Nama : Nurul Amelia

NIM : 1103194032

[UAS Robot Autonomy] – Webots Tutorial.

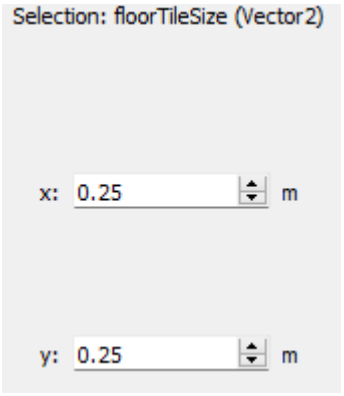
[Tutorial – 1]

Simulasi pertama akan berisi *environment* sederhana yaitu arena dengan lantai dan dinding, beberapa kotak, robot e-puck dan program pengontrol yang akan membuat robot bergerak.

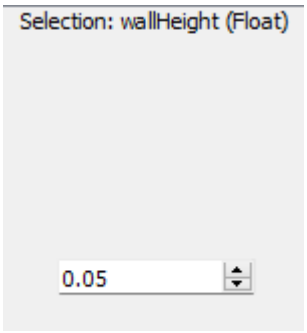
Robot e-puck.

Pada robot e-puck mempunyai 2 motor yang berada disebelah kanan dan kiri (right motor and left motor) yang berfungsi sebagai *actuator* pada robot tersebut. Motor sebagai *actuator* untuk memuat roda robot. Selain itu, robot e-puck juga mempunyai beberapa sensor diantaranya adalah sensor *accelerometer*, yang berfungsi sebagai sensor untuk mengukur percepatan atau perubahan kecepatan dari robotnya. Kemudian, terdapat beberapa sensor seperti *Ultrasound Sensor* yang digunakan untuk mengukur jarak (terdapat 8 buah sensor *ultrasound* yang terdapat pada robot e-puck). Pada robot e-puck juga terdapat kamera dengan revolusi 52 x 39 pixel.

1. Ubah *floorTileSize* pada *RectangleArena* menjasi 0,25 0,25 agar tampilan lebih terlihat seperti 3D.



2. Agar dinding arena terlihat lebih rendah ubah *wallHeight* di *RectangleArena* menjadi 0,05.



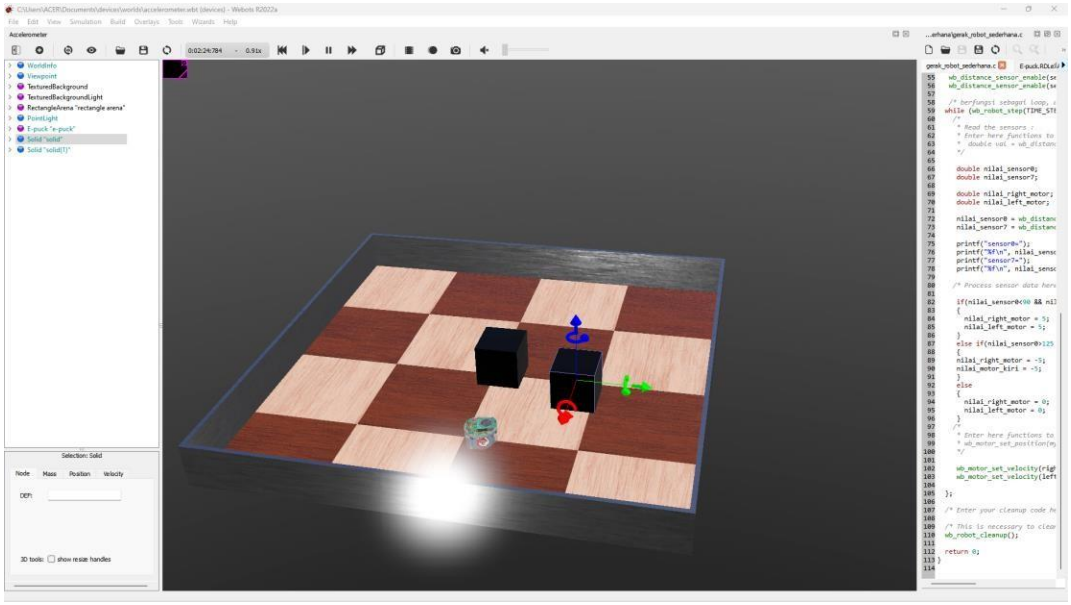
Sebelumnya saya pernah mencoba agar robot e-puck bergerak dengan menghindari rintangan, sehingga pada tutorial-1 ini, saya menggunakan program tersebut. Tetapi, untuk UI nya saya mengikuti tutorial dengan beberapa perubahan. Untuk penjelasannya sebagai berikut :

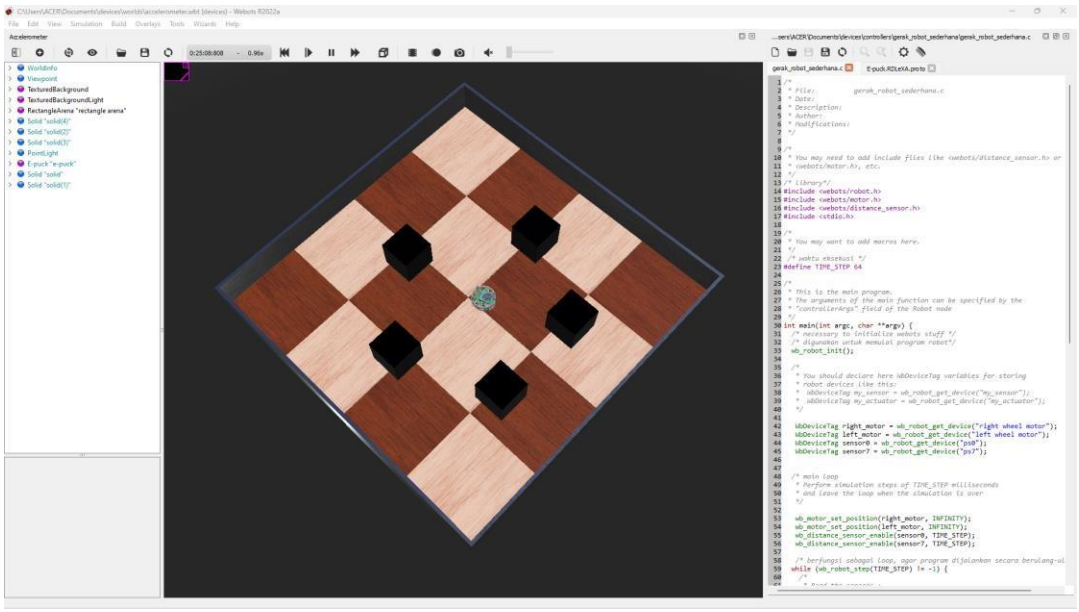
1. Pada robot, terdapat penambahan *source code* seperti dibawah ini :

```
else if(nilai_sensor0>125 || nilai_sensor7>125)
{
nilai_right_motor = -5;
nilai_motor_kiri = -5;
}
```

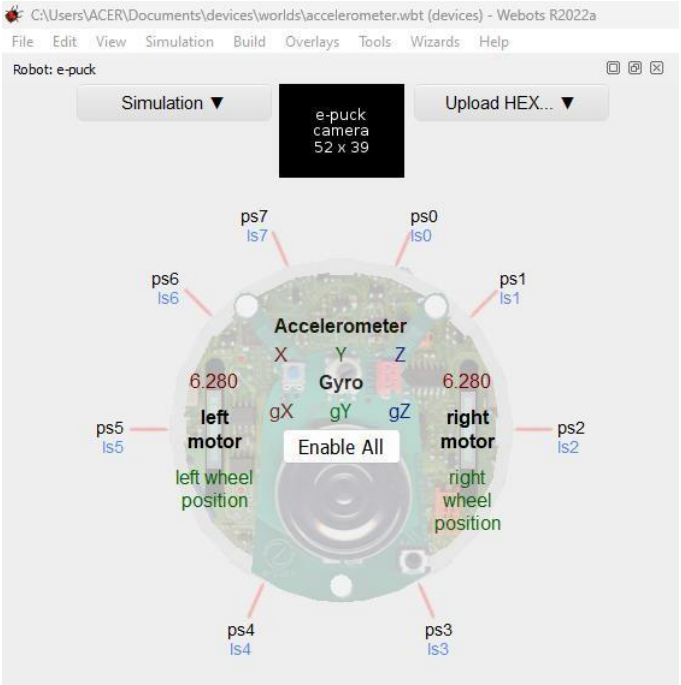
Jika halangannya terlalu dekat maka mundur, jika bukan dua-duanya maka berhenti.

2. Menambahkan objek :





3. Menghindari tabrakan dengan berbelok.



Jika ingin mengindari rintangan idealnya menggunakan lebih dari 2 sensor, sehingga tidak hanya bisa menghindari rintangan di depannya saja melainkan disampingnya. Disini menggunakan 4 buah sensor yaitu sensor 0, 1, 6 dan 7.

4. Mnambahkan sensor 1 dan 6.

```
WbDeviceTag right_motor = wb_robot_get_device("right wheel motor");
WbDeviceTag left_motor = wb_robot_get_device("left wheel motor");
WbDeviceTag sensor0 = wb_robot_get_device("ps0");
WbDeviceTag sensor7 = wb_robot_get_device("ps7");
WbDeviceTag sensor1 = wb_robot_get_device("ps1");
WbDeviceTag sensor6 = wb_robot_get_device("ps6");
```

5. Melakukan *inable sensor* yang aktif.

```
wb_motor_set_position(right_motor, INFINITY);
wb_motor_set_position(left_motor, INFINITY);
wb_distance_sensor_enable(sensor0, TIME_STEP);
wb_distance_sensor_enable(sensor7, TIME_STEP);
wb_distance_sensor_enable(sensor1, TIME_STEP);
wb_distance_sensor_enable(sensor6, TIME_STEP);
```

6. Melakukan penambahan pendefisian sensor.

```
double nilai_sensor0;
double nilai_sensor7;
double nilai_sensor1;
double nilai_sensor6;
```

Membaca nilai sensor.

```
nilai_sensor0 = wb_distance_sensor_get_value(sensor0);
nilai_sensor7 = wb_distance_sensor_get_value(sensor7);
nilai_sensor1 = wb_distance_sensor_get_value(sensor1);
nilai_sensor6 = wb_distance_sensor_get_value(sensor6);
```

7. Menampilkan nilai sensor.

```
printf("sensor0=");
printf("%f\n", nilai_sensor0);
printf("sensor7=");
printf("%f\n", nilai_sensor7);

printf("sensor1=");
printf("%f\n", nilai_sensor1);
printf("sensor6=");
printf("%f\n", nilai_sensor6);
```

8. Logikanya : selama di depan dan di samping robot (ke-4 sensor tidak mendeteksi objek atau objek makin jauh, maka robot akan diperintahkan untuk maju).

```
if(nilai_sensor0<90 && nilai_sensor7<90 && nilai_sensor1<90 && nilai_sensor6<90)
```

Jika ingin memutar ditempat, maka kecepatan nilai_right_motor dan nilai_left_motor harus sama, tetapi arahnya berbeda.

Jika belok kanan, roda kanan harus maju, roda kiri harus mundur. Sehingga, nilai motor kanan positif dan nilai motor kiri negatif.

```
else
{
  nilai_right_motor = 5;
  nilai_left_motor = -5;
}
```

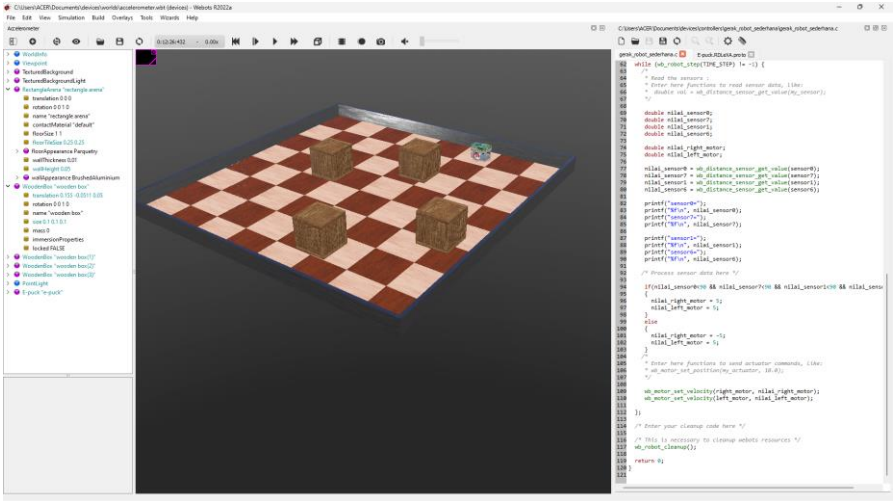
Penjelasan :

Jika tidak ada objek dimanapun, didepan ataupun kanan kiri depan robotnya, maka robot diperbolehkan maju. Tapi, jika ada objek di depan kanan kirinya, maka robotnya jangan maju dan harus memutar. Jika masih ada objek memutar terus, setelah tidak ada objek didepan atau samping kanan kirinya baru robot tersebut akan maju.

9. Jika robot ingin berbelok ke kanan, maka nilainya tinggal ditukar saja.

```
else
{
  nilai_right_motor = -5;
  nilai_left_motor = 5;
}
```

10. Mengganti box sebelumnya dengan WoodenBox, seperti gambar dibawah ini :

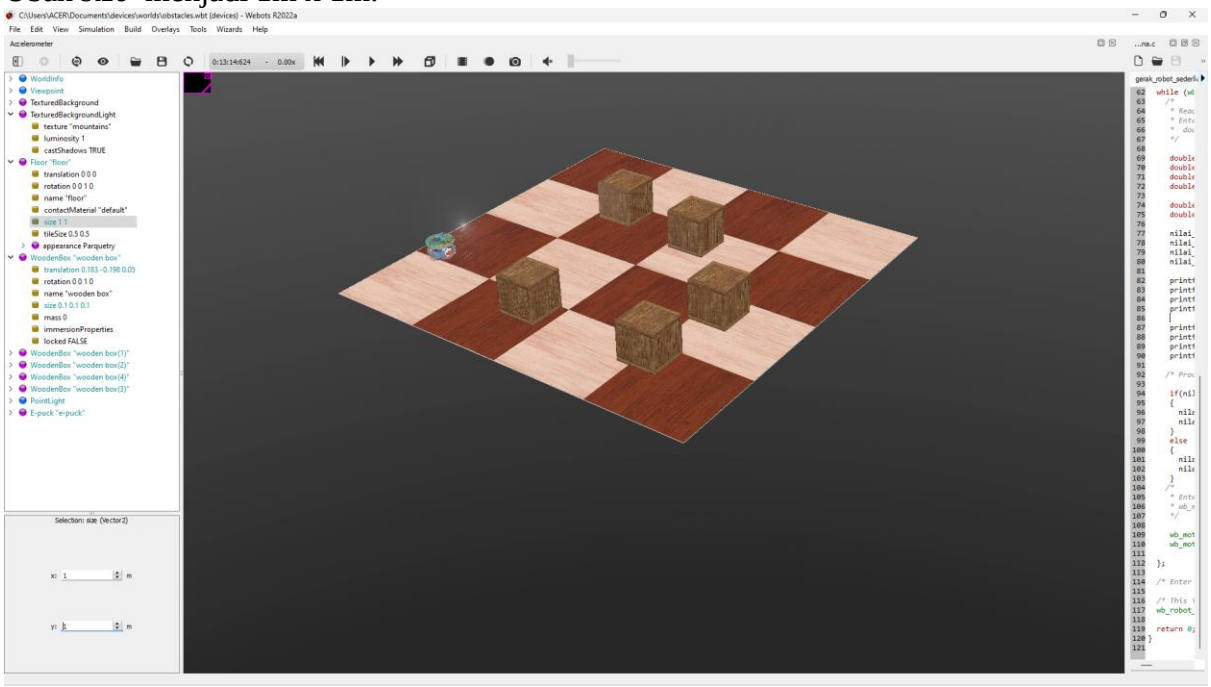


WoodenBox terletak di *PROTO nodes* (*Webots Projects*) / *objects* / *factory* / *containers* /

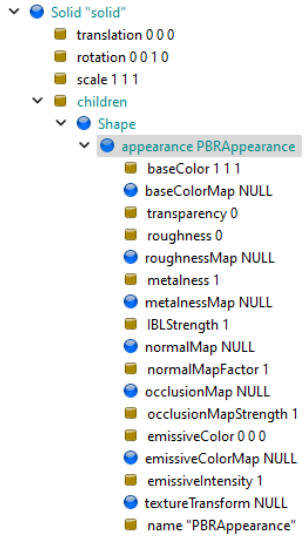
WoodenBox (*Solid*).

[Tutorial – 2]

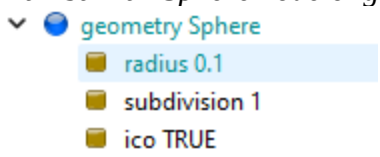
1. Hapus *RectangleArena* dan ubah di *TexturedBackgroundLight* node dengan menambahkan *Add button* dan pilih *PROTO nodes* (*Webots Projects*) / *objects* / *floors* / *Floor* (*Solid*). Ubah *size* menjadi 1m x 1m.



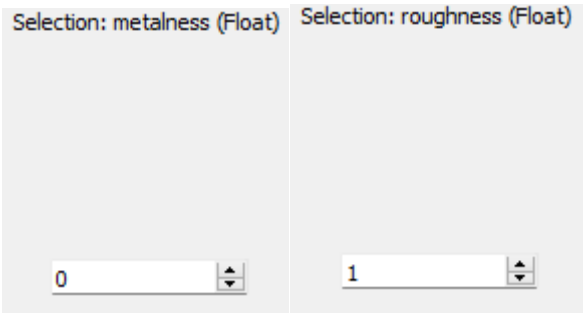
2. Menambahkan *Bases nodes* di *Solid node*. Pilih *children* dan tambahkan *Shape node*, setelah itu tambahkan *PBRAppearance node*.



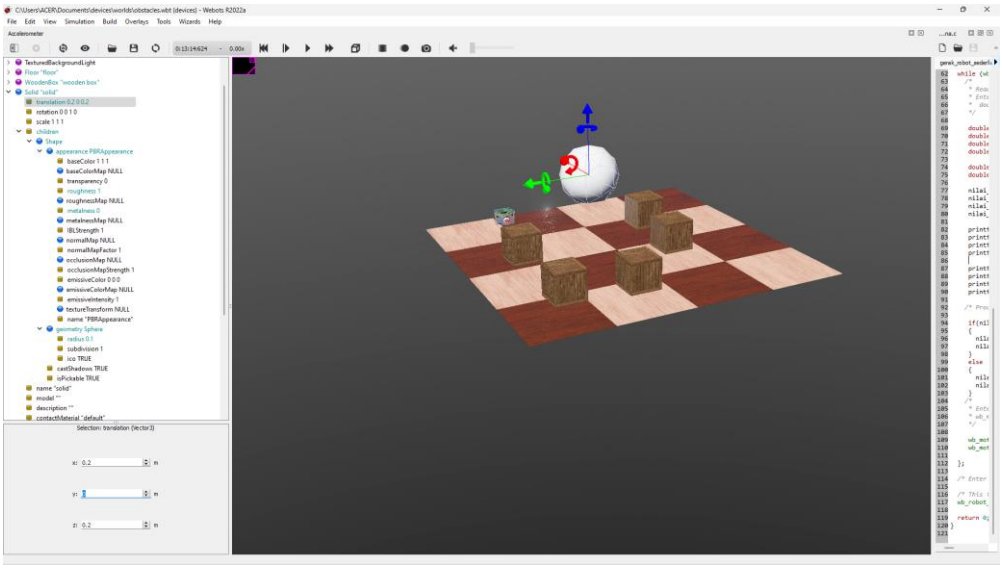
3. Tambahkan *Sphere node* di *geometry* dengan membuat *Shape node* baru.



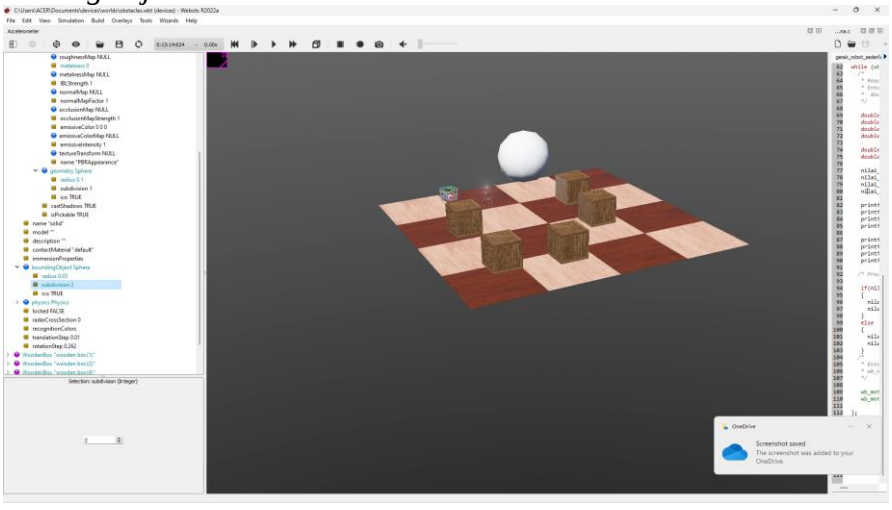
4. Ubah *metalness* menjadi 0 dan *roughness* menjadi 1 pada *PBRAppearance*.



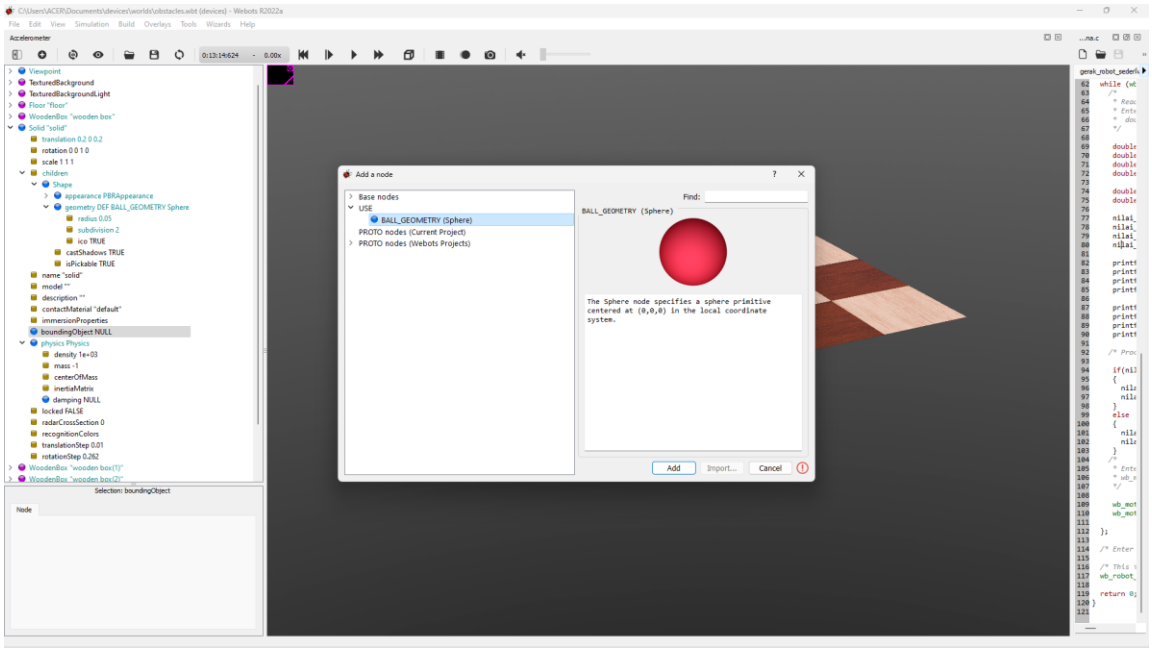
5. Tambahkan *Sphere* lain di *boundingObject* pada *Solid*.
6. Tambahkan *Physics node* pada *physics Solid*.
7. Modifikasi *translation field* pada *Solid node* dengan 0.2, 0, 0.2.



8. Langkah selanjutnya, ubah *radius* menjadi 0,05 dan *subdivision* menjadi 2 pada *Sphere* di *children boundingObject*.



9. Menggunakan fungsi DEF, memungkinkan untuk mendefinisikan *node* di satu tempat dan menggunakan kembali definisi tersebut di tempat lain dalam *scene tree*. Hal ini berguna untuk menghindari duplikasi *node* yang identik dalam *world file*. Selain itu, ini juga memungkinkan pengguna untuk memodifikasi beberapa objek pada saat yang bersamaan.

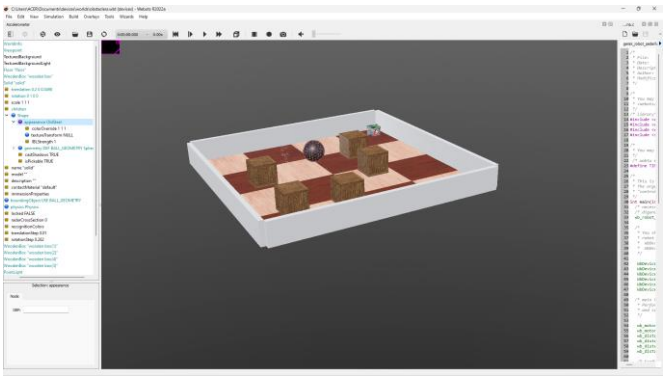


Hapus *boundingObject* dan buat DEF BALL_GEOMETRY pada *geometry* bagian *Shape Children*. Kemudian, *add new* pada *boundingObject* dengan menambahkan USE BALL_GEOMETRY seperti gambar diatas.

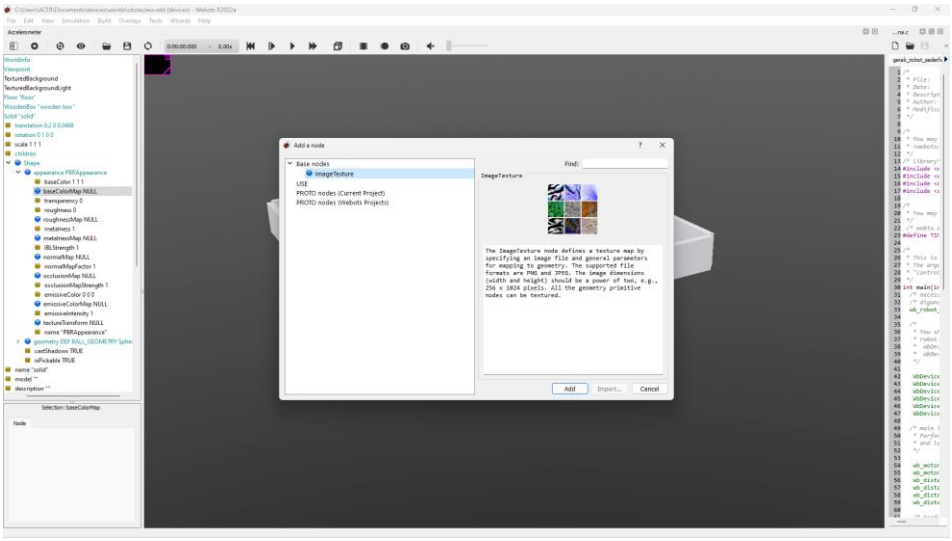
10. Memberikan empat dinding untuk mengelilingi *environment*, dengan menggunakan fungsi DEF.
    WALL_SHAPE berisi :
    PBRAppearance dengan *baseColor* 0.8 0.8 0.8, *roughness* 1, *metalness* 0 dan *geometry Box size* 0.02 0.98 0.1.
    DEF WALL_1 *Solid translation* 0.5 0 0.05.
    DEF WALL_2 *Solid translation* -0.5 0 0.05.
    DEF WALL_3 *Solid translation* 0 -0.5 0.05 dan *rotation* 0 0 1 1.5708.
    DEF WALL_4 *Solid translation* 0 0.5 0.05 dan *rotation* 0 0 1 1.5708.

[Tutorial – 3]

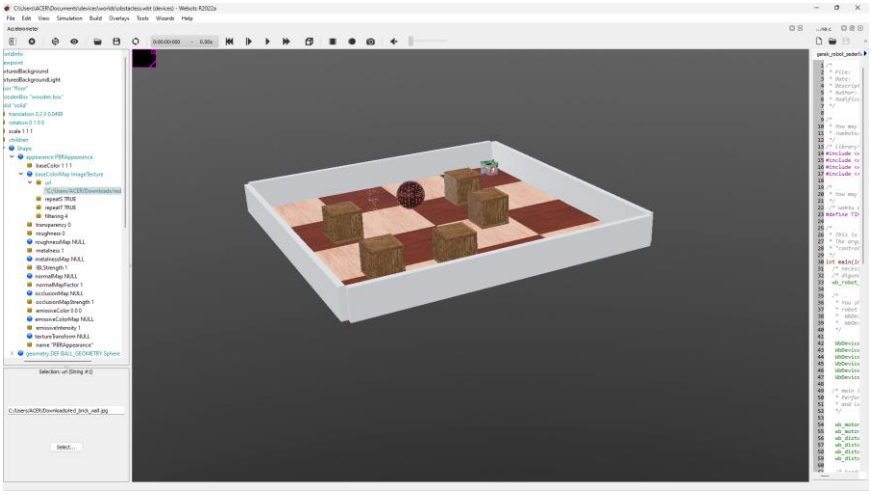
1. *Remove PBRAppearance*, setelah di *remove Appearace* menjadi NULL.
2. *Add new* pada *PROTO nodes* (*Webots Projects*), klik *appearances* dan pilih *OldSteel* (*PBRAppearance*).



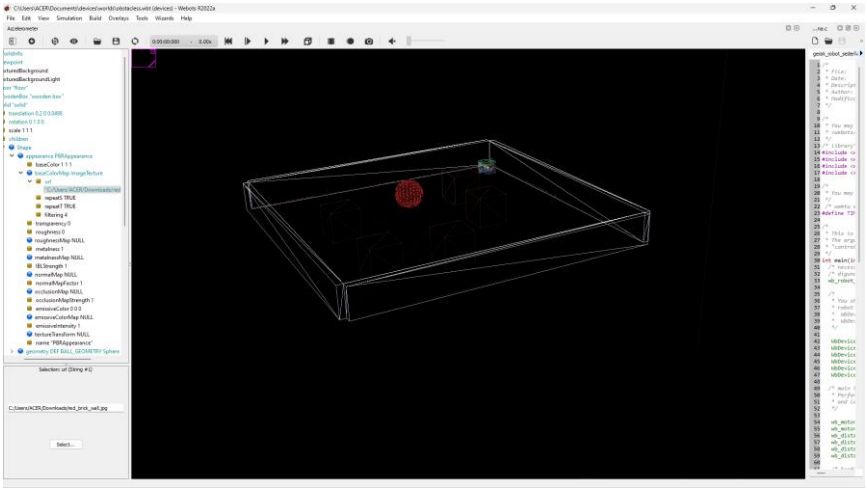
3. *Remove appearance* yang ditambahkan tadi dan ganti menjadi *PBRAppearance*.
4. Tambahkan *ImageTexture* di *baseColorMap* pada *PBRAppearance node*.



5. Tambahkan *url* dan klik *select* untuk menambahkan *path*.



6. *View* simulasi dengan *Wirefram Rendering*.



[Tutorial – 4]

Menambahkan program bahasa C pada .c yang telah dibuat, seperti dibawah ini :

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

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

#define MAX_SPEED 6.28

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

  // internal variables
  int i;
  WbDeviceTag ps[8];
  char ps_names[8][4] = {
    "ps0", "ps1", "ps2", "ps3",
```

```
  "ps4", "ps5", "ps6", "ps7"
};

// initialize devices
for (i = 0; i < 8 ; i++) {
  ps[i] = wb_robot_get_device(ps_names[i]);
  wb_distance_sensor_enable(ps[i], TIME_STEP);
}

WbDeviceTag left_motor = wb_robot_get_device("left wheel motor");
WbDeviceTag right_motor = wb_robot_get_device("right wheel motor");
wb_motor_set_position(left_motor, INFINITY);
wb_motor_set_position(right_motor, INFINITY);
wb_motor_set_velocity(left_motor, 0.0);
wb_motor_set_velocity(right_motor, 0.0);

// feedback loop: step simulation until an exit event is received
while (wb_robot_step(TIME_STEP) != -1) {
  // read sensors outputs
  double ps_values[8];
  for (i = 0; i < 8 ; i++)
    ps_values[i] = wb_distance_sensor_get_value(ps[i]);

  // detect obstacles
  bool right_obstacle =
    ps_values[0] > 80.0 ||
    ps_values[1] > 80.0 ||
    ps_values[2] > 80.0;
  bool left_obstacle =
    ps_values[5] > 80.0 ||
```

```
    ps_values[6] > 80.0 ||
    ps_values[7] > 80.0;

  // initialize motor speeds at 50% of MAX_SPEED.
  double left_speed  = 0.5 * MAX_SPEED;
  double right_speed = 0.5 * MAX_SPEED;

  // modify speeds according to obstacles
  if (left_obstacle) {
    // turn right
    left_speed  = 0.5 * MAX_SPEED;
    right_speed = -0.5 * MAX_SPEED;
  }
  else if (right_obstacle) {
    // turn left
    left_speed  = -0.5 * MAX_SPEED;
    right_speed = 0.5 * MAX_SPEED;
  }

  // write actuators inputs
  wb_motor_set_velocity(left_motor, left_speed);
  wb_motor_set_velocity(right_motor, right_speed);
 }

 // cleanup the Webots API
 wb_robot_cleanup();
 return 0; //EXIT_SUCCESS
}
```

Untuk penjelasannya seperti dibawah ini :

- *controllerArgs* meneruskan *argument* ke fungsi utama.
- *wb_robot_init* digunakan untuk menginisialisasi *API Webots* dan dibersihkan dengan

fungsi *wb_robot_cleanup*.

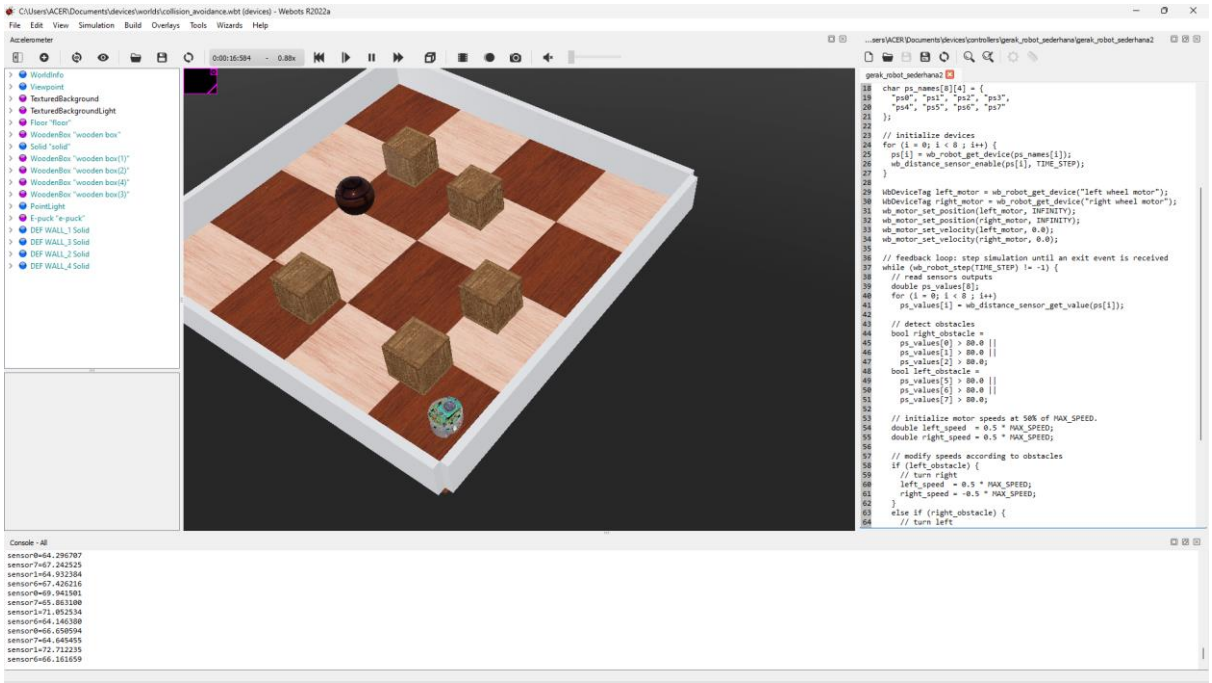
Perangkat robot direferensikan oleh *WbDeviceTag*. *WbDeviceTag* akan diambil oleh fungsi *wb_robot_get_device*. Kemudian, digunakan sebagai *argument* pertama dalam setiap pemanggilan fungsi mengenai perangkat ini. *Sensor Distance* harus diaktifkan sebelum digunakan. *Argument* kedua dari fungsi *anable* mendefinisikan pada tingkat mana sensor akan di *refresh*.

Ringkasan Tutorial – 4 :

Berikut ini adalah ringkasan singkat dari poin-poin penting yang baru saja diperlajari :

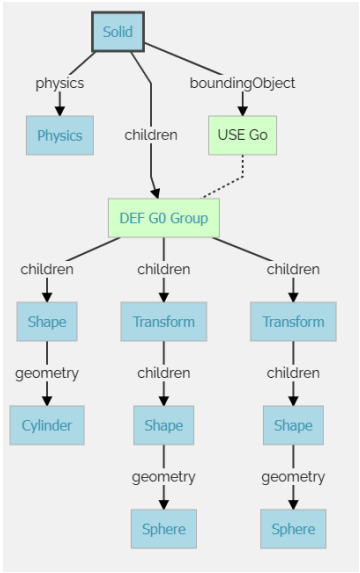
- Titik masuk pengontrol adalah fungsi utama seperti program C standar lainnya.
- Tidak ada fungsi *Webots API* yang harus dipanggil sebelum pemanggilan fungsi *wb_robot_init*.
- Fungsi terakhir yang harus dipanggil sebelum meninggalkan fungsi utama adalah fungsi *wb_robot_cleanup*.
- Sebuah perangkat direferensikan oleh bidang nama *node* perangkatnya. Referensi *node* dapat diambil berkat fungsi *wb_robot_get_device*.
- Setiap program pengontrol dieksekusi sebagai proses anak dari proses *Webots*. Proses pengontrol tidak berbagi memori dengan *Webots* (kecuali gambar kamera) dan dapat berjalan pada CPU lain (atau inti CPU) daripada *Webots*.
- Kode pengontrol dihubungkan dengan pustaka dinamis "libController".

Berikut ini hasil *running* program :

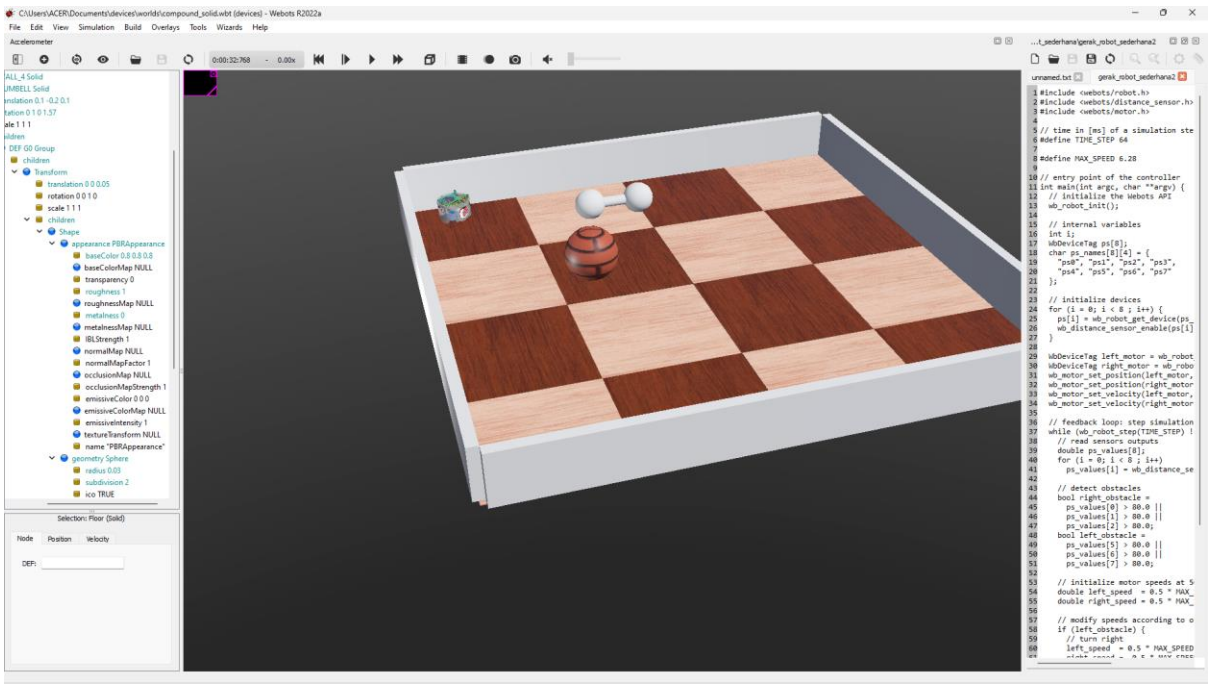


[Tutorial – 5]

Pada tutorial ke-5, hapus *WoodenBox* terlebih dahulu. Kemudian, tambahkan objek yang diperlukan mengikuti langkah yang diberikan, seperti pada bagan di bawah ini :



Untuk hasilnya seperti pada gambar dibawah ini :



Dengan membuat DEF DUMBBELL SOLID, dengan komposisi sebagai berikut :

Translation 0.1 -0.2 0.1

Rotation 0 1 0 1,5708.

Di dalam *children* DEF DUMBBLLE Solid berisi DEF G0 Group. *Children* DEF G0 Group berisi :

- *Shape*

PBRAppearance dengan *baseColor* 0.8 0.8 0.8, *roughness* 1 dan *metalness* 0.
Geometry Cylinder dengan *height* 0.1 dan *radius* 0.01.

- *Transform*

Translation 0 0 0.05.
Memiliki *children* yang berisi :
PBRAppearance dengan *baseColor* 0.8 0.8 0.8, *roughness* 1 dan *metalness* 0.
Geometry Sphere dengan *radius* 0.03 dan *subdivision* 2.

- *Transform*

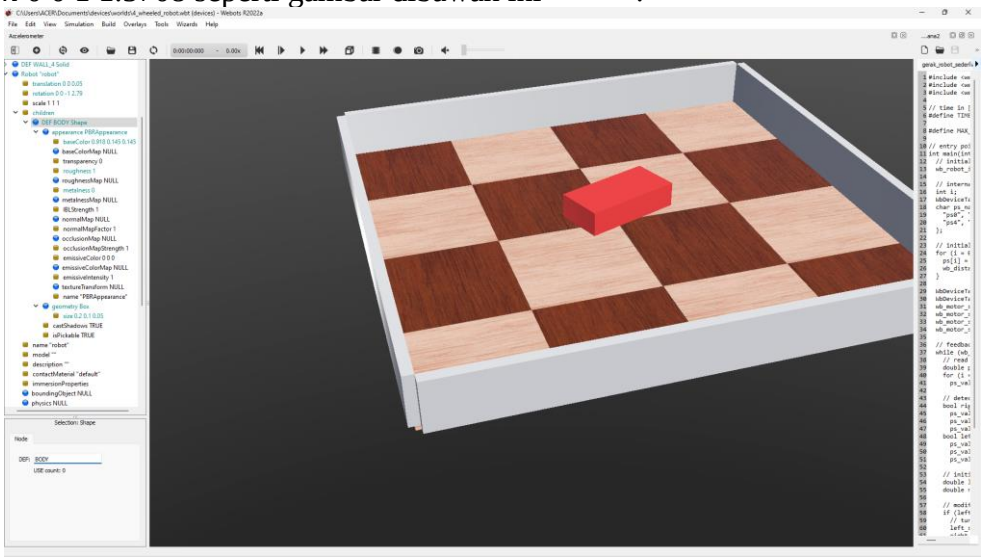
Translation 0 0 -0.05.
Translation 0 0 0.05.
Memiliki *children* yang berisi :
PBRAppearance dengan *baseColor* 0.8 0.8 0.8, *roughness* 1 dan *metalness* 0.
Geometry Sphere dengan *radius* 0.03 dan *subdivision* 2.

Name "solid(5)", *contactMaterial* "dumbbell", *boundingObject* USE G0 dengan *physics Physics. Physics* memiliki *density* -1, *mass* 2 dan *centerOfMass* 0 0 -0.02.

[Tutorial – 6]

Membuat robot dari awal yang terdiri dari sebuah badan, empat roda dan dua sensor jarak.

1. Tambahkan *Shape*, buat fungsi Shape DEF BODY dengan *translation* 0 0.5 0.05 dan *rotation* 0 0 1 1.5708 seperti gambar dibawah ini :



Dengan *Shape* :

- *PBRAppearance* – *baseColor* 0.917647 0.145098 0.145098, *roughness* 1 dan *metalness* 0.
- *geometryBox size* 0.2 0.1 0.05.

2. Tambahkan *HingeJoint* pada *Shape* dan buat DEF WHEEL1 *HingeJoint* dengan tambahkan *HingeJointParameters*, dengan *axis* 0 1 0 dan *anchor* 0.05 0.06 0. *Device* menggunakan *RotationMotor* dengan nama "wheel1". Pada *endpoint translation* 0.05 0.06 0 dan *rotation* 1 0 0 1.5708 yang mempunyai *children* DEF WHEEL SHAPE. Pada DEF WHEEL SHAPE menggunakan *PBRAppearance* dengan *baseColor* 0.305882 0.898039 0.25098, *roughness* 1 dan *metalness* 0. *Geometry Cylinder* dengan *height* 0.02, *radius* 0.04 dan *subdivision* 24.

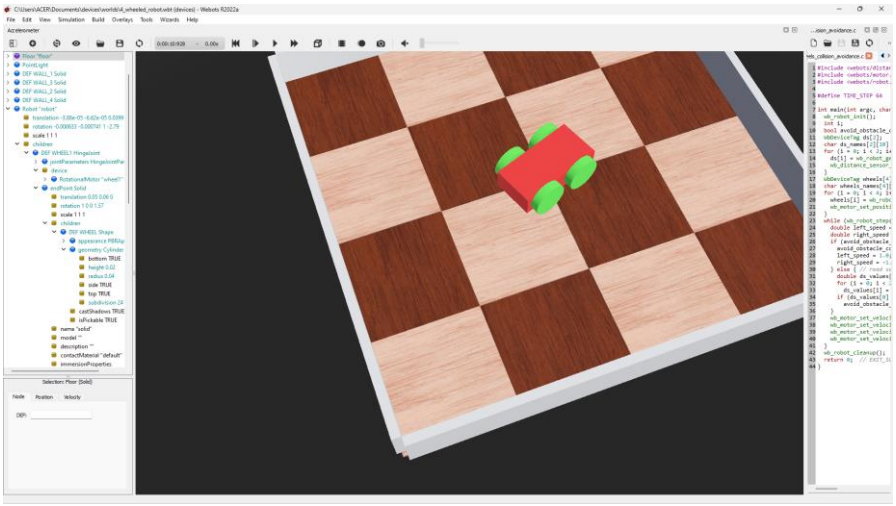
3. Tambahkan fungsi DEF WHEEL2 *HingeJoint* dengan *HingeJointParameters* yang memiliki *axis* 0 1 0 dan *anchor* 0.05 -0.06 0. *Device* menggunakan *RotationMotor* dengan nama “wheel2”. *Endpoint* menggunakan *translation* 0.05 -0.06 0 dan *rotation* 1 0 0 1.5708. *Children* menggunakan USE WHEEL.

4. Tambahkan fungsi DEF WHEEL3 *HingeJoint* dengan *HingeJointParameters* yang memiliki *axis* 0 1 0 dan *anchor* -0.05 0.06 0. *Device* menggunakan *RotationMotor* dengan nama “wheel3”. *Endpoint* menggunakan *translation* -0.05 0.06 0 dan *rotation* 1 0 0 1.5708. *Children* menggunakan USE WHEEL.

5. Tambahkan fungsi DEF WHEEL4 *HingeJoint* dengan *HingeJointParameters* yang memiliki *axis* 0 1 0 dan *anchor* -0.05 -0.06 0. *Device* menggunakan *RotationMotor* dengan nama “wheel3”. *Endpoint* menggunakan *translation* -0.05 -0.06 0 dan *rotation* 1 0 0 1.5708. *Children* menggunakan USE WHEEL.

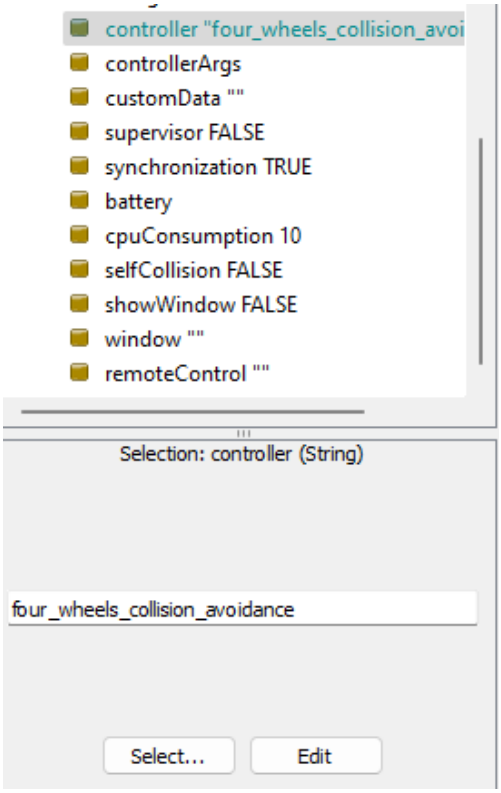
6. Tambahkan *DistanceSensor* dan buat fungsi DEF DS_RIGHT dengan *translation* 0.1 -0.03 0 dan *rotation* 0 0 1 -0.3. *Children* mengandung *Shape PBRAppearance* dengan *baseColor* 0.184314 0.596078 0.847059, *roughness* 1 dan *metalness* 0. *Geometry Box size* 0.01 0.01 0.01 serta nama “ds_right”.

7. Tambahkan *DistanceSensor* dan buat fungsi DEF DS_LEFT dengan *translation* 0.1 0.03 0 dan *rotation* 0 0 1 0.3. *Children* mengandung *Shape PBRAppearance* dengan *baseColor* 0.184314 0.596078 0.847059, *roughness* 1 dan *metalness* 0. *Geometry Box size* 0.01 0.01 0.01 serta nama “ds_left”.

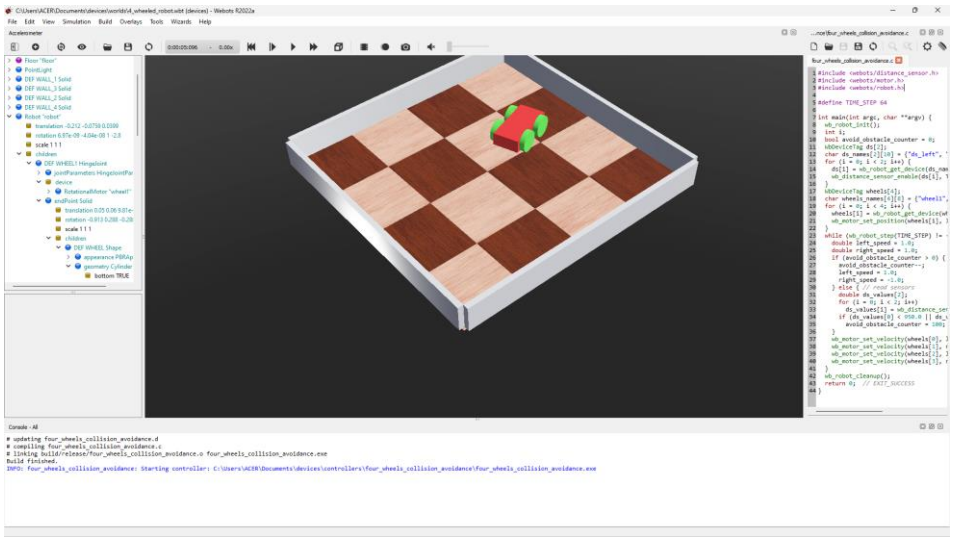


8. Atur *density* -1 dan *mass* 1 pada *Physics*.

9. Select *file controller* seperti gambar dibawah ini :



10. Hasil *running* seperti gambar dibawah ini :



Berikut ini file programnya :

```
#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 = 1.0;
    double right_speed = 1.0;
    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
     }
```

[Tutorial – 7]

1. Membuat folder *protos* yang digunakan untuk membuat *file* .protos. Nama *file* yang dibuat yaitu FourWheelsRobot.proto yang berisi untuk mendefinisikan robot.
2. *File* FourWheelsRobot.proto berisi program seperti dibawah ini :

```
PROTO FourWheelsRobot [
  field SFVec3f    translation  0 0 0
  field SFRotation rotation     0 0 1 0
  field SFFloat    bodyMass     1
]
{
```

```
Robot {
  translation IS translation
  rotation IS rotation
  children [
    DEF BODY Shape {
      appearance PBRAppearance {
        baseColor 0.917647 0.145098 0.145098
        roughness 1
        metalness 0
      }
      geometry Box {
        size 0.2 0.1 0.05
      }
    }
    DEF WHEEL1 HingeJoint {
      jointParameters HingeJointParameters {
        axis 0 1 0
        anchor 0.05 0.06 0
      }
      device [
        RotationalMotor {
          name "wheel1"
        }
      ]
      endPoint Solid {
        translation 0.05 0.06 0
        rotation 1 0 0 1.5708
        children [
          DEF WHEEL Shape {
            appearance PBRAppearance {
              baseColor 0.305882 0.898039 0.25098
              roughness 1
              metalness 0
            }
            geometry Cylinder {
              height 0.02
              radius 0.04
              subdivision 24
            }
          }
        ]
        boundingObject USE WHEEL
        physics Physics {
        }
      }
    }
    DEF WHEEL2 HingeJoint {
      jointParameters HingeJointParameters {
        axis 0 1 0
        anchor 0.05 -0.06 0
```

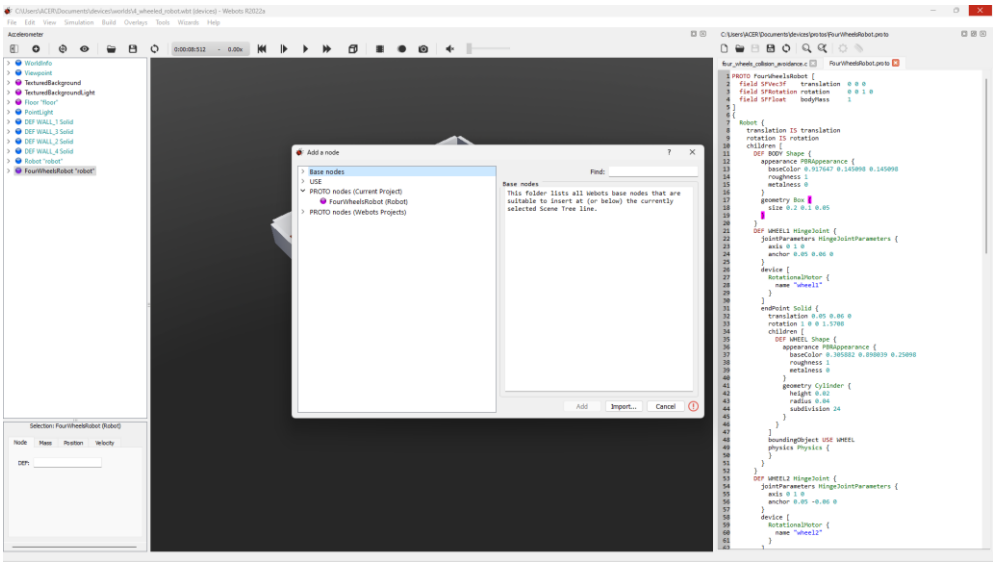
```
      }
      device [
       RotationalMotor {
        name "wheel2"
       }
      ]
      endPoint Solid {
       translation 0.05 -0.06 0
       rotation 1 0 0 1.5708
       children [
        USE WHEEL
       ]
       name "solid(1)"
       boundingObject USE WHEEL
       physics Physics {
       }
      }
     }
     DEF WHEEL3 HingeJoint {
      jointParameters HingeJointParameters {
       axis 0 1 0
       anchor -0.05 0.06 0
      }
      device [
       RotationalMotor {
        name "wheel3"
       }
]

      endPoint Solid {
       translation -0.05 0.06 0
       rotation 1 0 0 1.5708
       children [
        USE WHEEL
       ]
       name "solid(2)"
       boundingObject USE WHEEL
       physics Physics {
       }
      }
     }
     DEF WHEEL4 HingeJoint {
      jointParameters HingeJointParameters {
       axis 0 1 0
       anchor -0.05 -0.06 0
      }
      device [
       RotationalMotor {
        name "wheel4"
       }
```

```
    ]
    endPoint Solid {
      translation -0.05 -0.06 0
      rotation 1 0 0 1.5708
      children [
        USE WHEEL
      ]
      name "solid(3)"
      boundingObject USE WHEEL
      physics Physics {
      }
    }
  }
  DEF DS_RIGHT DistanceSensor {
    translation 0.1 -0.03 0
    rotation 0 0 1 -0.3
    children [
      Shape {
        appearance PBRAppearance {
          baseColor 0.184314 0.596078 0.847059
          roughness 1
          metalness 0
        }
        geometry Box {
          size 0.01 0.01 0.01
        }
      }
    ]
    name "ds_right"
  }
  DEF DS_LEFT DistanceSensor {
    translation 0.1 0.03 0
    rotation 0 0 1 0.3
    children [
      Shape {
        appearance PBRAppearance {
          baseColor 0.184314 0.596078 0.847059
          roughness 1
          metalness 0
        }
        geometry Box {
          size 0.01 0.01 0.01
        }
      }
    ]
    name "ds_left"
  }
]
boundingObject USE BODY
physics Physics {
```

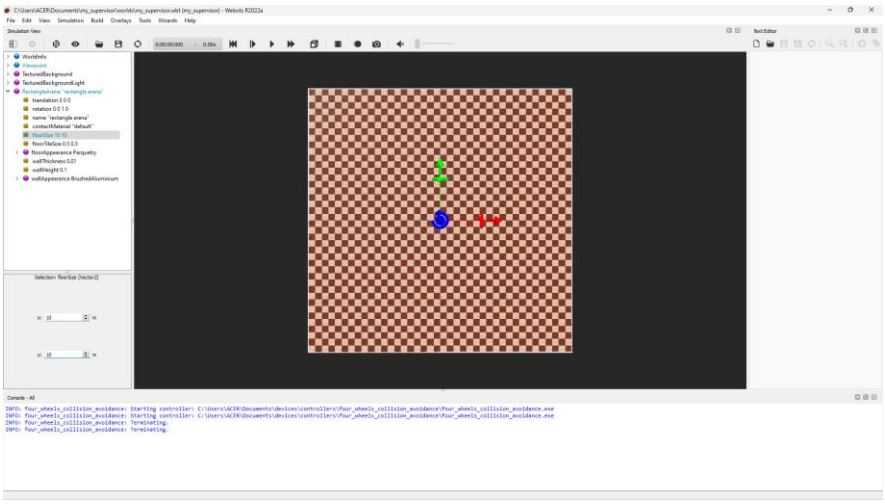
```
      density -1
      mass IS bodyMass
    }
    controller "four_wheels_collision_avoidance"
  }
}
```

3. Tambahkan *node proto* yang telah dibuat.

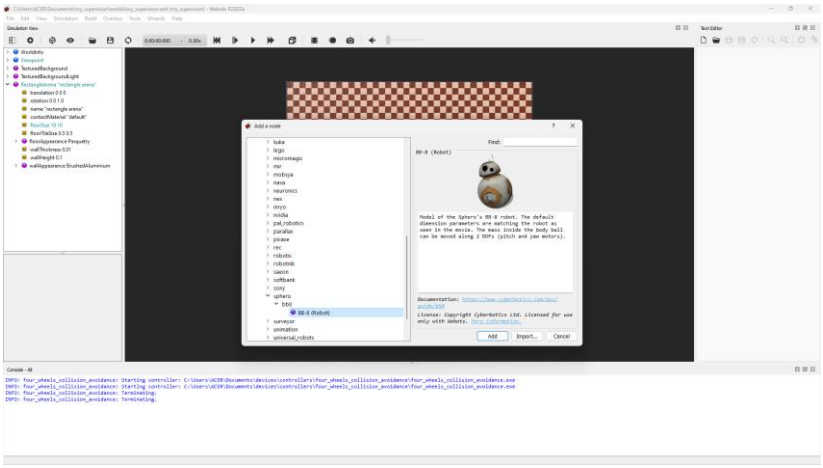


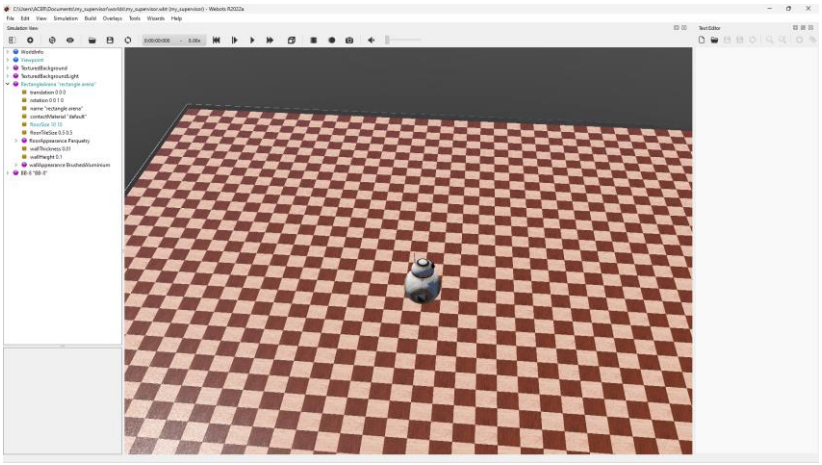
[Tutorial – 8]

1. Buat *project directory* dengan nama *my_supervisior*.
2. Beri nama *world file* dengan my_supervisor.wbt.
3. Pada *rectangle arena* ubah ukuran pada *floorSize* menjadi 10x10 *meters*.

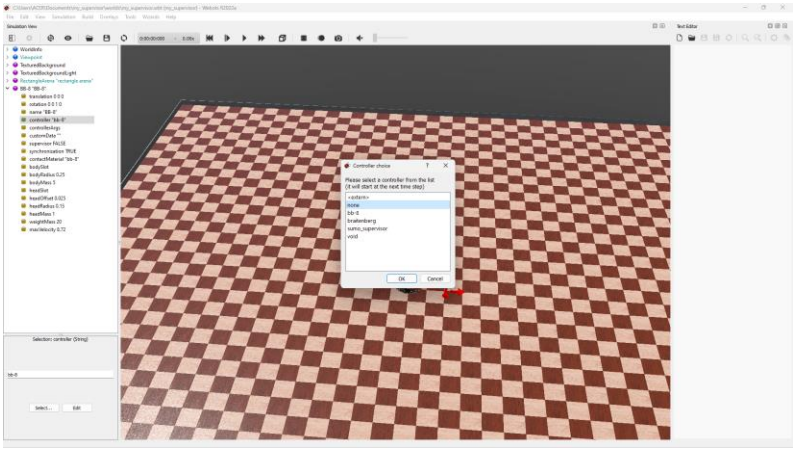


4. Tambahkan robot BB-8 dengan menambahkan mengklik tombol *button* dan tambahkan pada *PROTO nodes* (*Webots projects*) / *robots* / *sphero* / bb8.

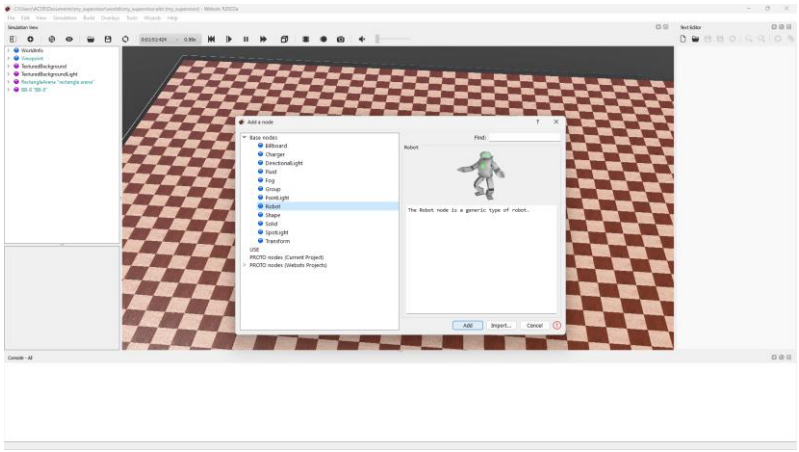


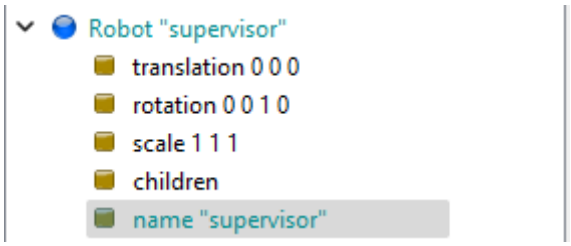


5. *Remove the default controller*, klik *Select* dan pilih *None*.

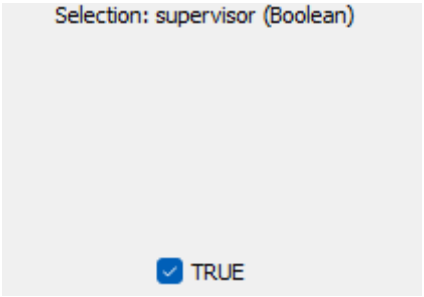


6. Tambahkan Robot sederhana yang dapat ditemukan pada *base nodes* dan ubah namanya menjadi *supervisor*.

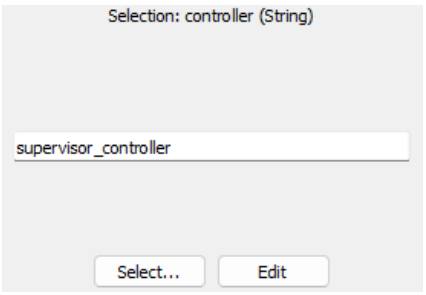




7. Meskipun nama Robot telah dirubah, tetapi node masih merupakan Robot. Sehingga, untuk mengubahnya menjadi *Supervisor* perlu mengarut *field supervisor* menjadi TRUE.

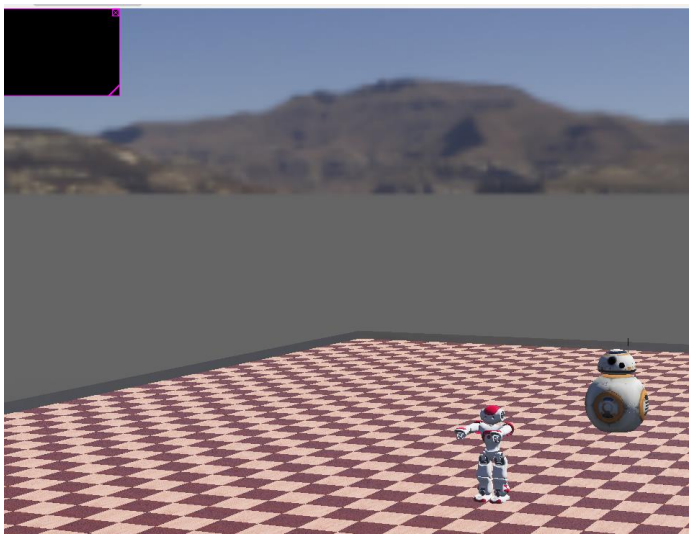


8. Ubah *controller* menjadi *supervisor_controller*.



9. Tambahkan *controller* pada *Wizard* / *New Robot Controller* dan pilih bahasa pemrograman menggunakan *Pyhton*. Beri nama *Controller* menjadi *supervisor_contoller*.
10. Identifikasikan BB-8 menjadi DEF.

11. Tambahkan Nao Robot dan ubah *translation* menjadi 2.5 0 0.334.



12. Tambahkan ball, dan ubah menjadi DEF BALL.
13. Berikut ini *syntax* pada *supervisor_controller*.

```
from controller import Supervisor

TIME_STEP = 32

robot = Supervisor()  # create Supervisor instance

# [CODE PLACEHOLDER 1]
bb8_node = robot.getFromDef('BB-8')
translation_field = bb8_node.getField('translation')

root_node = robot.getRoot()
children_field = root_node.getField('children')

children_field.importMFNodeFromString(-1, 'DEF BALL Ball { translation 0 1 1 }')
ball_node = robot.getFromDef('BALL')
color_field = ball_node.getField('color')

i = 0
while robot.step(TIME_STEP) != -1:
  # [CODE PLACEHOLDER 2]
  if (i == 0):
    new_value = [2.5, 0, 0]
    translation_field.setSFVec3f(new_value)

  if i == 10:
      bb8_node.remove()

  if i == 20:
    children_field.importMFNodeFromString(-1, 'Nao { }')

  position = ball_node.getPosition()
```

```
print('Ball position: %f %f %f\n' %(position[0], position[1], position[2]))

if position[2] < 0.2:
  red_color = [1, 0, 0]
  color_field.setSFColor(red_color)

    i += 1
```

14. Hasil *running*, Robot BB-8 akan menghilang.