Webots Tutorial

diajukan untuk memenuhi UAS mata kuliah Robotika dan Sistem Cerdas

oleh:

MUHAMMAD RAKAN FAWWAZ (1103194031)

PROGRAM STUDI S1 TEKNIK KOMPUTER

FAKULTAS TEKNIK ELEKTRO

UNIVERSITAS TELKOM

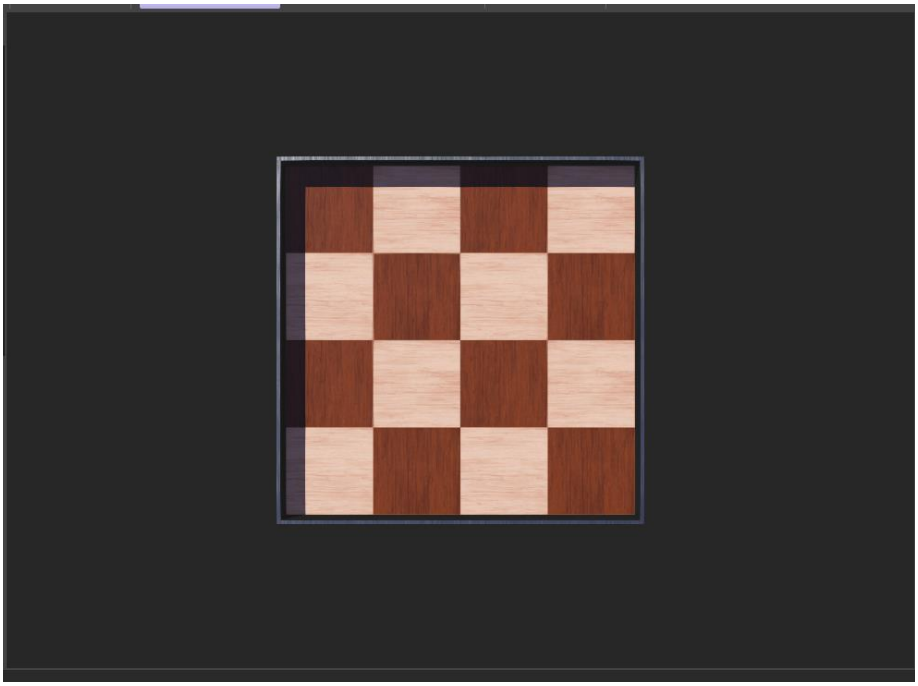
BANDUNG

2023

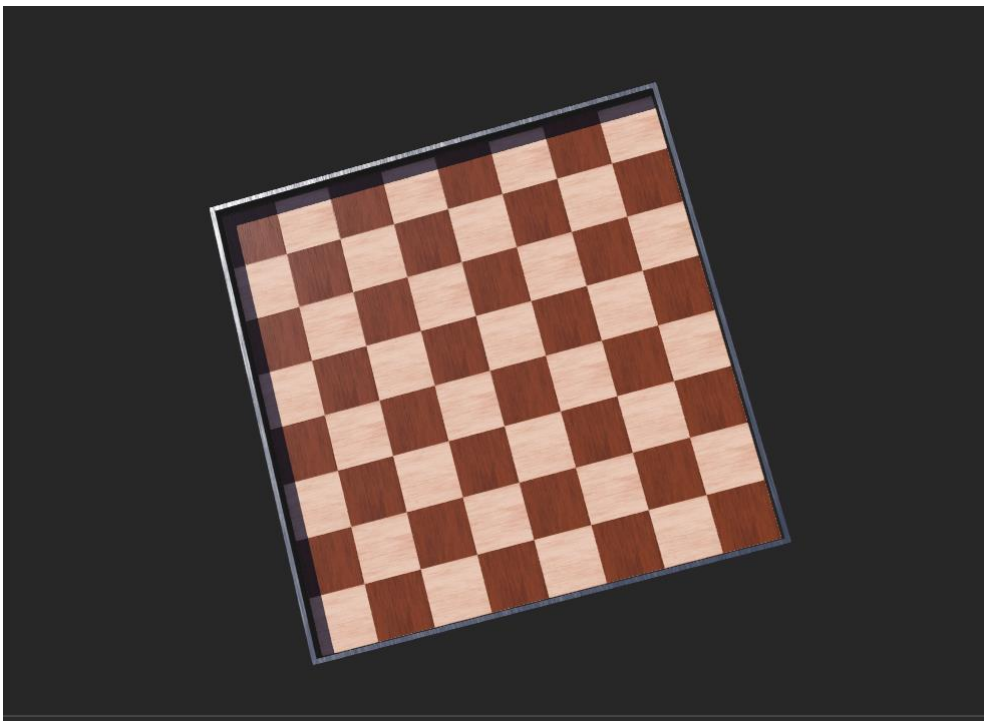
1. Your First Simulation in Webots

a. Create a New World

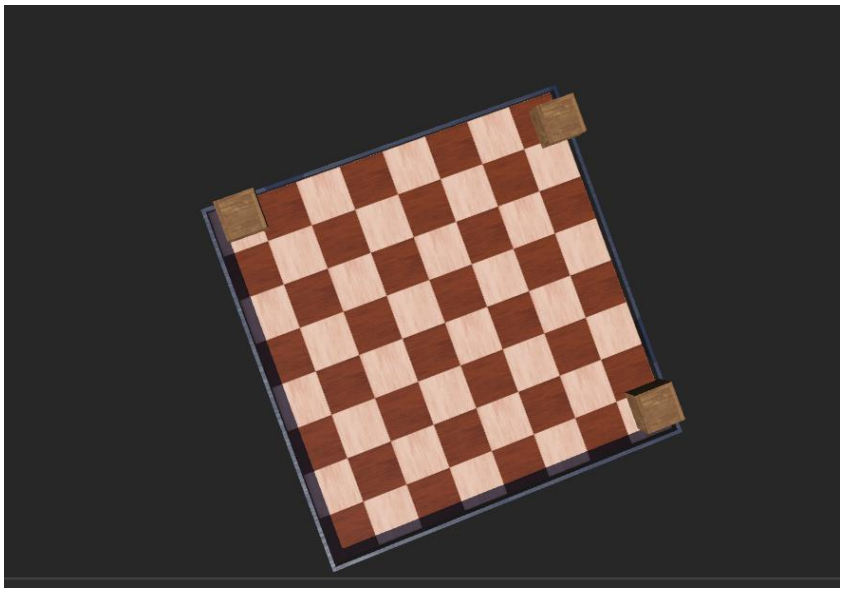
Gambar 1 Hands-on 2



Gambar 2 Hands-on 3

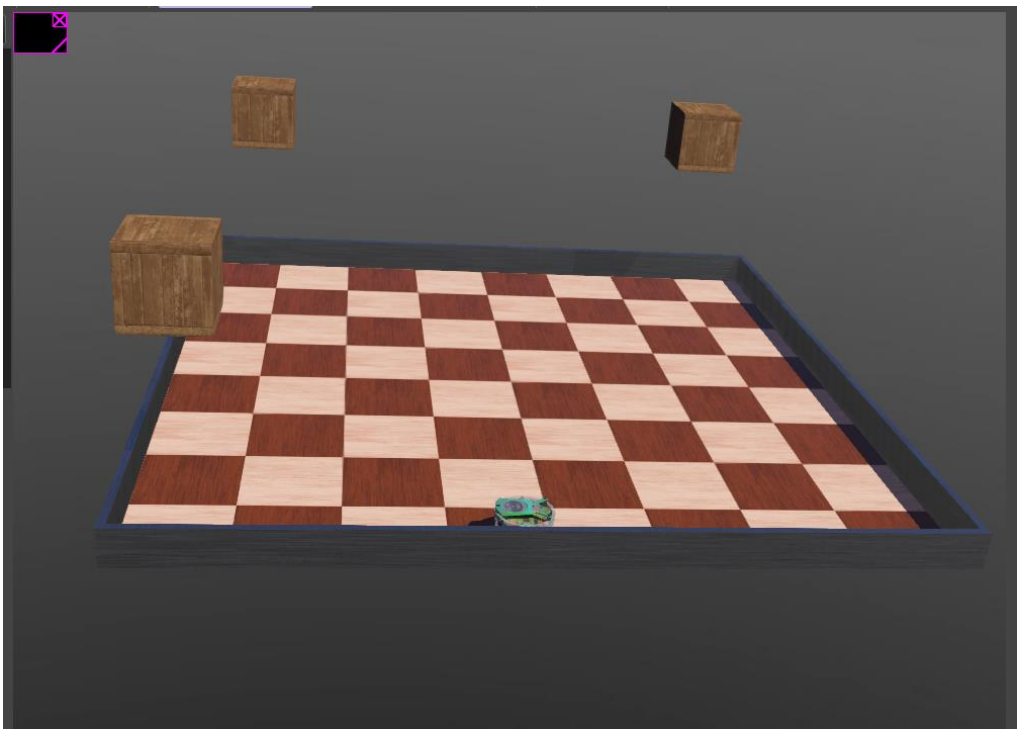


Gambar 3 Hands-on 4

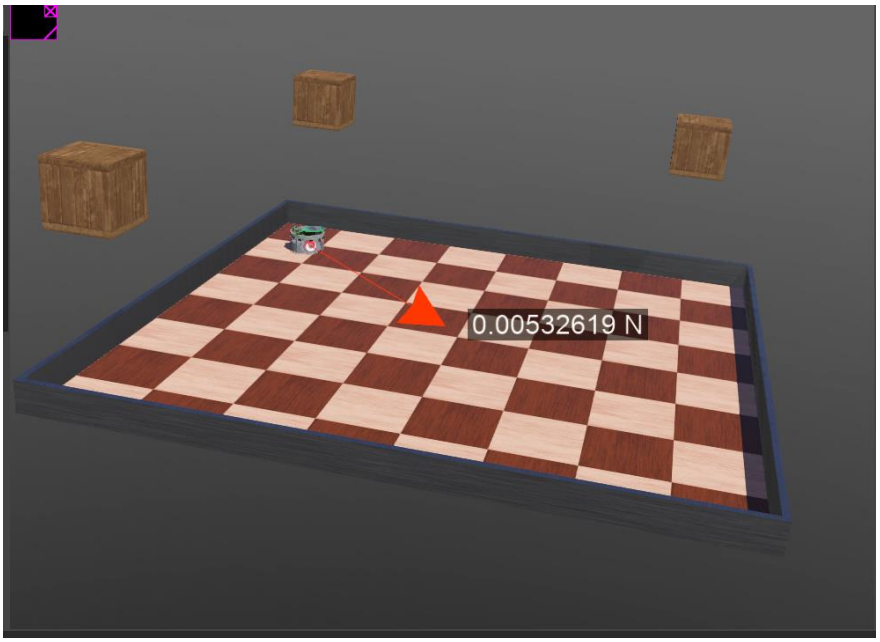


b. Add an e-puck Robot

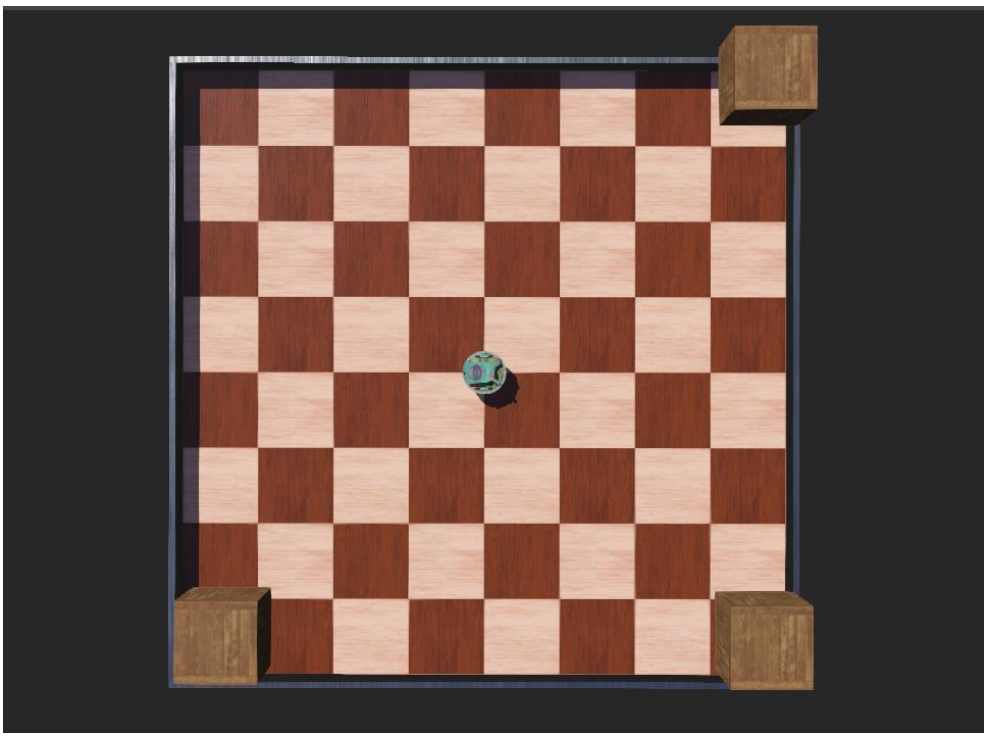
Gambar 4 Hands-on 5



Gambar 5 Hands-on 6

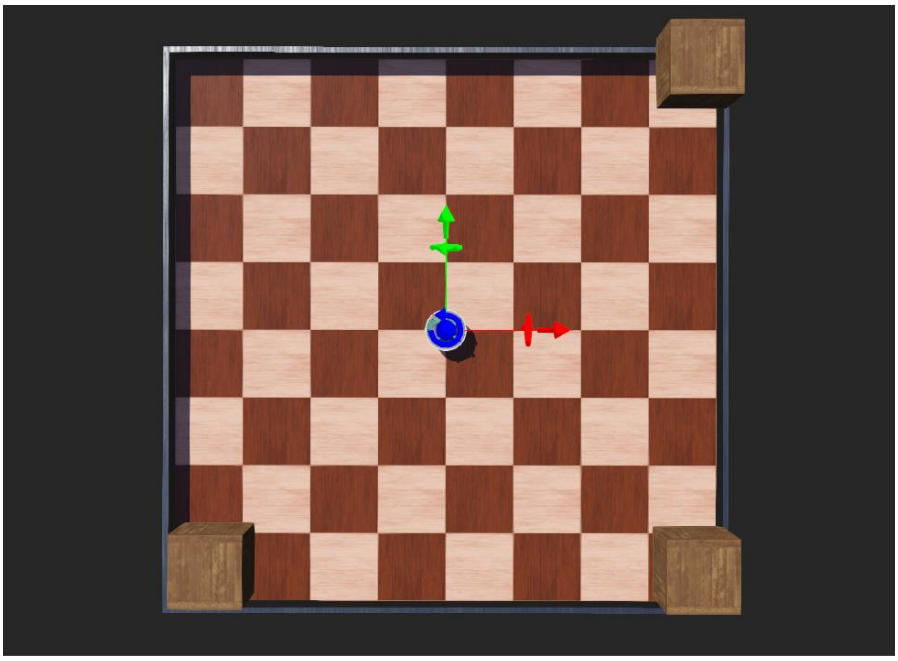


Gambar 6 Hands-on 7

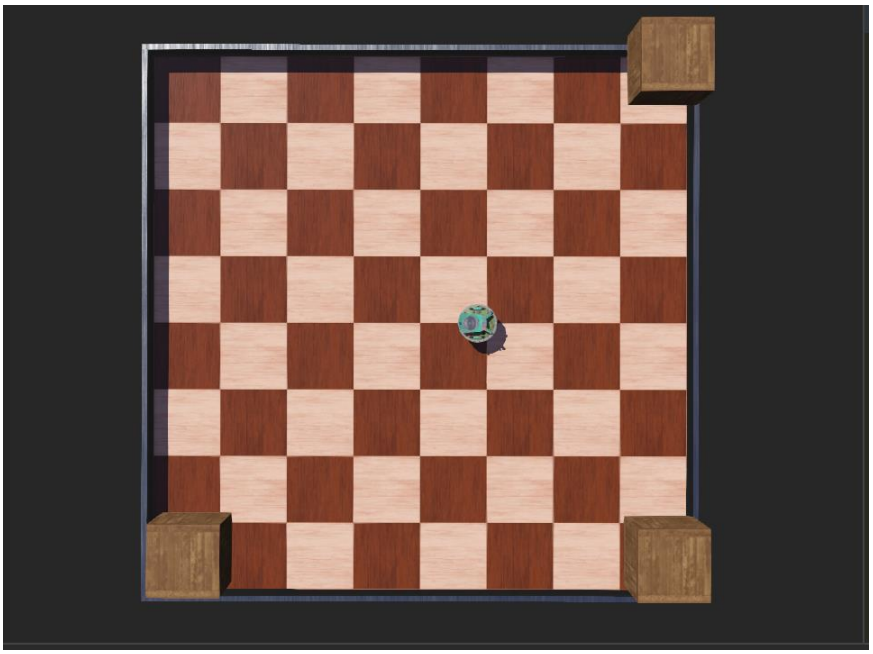


c. Create a New Controller

Gambar 7 Hands-on 8

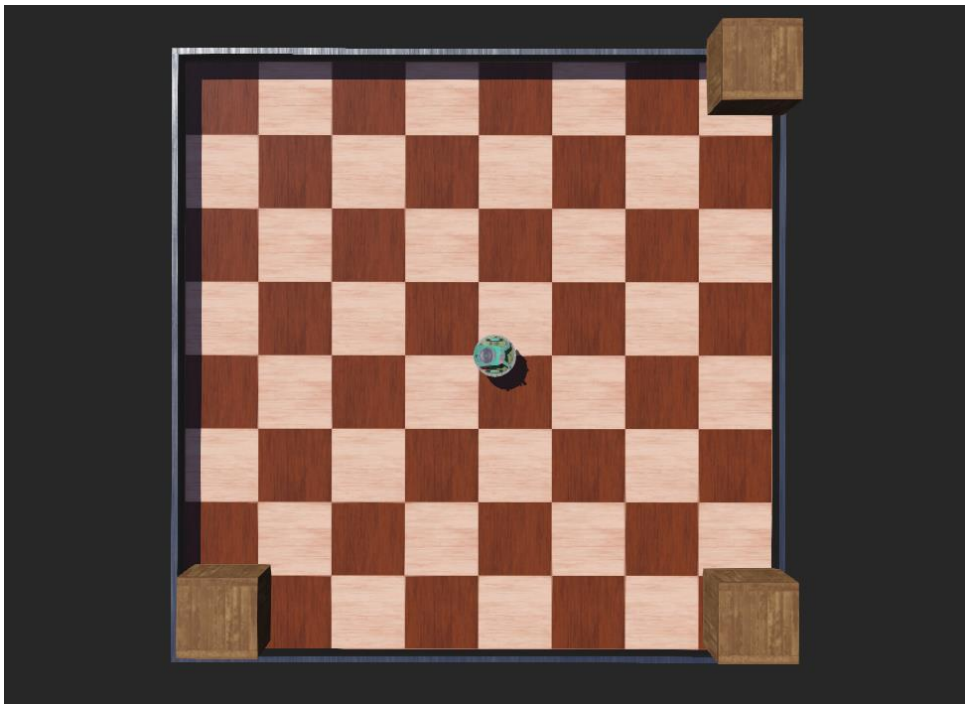


Gambar 8 Hands-on 9



d. Extend the Controller to Speed Control

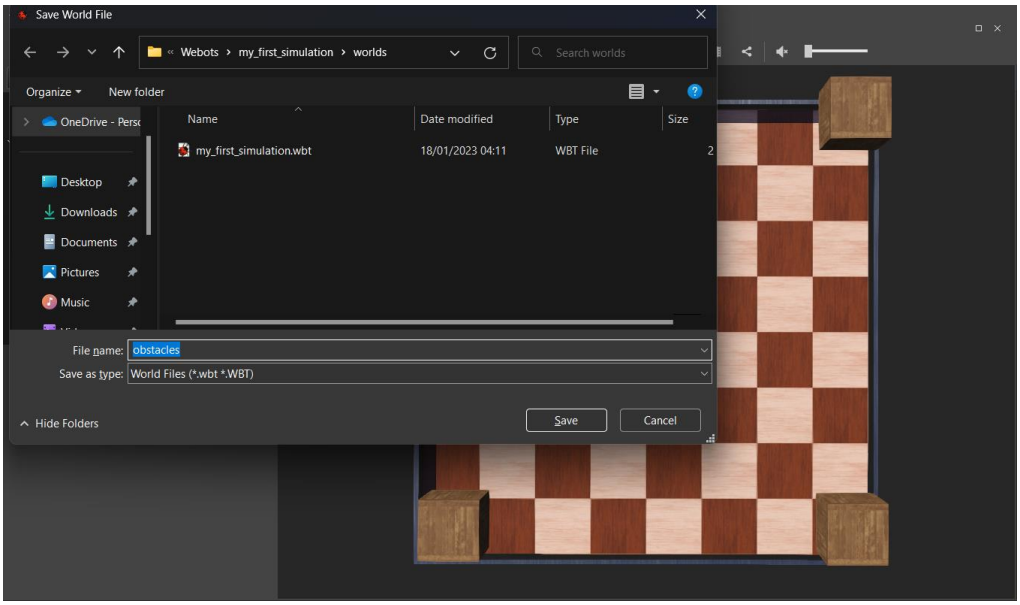
Gambar 9 Hands-on 10



2. Modification of the Environment

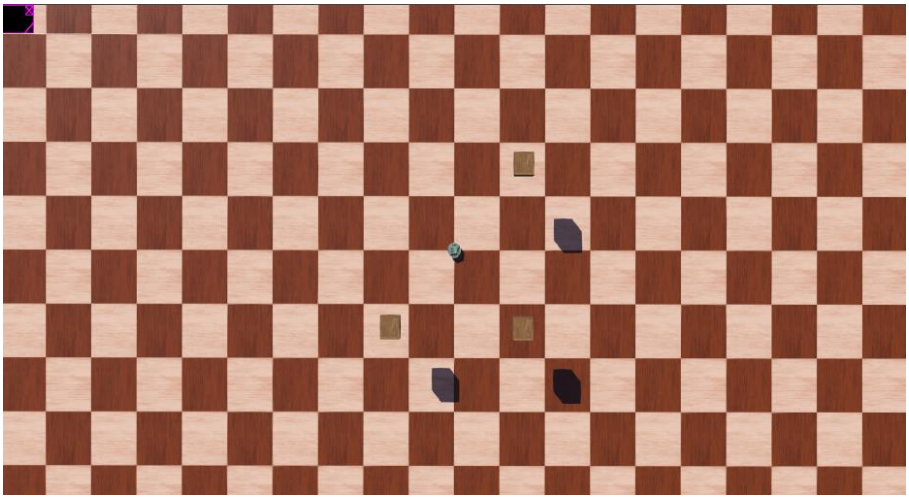
a. A New Simulation

Gambar 10 Hands-on 1



b. Modifying the Floor

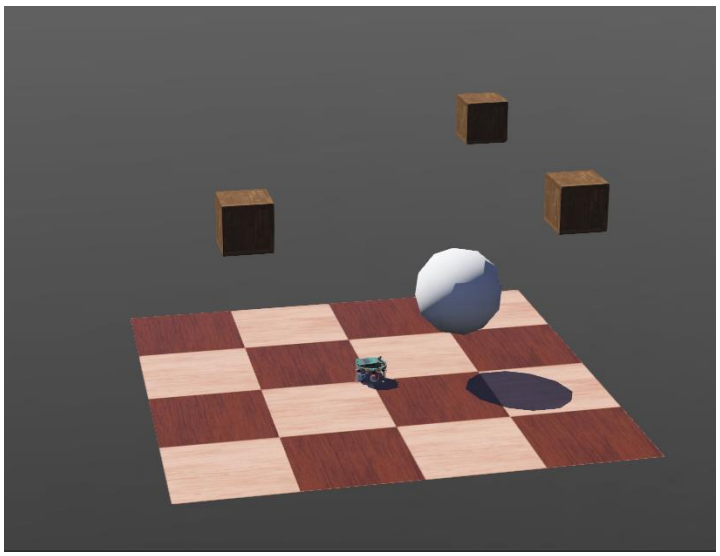
Gambar 11 Hands-on 2



ASGambar 12 Hands-on 3

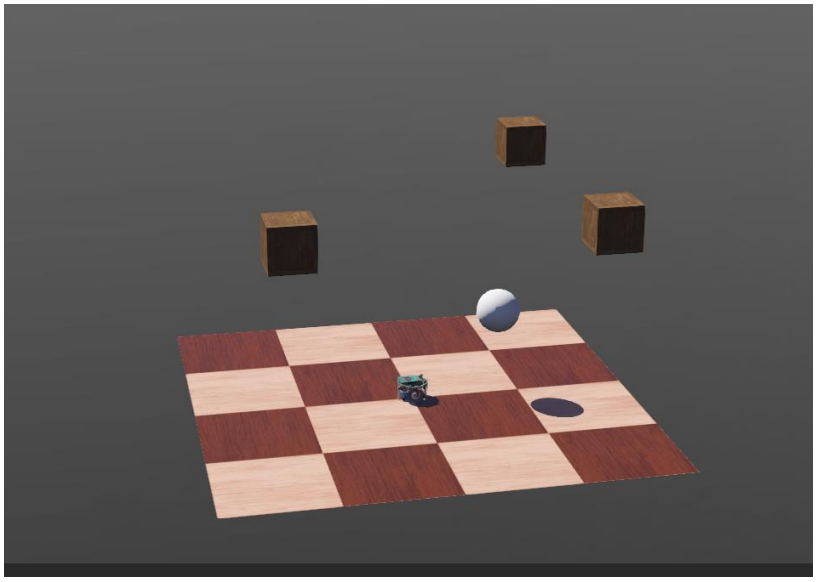
c. Create a Ball

Gambar 13 Hands-on 4



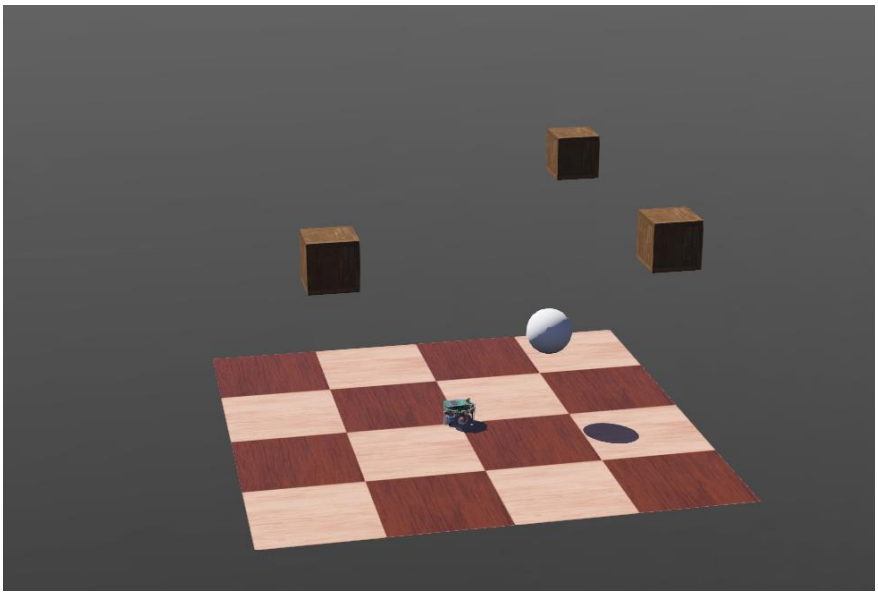
d. Geometries

Gambar 14 Hands-on 5

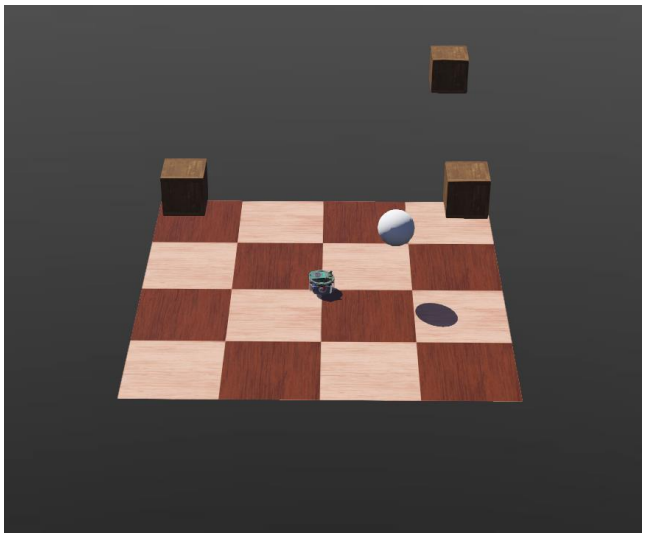


e. DEF-USE Mechanism

Gambar 15 Hands-on 6

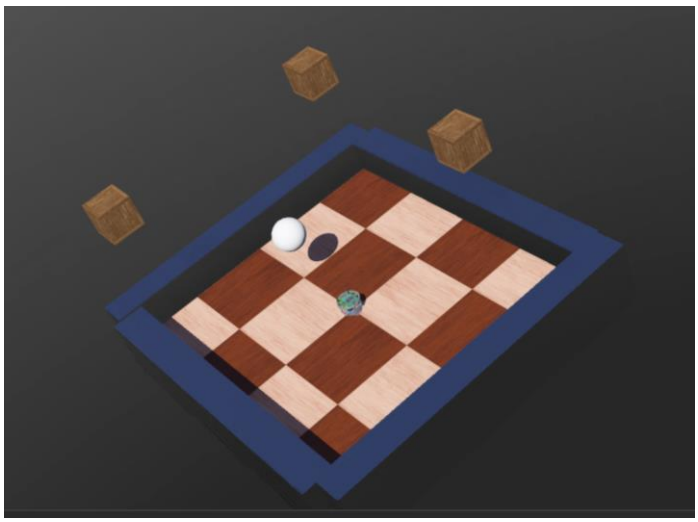


Gambar 16 Hands-on 7



f. Add Walls

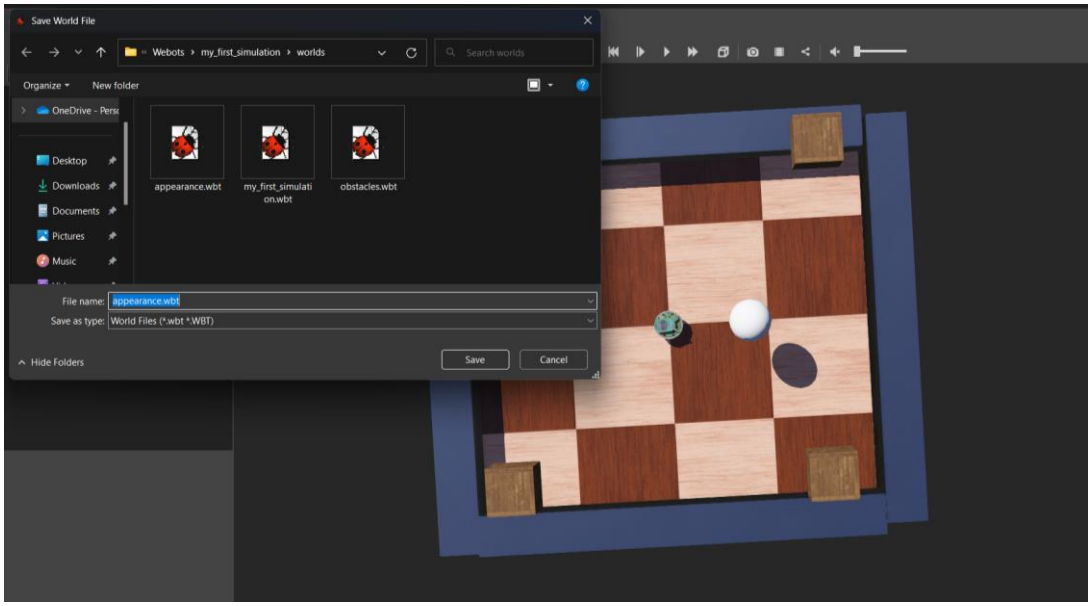
Gambar 17 Hands-on 8



3. Appearance

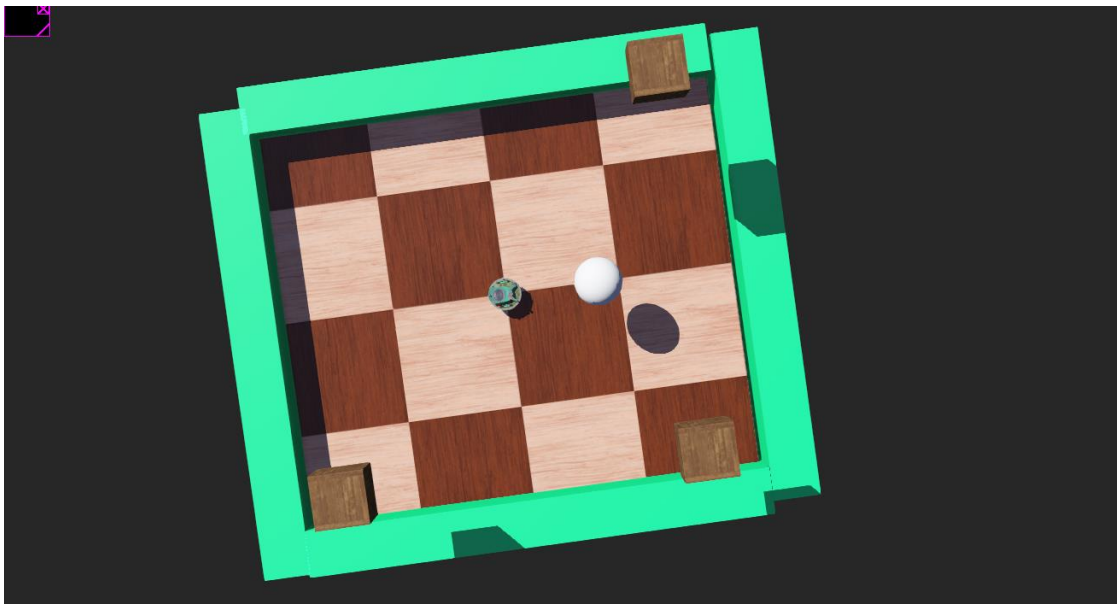
a. New Simulation

Gambar 18 Hands-on 1



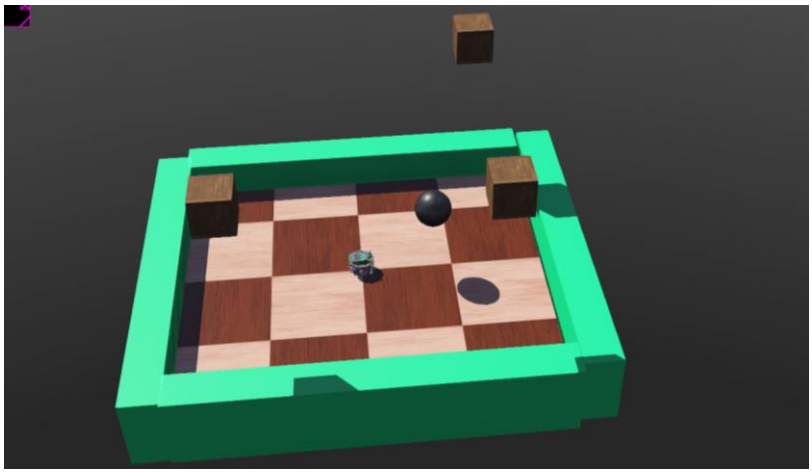
b. Modify the Appearance of the Walls

Gambar 19 Hands-on 2



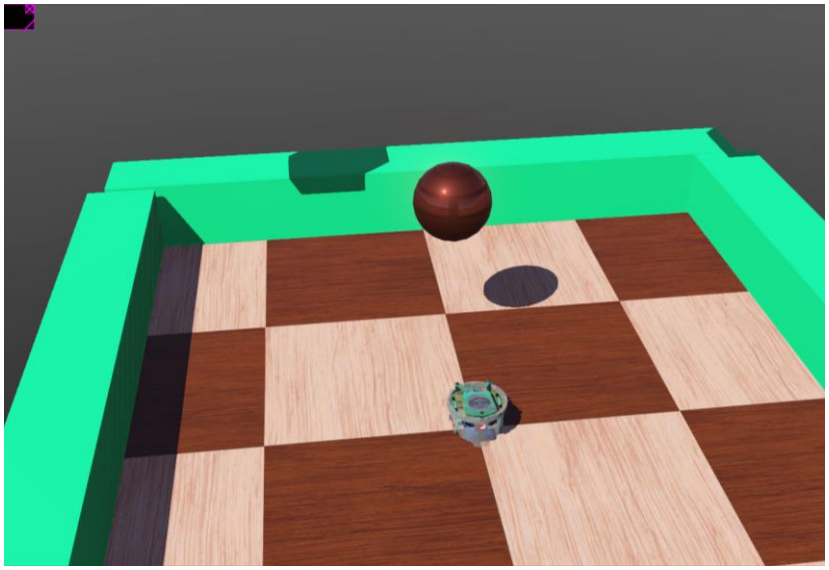
c. Add an Existing Appearance to the Ball

Gambar 20 Hands-on 3



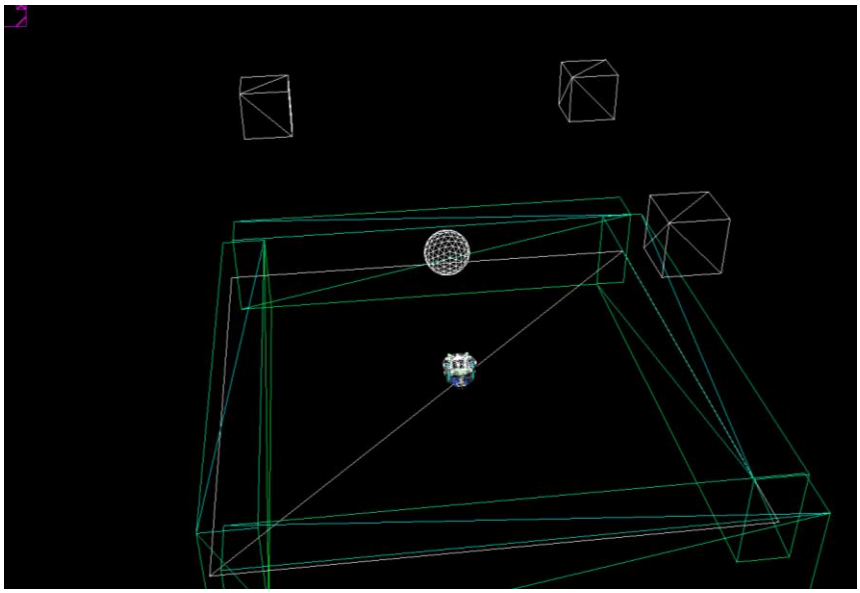
d. Add a Texture from Disk

Gambar 21 Hands-on 4



e. Rendering Options

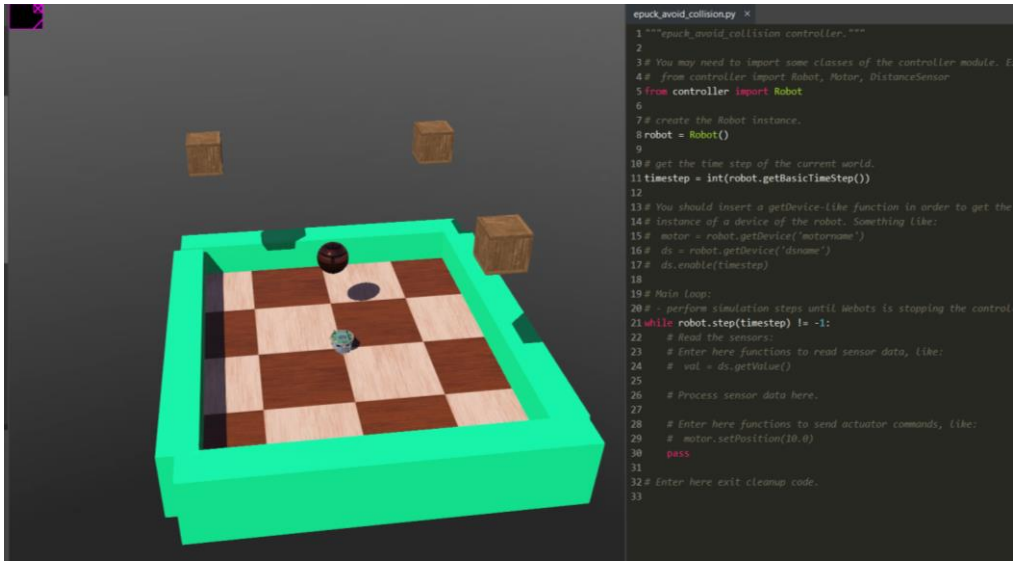
Gambar 22 Hands-on 5



4. More about Controllers

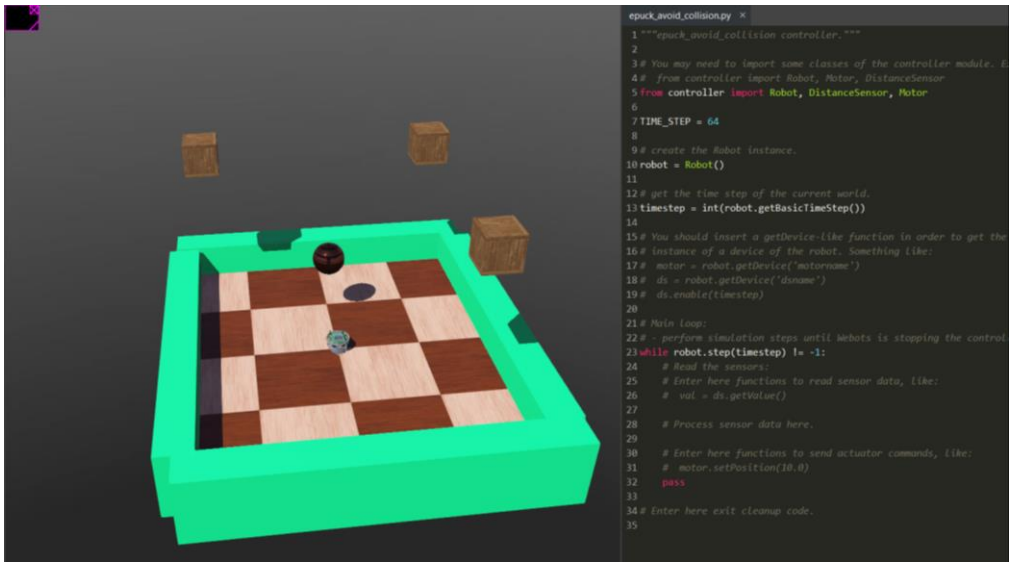
a. New World and New Controller

Gambar 23 Hands-on 1

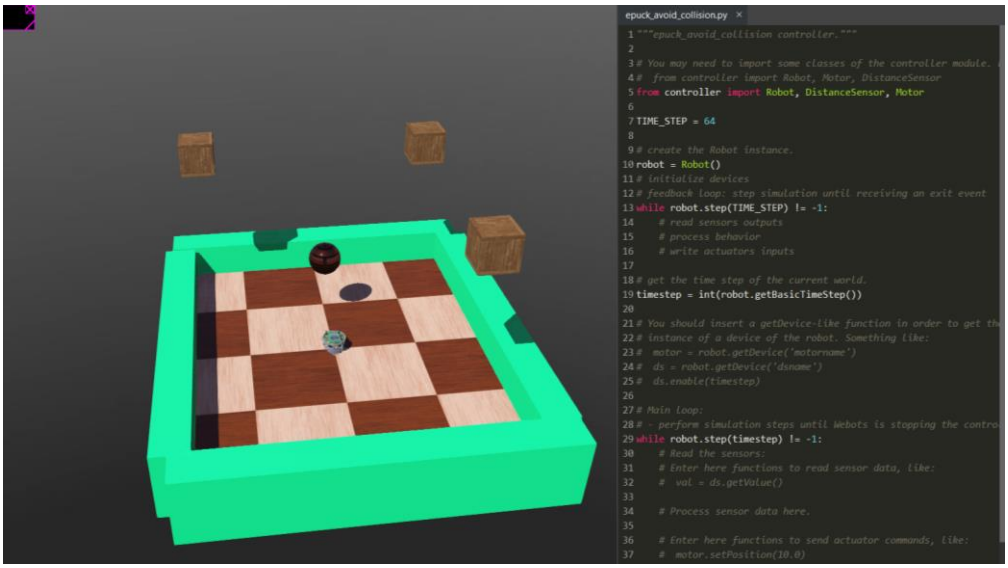


b. Program a Controller

Gambar 24 Hands-on 2



Gambar 25 Hands-on 3



```
epuck_avoid_collision.py
"""epuck_avoid_collision controller."""

# You may need to import some classes of the controller module.
#  from controller import Robot, Motor, DistanceSensor
from controller import Robot, DistanceSensor, Motor

TIME_STEP = 64

# create the Robot instance.
robot = Robot()
# initialize devices
# feedback loop: step simulation until receiving an exit event
while robot.step(TIME_STEP) != -1:
    # read sensors outputs
    # process behavior
    # write actuators inputs

# get the time step of the current world.
timestep = int(robot.getBasicTimeStep())

# You should insert a getDevice-like function in order to get th
# instance of a device of the robot. Something like:
#  motor = robot.getDevice('motorname')
#  ds = robot.getDevice('dsname')
#  ds.enable(timestep)

# Main Loop:
# - perform simulation steps until Webots is stopping the contro
while robot.step(timestep) != -1:
    # Read the sensors:
    # Enter here functions to read sensor data, like:
    #  val = ds.getValue()

    # Process sensor data here.

    # Enter here functions to send actuator commands, like:
    #  motor.setPosition(10.0)
```

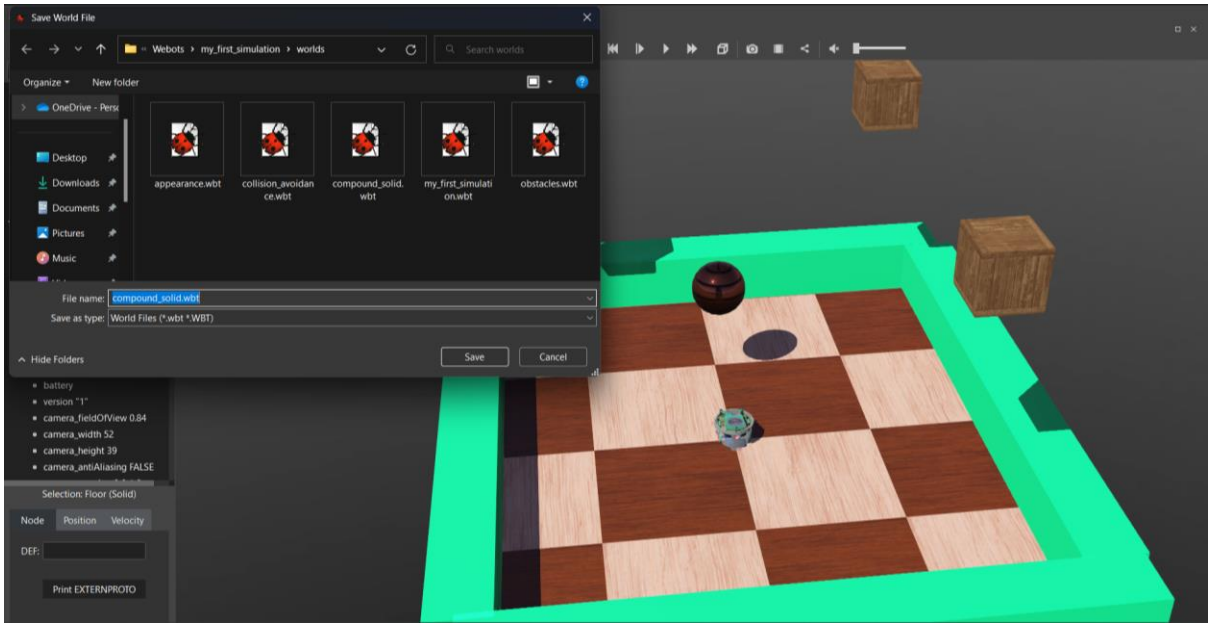
Gambar 26 Hands-on 4

```
epuck_avoid_collision.py
#  from controller import Robot, Motor, DistanceSensor
from controller import Robot, DistanceSensor, Motor

TIME_STEP = 64

MAX_SPEED = 6.28

# create the Robot instance.
robot = Robot()
# initialize devices
ps = []
psNames = [
    'ps0', 'ps1', 'ps2', 'ps3',
    'ps4', 'ps5', 'ps6', 'ps7'
]

for i in range(8):
    ps.append(robot.getDevice(psNames[i]))
    ps[i].enable(TIME_STEP)

leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')
leftMotor.setPosition(float('inf'))
rightMotor.setPosition(float('inf'))
leftMotor.setVelocity(0.0)
rightMotor.setVelocity(0.0)

# feedback loop: step simulation until receiving an exit event
while robot.step(TIME_STEP) != -1:
    # read sensors outputs
    psValues = []
    for i in range(8):
        psValues.append(ps[i].getValue())
    # process behavior
    # detect obstacles
right_obstacle = psValues[0] > 80.0 or psValues[1] > 80.0 or psV
left_obstacle = psValues[5] > 80.0 or psValues[6] > 80.0 or psVa
```

5. Compound Solid and Physics Attributes

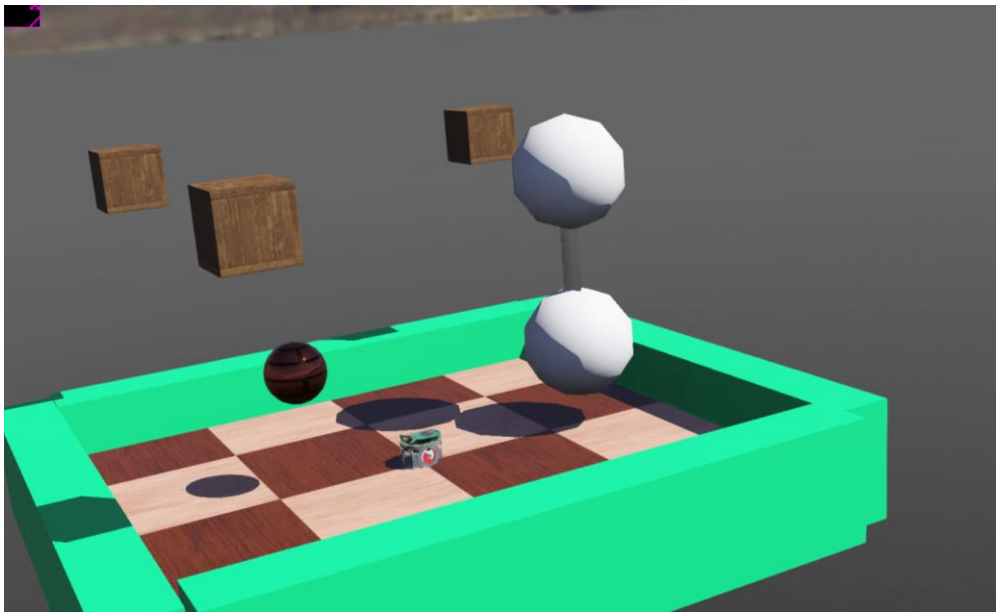
a. New Simulation

Gambar 27 Hands-on 1



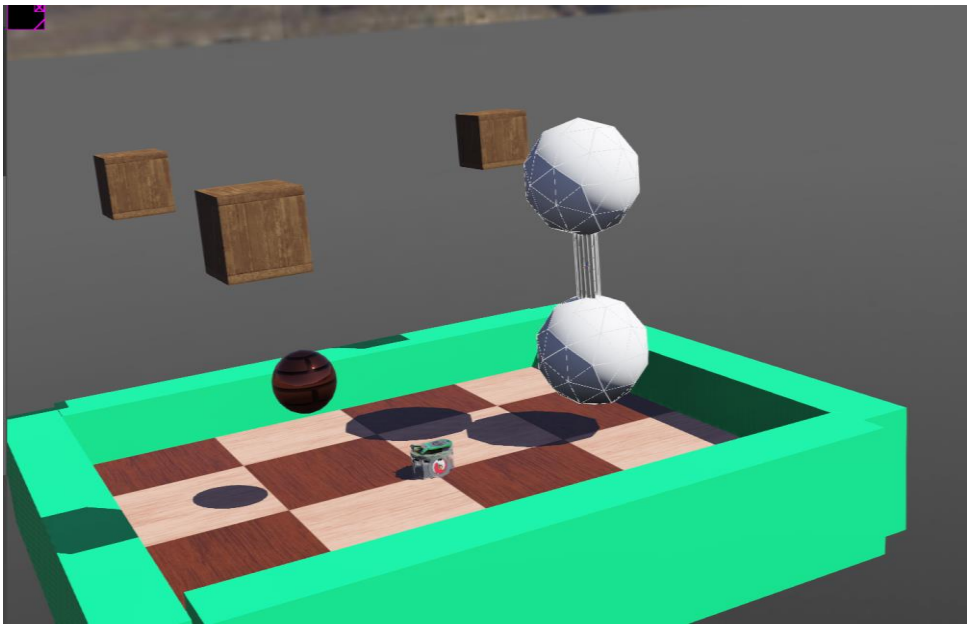
b. Compound Solid

Gambar 28 Hands-on 2

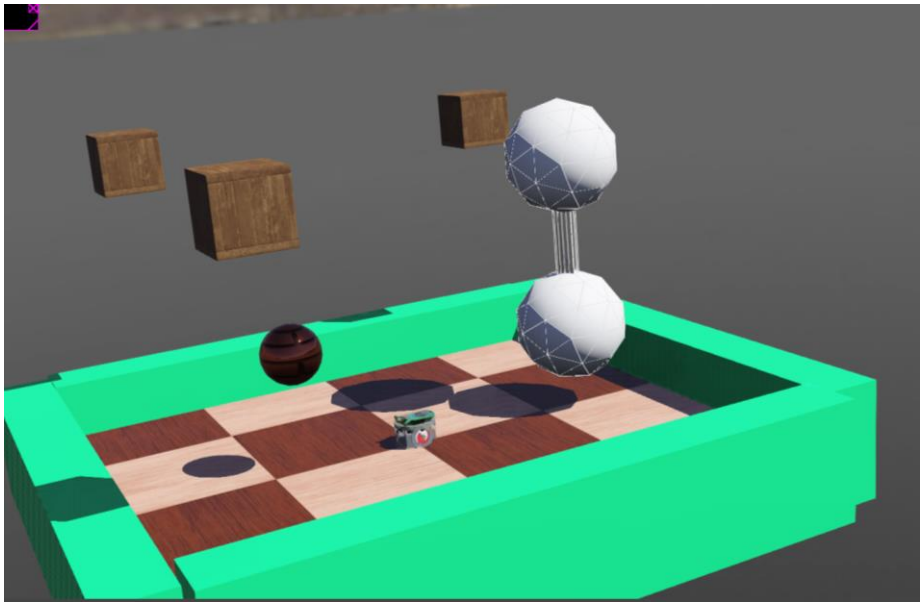


c. Physics Attributes

Gambar 29 Hands-on 3

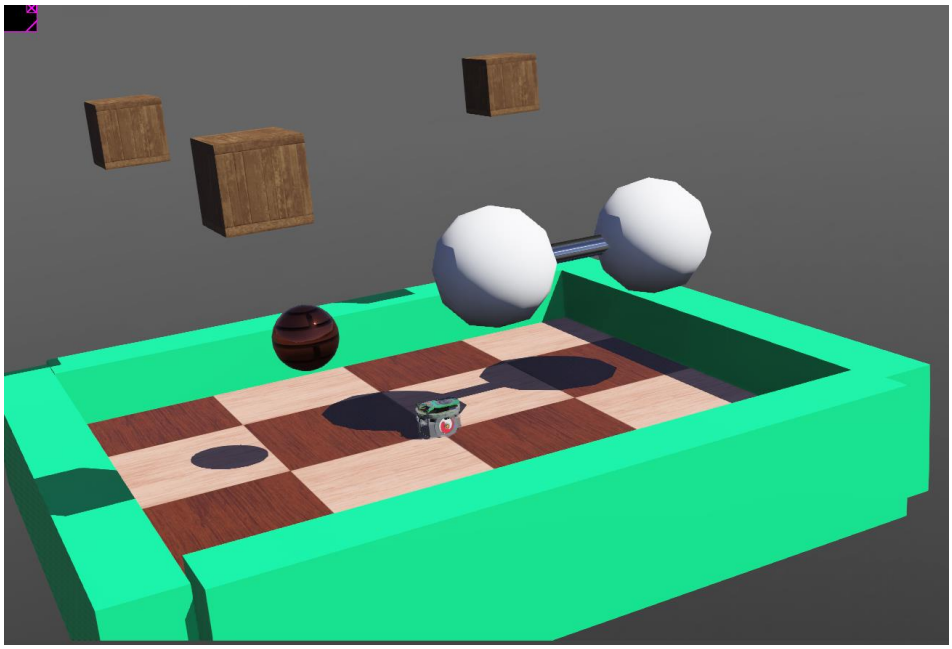


Gambar 30 Hands-on 4



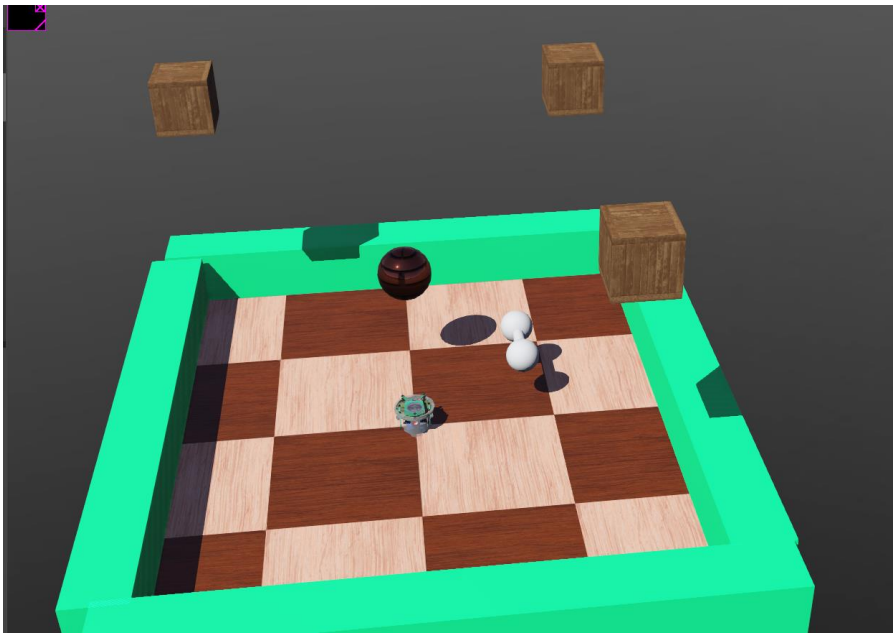
d. The Rotation Field

Gambar 31 Hands-on 5



e. Contacts

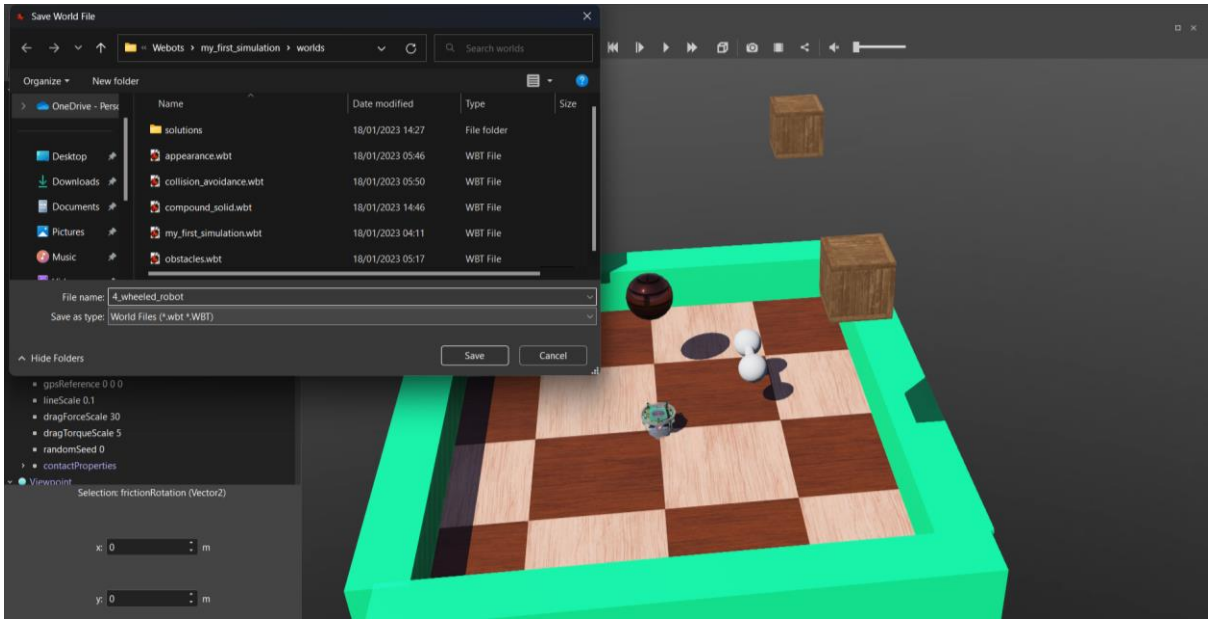
Gambar 32 Hands-on 6



6. 4-Wheeled Robot

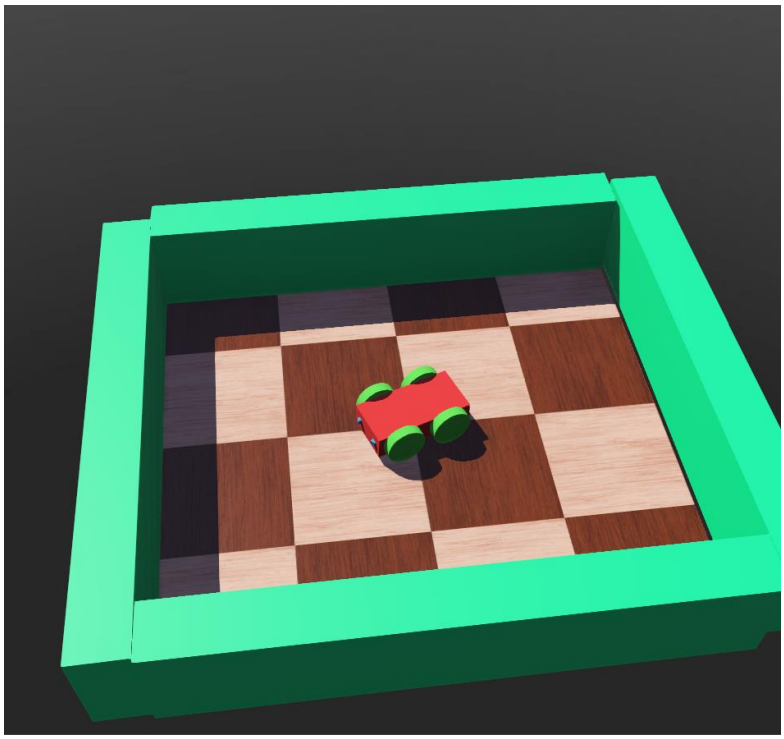
a. New Simulation

Gambar 33 Hands-on 1



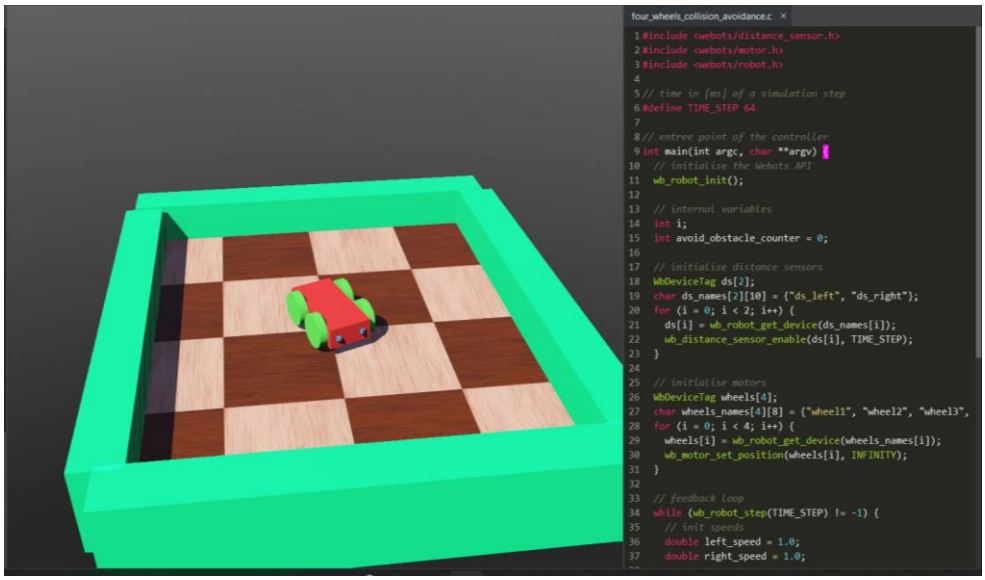
b. Separating the Robot in Solid Nodes

Gambar 34 Hands-on 2



c. Controller

Gambar 35 Hands-on 3

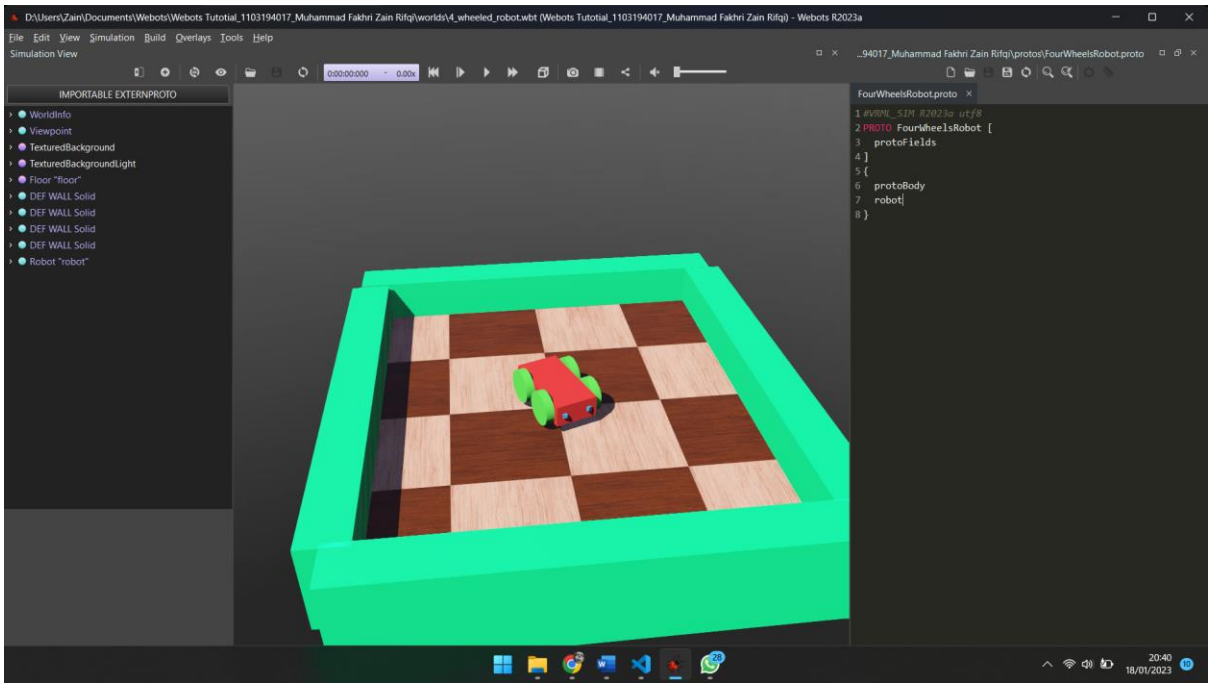


four_wheels_collision_avoidance.c

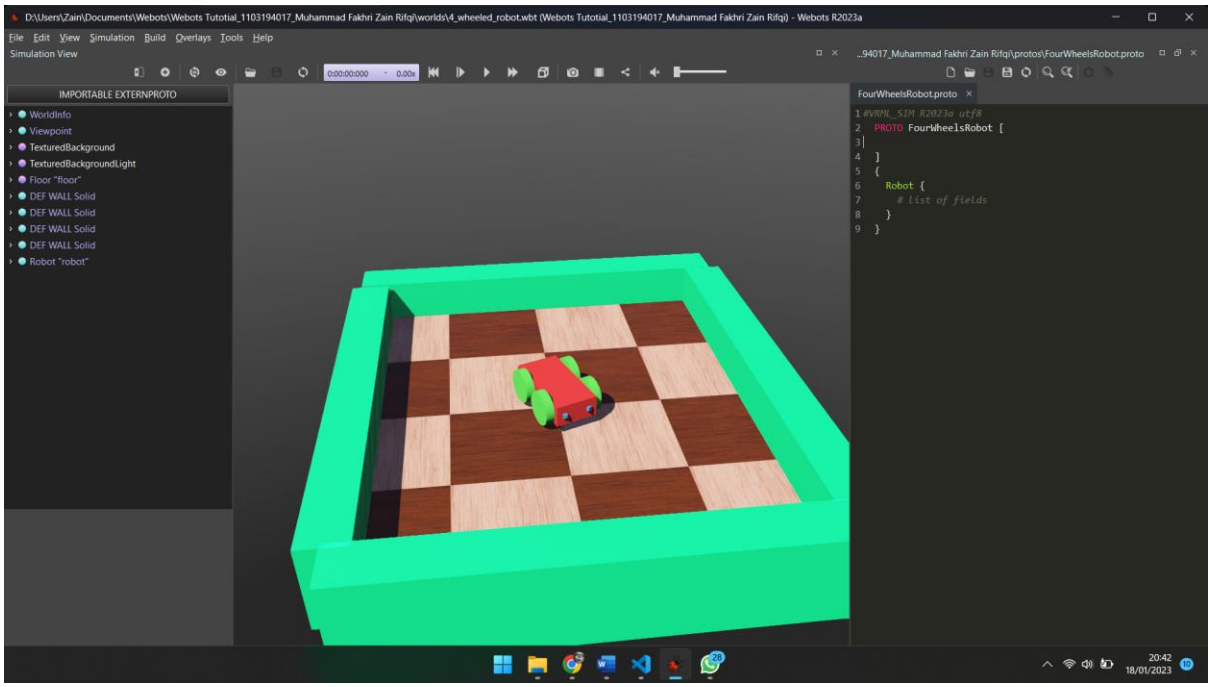
```c
#include <webots/distance_sensor.h>
#include <webots/motor.h>
#include <webots/robot.h>

// time in [ms] of a simulation step
#define TIME_STEP 64

// entree point of the controller
int main(int argc, char **argv) {
  // initialise the Webots API
  wb_robot_init();

  // internal variables
  int i;
  int avoid_obstacle_counter = 0;

  // initialise distance sensors
  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);
  }

  // initialise motors
  WbDeviceTag wheels[4];
  char wheels_names[4][8] = {"wheel1", "wheel2", "wheel3",
  for (i = 0; i < 4; i++) {
    wheels[i] = wb_robot_get_device(wheels_names[i]);
    wb_motor_set_position(wheels[i], INFINITY);
  }

  // feedback loop
  while (wb_robot_step(TIME_STEP) != -1) {
    // init speeds
    double left_speed = 1.0;
    double right_speed = 1.0;
```

7. Your First PROTO

a. Copy the Robot Definition



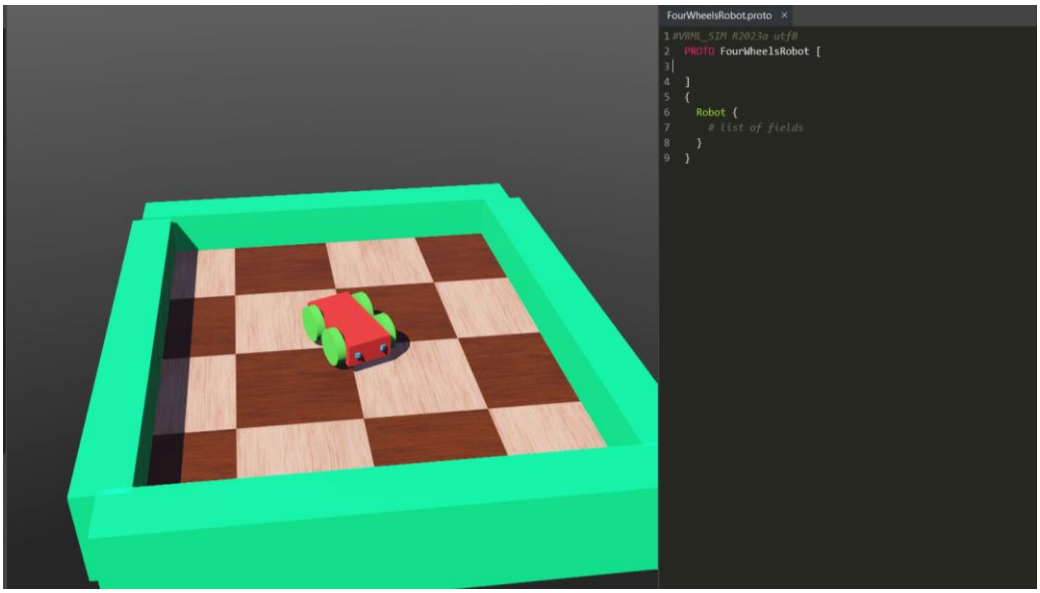
Gambar 36 Hands-on 1



Gambar 37 Hands-on 2

b. Use the PROTO Node

Gambar 38 Hands-on 3



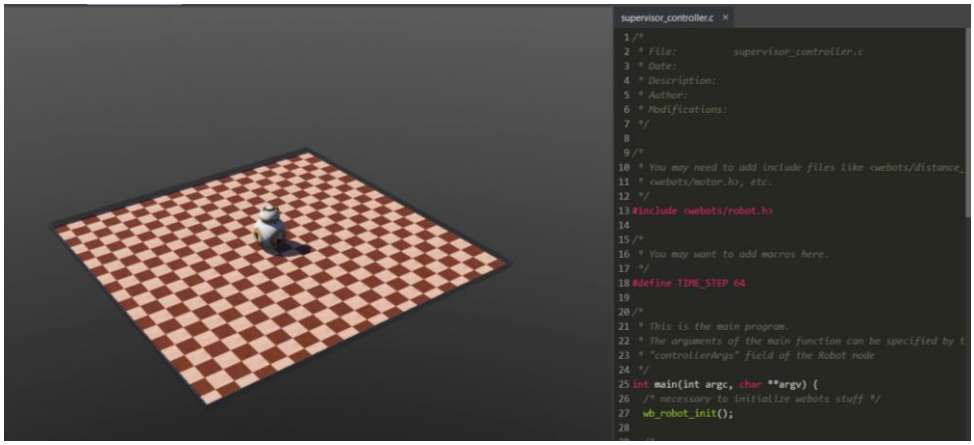
c. Adding Fields

Gambar 39 Hands-on 4

8. the Supervisor

a. Setting up the Environment and Adding a Supervisor

Gambar 40 Hands-on 1



b. Putting Everything Together

Gambar 41 Hands-on 2