Workshop 1 : Workshopworld

(**Let op**: weg is weg, er is geen undo, de save-knop is je vriend :-))

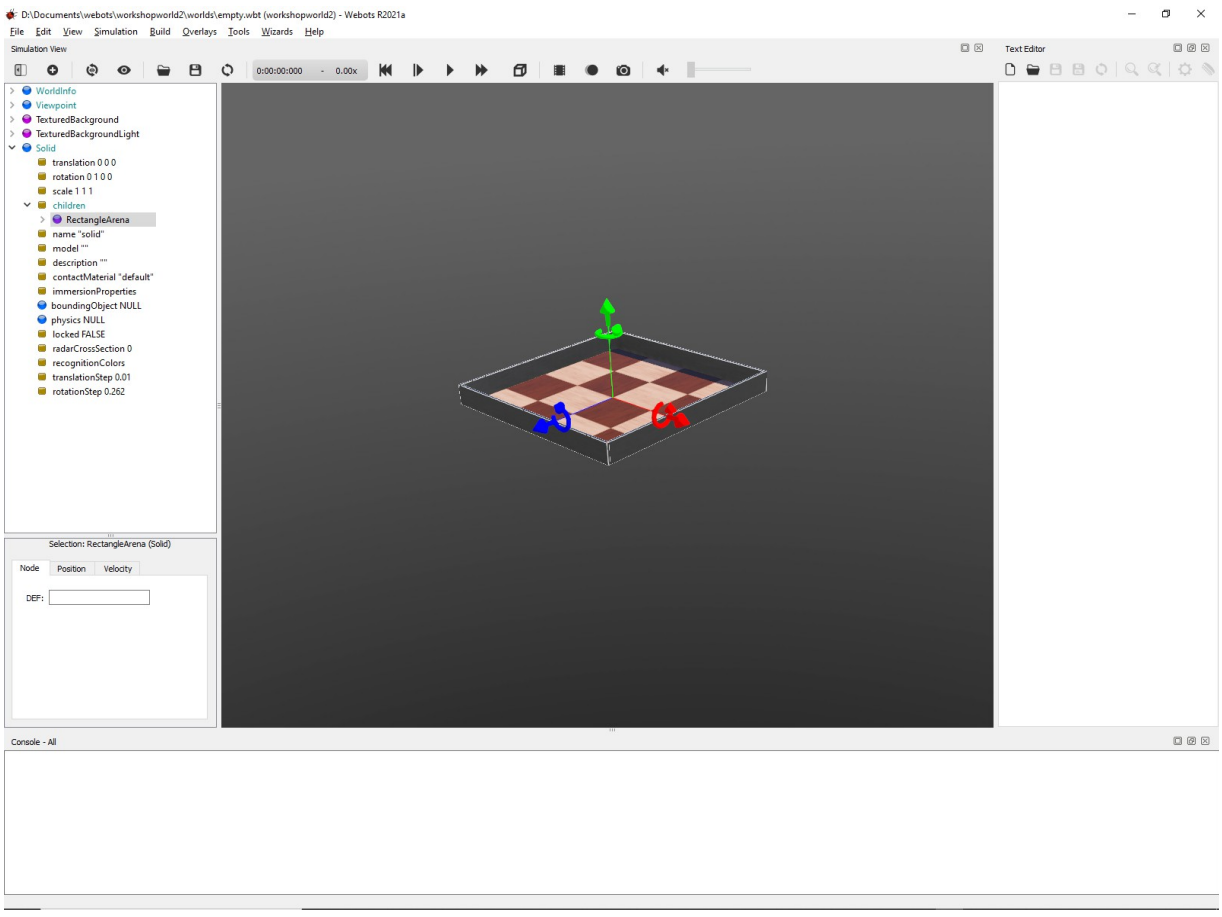
- Start Webots
- Kies : Wizards->New Project Directory...
- Vul in als directory naam aan het einde : workshopworld en klik op Next
- Vul in : workshopworld.wbt en klik op Next
- Klik op Finish

Nu is de wereld nog helemaal leeg.

Bak toevoegen:

- Klik op de laatste regel links in de lijst, button '+' wordt actief.
- Klik op de '+' button.
- Kies: Base nodes->Solid
- Klap de nieuwe solid open
- Kies 'children'
- Kies '+'
- Kies 'PROTO nodes (Webots Projects)'
- Kies objects
- Kies floors
- Kies RectangelArena
- Kies Add

Je ziet nu:



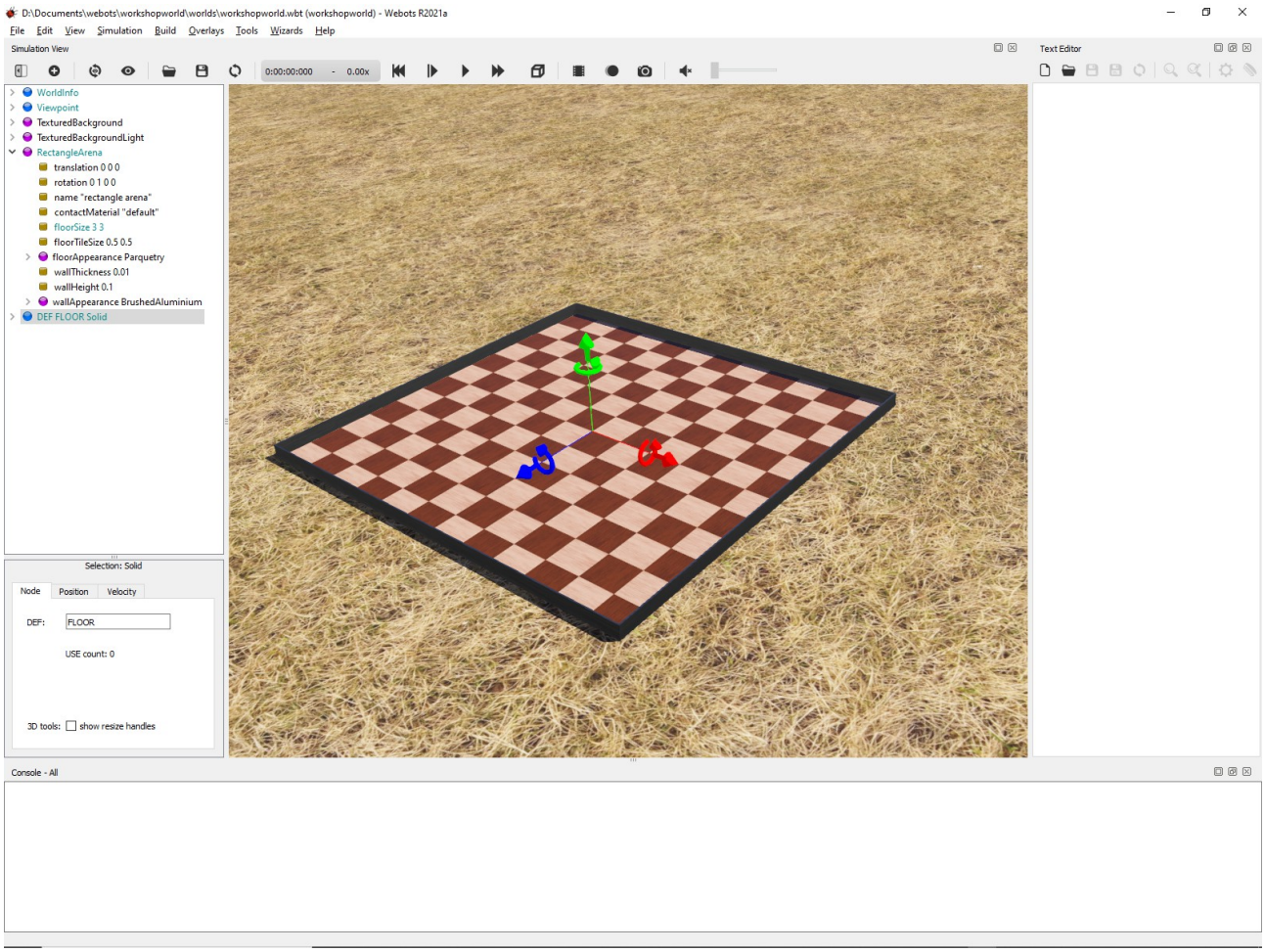
De grootte van de bak aanpassen:

- Kies: Solid->children->RectangleArena->floorSize
- Verander van 1 m naar 2 m in vierkant
- Klik op het 'save' icoon :

Gras toevoegen:

We gaan het gras gebruiken uit een ander voorbeeld project.

- Open 'File->Open Sample World->vehicles->boomer.wbt'
- Klik links in de lijst op 'DEF FLOOR Solid'. Dit is de grasgrond.
- Copieer die door rechtermuis en dan Copy te kiezen
- Ga naar 'File->Open recent worlds->workshopworld.wbt'
  Je ziet nu je eigen wereld weer terug.
- Selecteer de laatste regel in de lijst links.
- Plak het gekopieerde gras erin met rechter muisknop->paste.
  Je zou nu dit moeten zien:



Wereld assenstelsel laten zien:

- Kies 'View->Optional Rendering->Show coordinate system'
- Rechtsonderaan je 3D view zie je nu een groen, rood en blauw assenstelsel.

Dit is handig om de oriëntatie van je wereld te weten. Het draait mee als je je wereld draait.

De wereld draaien:
- Klik op de bak met de linker muisknop.
- Hou de linker muisknop vast en beweeg de muis.

De wereld verplaatsen:
- Hou de rechtermuisknop vast en verplaats de muis.

In/uitzoomen:
- Gebruik je scrolwiel om in en uit te zoomen.

Toets om top view te laten zien:
- Gebruik de toetsencombinatie 'Alt' tegelijk met '1'.
  De wereld draait nu zo dat je vanaf boven naar de wereld kijkt. Dit kan handig zijn als je wereld wat vreemd op je scherm staat en je weer terug wilt naar een goed beginpunt.

Objecten draaien en verplaatsen binnen de wereld:
- Selecteer het bak object door in het scherm links op 'RectangleArena' te klikken.
- Groene, rode en blauwe pijlen verschijnen op het object.
  - Klik muis links op een draaiende pijl om het object langs die as te draaien.
  - Klik muis links op een pijl om het object te verplaatsen.
    Groen is de Y-as (omhoog), Rood is de X-as en Blauw is de Z-as.

Bijzonderheid: hoeken worden binnen webots in radialen aangegeven, niet in graden. Eigenlijk komt het erop neer dat 360 graden gelijk is aan 2PI radialen. 180 graden is PI radialen en 90 graden is 0.5PI radialen. Door met het 3D scherm te werken kun je visueel zien hoe een hoek uitpakt en heb je hier niet zoveel last van.

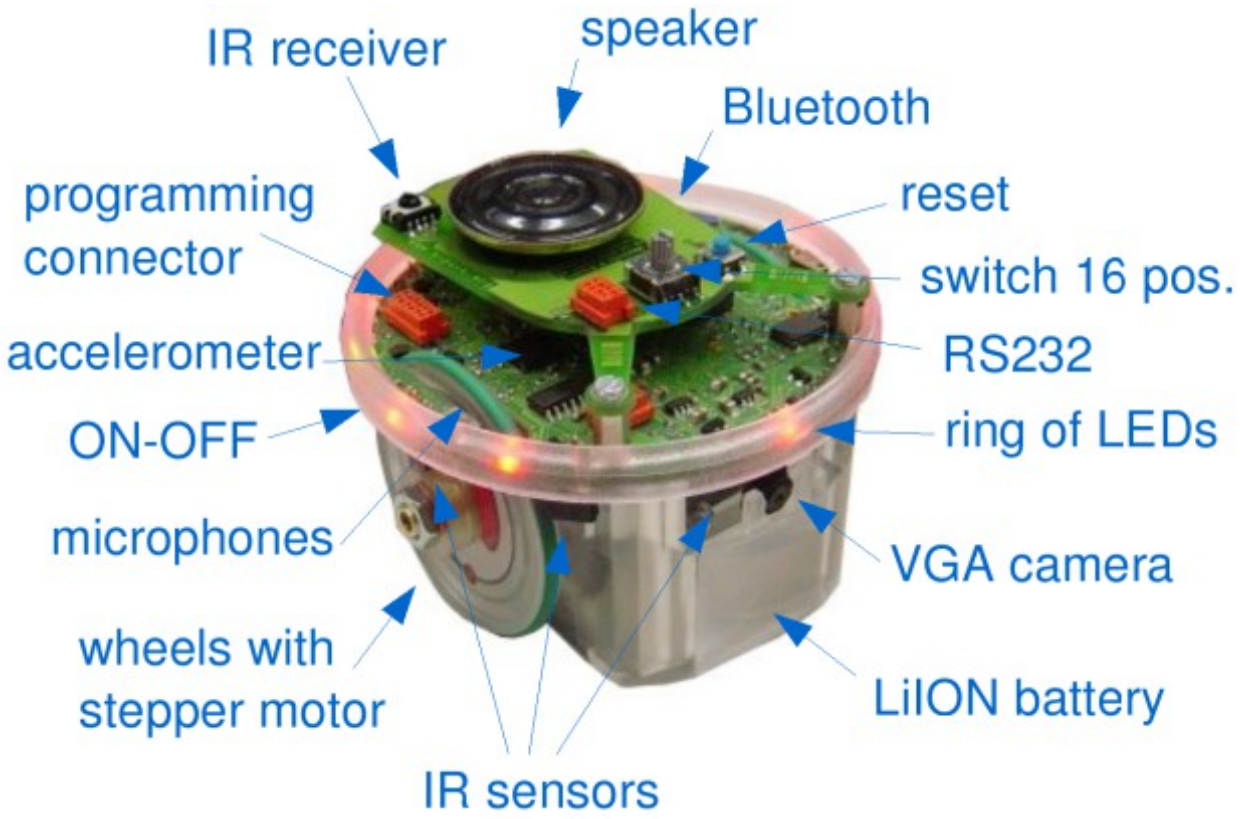
Voorgedefinieerd object toevoegen
Nu gaan we een paar objecten aan onze wereld toevoegen vanuit de rijke beschikbare verzameling objecten binnen webots:

- Klik op het laatste veld in de lijst links om het '+' teken actief te laten worden.
- Klik op de '+'.
- Kies 'PROTO nodes (Webots Projects)->objects->factory->containers->WoodenBox (Solid)'
- Klik 'Add'
- Zoals je ziet is deze wooden box nogal groot met 60 cm per kant. De e-puck robot is 7 bij 5 cm. We willen dus deze box wat kleiner maken.
- Klap links in de lijst de node 'WoodenBox' open.
- Kies 'size'.
- Zet het veld x,y en z op 0.1 m
- Nu hangt de box nog in de lucht, verplaats hem naar het veld met behulp van de groene pijl.
- Een andere mogelijkheid om de box omlaag te krijgen is door zijn y waarde aan te passen:
  - Klik op de wooden box in de lijst links.
  - Klik op translation
  - Vul in het veld linksonder voor de y waarde 0.05 in. Dit is dus 5 cm omhoog.
- Gebruik copieren en plakken op deze box om nog drie boxen aan te maken en plaats die op je veld.
- Het lijkt alsof je nog steeds maar 1 box ziet maar dat komt omdat ze allemaal op dezelfde plek staan.
- Kies de laatste box in de lijst links en verplaats die met de pijlen.

- Doe dit ook voor de andere boxen.
- Klik op het save icoon om je werk te bewaren.

Workshop 2 : Webotsrobot

In deze workshop gaan we aan de slag met de e-puck robot maar er zijn nog vele tientallen andere robots beschikbaar binnen webots.
Dit is een foto van de echte e-puck robot, die is 7cm breed bij 5 cm hoog :



(Bron: https://www.generationrobots.com/en/401410-e-puck-programmable-robot-with-battery.html)

De e-puck heeft onder meer acht IR sensoren rondom om de nabijheid van andere objecten te kunnen vaststellen. Verder kunnen in een uitbreidingsslot aan de voorkant van deze robot nog een drietal IR sensoren geplaatst worden die naar de grond kijken. Dit is bijvoorbeeld om vast te stellen of die sensor een witte of een zwarte ondergrond ziet. (Handig voor in de sumoring :-))
Al zijn sensoren, leds en motoren kun je via webots gebruiken.

In deze workshop gaan we de taal C++ gebruiken om de robot te besturen. (Als je gewent bent om met Arduino te werken dan heb je al een tijd in C++ geprogrammeerd :-)). (Het is ook mogelijk om een aantal andere programmeertalen te gebruiken voor het besturen van een webots robot).

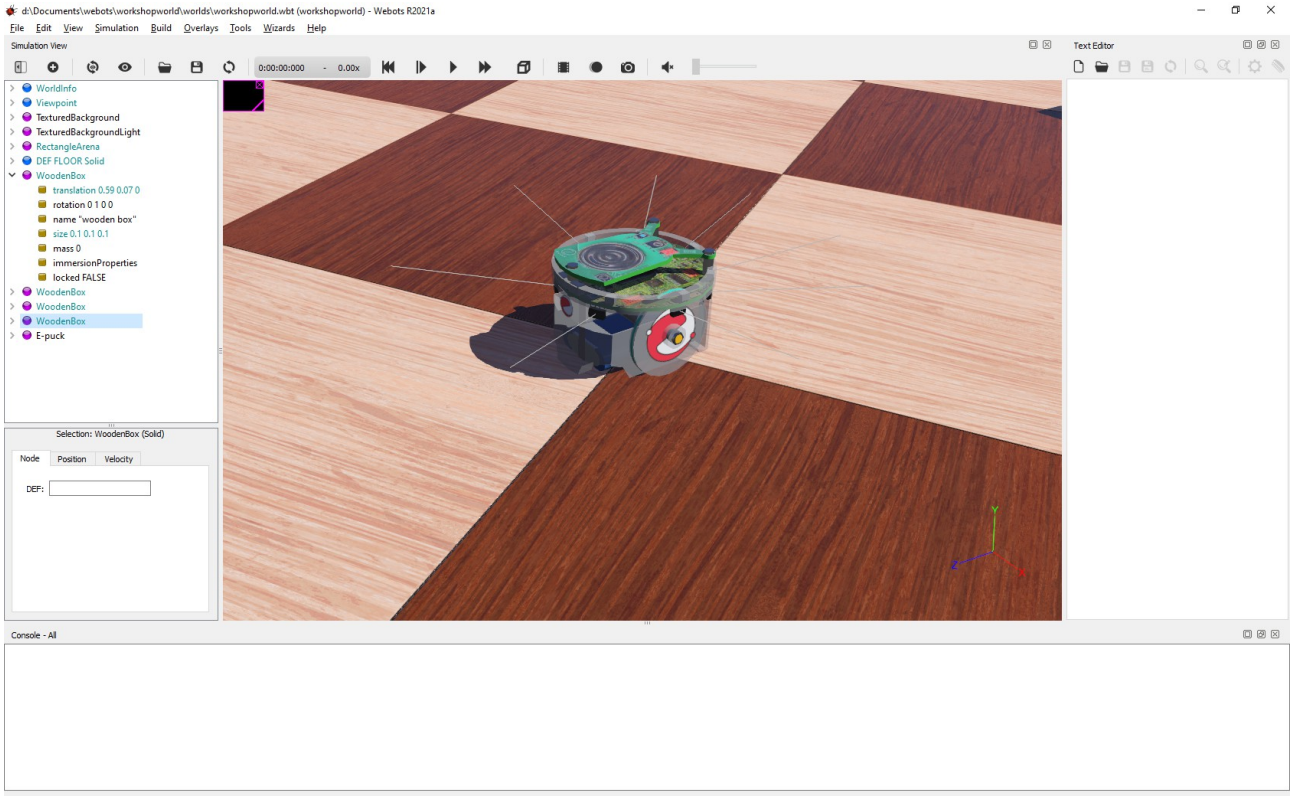
We gaan een e-puck robot op onze wereld plaatsen:

- Klik op de '+' knop.
- Kies 'PROTO nodes (Webots Projects)->robots->gctronic->e-puck->E-puck (Robot)'.
- Klik 'Add'.
- In het midden van je 3D scherm zie je je nieuwe robot staan.

- Zoom in om de robot van dichtbij te bekijken.

Om het werkingsgebied van de IR sensoren zichtbaar te maken doen we het volgende:

- Kies 'View->Optional Rendering->Show DistanceSensor Rays'
- Dat ziet er dan zo uit:



(Let op de grijze lijntjes, de sensor rays, die nu vanuit de robot komen per sensor)
De robot kan per IR sensor tot aan het einde van de sensor ray een voorwerp detecteren.

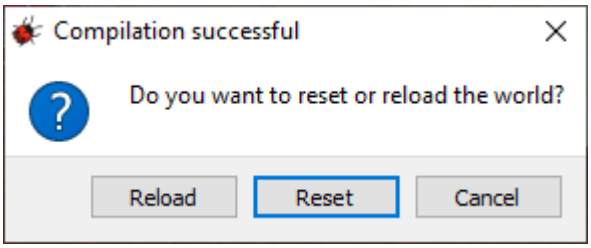
Een eigen robot met C++ besturen binnen de webots omgeving:

- We gaan de taal C++ gebruiken voor het besturen van de e-puck.
- Een beginpunt voor deze code kunnen we met de webots wizards aanmaken:
- Klik op 'Wizards->New Robot Controller...'
- Klik 'Next'
- Kies 'C++' en 'Next'
- Kies 'Webots (gcc/Makefile)' en 'Next'
- Geef een 'Controller name:' op van 'workshop_controller' en 'Next'
- Check dat het 'Open 'workshop_controller.cpp' in Text Editor' vinkje aangevinkt staat.
- Klik op 'Finish'

Je hebt nu een leeg bestuur programma voor de e-puck robot aangemaakt. Als test gaan we dit lege programma compileren en laten lopen op de robot.

- Open de 'e-puck' tab
- Klik het veld 'controller'
- Klik op de button 'Select...'
- Scrol omlaag en kies de 'workshop_controller' die we net aangemaakt hebben.

- Sla je werk op door op het 'save' icoon te klikken.
- Druk op 'F7' om het compileren te starten, linksonder in het Console venster verschijnen wat compileer boodschappen en uiteindelijk het volgende venster:



- Kies 'Reset', de 'Reload' optie haalt de simulatie van disk en is trager in gebruik.

Workshopworld 1 programma

Nu gaan we de e-puck robot ook echt wat laten doen. Eerst alleen de linker en rechter motor aanzetten. De programmacode hiervoor is:

```
// Workshopworld 1
// See e-puck documentation: https://cyberbotics.com/doc/guide/epuck
#include <webots/Robot.hpp>
#include <webots/Motor.hpp>
#include <webots/DistanceSensor.hpp>

using namespace webots;

int main(int argc, char **argv) {
// create the Robot instance.
Robot *robot = new Robot();

// get the time step of the current world.
int timeStep = (int)robot->getBasicTimeStep();

// Robot parts initialisation
Motor *leftMotor = robot->getMotor("left wheel motor");
Motor *rightMotor = robot->getMotor("right wheel motor");
leftMotor->setPosition(INFINITY);
rightMotor->setPosition(INFINITY);
double MAXSPEED = 6.28;
leftMotor->setVelocity(MAXSPEED);
rightMotor->setVelocity(MAXSPEED);

while (robot->step(timeStep) != -1) {
// Robot loop
};

delete robot;
return 0;
}
```

- Kopieer deze code in zijn geheel naar het rechter scherm met de tab 'workshop_controller.cpp'.

- Bewaar de code op disk door op dit icoon boven het bestand te klikken:

- Druk op F7 om de code te compileren.
- Laat de simulatie lopen door op het play icoon te klikken.

- Je robot zou nu zijn motoren moeten starten en als een blind paard voorwaarts moeten gaan.
- Stop de simulatie met het pauze icoon:

- Reset je simulatie met het terugspoel icoon:

Workshopworld 2 programma

Het zou natuurlijk wel handig zijn als de robot kon stoppen als hij een object tegenkomt. We gaan de code nu zo aanpassen dat als de vooruit kijkende sensor een object ziet, dat de robot zijn motoren uitzet en een LED aanzet. De e-puck heeft 8 IR afstand sensoren (DistanceSensor). We gaan de eerste DistanceSensor met label 'ps0', aan de voorkant, gebruiken om te bepalen of we een object tegenkomen.

```
// Workshopworld 2
// See e-puck documentation: https://cyberbotics.com/doc/guide/epuck
#include <webots/Robot.hpp>
#include <webots/Motor.hpp>
#include <webots/LED.hpp>
#include <webots/DistanceSensor.hpp>

using namespace webots;

int main(int argc, char **argv) {
// create the Robot instance.
Robot *robot = new Robot();

// get the time step of the current world.
int timeStep = (int)robot->getBasicTimeStep();

// Robot parts initialisation
LED *led0 = robot->getLED("led0");
Motor *leftMotor = robot->getMotor("left wheel motor");
Motor *rightMotor = robot->getMotor("right wheel motor");
led0->set(false); // LED off
leftMotor->setPosition(INFINITY);
rightMotor->setPosition(INFINITY);
double MAXSPEED = 6.28;
leftMotor->setVelocity(MAXSPEED);
rightMotor->setVelocity(MAXSPEED);

DistanceSensor *ps0 = robot->getDistanceSensor("ps0");
ps0->enable(timeStep); // always need to enable sensor

while (robot->step(timeStep) != -1) {
// Robot loop
printf("distance:%f\n", ps0->getValue());
if (ps0->getValue() > 100) {
led0->set(true); // LED on
// Engines off
leftMotor->setVelocity(0);
rightMotor->setVelocity(0);
}
};

delete robot;
return 0;
}
```

- Kopieer deze code in zijn geheel naar het rechter scherm met de tab 'workshop_controller.cpp'.

- Bewaar de code op disk door op dit icoon boven het bestand te klikken:

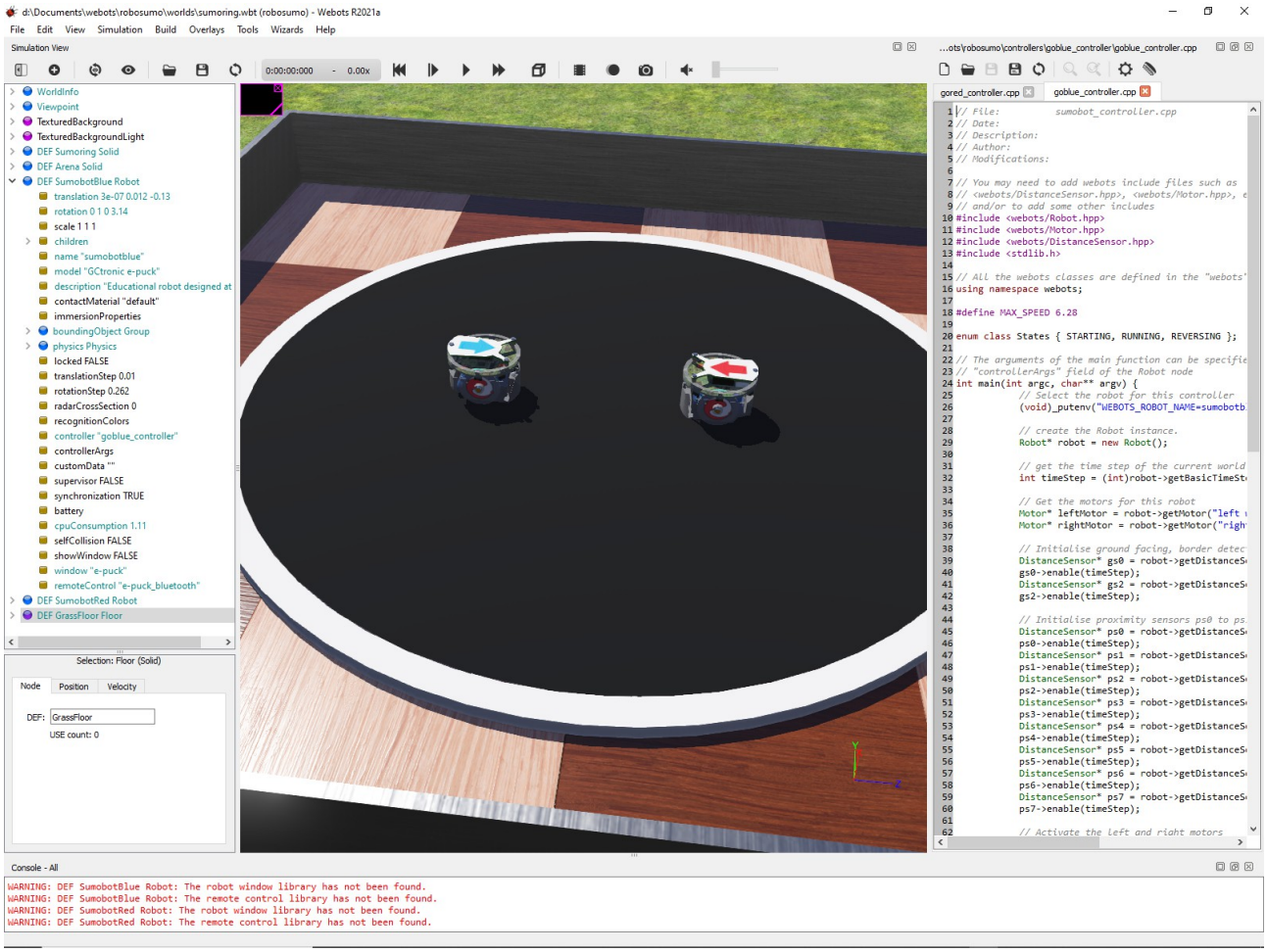
- Druk op F7 om de code te compileren.
- Laat de simulatie lopen door op het play icoon te klikken.

- Je robot zou nu zijn motoren moeten starten en bij een obstakel moeten stoppen.
- Stop de simulatie met het pauze icoon:

- Reset je simulatie met het terugspoel icoon:

Demonstratie RoboSumo in Webots en Feedback

De installatie handleiding voor onder meer de robosumo webots wereld is te vinden op:
https://github.com/roboticas/virtual



Handige webots links:

- RoboSumo webots wereld:
  https://github.com/roboticas/virtual

- Nederlandstalige webots handleiding:
  https://github.com/roboticas/virtual/blob/main/webots_handleiding_v1_2.pdf

- HCC robotica forum:
  https://groups.google.com/g/hcc_robotmc

Optioneel maar wel erg krachtig:
Visual Studio gebruiken met webots geeft **debug** mogelijkheden, zie de **Nederlandstalige handleiding over hoe je dit opzet**.

Vragen, suggesties of opmerkingen:
Rob van der Ouderaa (rouderaa@hccnet.nl)

Veel plezier met webots !