Collision Avoidance

Homework 3 Report
ENPM 690 - Robot Learning

Nikhil Lal Kolangara (116830768)

GitHub:https://github.com/nikhil-kolangara/ENPM-690-Robot-Learning_Homework-3

Date : 02/24/2020

Contents

1 Problem 1

Question 1 *Program a simple robot vehicle in a simulated environment (robot simulation tools and libraries may be used). Your simulated robot should exhibit at least one sensor input (e.g., forward-looking range sensor that returns the distance to the nearest obstacle) and two control outputs (e.g., left and right wheels, or speed and direction of vehicle motion). Show that you can drive your robot around through mouse or keyboard inputs.*

Approach 1 *Robot Description:*

The Robot is made using the Webots simulator. It is a simple robot with a rectangular body and four wheels. The Body of the robot was constructed by importing the BODY Node in which the child Node parameters are its shape (box) and the color (red). The next parameters are the wheels which are basically 4 cylinders connected to the rectangular box using Hinge Joints. Two distance sensors are implemented on the front of the robot which measures the distance of the robot from the nearest obstacles such as walls and boxes. The wheels are driven using the Rotational Motor Node. The wheel controller module are programmed in C for the keyboard control of the robot. The controller outputs are wheel velocity and direction of vehicle motion.

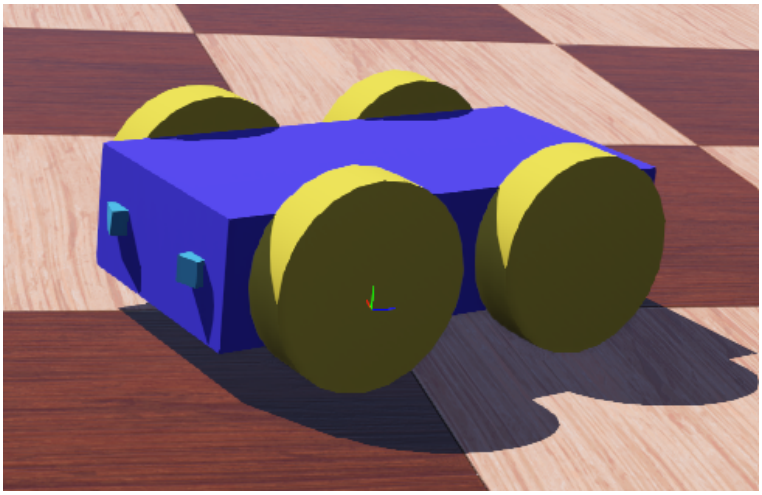

Figure 1: Robot developed using Webots

Webots: It is an open source and multi-platform desktop application used to simulate robots. It provides a complete development environment to model, program and simulate robots. It has been designed for a professional use, and it is widely used in industry, education and research. Cyberbotics Ltd. maintains Webots as its main product continuously since 1998.

Source: `https://cyberbotics.com/`

Controlling the Robot using Keyboard:

Step 1: Created the Robot and added controllers to the wheels.

Step 2: While creating the robot, the Keyboard Action controller was created to program the controller to drive the robot using the keyboard actions.

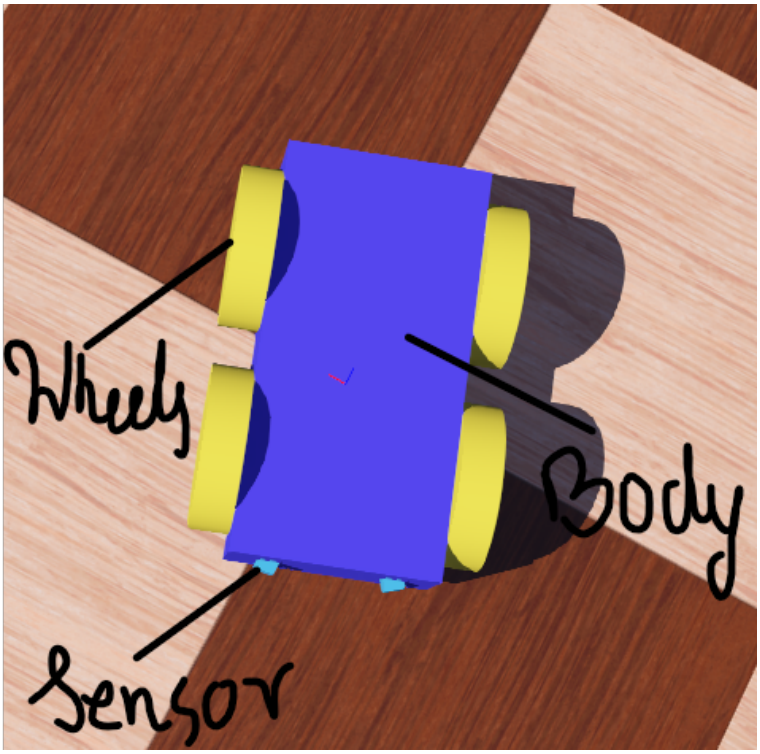

Figure 2: Robot's Top View

Step 3: The keyboard action controller is implemented in C++.

Step 4: The structure of the code is described in next steps.

Step 5: The appropriate header files are included especially, "Keyboard.hpp" under webots namespace.

Step 6: While the TIME STEP is not equal to -1, a switch case is implemented for all actions namely, UP, DOWN¡ LEFT and RIGHT.

Step 7: In the switch case, the pressed key is received through a function named "getKey()".

Step 8: The cases are actions specifies under the Keyboard class. For each action, the left and right wheel's values are changed.

Step 9: Therefore for LEFT, the values of left wheel is set to -1 and right wheels as 1 so as to rotate the left wheels backwards and right wheels forwards. Similarly for every action, the values are set accordingly.

Step 10: Finally, the keyboard inputs are disabled and the environment cleaned up by implementing "delete robot".

Simulation Video Snap:

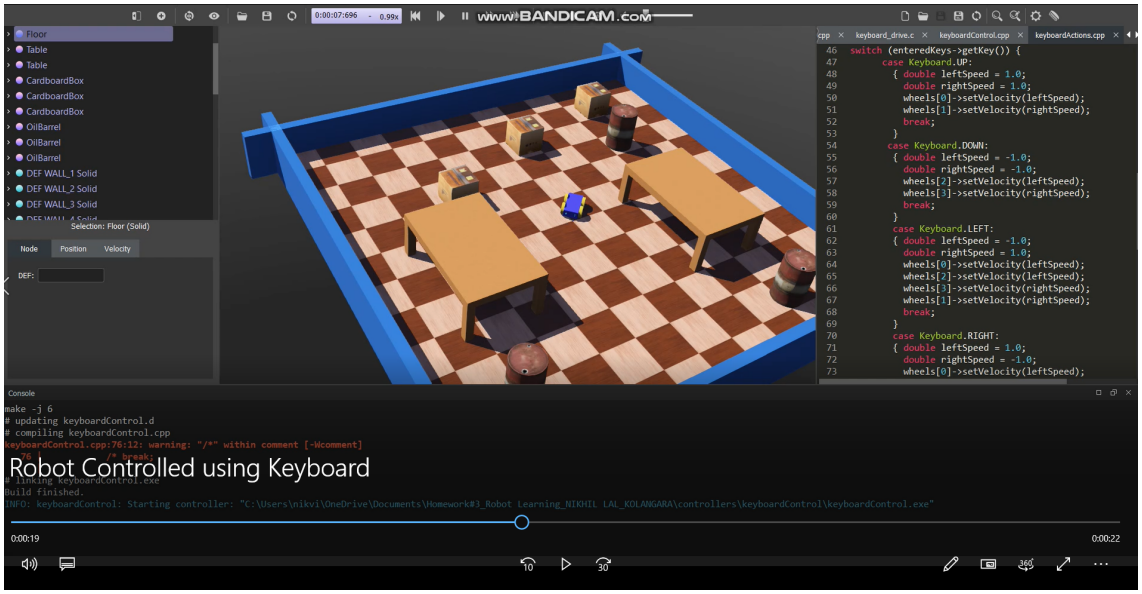

Figure 3: Video Simulation of Problem 1

2 Problem 2

Question 2 *Add a programmed behavior to your robot, such as following (or avoiding) a light, or wandering, while avoiding collisions with obstacles.*

Approach 2 *For this problem, the developed Robot is programmed to wander a simulated environment. The environment contains few objects such as Oil Barrels, Desks and Cardboard Boxes. The environment is also surrounded by solid Walls. The robot's controller is programmed in C++ to detect and avoid the obstacles.*

Steps:

1. Used the robot created in previous problem, which exhibits two sensor inputs i.e. distance sensors and two control outputs such as speed and vehicle direction.

2. Implemented a controller by adding a new controller from the "Wizards" tab in the Webots simulator.

3. The controller node is one of the robot parameters. Under the controller node, a programmable file is selected which is used to drive the robot to avoid obstacles whilst taking inputs from the sensors.

4. The controller is programmed in C++. The structure of the code is described in next steps.

5. First the header files are called which contains codes to read data from different sensors and controllers and pass it as arguments to the function.

6. The number of wheels and distance sensors used are specifies in each array respectively.

7. The position of the wheels are set to INFINITY such that the robot wheels are actuated using its velocity and not by the position it has to translate to.

8. An avoidObstacleCounter is specified and initialised to 0. A while loop is run until the TIME STEP variable is not equal to -1.. This variable is predefined as 64.

9. If the avoidObstacleCounter is greater than 0 then it is decremented and the wheels are controlled to take a turn.

10. The avoidObstacleCounter is set to 100 as soon as it reads inputs from distance sensors about the objects.

Simulation Video Snap:

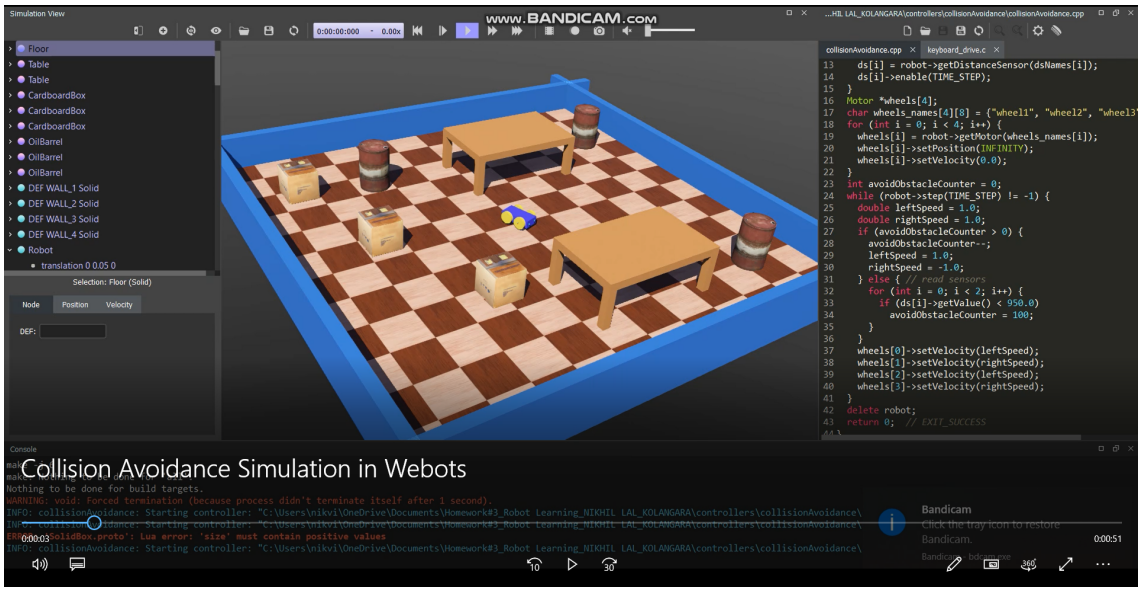

Figure 4: Video Simulation of Problem 2

NOTE: The video file is large to be viewed in GitHub. Therefore please select "View raw" option which will download the video to be played in any video player in your system.

3 Conclusion

The homework helped in understanding how to use a Robotics simulator to build a robot from scratch robot, add sensors and controllers, build an environment and test the programmed features in the constructed environment.