Robo Games - 2020

With Webots – University Category

About Webots

Webots is an open-source three-dimensional mobile robot simulator.
It was originally developed as a research tool to investigate various control algorithms in mobile robotics. Since December 2018, Webots is released as an open source software under the Apache 2.0 license.
This user guide will get you started using Webots for Robo Games 2020.
Here we Go!

Introduction

Once you have successfully installed Webots software in your PCs, you can start designing your robot.
For our competition you need to program and design a robot which satisfies the requirements of the competition.
Since Webots offer many numbers of robots, libraries, and sensors you are free to design on your own.
We provide you an environment in which your robot must complete our tasks.
We will evaluate the robots according to our grading criteria.

(PS : For inquiries related to installing Webots, check https://cyberbotics.com/doc/guide/index)

System Requirements

RAM (Minimum) : 2 GB
Storage (Minimum) : 2 GB
Graphics : NVIDIA or
AMD OpenGL (minimum version 3.3)
Operating Systems :
Ubuntu (Versions after 16.04)
Windows (10 or 8.1 only 64-bit)
MacOS 10.15 or 10.14

(PS : For inquiries related to installing Webots, check https://cyberbotics.com/doc/guide/index)

Environment

Webots offers many environments, but for RoboGames 2020, we will offer you an environment in which the task must be done.
But for practice, you can use any of the platforms available in Webots.

New to Webots?

Refer to the website https://cyberbotics.com
In this site, there are tutorials which will be fit for those people who are new to Webots.
And also, there are many example simulations which you can use to get an abstract idea about working with Webots.

The tutorials cover mostly all the functions and attributes which are available in their platform.

The Challenge

The recent outbreak of COVID-19 has drastically changed pretty much everyone's lifestyle. Even usual tasks such as going to the store to purchase one's food requirements has become a risky activity. Mazeville is a city that has been severely affected by the outbreak. Unfortunately, despite the many warnings that have been issued, there are still people walking in the streets of Mazeville without even wearing a mask. You have been given the task of creating a robot that can navigate through the city, obtain an item from the store, and safely return back home as soon as possible, with minimal risk.

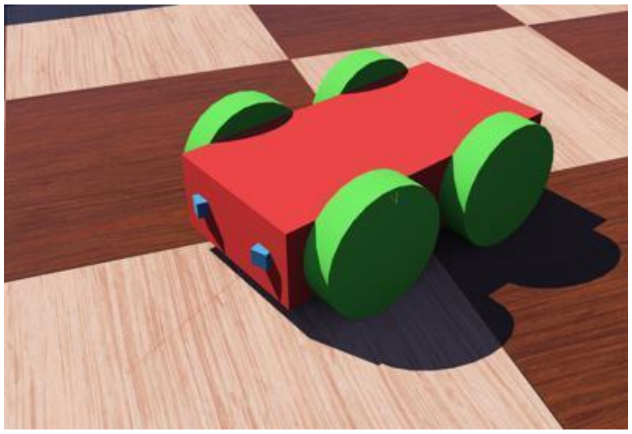
Note: Throughout the rest of this document, you will be provided with links to the official Webots documentation for reference.

Robot

The base model for the robot should be the **4-Wheeled Robot** available by default within webots (Add a node -> PROTO nodes (Webots Projects) -> samples -> tutorials -> FourWheelsRobot (Robot)). You may also build it on your own if you wish by following this tutorial. We also recommend that you take a look at the rest of the tutorials available on the Webots User Guide.

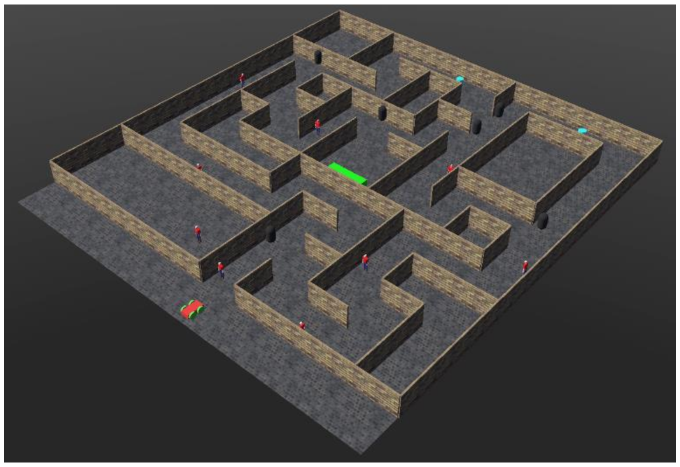
You are free to use any sensors you wish. However, the position of all sensors or any other node that you attach to the robot should remain inside or on the surface of the robot. Suggestions on the types of sensors that might help you can be found on the Environment Specifications section.

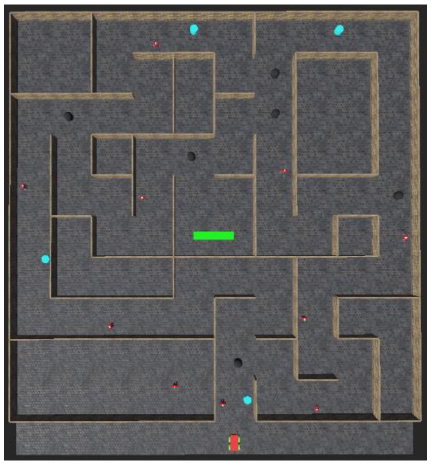
By default, the 4-wheeled Robot comes attached with a script written in the C language. However, you are free to use any language which Webots supports (C, C++, Python, Java, MATLAB, ROS).

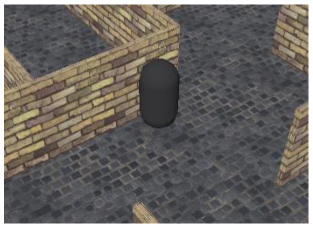


Environment Specifications

The layout of the city is like a maze (similar to the practice platform). The practice platform is a 5x5 m^2 maze while the final platform used at the competition will be a 7.5x7.5 m^2 maze. Each road has a width of 0.5m including wall (each with a width of 0.01m).







There is an emitter at the center of each junction that the robot may come across. It emits a signal that indicates how long the shortest path to the store is, depending on the direction of the turn the robot takes at the junction. To receive this signal, you are required to attach a receiver to your robot (pay attention to the emitter's parameters within the practice platform such as "channel" and "range"). The signal string from the emitter is in the following format: "N, E, S, W" where N, E, S and W represent the lengths of the shortest paths when taking each of the North, East, South or West paths. If the value of a given path is zero, this indicates that the robot is not permitted to go in that direction, either due to the presence of a wall, or if the given road is a one-way street. We recommend that you add a Compass to the robot to help keep track of the direction in which the robot is currently facing.

For example: If the signal emitted by the emitter is "0, 10, 0, 25", this indicates that you are not permitted to take a turn towards the North or South at this junction. The Eastern path is closer (shorter distance) to the store than the Western path.

Consider using Distance Sensors to ensure that the robot stays on the path and doesn't collide with walls or any other objects.

You must also add a Camera node to the robot in order to detect the following. You should use the image from the camera and detect the following using its pixels (image processing):

- Pedestrians
- Sanitizers
- Store

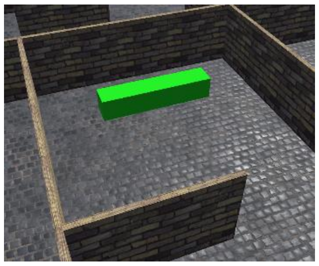
Pedestrians

The shorter paths tend to have more pedestrians and each pedestrian has a higher risk of infection compared to those of longer paths. Therefore, choose the longer or shorter path depending on how good your robot is at avoiding pedestrians. You are required to maintain a distance of at least 0.2m between your robot and the pedestrian to get zero risk of infection. The closer you get to the pedestrian, the higher the risk of infection. Pedestrians always have a **red** t-shirt.

Sanitizers

There are also bottles of sanitizers on the side of streets that can be picked. These are represented by **cyan** colored cylinders. Getting close to them would reduce your risk of infection (the sanitizer will disappear after being picked).

Store



The store is represented by a **green** box at the center of the map. The robot should return back home after detecting this box. We recommend that you follow back the path you took to get to the store in order to avoid getting stuck in loops or taking too long. You should avoid the pedestrians on your way back as well.

You will be evaluated based on your time and the risk of your robot having infectious particles on it (how close your robot got to the pedestrians). The lower the better. Decide which of the above two evaluating bases your robot will focus on. A robot who is good at avoiding the pedestrians is better off taking the shorter paths while robots that can move fast but are not the best at avoiding the pedestrians should focus on the longer paths. It may even be that a combination of the above two will be the best for your robot. Use the practice platform that will be provided to you, to test out your robot.

- **Grading criteria will be released after the registration deadline**

Eligibility

- Participants are advised to form a team of up to 5 undergraduates.
- Any number of teams from a university can enroll in the competition.
- All team members should be undergraduates of the same university at the time of their participation in the competition.
- Each team should provide valid identification documents from their university on the competition day to prove their eligibility to participate in the competition.
- Students who sat for the 2019 A/Ls fall under the School Category.

Task Procedure

- You will receive a link to which you must submit your Webots project folder. You may submit until the day before the competition after which no more submissions will be accepted.
- You will also receive a link using which you can watch the competition unfold.
- On the day of the competition, we will check if all the robots in the competition abides by our rules.
- Afterwards, each robot will be placed on our final platform and the simulation will begin, one at a time.
- You will be able to navigate and spectate within the platform but you will not be able to interact with the environment.