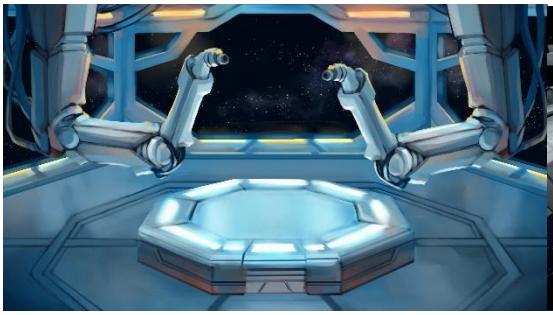
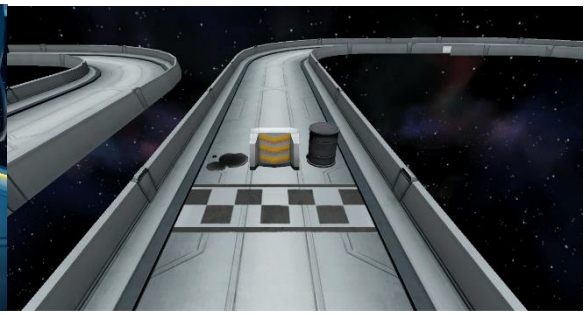
Robolocode – Control your robot and win the race!

Teachers Manual





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

Index

Introduction

This manual is meant for teachers, to support them in using the mobile game Robolocode as a learning tool with their students. The game can be played on mobile devices, where students will learn the basics of computational thinking and coding.

The goal of the game is for the player (the student) to teach the robot (by simple programming) how to move around on a track, avoiding obstacles, hinder opponents and reaching the finish-line first.

Robolocode is a mobile game in which players assemble a robot and program it to race along a racetrack against other robots. The programming part starts off as a simple code to guide the robot along the track to make sure it reaches the finish line. However, it gets more and more complicated when more robots compete and get in the way of each other, especially when they are equipped with offensive and defensive devices that can impact the race.

The goal of the game is to win races, level up and improve the robot further (upgrades) for continuing races. However, the key in winning the races lies in writing good code! ☺.

Purpose of this handbook

This handbook is made for teachers who are using the Robolocode game as part of their learning activities, teaching their students the basics of computational thinking, programming simple commands, creating loops, defining conditions and more.

The handbook gives both theoretical and hands-on information on the use of the Robolocode game for learning purposes, looking at implementation strategies, teacher roles and giving examples on how the game can be used as a learning tool with the target audience.

It is advised that you, as a teacher, read the handbook carefully before planning the use of the game in your learning activity with your students.

Target Group / Audience

The game is made for primary education and lower secondary education.

Learning objectives

- Computational thinking
- Programming language Python (not a learning objective within the game, but as part of the learning activities)
- …

Robolocode – The game

Welcome

Dear teacher, while you read this, we want you to explore and see how you make use of and implement the game as part of your learning environment.

What you need to know (as a teacher) - to prepare and to help your students

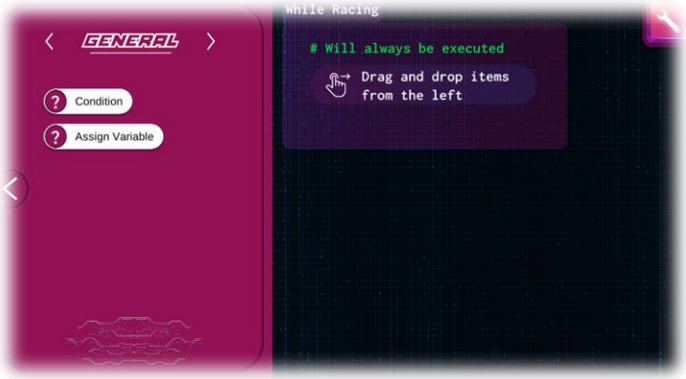
The game is divided into three main parts: The **Garage**, the **Coding** interface and the **Racing** (Sandbox mode).

The Garage:

In this part of the game, you will design your robot and add modules to it. Modules that can be added to the robot are **B**attery, **E**ngine and **H**ull. The properties of the different modules are described further down.

The Coding:

In this part of the game, the interface lets you generate code that will help your robot maneuver on the track, avoiding obstacles, competing with opponents and more.



The Sandbox (Race tracks):

In this part of the game you can let your robot enter the race tracks and let the robot compete against AI robots or against other players (in multiplayer mode). As you skill up and compete with others (multiplayer mode), you will be able to add new components to your robot, making it faster, stronger, more competitive and more.

During the race, the robot will face different types of obstacles. These are described below:

Game Obstacles

Whilst racing, the robot will face several obstacles. As the owner of the robot, you - the player need to add code to the robot to make sure that it avoids obstacles in the racing track. The table below shows the list of obstacles and how these obstacles "behave".

Barrel
Could contain oil (random). When it is crashed into, it causes random movement to a lane. If the robot is on fire and hits a barrel with oil, causes an area explosion and leave oil behind, that will then burn out leaving the lane cleared

Flames
Intermittent activation. Shoot directly upward and gives instantaneous damage. The robot will be visually on flames for some seconds.

Oil
Can be lit on fire by a robot that is on fire. It will burn for about 5 seconds. When burning it will have the same effect as the flames. When the fire goes out the oil is also gone. In a curve it will cause the robot to slide to an outside lane (unless the robot has chains or legs). If the robot is on the outer lane it will slide and hit the side/wall, causing damage. Randomly the oil can increase the speed of the robot if it is not in a curve.

Spikes
Cause damage to wheels and has intermittent activation. A robot with chains destroys the spikes. A robot with legs will slow down but does not take damage or destroy the spikes.

Robot Modules

The Robot can be upgraded by adding different types of modules to it. Some modules will *boost the effect* of the robot, some modules will give the robot *offensive abilities* and some modules will give the robot *defensive abilities*.

The different modules will unlock as the player progresses in the game, competing with peer players (in multiplayer mode).

The modules are divided into the categories **M**ovement, **U**pgrades, **S**ensors and **R**obot tools (**O**ffensive tools and **D**efense **T**ools).

When the player has added a module to the robot, the module needs to be coded so that the robot will understand how to use it. If the code is not made to activate the modules, the module will not be activated. The different modules are described below.

Movement (Speed)

The robot is equipped with standard wheels from the beginning. It can be upgraded with **C**hains or **O**mni wheels. These different movement modules have different abilities, with advantages and disadvantages. These are listed below:

- Wheels: Fastest movement. Affected by all obstacles
- Chains: Slower than wheels. Not affected by spikes
- Omni wheels: Slower than wheels. Not affected by oil

Robot properties

The robot can be upgraded with several modules. The **H**ull module will increase the protection of the robot, the **E**ngine module will increase the power of the engine - making the robot go faster, the **B**attery module is integrated with the defense and attack modules, increasing the "life-span" for the usage of the different modules.

Hull / Casing

1. Default mode - Standard hull
2. Extended mode - Reinforced Hull
3. One punch mode - Heavy hull (more weight)

Engine

1. Default mode - Standard engine
2. V8 mode - More powerful engine
3. Rocket mode - Big engine (more weight)

Battery

1. Default mode - Standard battery
2. Alkaline mode - Bigger battery
3. Monster mode - Biggest battery (more weight)

Sensors (others category)

The robot can also be upgraded with sensors to detect obstacles, heat and how the opponent robots are equipped. Level 1 sensors weigh less than level 2 sensors and level 3 sensors weigh more than level 2 sensors. Note, the weight will affect the speed of the robot! A list of sensors are shown below.

Obstacle sensor

- Level 1 - Only sees obstacles directly in front
- Level 2 - Detect the type of obstacle
- Level 3 - Can check the lanes for obstacles

Heat sensor

- Level 1 - Detects if an obstacle is on fire directly in front
- Level 2 - Can predict if there will a fire in that lane
- Level 3 - Can check the lanes for obstacles on fire and predict them

Opponent *(Property)* sensor

- Level 1 - Can detect position
- Level 2 - Can detect armament, type of movement (wheels, chains, omni wheels)
- Level 3 - Can detect properties (energy, hull points). Can be installed on the sides, front and back. Can see in 360º

Note! Adding it to the robot adds new possibilities to the conditions. The levels are cumulative, example level 3 has level 2 capabilities and level 1.

Robot tools / gadgets

The levels are cumulative, example level 3 has level 2 capabilities and level 1.

Hammer

Can only be used on the side where it's placed

1. Plastic hammer - Standard damage
2. Wooden hammer - Extra damage, more weight
3. Steel hammer - More weight and more energy efficient

Spikes

Only affects the give lane and the robots that are behind

1. Small spikes - Standard
2. Brick spikes - Extra damage
3. Lego spikes - Hurts like hell

Tractor beam

Affects the robot directly in front. Requires a certain amount of energy

1. Basic beam - Slows down opponent in front
2. Extended beam - Increases the slow down effect
3. Awesome beam - More energy efficient

Shield

The shield module defends towards other robots, not obstacles.

1. Wooden shield - Takes less damage from the hammer
2. Steel shield - Decreases damage from crashes
3. Titanium shield - Decreases the slow down from a tractor beam

Boosts (other)

- Speed boost. Is tied to the battery and a part of the main robot abilities.

In-game Functions

The game has its own syntax for the coding. The list below shows what types of functions are available in within the coding interface of the game:

General

- Condition
- Assign Variable
- Enable autodrive

Movement functions

- Start Moving
- Stop Moving
- Move left
- Move right

Gadgets

- Hammer Left
- Hammer Right
- Tractor Beam
- Drop Spikes
- Shield Left
- Shield Right

Sensor functions (Edit condition)

- is_robot_on_left
- is_robot_on_right
- is_robot_in front
- is_robot_behind
- is_obstacle_on_left
- is_obstacle_on_right
- is_obstacle_in_front
- is_obstacle_behind
- is_driving
- current_lane

Example of coding, to initiate some functions

As listed in the previous chapter, there are several functions that can be used and added as code to the robot. The further part will showcase some examples of simple code snippets that initiate the use of robot items to avoid obstacles, use the hammer and more.


GENERAL
Condition
Assign Variable
MOVEMENT
Start Moving
Stop Moving
Move Left
Move Right
DEFENSE
Shield Left
Shield Right
While Racing
Will always be executed
Drag and drop items from the left
Condition
if is_robot_on_right == true :
If Condition is met
use_hammer_right( ):
Drag and drop items from the left
Add Else
if is_obstacle_in_front == true :
If Condition is met
Condition
if current_lane == 0 :
If Condition is met
move_one_to_left( ):
Drag and drop items from the left
else :
If Condition is not met
move_one_to_right( ):
Drag and drop items from the left

Starting the race

(in multiplayer mode)

Teacher role

When implementing the game in the classroom with your students, you need to make sure that everyone has enough time to explore the game and get to know how to maneuver through the different menus, how to build their own robot and how to add code to the robot, before testing the robot (and the given code) in a race. The below setup gives an idea on how the teacher can plan a first-time implementation of the game with their students

First time implementation

Step	Task	Teacher role	Time
1	Give an introduction (presentation) of the tool	Present the tool, show screenshots and quickly show how the interface works, show a simple code snippet for obstacle avoidance and show how you can get started with the racing track	15 minutes
2	Let the students explore the UI	Be there as a facilitator while the students explore the game by themselves, with no specific task	15 minutes
3	Short summary of experiences	Teacher ask students about the game, what they have done so far and helps summarize the experience (as a plenary session)	15 minutes
Pause			
4	Prepare for racing. Let the students code their robots and prepare for a race	Facilitate the students and help them organize their code	15 minutes
5	Set up multiplayer event	Help students join in on multiplayer races, … and start up the races	10 minutes
6	Do an in-class reflection where students present how it went and then discuss how the code can be improved	Teacher steering the process and helping the group learn from each other.	20 minutes

Possible didactical approaches

To implement the game for learning activities, there are several approaches the teacher can take on. In this manual, we suggest three approaches, described below.

Individual approach

Let the students explore the game by themselves, with the teacher as a facilitator, where the teacher will facilitate each individual on the coding and helping the students get started with the multiplayer racing, doing plenary reflection regularly.

Group-based approach

Organize the students in groups (by three-four) and have them explore the app together. The teacher should then help facilitate the groups with in-group communication and then help the teams/groups start up the multiplayer racing – competing with each other on a group level. Follow-up with a both a group-based and a plenary reflection.

Class-based approach

For this approach, the teachers should set up a collaboration with other schools and then see if groups or whole classes from the different schools can compete with each other, to make it all into a “school league”, where classes (from different schools) are competing with each other. Events like this should also be followed up with reflection sessions