Rescue Simulation - Super Team 2026

RoboCupJunior Rescue Committee

Contents

The RoboCup Materials Storage

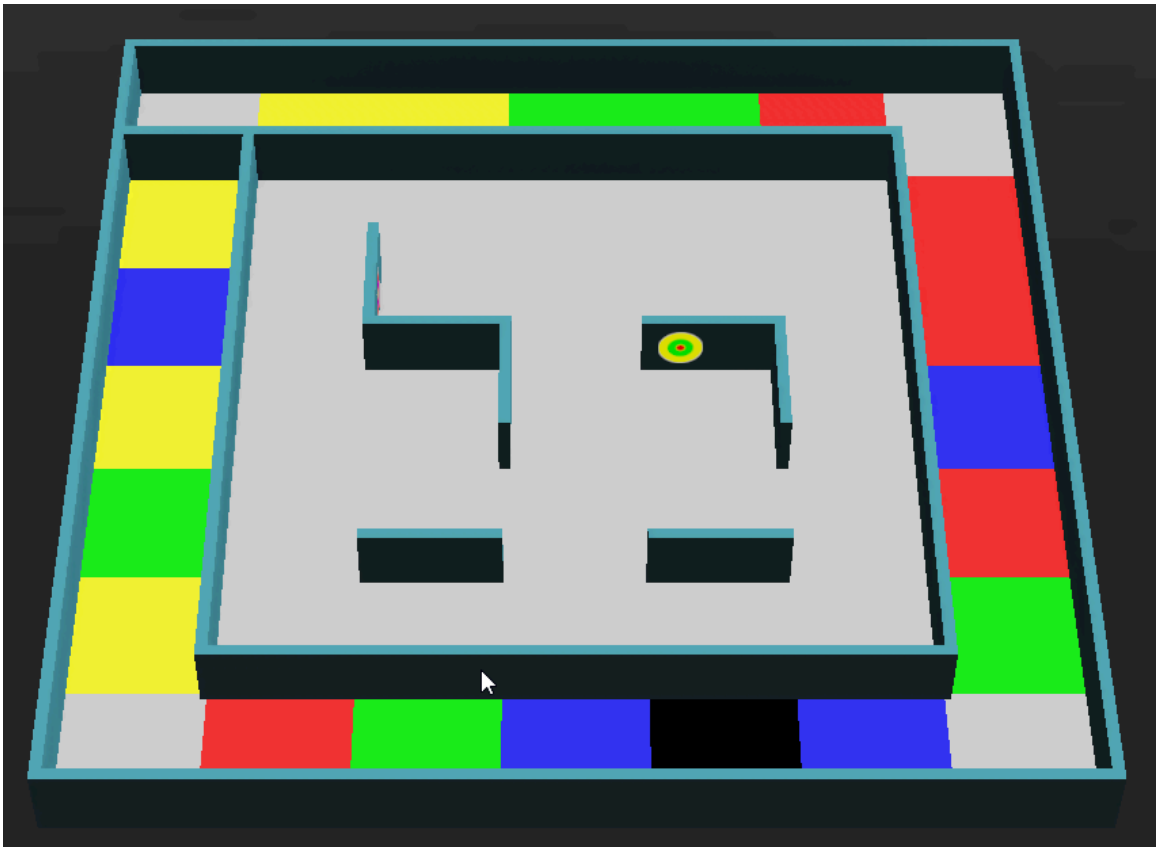
Scenario

Inside the RoboCup materials storage facility, order has broken down. Hundreds of elements are stored and labeled using encoded visual systems, designed for efficient robotic handling. However, due to system failures, the organization has been lost.

Two cooperative robots are deployed to restore order. One robot operates outside the storage maze, navigating a path of colored tiles that encode sequences of target identifiers. The other robot operates inside the maze, where multiple cognitive targets (CTs) are placed, each represented by a sign containing five concentric colored circles.

The outer robot acts as the “decoder,” while the inner robot acts as the “operator.”

Only through precise communication, strict sequencing, and correct interpretation can the system be restored.



Field

The field consists of two main areas:

- **Outer Navigation Zone**
  A continuous path of colored tiles runs along the outer perimeter. The tiles use the following colors: black, yellow, red, green, and blue.
  These tiles form ordered sequences of five colors. A white tile indicates the termination of a sequence.

- **Inner Maze Zone**
  A maze structure with walls and corridors occupies the center of the field. Inside the maze, multiple cognitive targets (CTs) are distributed. Each CT consists of a vertical sign displaying five concentric colored circles. The colors represent encoded identifiers, arranged from the outermost circle to the innermost.

Start of Game

The robots start simultaneously when the referee initiates the run.

- The **outer robot** starts at the beginning of the tile path, on its start tile.
- The **inner robot** starts on a **start tile**, which is not predefined and may vary between runs.

The robots must immediately begin cooperation to interpret sequences and locate targets.

Rules

The objective is to correctly identify, locate, and report cognitive targets based on encoded sequences.

- The outer robot reads sequences of colored tiles in order.
- Each valid sequence contains exactly **five colors**.
- A **white tile** marks the end of a sequence.
- Each sequence corresponds to one cognitive target.

The mapping rule is:

- Tile colors represent the CT's concentric circles **from outermost to innermost**.

Example:

- Tile sequence: Yellow → Yellow → Green → Green → Red
- Target: A CT with circles (outer → inner): Yellow, Yellow, Green, Green, Red

Cooperation Mechanism

- The outer robot communicates the sequence to the inner robot.
- The inner robot navigates the maze to find the matching CT.
- Upon reaching the correct CT, the inner robot must send a message to the supervisor indicating that the corresponding object has been collected.

Task Completion

A task is considered complete when:

- The inner robot is positioned at the correct CT, and
- The correct message associated with the sequence is sent to the supervisor

Each CT is associated with the same letter as in the rounds.

Order Constraint

- Targets **must be completed strictly in order**.
- The inner robot cannot report a target unless all previous targets have been correctly completed.

Scoring

Scoring is based on correctness of both order and message.

- Correct order, correct message: **+50 points**
- Correct message, incorrect order: **+5 points (Note, the CT is shutting down!)**

Penalties:

- Incorrect message: **−10 points**

Additional rules:

- Score can **never drop below 0**

Of the two rounds, the one with the higher score is used to rank. If two teams have the same score in the standings, the team that completed the round in the lower time will win.

Lack of Progress (LoP)

LoP may be declared by the team.

When LoP is called:

- The robot in the outside zone returns to its **start tile or the last checkpoint.**
- The robot in the inside zone returns to its **start tile**.

Penalty:

- **Subtract 1% from the end bonus (limit 0%)**

End of Game

The game ends when:

- The time limit of **6 minutes** is reached, or
- The outer robot sends an end message to the supervisor.

Bonus Condition

- If the outer robot sends the exit signal **when it is on its start tile**, an additional **10% bonus (or less, depending on the amount of LoP)** is added to the total score achieved at that moment

The final score is the total accumulated points, including any bonus.

Technical specifications

Controllers for testing

Inside the controllers folder, you'll find a debugging0.py controller and a debugging1.py controller that you can use to test the environment. The instructions appear in the console. Note that there is only one message that can be sent.

Remote Controller

The remote controller connection robot name has been changed to allow for multiple robots to connect:
To connect to the webots server, you can use the following command:
<Webots Controller Path>/webots-controller.exe --protocol=tcp --ip-address=<Server IP Address> --robot-name=<robot-name> <Controller Path>

****robot-name needs to be "Erebus_Bot_0" or "Erebus_Bot_1"***

Supervisor messages changes

Erebus supervisor only accepts messages 'L' (for lack of progress), 'G' (to get game info) and cognitive target messages.

The sign message structure has changed:

message = struct.pack('i i c i', x, y, letter, robot_playing)

when ***x***, ***y*** and ***letter*** are the same as in the standard game, and ***robot_playing***, the number of the robot (0 blue, 1 red).

Messages between robots

To send a message to another robot, you can use the emitter of the robot. In this case, you need to pack the message with the next method of the module ***struct*** :

struct.**pack***(format, v1, v2, ...)*

Return a bytes object containing the values v1, v2, … packed according to the format string format. The arguments must match the values required by the format exactly.

To receive a message, you can use the receiver of the robot. When it receives a packet, you need to unpack the message, using the method unpack of the same module:

struct.**unpack***(format, buffer)*

Unpack from the buffer buffer (presumably packed by pack(format, ...)) according to the format string format. The result is a tuple even if it contains exactly one item. The buffer's size in bytes must match the size required by the format, as reflected by calcsize().

See https://docs.python.org/3/library/struct.html