ROBOTICS PROJECT

Obstacle Avoidance Robot

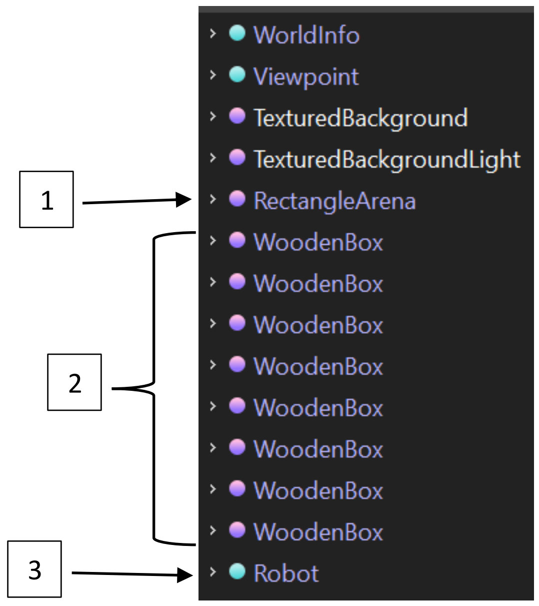
Abstract

Obstacle detection and avoidance are the central issues in designing mobile robots. This technology provides the robots with senses that they can use to traverse unfamiliar environments without damaging themselves. In this project, an Obstacle Avoiding Robot is designed in **Webots** to detect obstacles in its path and maneuver around them without collapsing. It is a robot vehicle that works in Webots and employs ultrasonic distance sensors to detect obstacles. In this Software C, C++, MatLab, ROS, and Python Languages were used to carry out the programming. In this project, we used C++ programming. The integration of ultrasonic distance sensors provides higher accuracy in detecting surrounding obstacles. Being a fully autonomous robot, it successfully maneuvered in unknown environments without any collision.

The Software used in this project is Webots. Webots 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 professional use and is widely used in industry, education, and research. Cyberbotics Ltd. maintains Webots as its main product continuously since 1998.

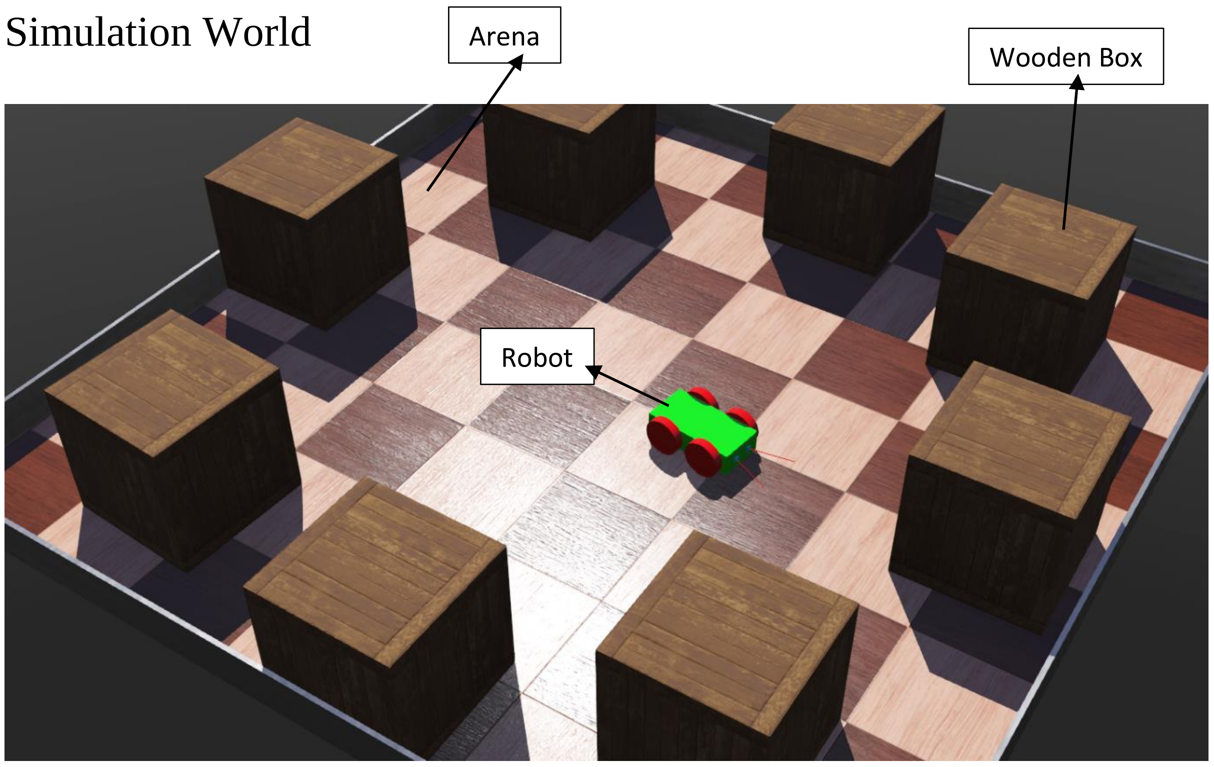
Simulation

Simulation World Components – Rectangular Arena, Wooden Box, and Robot.

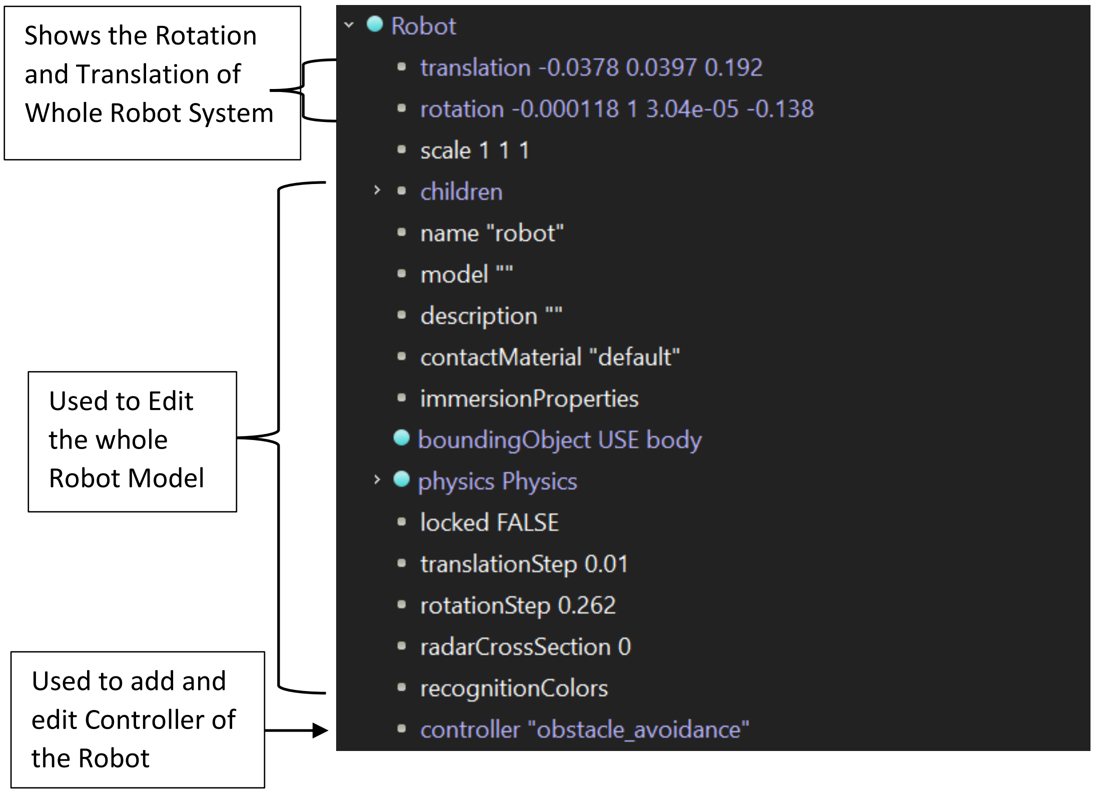


WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena
WoodenBox
WoodenBox
WoodenBox
WoodenBox
WoodenBox
WoodenBox
WoodenBox
WoodenBox
Robot

Simulation World

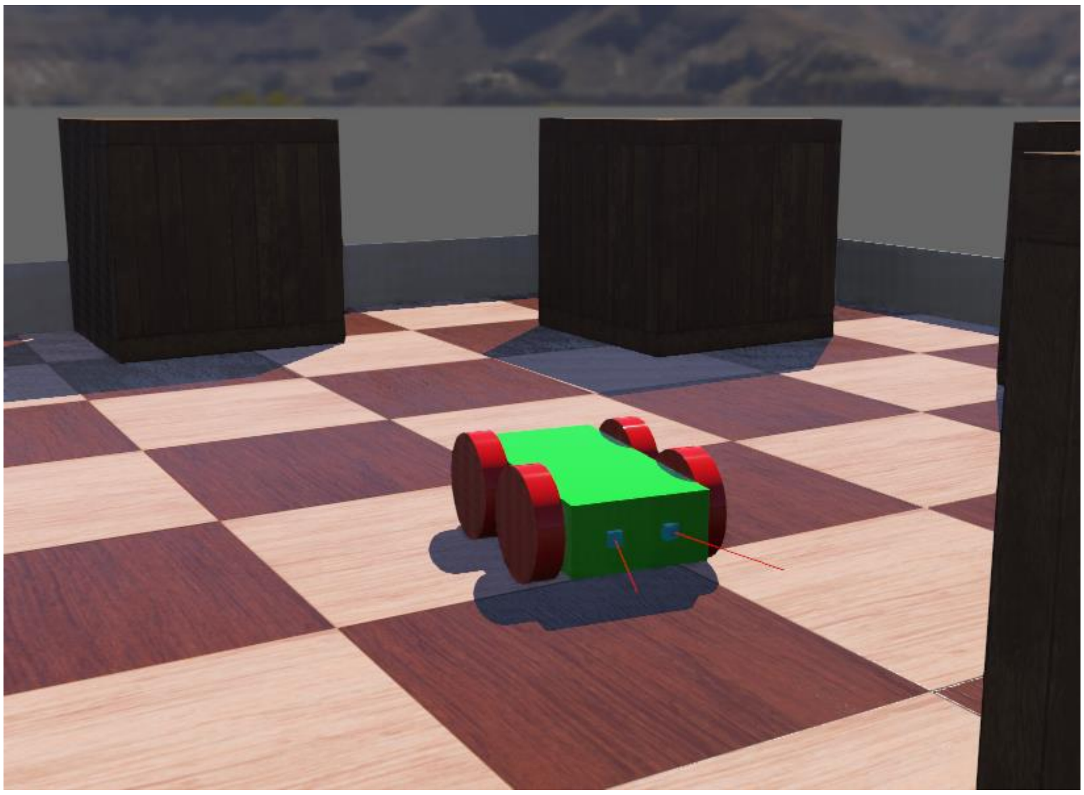


Robot Parameters Info

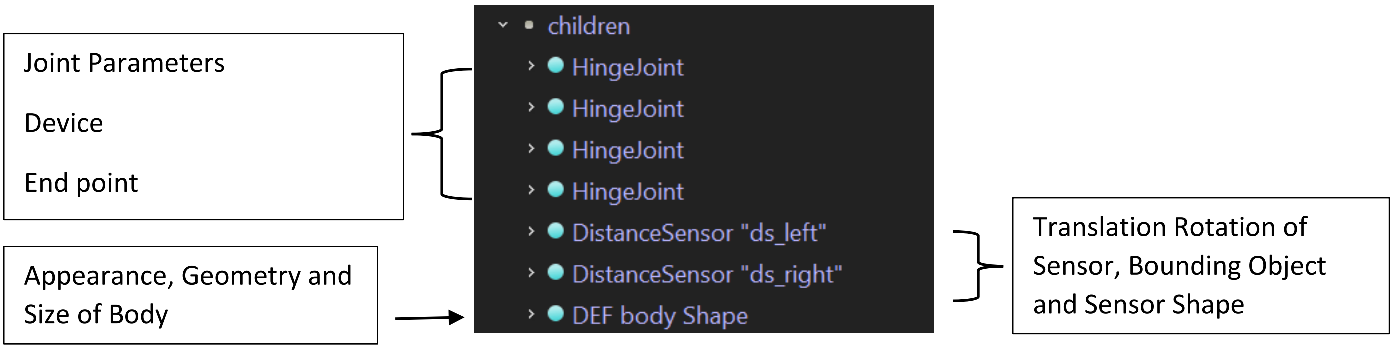

Shows the Rotation and Translation of Whole Robot System
Used to Edit the whole Robot Model
Used to add and edit Controller of the Robot

```
Robot
    translation -0.0378 0.0397 0.192
    rotation -0.000118 1 3.04e-05 -0.138
    scale 1 1 1
  › children
    name "robot"
    model ""
    description ""
    contactMaterial "default"
    immersionProperties
    boundingObject USE body
  › physics Physics
    locked FALSE
    translationStep 0.01
    rotationStep 0.262
    radarCrossSection 0
    recognitionColors
    controller "obstacle_avoidance"
```

Final Robot Model

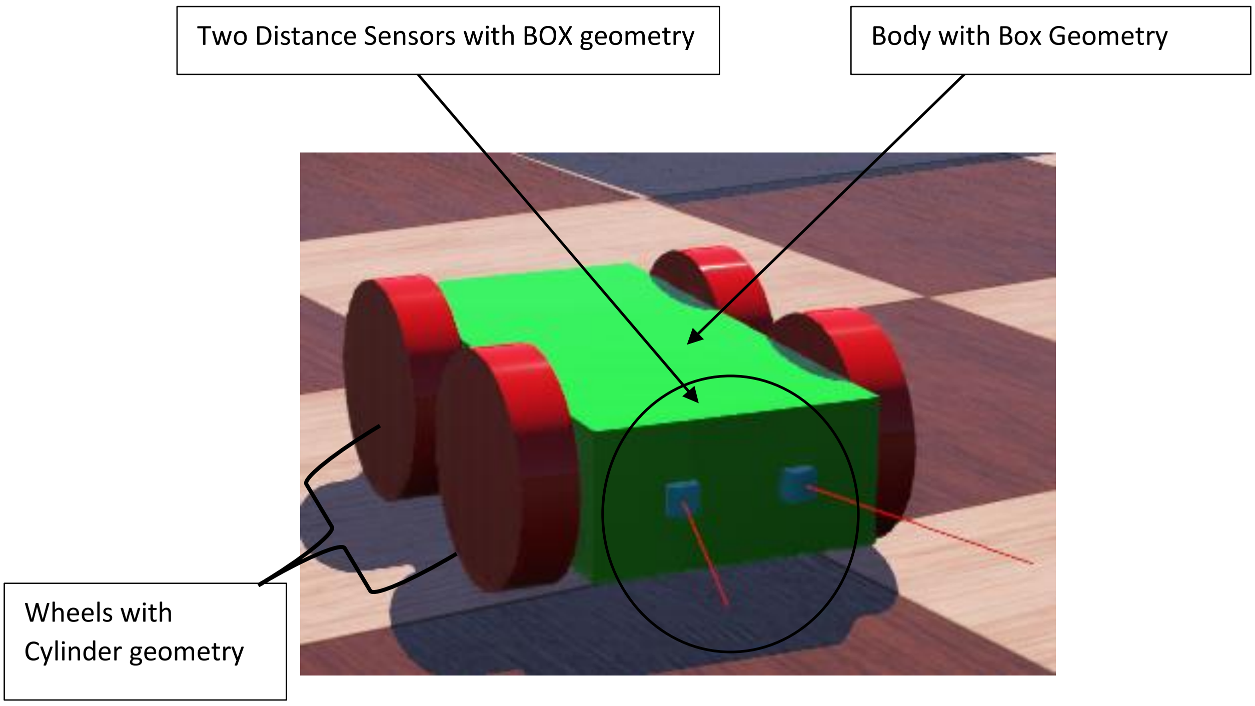


Robot Model Parameters


Joint Parameters
Device
End point
Appearance, Geometry and Size of Body
Translation Rotation of Sensor, Bounding Object and Sensor Shape

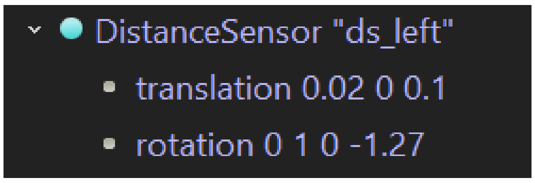
```
children
  HingeJoint
  HingeJoint
  HingeJoint
  HingeJoint
  DistanceSensor "ds_left"
  DistanceSensor "ds_right"
  DEF body Shape
```

Joint Parameters – Hinge Joint Parameters
Device – Rotational Motor (Wheel/Motor)
Endpoint Solid – Position of Wheel and children-Use wheel
Appearance – PBR Appearance
Geometry of Body – BOX
Size of the Body – 0.1 0.05 0.2
Geometry of Sensor – BOX
Size of the Sensor – 0.01 0.01 0.01
Geometry of Wheel – Cylinder
Dimensions of Cylinder – Height - 0.02 and Radius – 0.04


Two Distance Sensors with BOX geometry
Body with Box Geometry
Wheels with Cylinder geometry

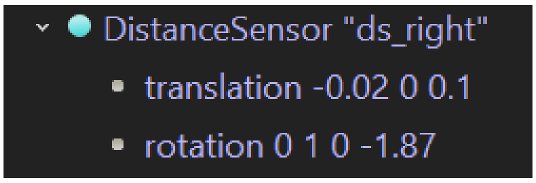
Position of Sensors

Left Sensor



```
DistanceSensor "ds_left"
    translation 0.02 0 0.1
    rotation 0 1 0 -1.27
```

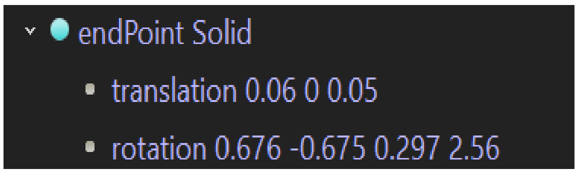
Right Sensor



```
DistanceSensor "ds_right"
    translation -0.02 0 0.1
    rotation 0 1 0 -1.87
```

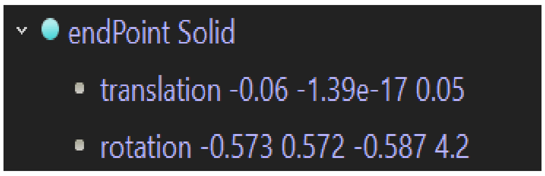
Position of Wheels

Front Left Wheel



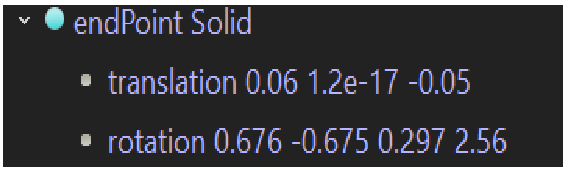
```
endPoint Solid
    translation 0.06 0 0.05
    rotation 0.676 -0.675 0.297 2.56
```

Front Right Wheel



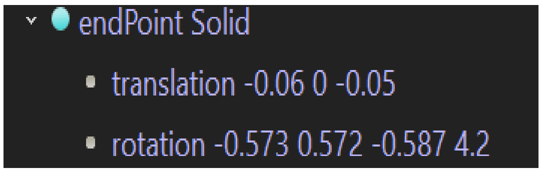
```
endPoint Solid
    translation -0.06 -1.39e-17 0.05
    rotation -0.573 0.572 -0.587 4.2
```

Back Left Wheel



```
endPoint Solid
    translation 0.06 1.2e-17 -0.05
    rotation 0.676 -0.675 0.297 2.56
```

Back Right Wheel

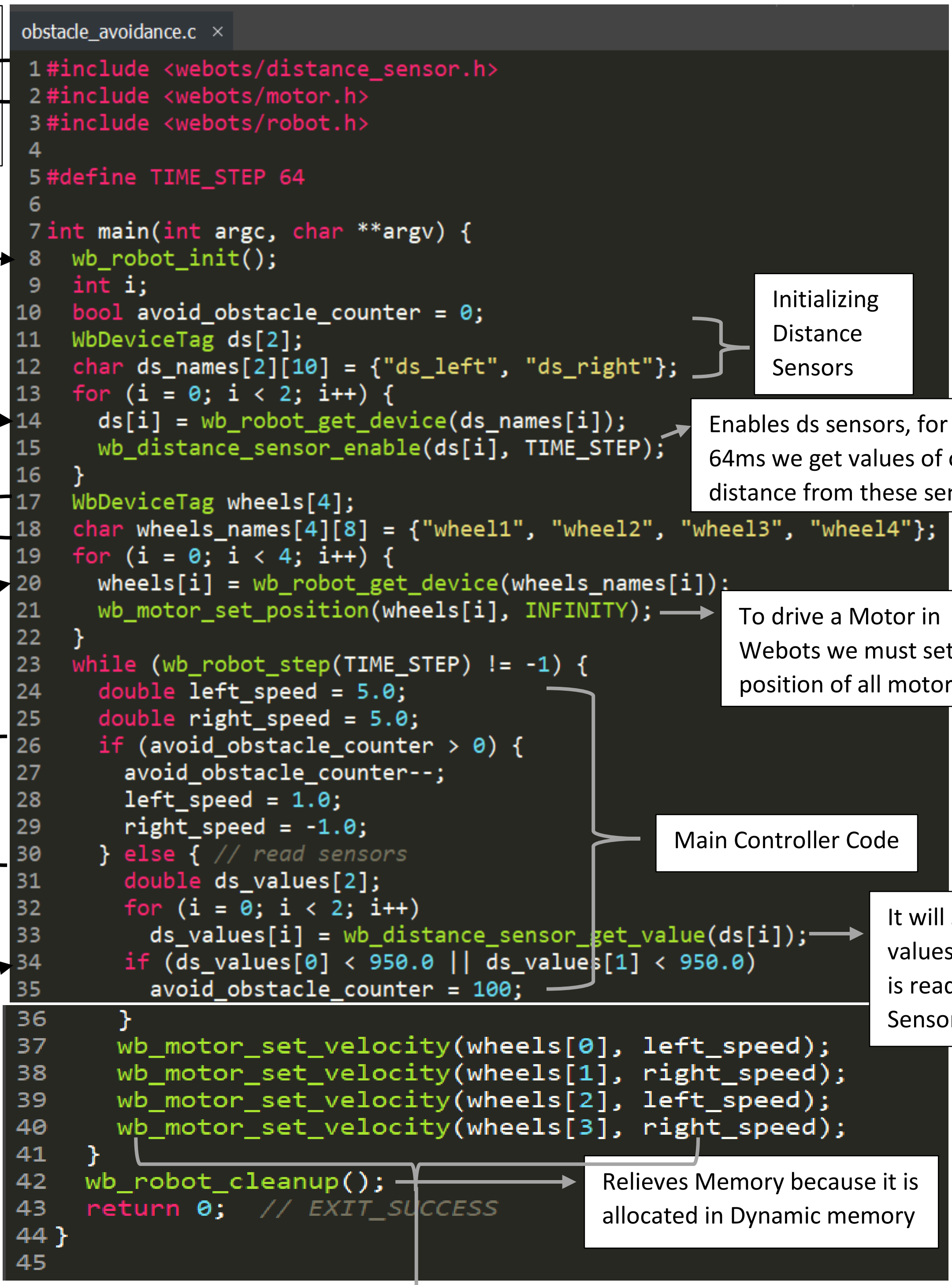


```
endPoint Solid
    translation -0.06 0 -0.05
    rotation -0.573 0.572 -0.587 4.2
```

Video Link

https://drive.google.com/file/d/1D1wwki6Q889jh5n3jpAmwiPbXKRxbCaf/view?usp=sharing

Controller C Language Code



```c
#include <webots/distance_sensor.h>
#include <webots/motor.h>
#include <webots/robot.h>

#define TIME_STEP 64

int main(int argc, char **argv) {
  wb_robot_init();
  int i;
  bool avoid_obstacle_counter = 0;
  WbDeviceTag ds[2];
  char ds_names[2][10] = {"ds_left", "ds_right"};
  for (i = 0; i < 2; i++) {
    ds[i] = wb_robot_get_device(ds_names[i]);
    wb_distance_sensor_enable(ds[i], TIME_STEP);
  }
  WbDeviceTag wheels[4];
  char wheels_names[4][8] = {"wheel1", "wheel2", "wheel3", "wheel4"};
  for (i = 0; i < 4; i++) {
    wheels[i] = wb_robot_get_device(wheels_names[i]);
    wb_motor_set_position(wheels[i], INFINITY);
  }
  while (wb_robot_step(TIME_STEP) != -1) {
    double left_speed = 5.0;
    double right_speed = 5.0;
    if (avoid_obstacle_counter > 0) {
      avoid_obstacle_counter--;
      left_speed = 1.0;
      right_speed = -1.0;
    } else { // read sensors
      double ds_values[2];
      for (i = 0; i < 2; i++)
        ds_values[i] = wb_distance_sensor_get_value(ds[i]);
      if (ds_values[0] < 950.0 || ds_values[1] < 950.0)
        avoid_obstacle_counter = 100;
    }
    wb_motor_set_velocity(wheels[0], left_speed);
    wb_motor_set_velocity(wheels[1], right_speed);
    wb_motor_set_velocity(wheels[2], left_speed);
    wb_motor_set_velocity(wheels[3], right_speed);
  }
  wb_robot_cleanup();
  return 0;  // EXIT_SUCCESS
}
```

Headers used to include/access the motor and sensor in webots

Initializing Motors and Sensors

Initializing names for ds sensors

Initializing Distance Sensors

Enables ds sensors, for every 64ms we get values of obstacle distance from these sensors

Initializing Wheels/Motors

Initializing names for wheels/motors

To drive a Motor in Webots we must set the position of all motor to INF

Till the counter gets "0" the robot will rotate with the given speed

Main Controller Code

It will set the values which is read by ds Sensors

Sensor values will be compared with a condition to avoid obstacle and the counter will be set to "100" if the obstacle is nearer to the robot

Relieves Memory because it is allocated in Dynamic memory

I will set the wheel velocities with the updated values of right wheel velocity and left wheel velocity