

Professional Workshop (2)
on
**Intelligent Soccer
Robotics: Build, Train,
Win!"**

Presented by Dr. Khaled Eskaf





- ❑ Ph.D in Computer Science from Salford University (UK), Professor in Intelligent System. Dean College of Informatics- Midocean University. More than 20 years in Teaching computer science courses.
- ❑ Training, Coaching Students in Robotics,
- ❑ Programming Languages (Java, python,Kotlin)
- ❑ Game Programming, Virtual Reality, Augmented Reality and Embedded System.
- ❑ Judging for many local and national Robotics competitions.

1**Day****2**

Day 1: Introduction to Robotics and Basic Programming

Session 1: Welcome and Introduction

Session 2: Basic Components of Robots

Session 3: Motors and Actuators

Session 4: Sensor Integration

Session 5: Introduction to Webots

- Overview of Webots Simulation Environment
- Setting Up Your First Webots Project
- Creating and Running Simple Robot Controllers

Hands-on Examples:

1. Obstacle Detection and LED Flashing with e-puck Robot
2. Capturing and Analyzing Images with e-puck Robot's Camera
3. Red Obstacle Detection and Navigation with e-puck Robot
4. Color Score-Based Speed Control for e-puck Robot

Day 2: Advanced Applications and Machine Learning

Session 1: Recap and Q&A

The First Virtual Robo Soccer Match

Put your robot to the test in a real-time soccer match simulation

Who is the Father of robotics ?

1- George Devol

2- Ismail al-Jazari

3- Isaac Newton



Who is the Father of robotics ?

1- George Devol



Devol's work in the 1950s and 1960s laid the groundwork for the robotics industry as we know it today

2- Ismail al-Jazari



"Father of Robotics." Al-Jazari was a 12th-century Islamic scholar and inventor who created numerous automatons and mechanical devices. His book, "The Book of Knowledge of Ingenious Mechanical Devices," described various innovative machines, including water clocks and automated musical instruments.

3- Isaac Newton



contributions to physics and mathematics, he is not typically associated with robotics.

Day
01



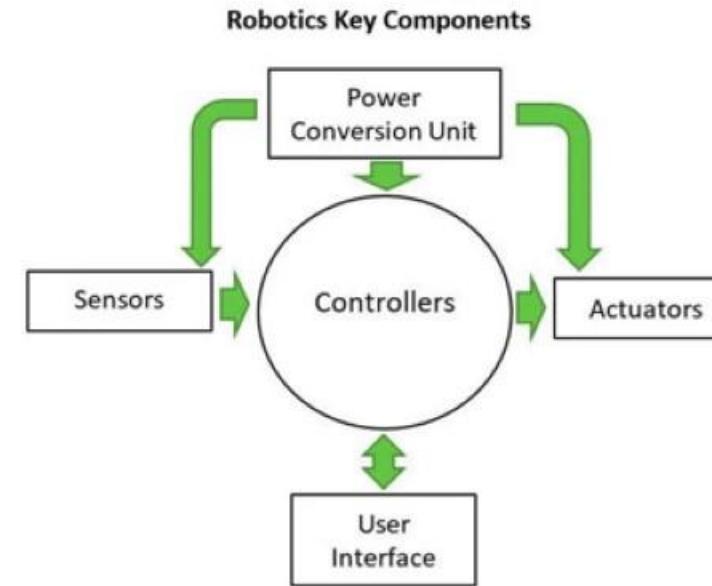
The Core of Robotics

Day

01

The Core Components of a Robot

An illustrative diagram showing the robot's core components: Computer Parts, Mechanical Components, Sensors, and Embedded Systems, leading to Human-Robot Interaction.



The Core of Robotics : Understanding Computer Parts

Day
01

Computer Parts in Robotics

MIDOCEAN
UNIVERSITY

Simplified explanations of CPUs, Memory, and Storage, using metaphors like the robot's "brain" and "memory bank."



Actuators: The Muscles of Robots

1. What Are Actuators?

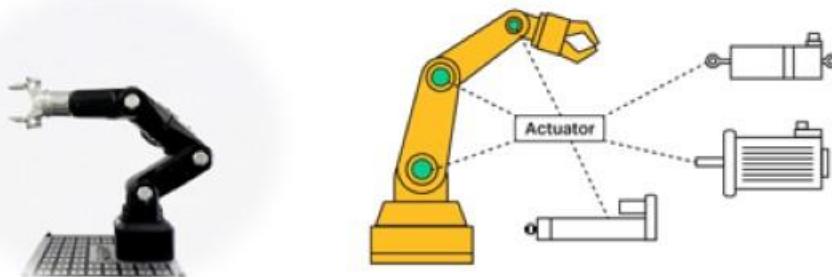
1. Brief Definition: Devices that **convert electrical energy** into **physical motion**.
2. Analogy: Just as muscles allow humans and animals to move, actuators enable robots to walk, grasp, lift, and perform other actions.

2. Types of Actuators

Electric Motors: The most common type, used in robotic arms, drones, and more. Illustrate with a snippet showing an electric motor in a robotic arm.

3. Why Actuators Are Important

Emphasize the role of actuators in expanding what robots can do, making them capable of interacting with the physical world in diverse and complex ways.



Day

01

Motors: Powering Robotic Movement

1. Introduction to Motors in Robotics

1. Brief explanation of how motors are the driving force behind robotic movement, acting as the "muscle" that powers everything from robotic arms to autonomous vehicles.

2. Types of Motors

1. Servo Motors:

1. Characterized by their ability to control angular position with precision. Commonly used in robotic arms, antennas, or any application where precise positioning is required.

2. Stepper Motors:

1. Known for their accuracy in controlling steps, making them ideal for applications requiring exact movements, such as 3D printers and CNC machines.

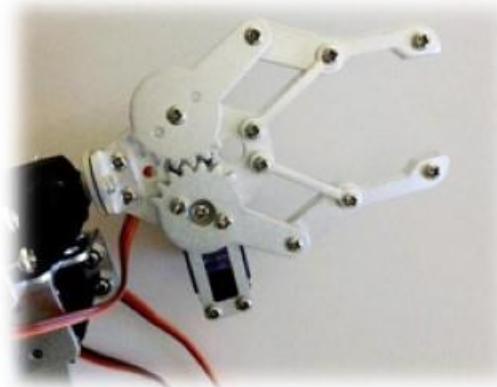
3. DC Motors:

1. Direct Current motors offer simple control and are used in a wide range of applications due to their speed variability and torque. Examples include hobby robots, drones, and electric vehicles.

Day

01

Servo Motor



Day

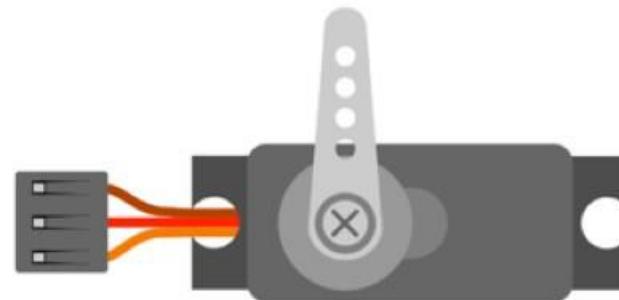
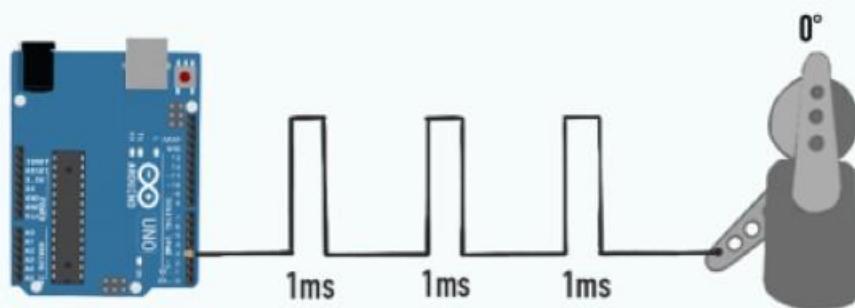
01

Servo motor

You can rotate the servo motor shaft from 0° to 180° by varying the PWM pulse width from 1 ms to 2 ms.

PWM (Pulse Width Modulation)

SERVO MOTOR CONTROL



NAME	WIRE COLOR	DESCRIPTION
PWM	Yellow	Input control signal for the motor
V+	Red	Positive supply terminal
Ground	Brown	Ground terminal

Day

01

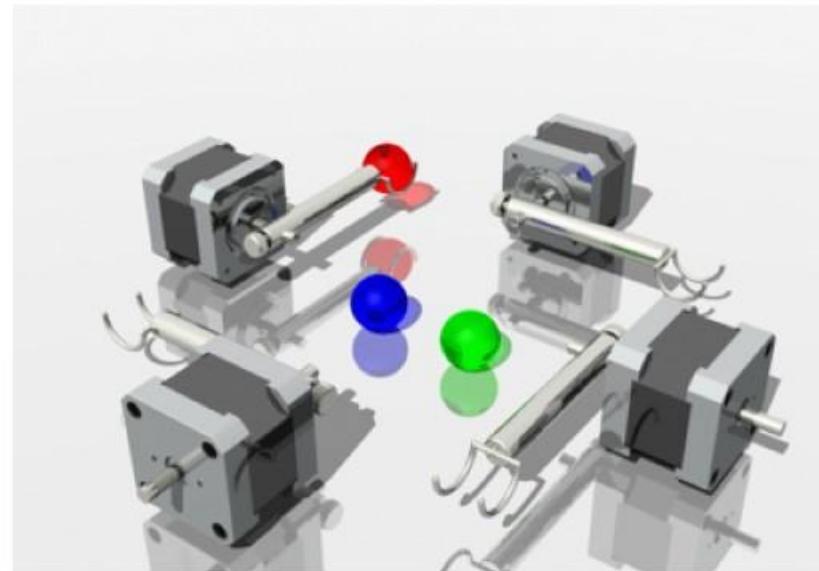
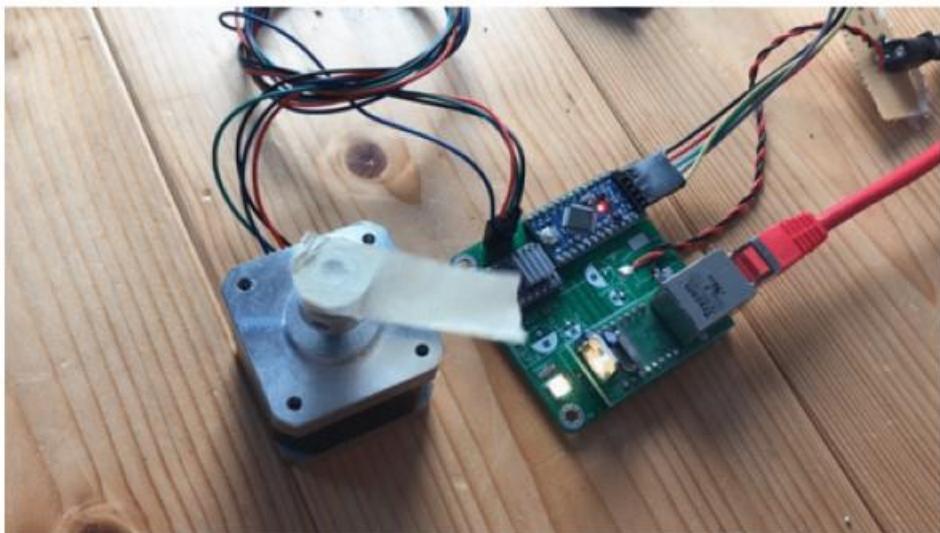
Stepper Motor



Day

01

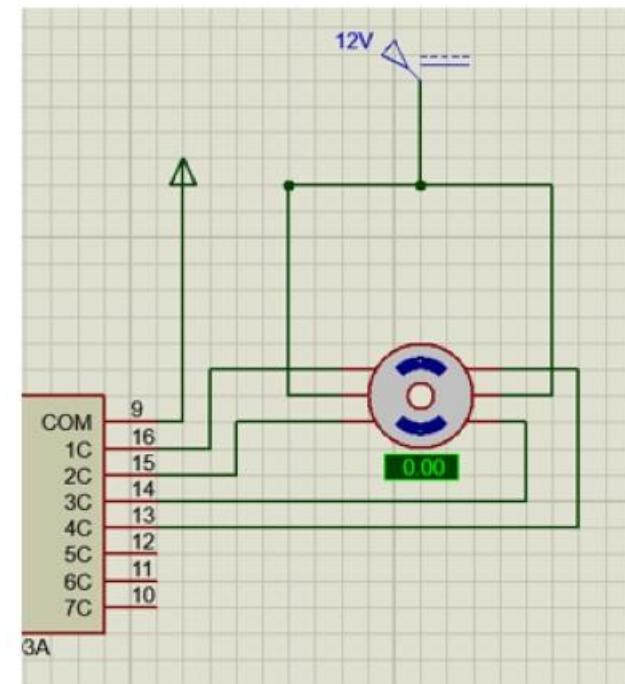
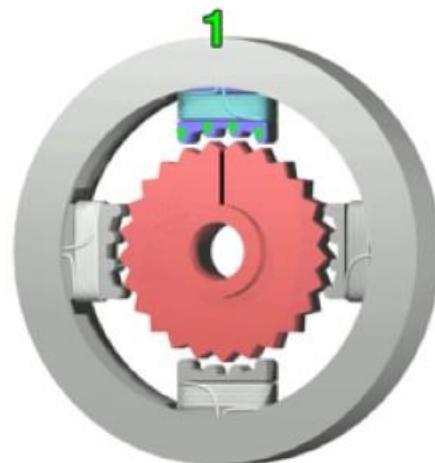
Stepper Motor



Day

01

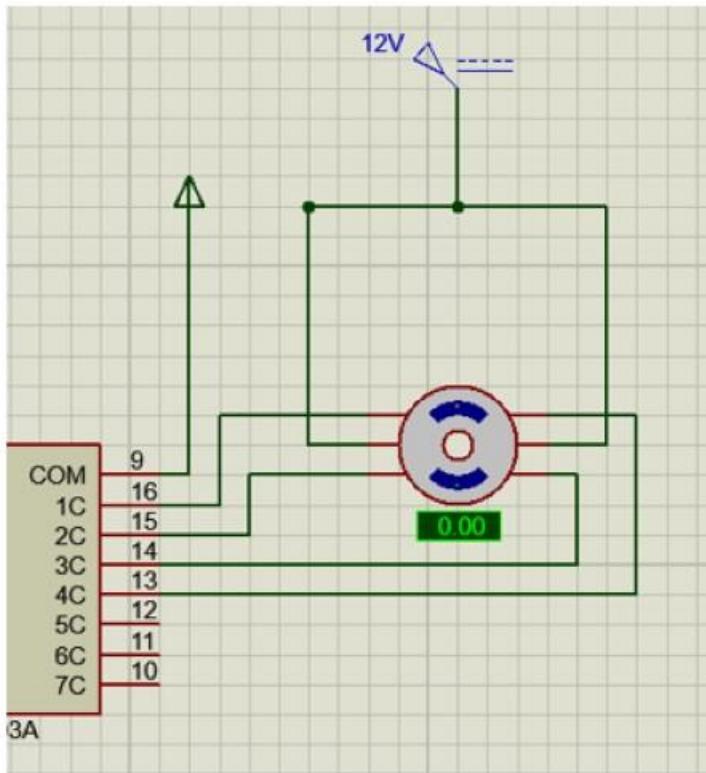
Stepper Motor



Day

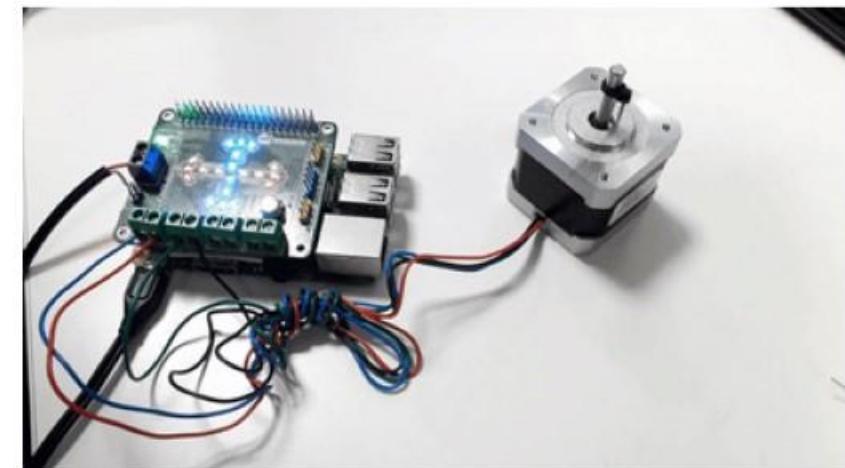
01

Stepper Motor



In the following table, A-B-C-D refers to the stator coils, that are to be energized sequentially in the manner and '1's and '0's refers to 'HIGH' and 'LOW' states.

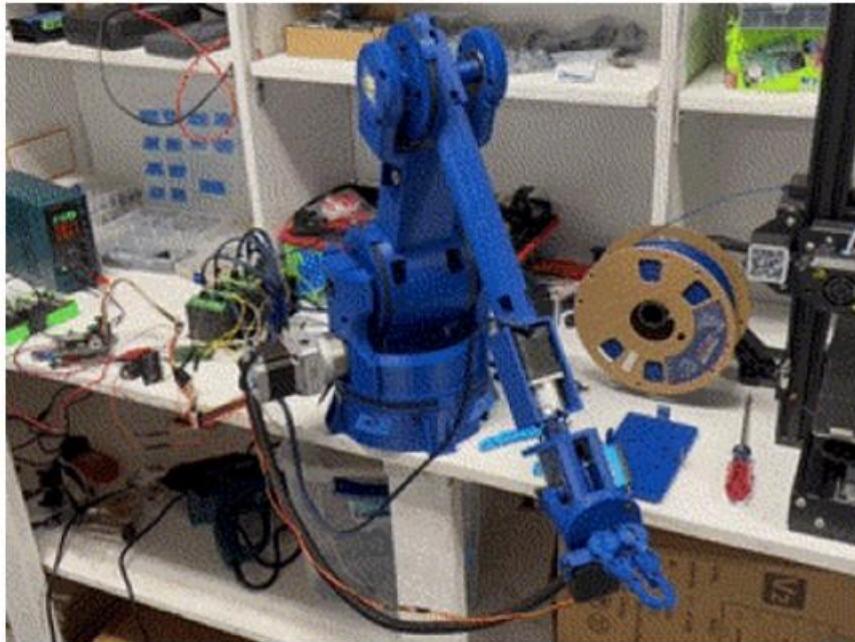
Steps	A	B	C	D	HEX
1	1	0	0	0	0x08
2	0	1	0	0	0x04
3	0	0	1	0	0x02
4	0	0	0	1	0x01



Day

01

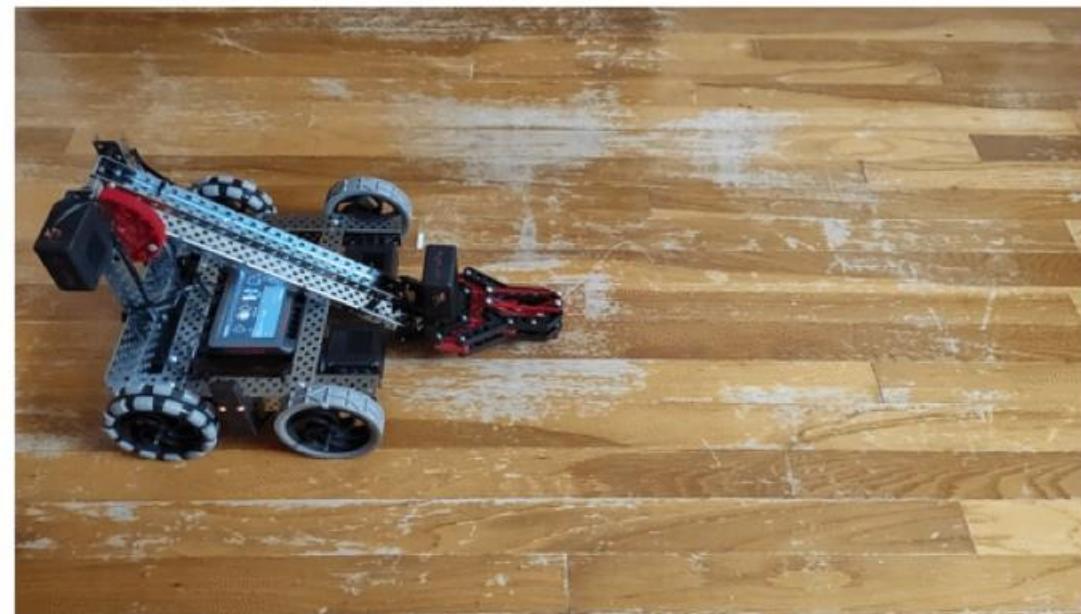
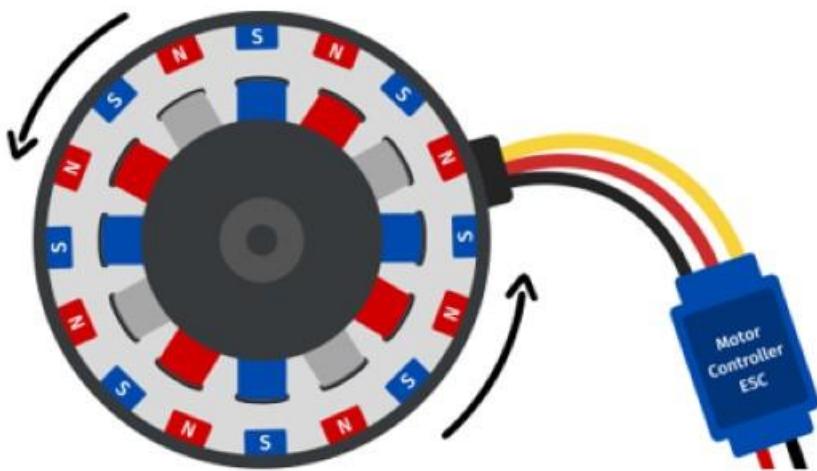
Stepper Motor



Day

01

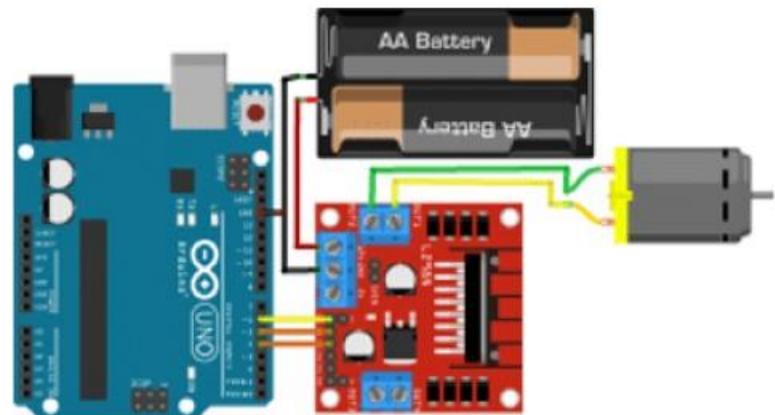
DC Motor



Day

01

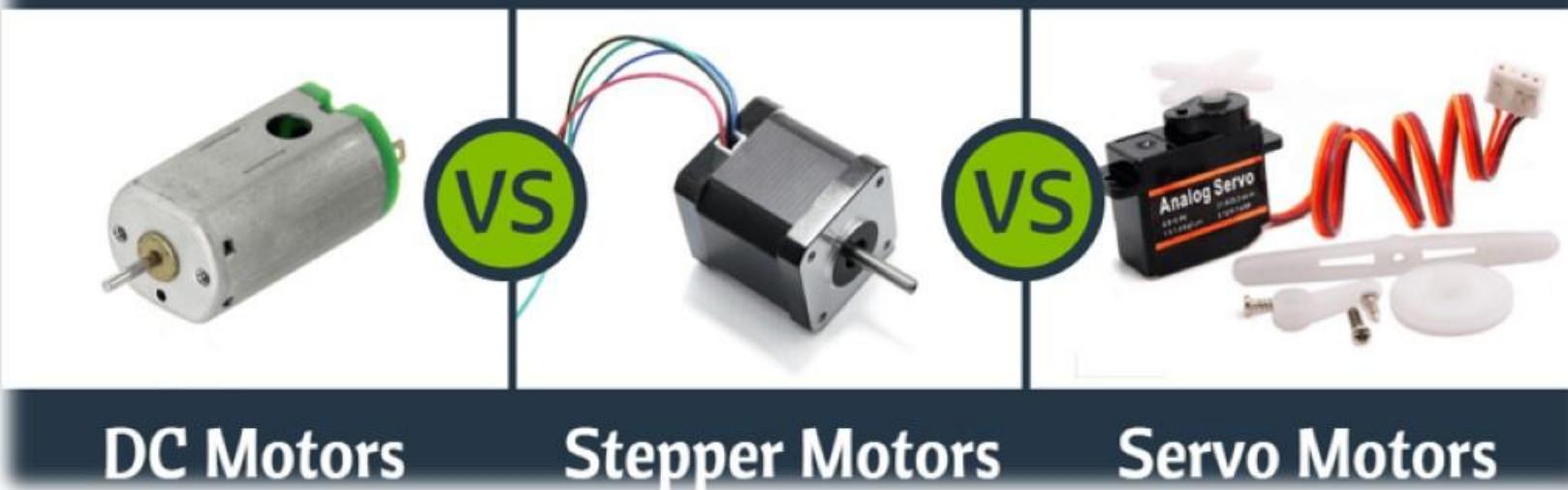
DC Motor



Arduino uno
D5
D4
GND

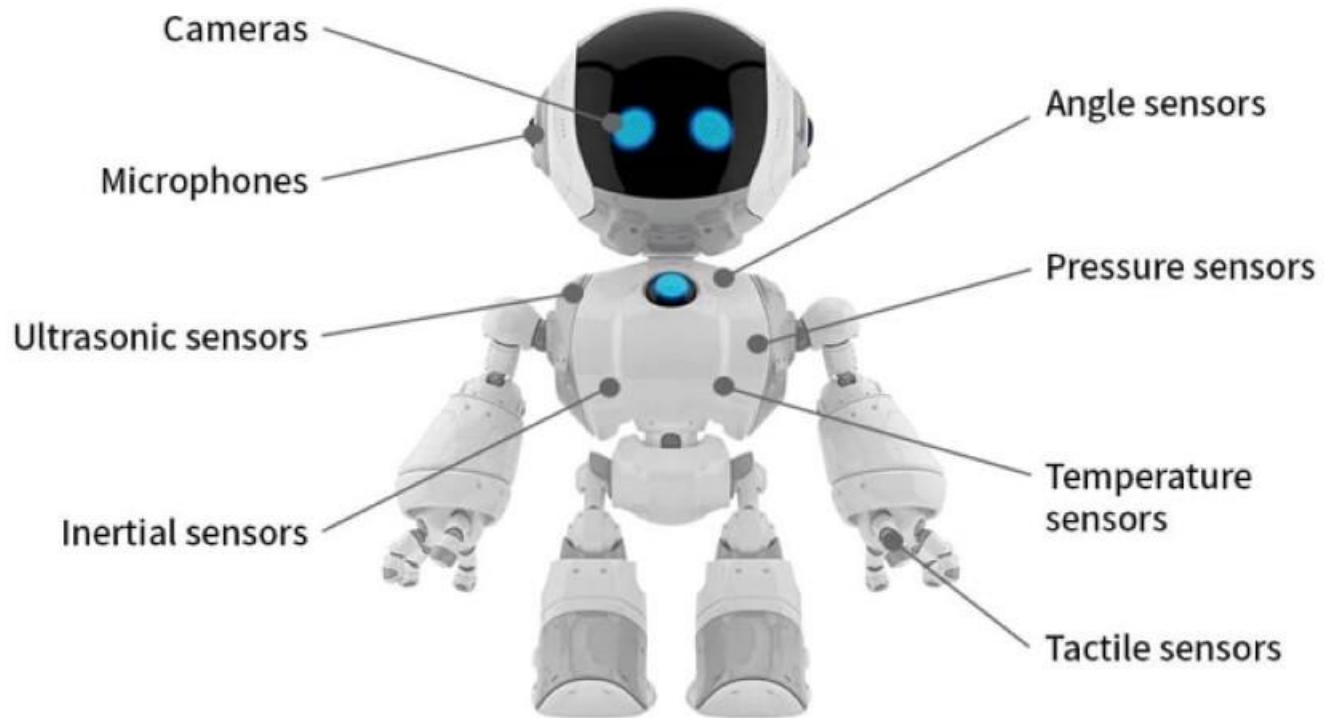
Motor driver
IN1
IN2
GND

The Difference Between



Fundamental differences between DC, Servo, and Stepper motors

Motor Type	Control	Precision	Speed	Torque	Primary Use in Robotics
DC Motor	Variable speed control	Low	High	Medium	Ideal for applications requiring variable speed and direction, such as robot wheels or drones.
Servo Motor	Positional control	High	Medium	High at low speeds	Best for precise control of movement, such as robotic arms or where specific positioning is crucial.
Stepper Motor	Step control	High	Low to Medium	High at low speeds	Suited for precise steps in motion, making them perfect for 3D printers, CNC machines, and robots requiring detailed positioning without feedback.



Vision Sensors: The Eyes of Robots

• **Overview:** Introduction to how vision sensors allow robots to perceive their environment visually, simulate human sight.

• **Types:**

- **Cameras (2D and 3D):** Capture visual information. 2D cameras provide flat images, while 3D cameras add depth perception, crucial for tasks requiring spatial understanding.
- **LIDAR (Light Detection and Ranging):** Uses laser pulses to map out the robot's surroundings in high detail, essential for navigation and obstacle avoidance.

• **Applications:**

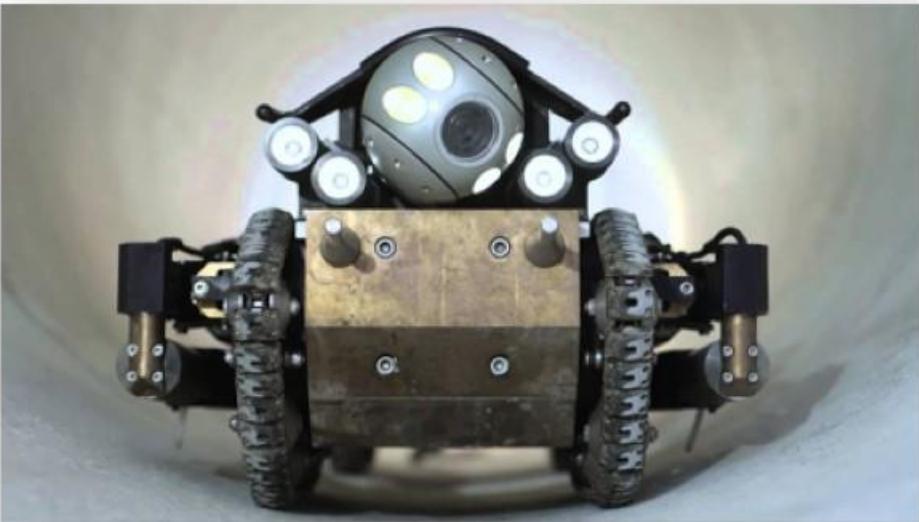
- Object recognition, navigation, and environmental mapping.



Day

01

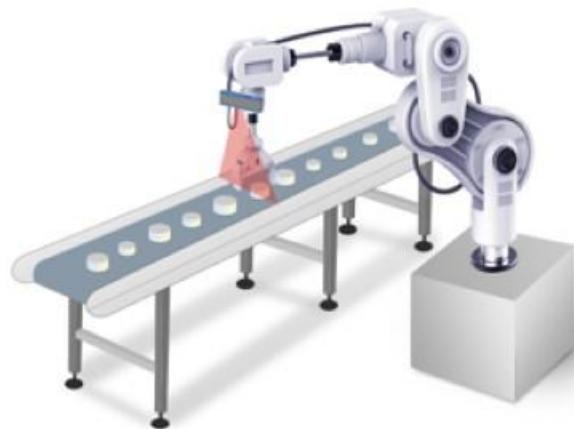
2D Camera Sensor



Day

01

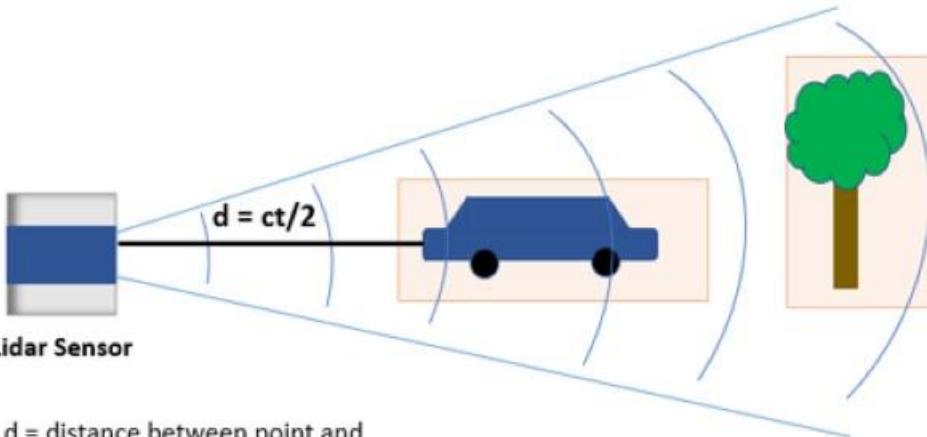
3D Camera Sensor



Day

01

LIDAR (Light Detection and Ranging):

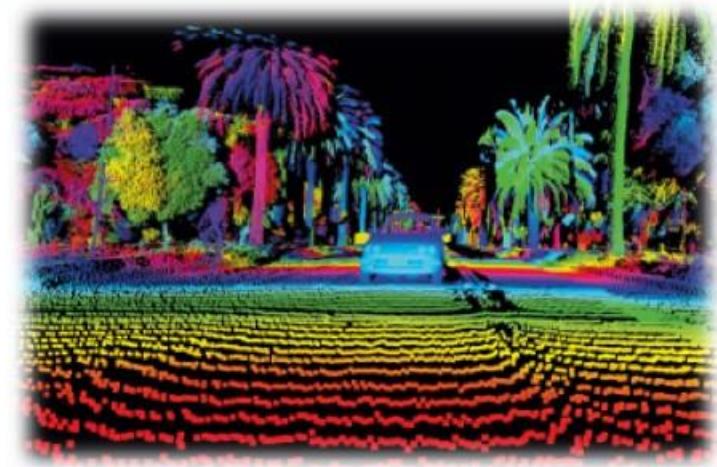


Lidar Sensor

d = distance between point and
the sensor

c = speed of light

t = time of flight



• **Overview:** Explaining the role of proximity and distance sensors in detecting the presence of objects and measuring the distance to them.

• **Types:**

- **Ultrasonic Sensors:** Emit ultrasonic waves and measure the reflection time to determine distance to objects. Widely used in robotic cars for obstacle detection.
- **Infrared Sensors:** Use infrared light to detect objects and estimate distance, useful in line-following robots and simple navigation tasks.

• **Applications:**

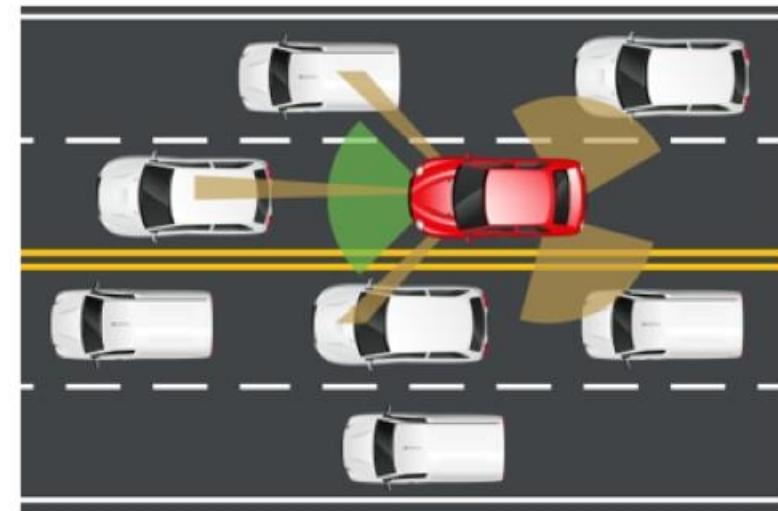
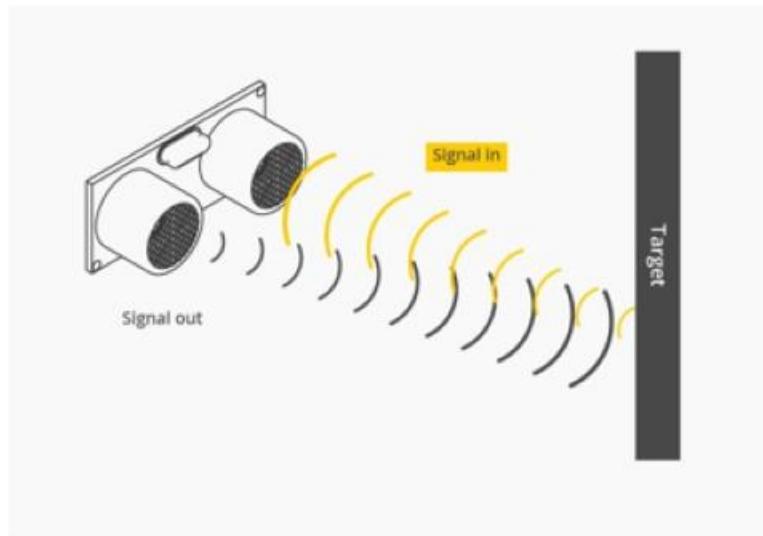
- Collision avoidance, area scanning, and precise movements in tight spaces.



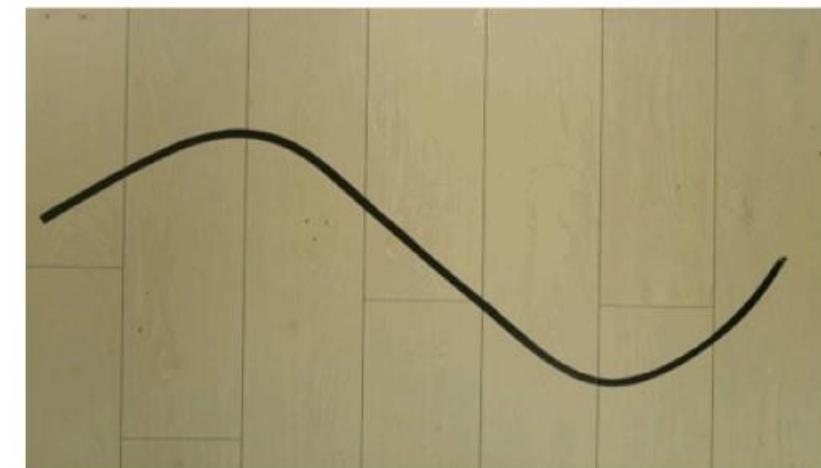
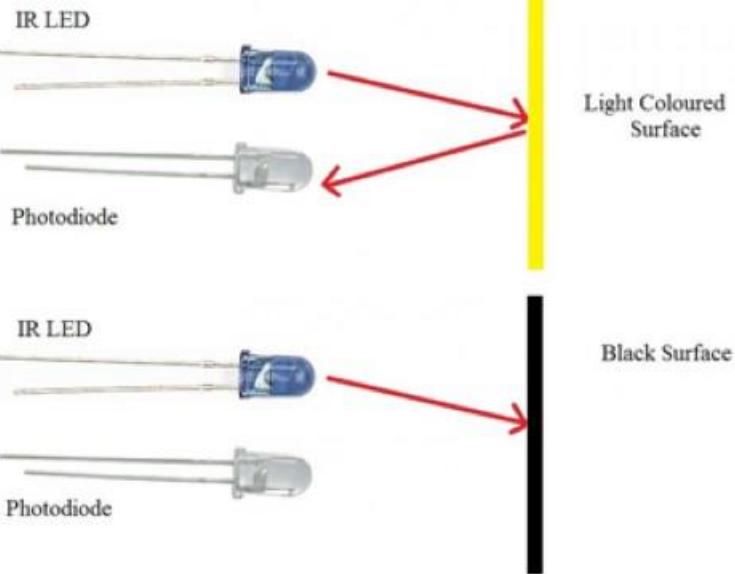
Day

01

Ultra Sonic Sensor



Infra red Sensor



Feeling and Interacting: Tactile and Environmental Sensors

- **Overview:** Highlighting sensors that allow robots to 'feel' their environment and respond to physical interactions or changes in their surroundings.

- **Types:**

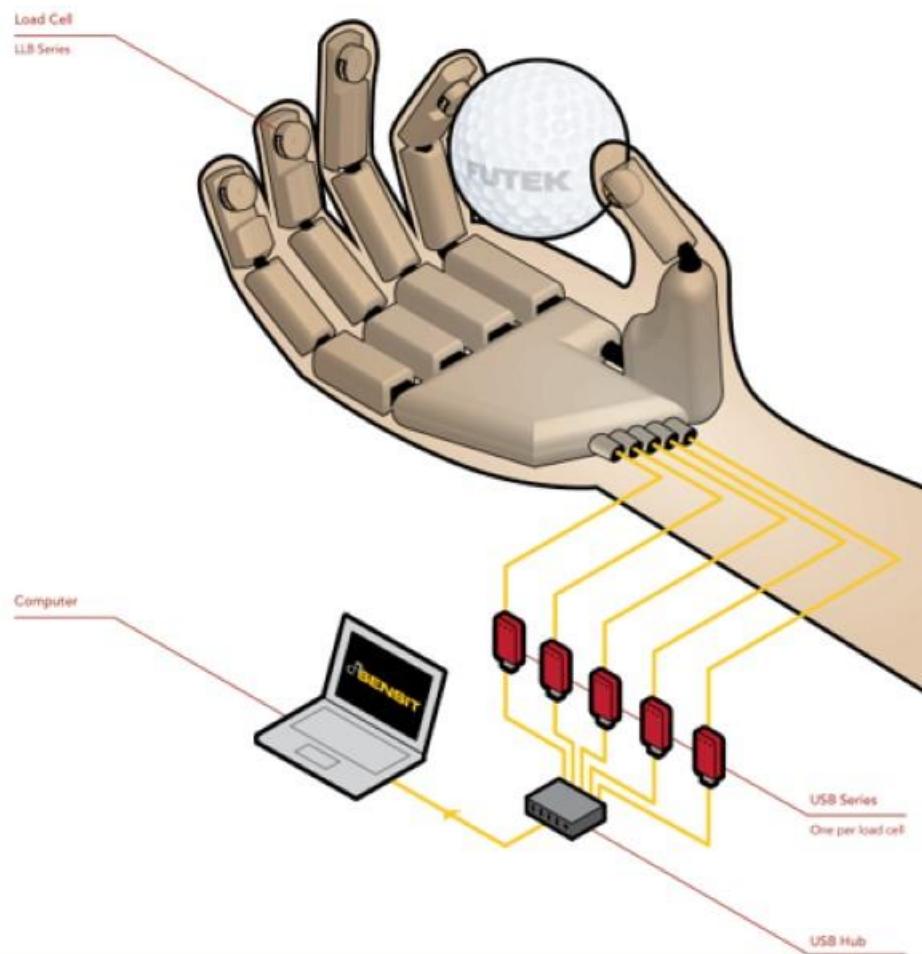
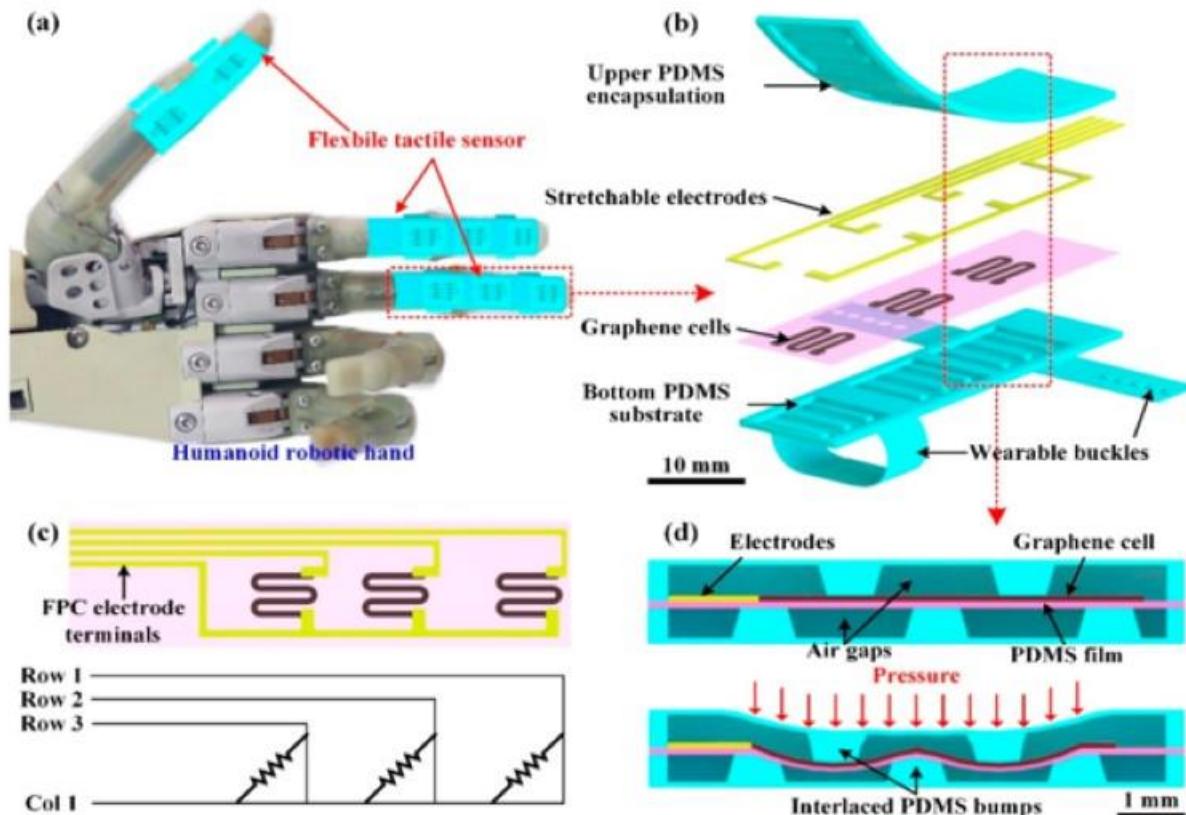
- **Tactile Sensors:** Enable robots to detect physical contact, pressure, and textures. Key in applications requiring delicate handling or manipulation, like robotic hands.
- **Environmental Sensors:** Measure temperature, humidity, or gas concentrations, critical for robots operating in varying or extreme conditions, such as disaster response robots.

- **Applications:**

- Hazard detection, material handling, and adapting to environmental conditions.



Tactile Sensors



Day

01

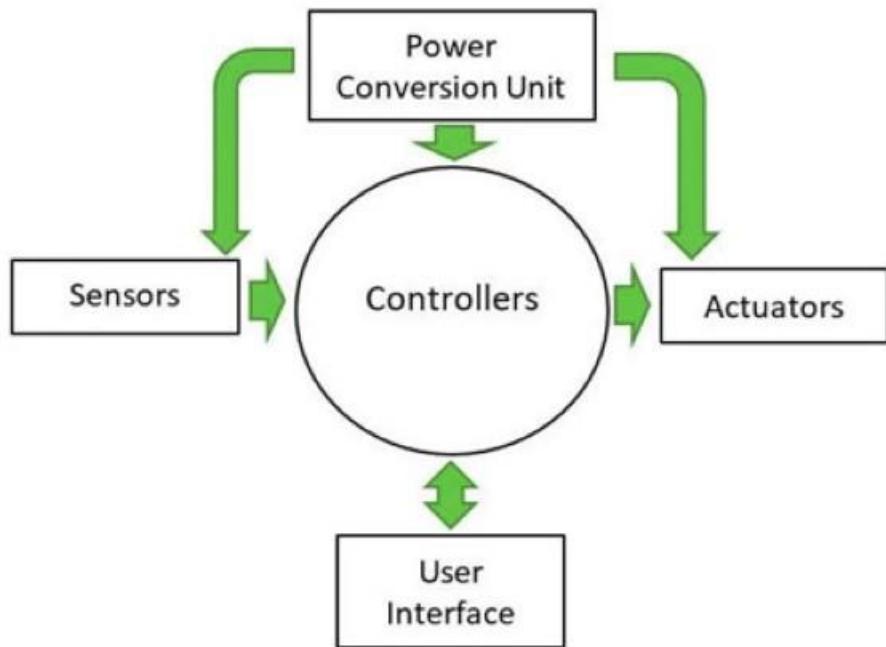
Environmental Sensors

MIDOCEAN
UNIVERSITY



Bringing It All Together

Robotics Key Components



Introduction to Webots



Webots

robot simulation

Webots is an open source and multi-platform desktop application used to **simulate** robots



<https://cyberbotics.com/>



Webots Open Source



1 model

2 program

3 simulate



4 transfer

Introduction to Webots

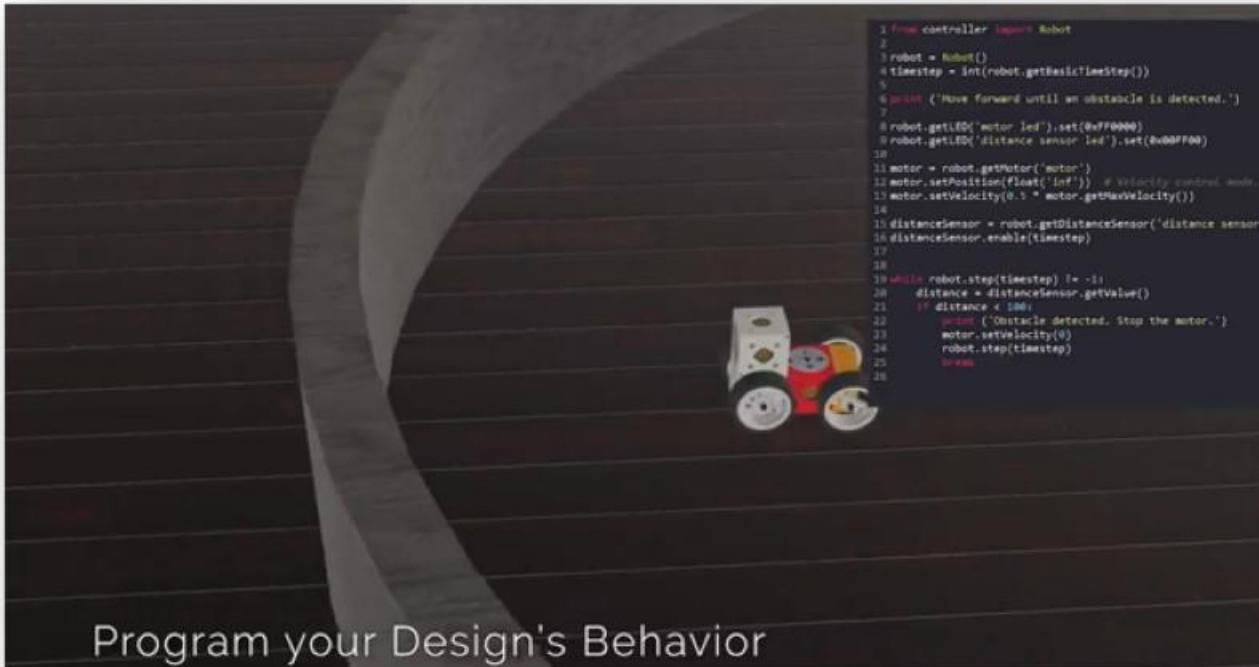


Used in Thousands of Universities Worldwide



Rapidly Design your Robot Model

Introduction to Webots



```
1 from controller import Robot
2
3 robot = Robot()
4 timestep = int(robot.getBasicTimestep())
5
6 print ('Move forward until an obstacle is detected.')
7
8 robot.getLED('motor led').set(0x770000)
9 robot.getLED('distance sensor led').set(0x00FF00)
10
11 motor = robot.getMotor('motor')
12 motor.setPosition(float('inf')) # Velocity control mode.
13 motor.setVelocity(0.5 * motor.getMaxVelocity())
14
15 distanceSensor = robot.getDistanceSensor('distance sensor')
16 distanceSensor.enable(timestep)
17
18
19 while robot.step(timestep) != -1:
20     distance = distanceSensor.getValue()
21     if distance < 100:
22         print ('Obstacle detected. Stop the motor.')
23         motor.setVelocity(0)
24     robot.step(timestep)
25
26
```

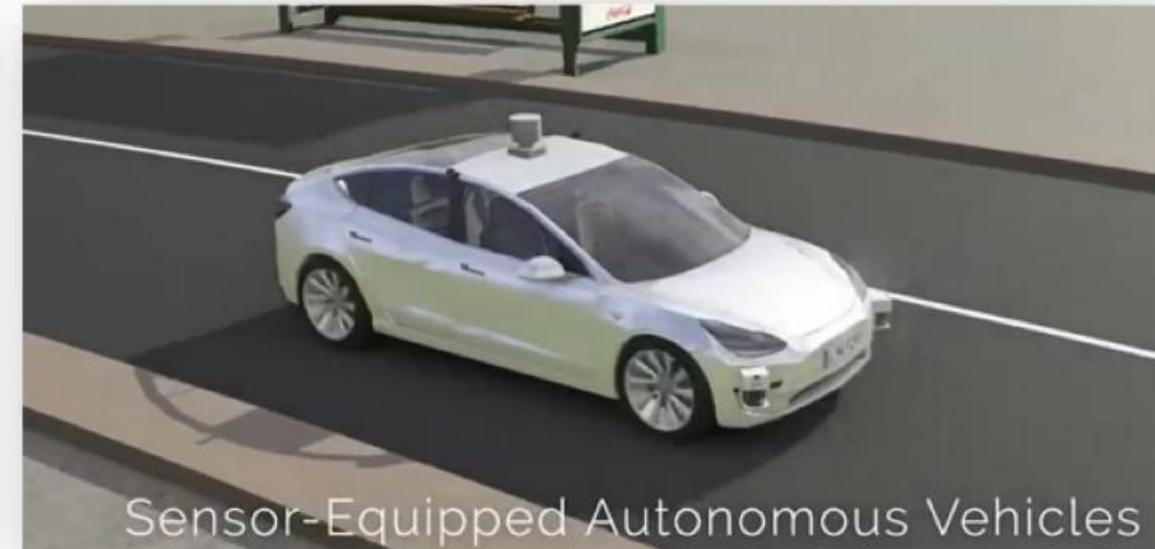
Program your Design's Behavior



Introduction to Webots



Advanced Automobile Simulation



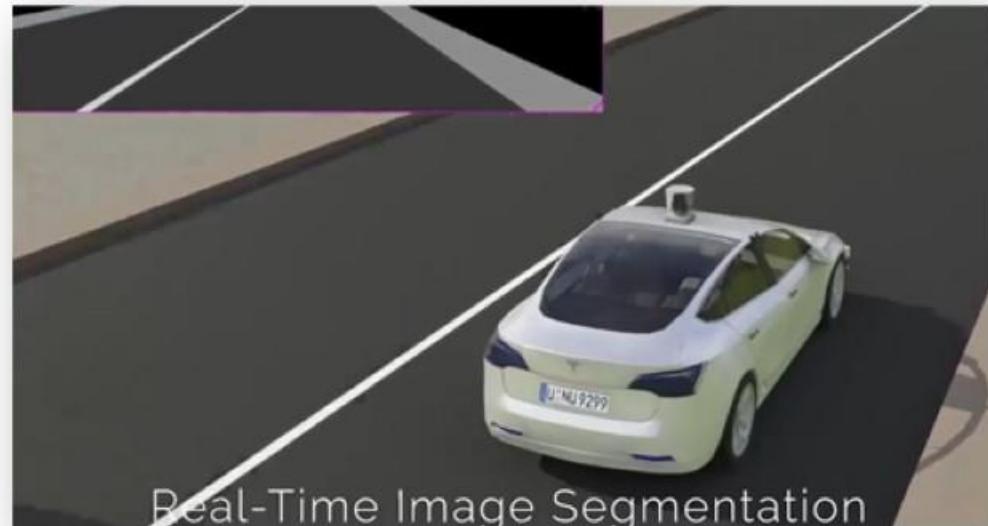
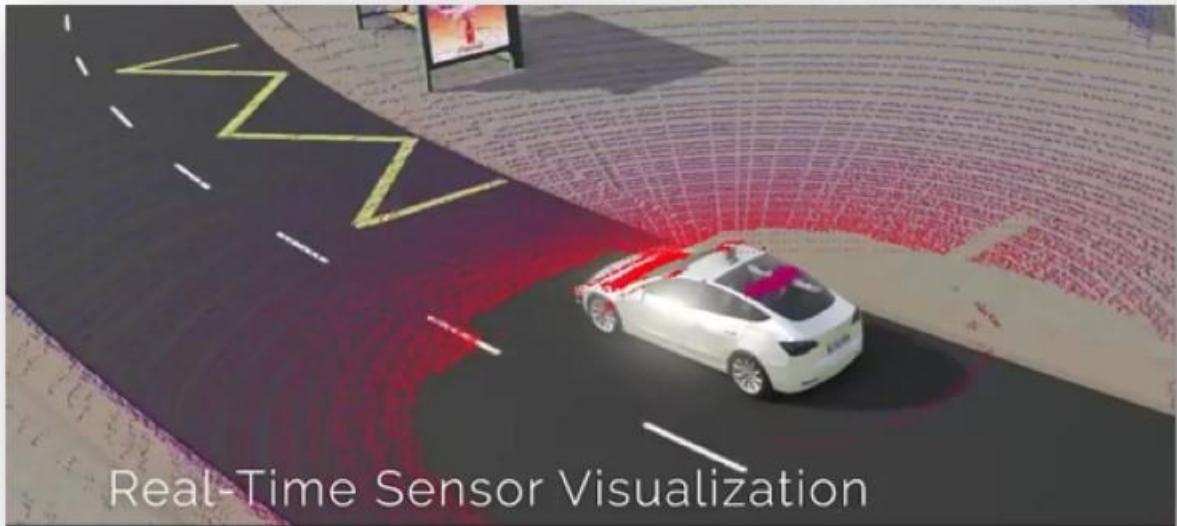
Sensor-Equipped Autonomous Vehicles

Day

01

Introduction to Webots

MIDOCEAN
UNIVERSITY



Webots is Amazing Because:

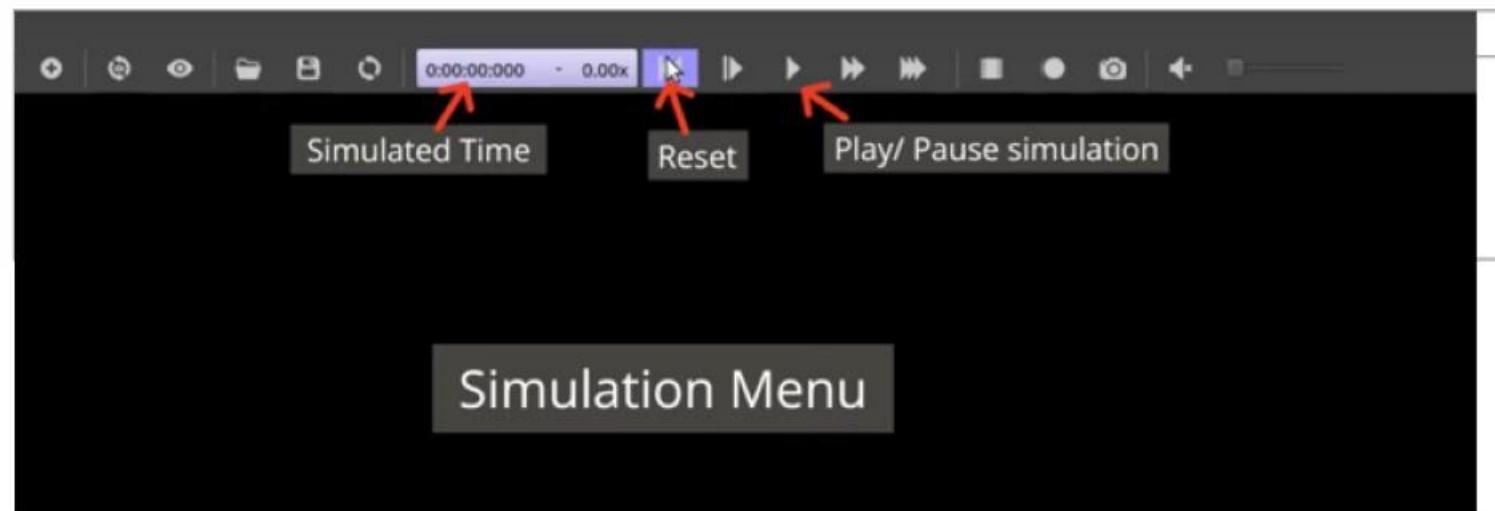
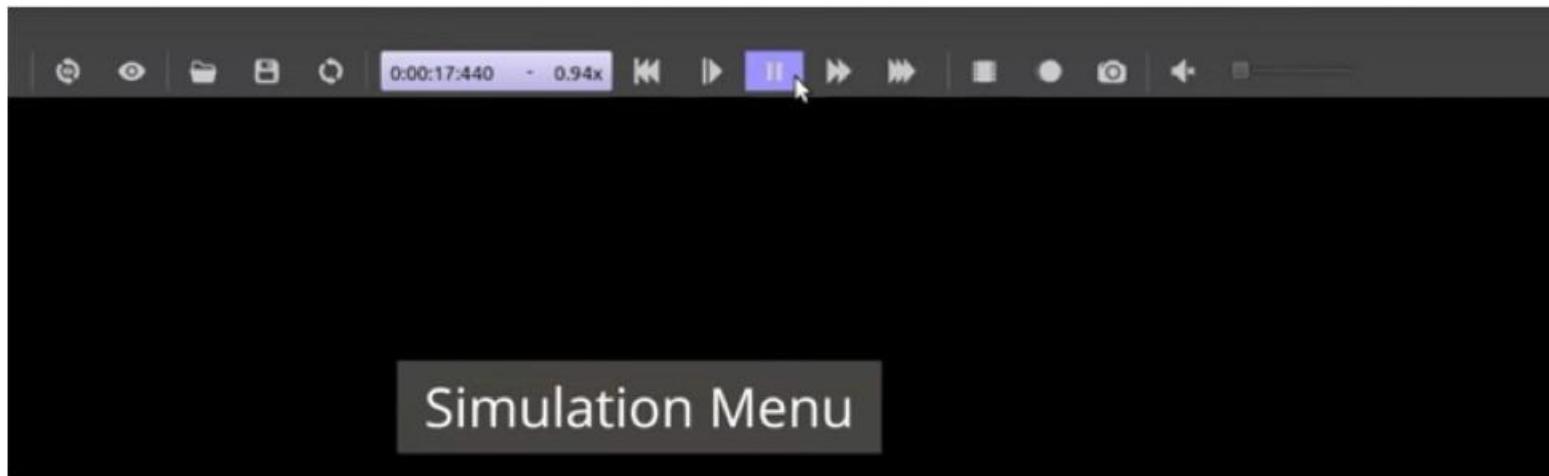
- **Real Robots on Your Screen:** Your virtual robots in Webots move just like real ones. It's like having a robot pal in your computer!
- **Create Your Own Robot Places:** Make all sorts of places for your robots to explore, like cities, parks, or space stations!
- **Loads of Ready-Made Robots:** Choose from many robots that are already built in Webots. Start playing and learning right away.
- **Works on Any Computer:** You can use Webots on Windows, Mac, or Linux. It's super flexible!
- **Easy Robot Coding with Python:** Use Python, a simple programming language, to tell your robots what to do. Perfect for beginners and experts!
- **Teach Robots to Think:** Make your robots smart with Python. They can learn to solve problems by themselves, like real AI!
- **Robots with Cool Parts:** Add neat parts to your robots, like cameras, hands, or sensors, so they can see, touch, and feel.
- **All-in-One Robot Kit:** Webots has everything you need to create and control amazing robots.
- **For School, Research, and Business:** You can play with Webots alone, use it in school, or even in big research projects and businesses.
- **Used by Top Universities and Companies:** Many of the best universities and big companies in America use Webots. Places like MIT, Stanford, and tech giants in Silicon Valley trust Webots for their robot projects.

Download & Install
User interface (UI)
Create your first project
Make your own object
Understand fundamentals
Run your first simulation
Add a robot & run simulation
Where is the code

Objective

User Interface (UI)

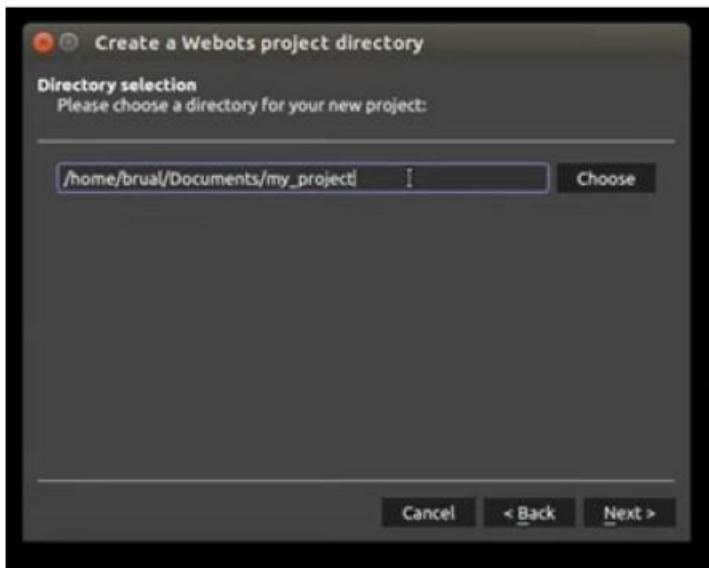
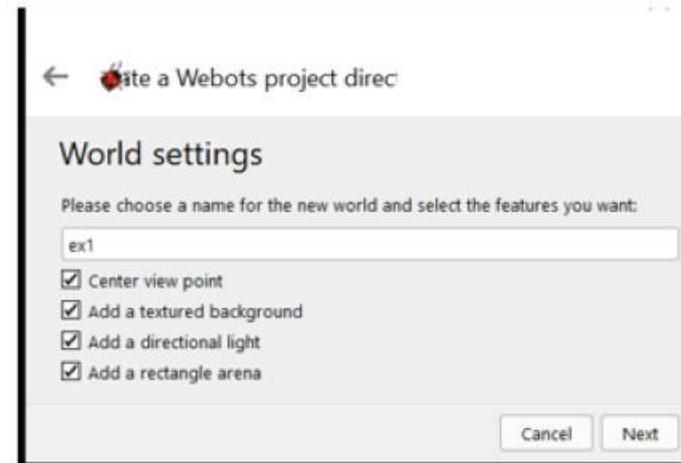
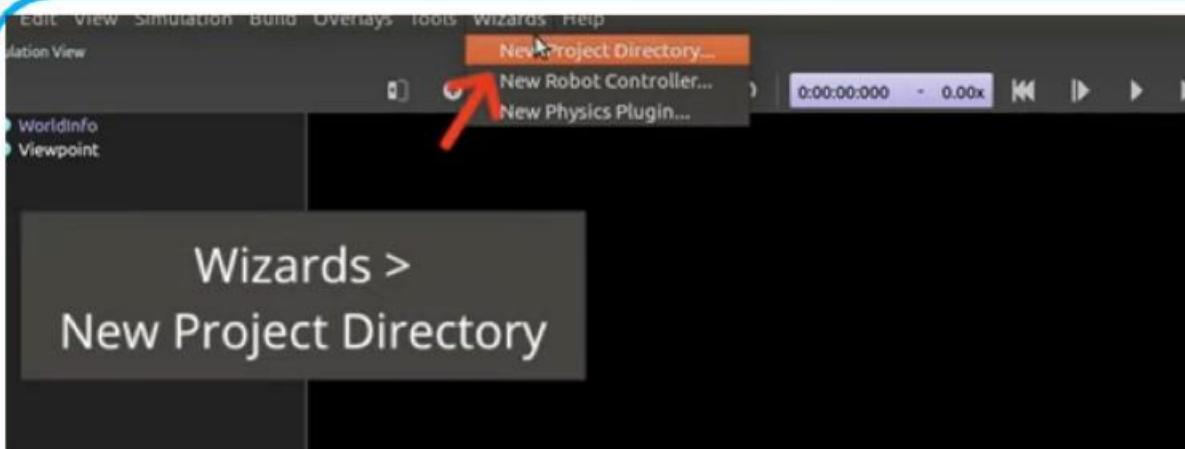
Create your first Project



Day

01

Create new Project



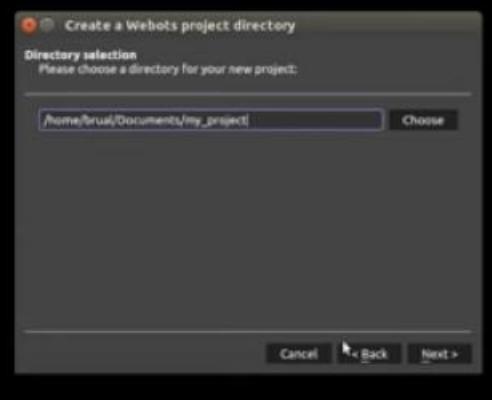
Day

01

Create new Project

(1) Use "Choose" to choose where to save your project.

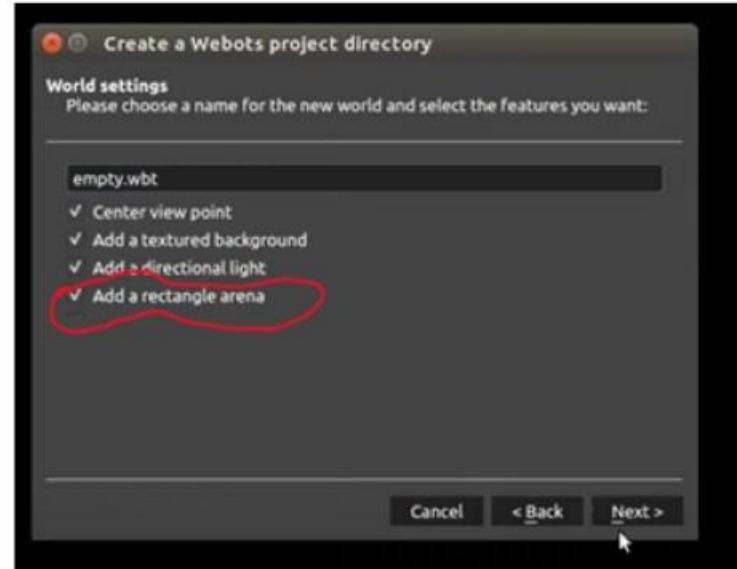
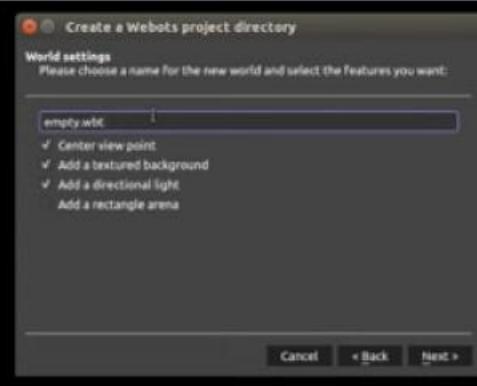
(2) Name your project.
Eg: my_project

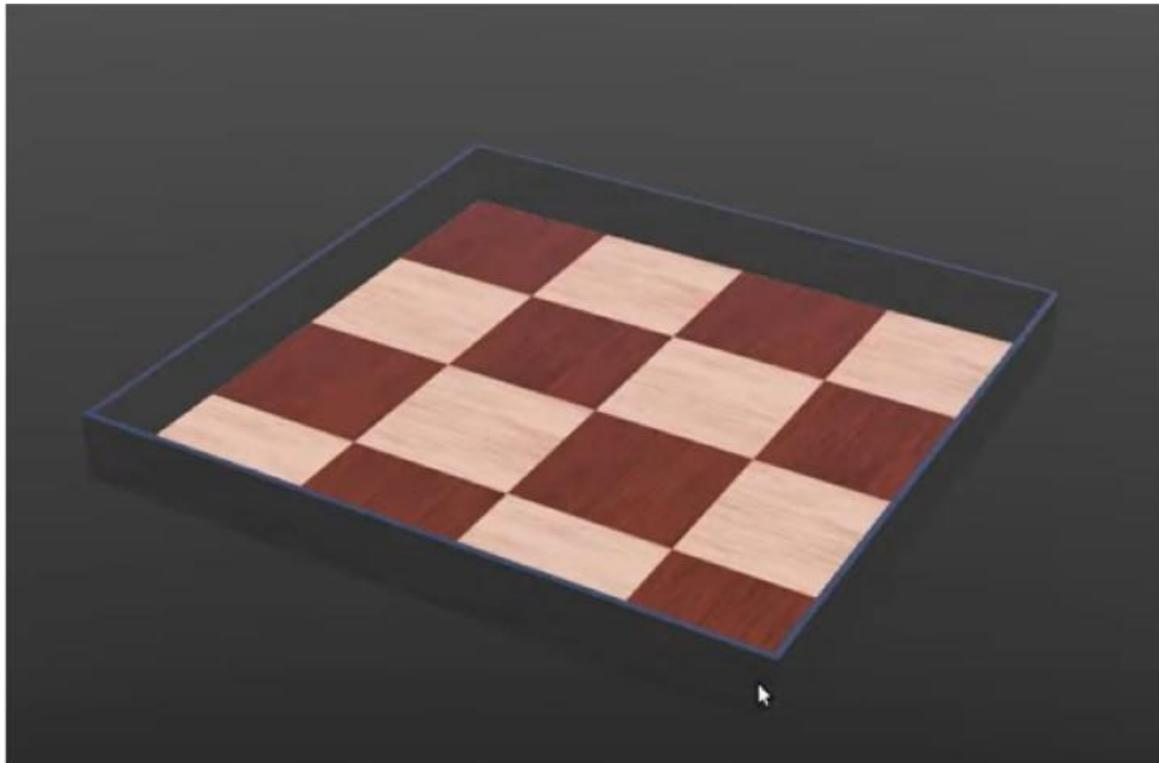


(3) Click Next

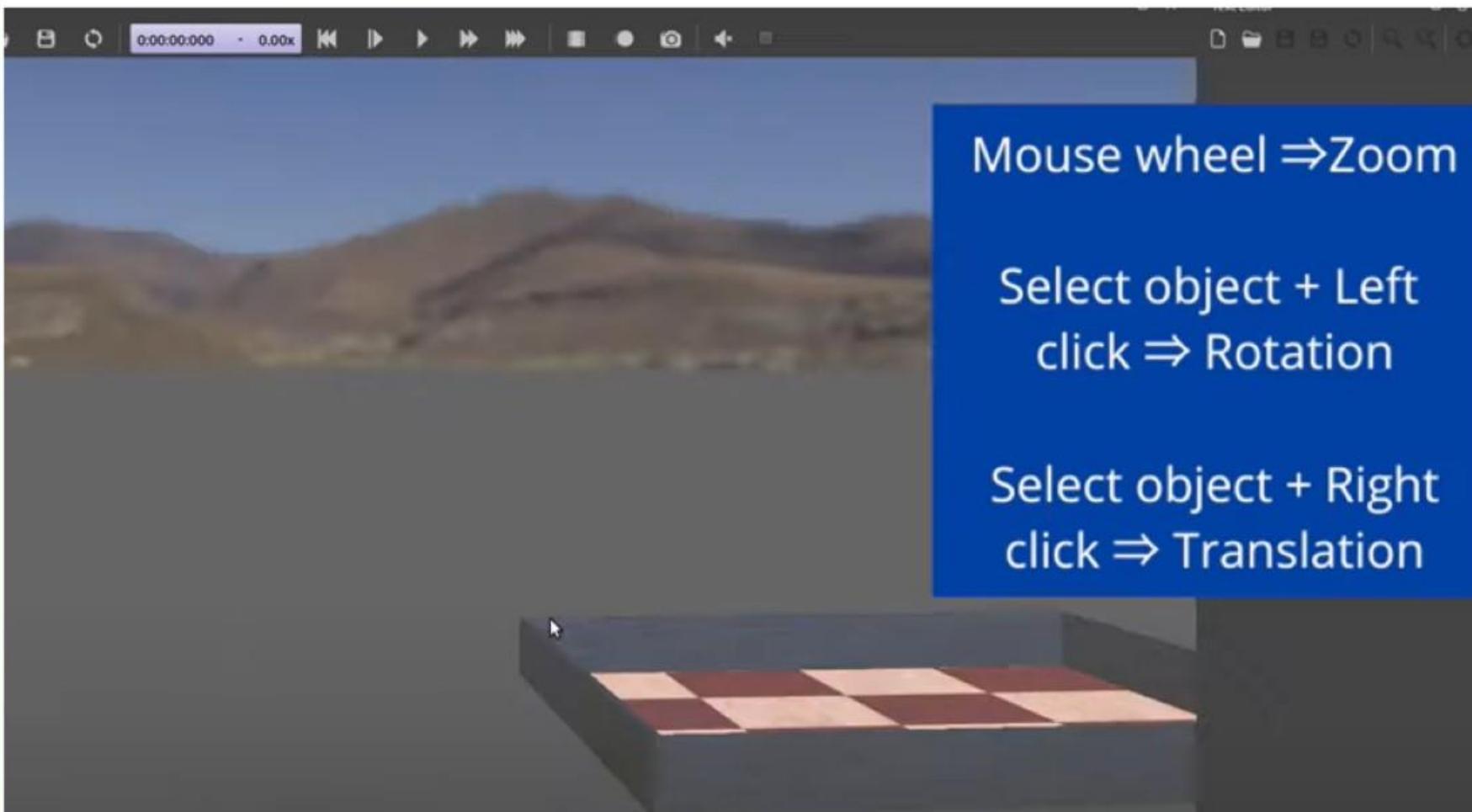
(4) Select "Add rectangle arena"

(5) Click Next & Finish

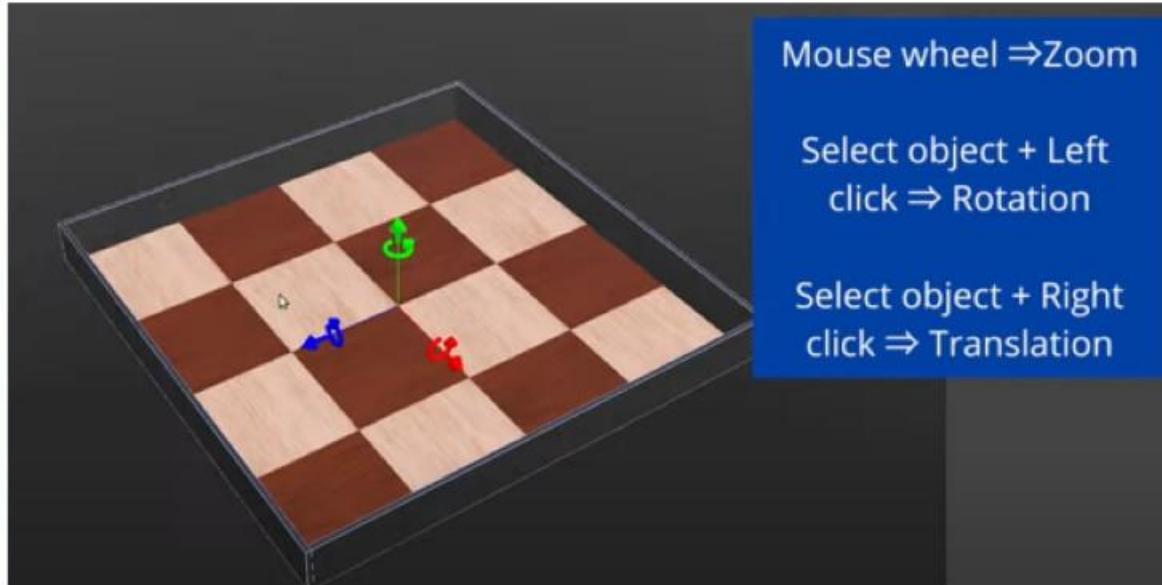




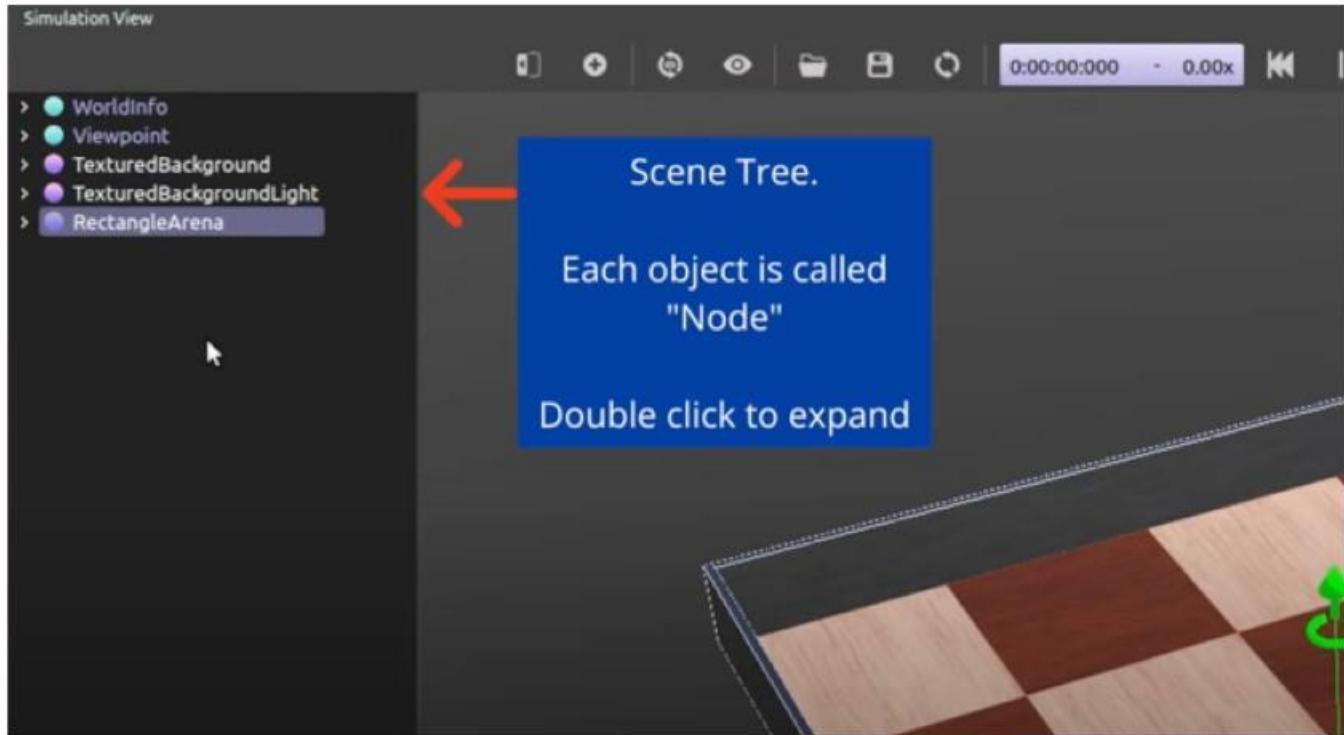
Loaded with a rectangular arena
essentially a floor with four walls.



learn how to zoom in
and zoom out this is
simply done by scrolling
in and out



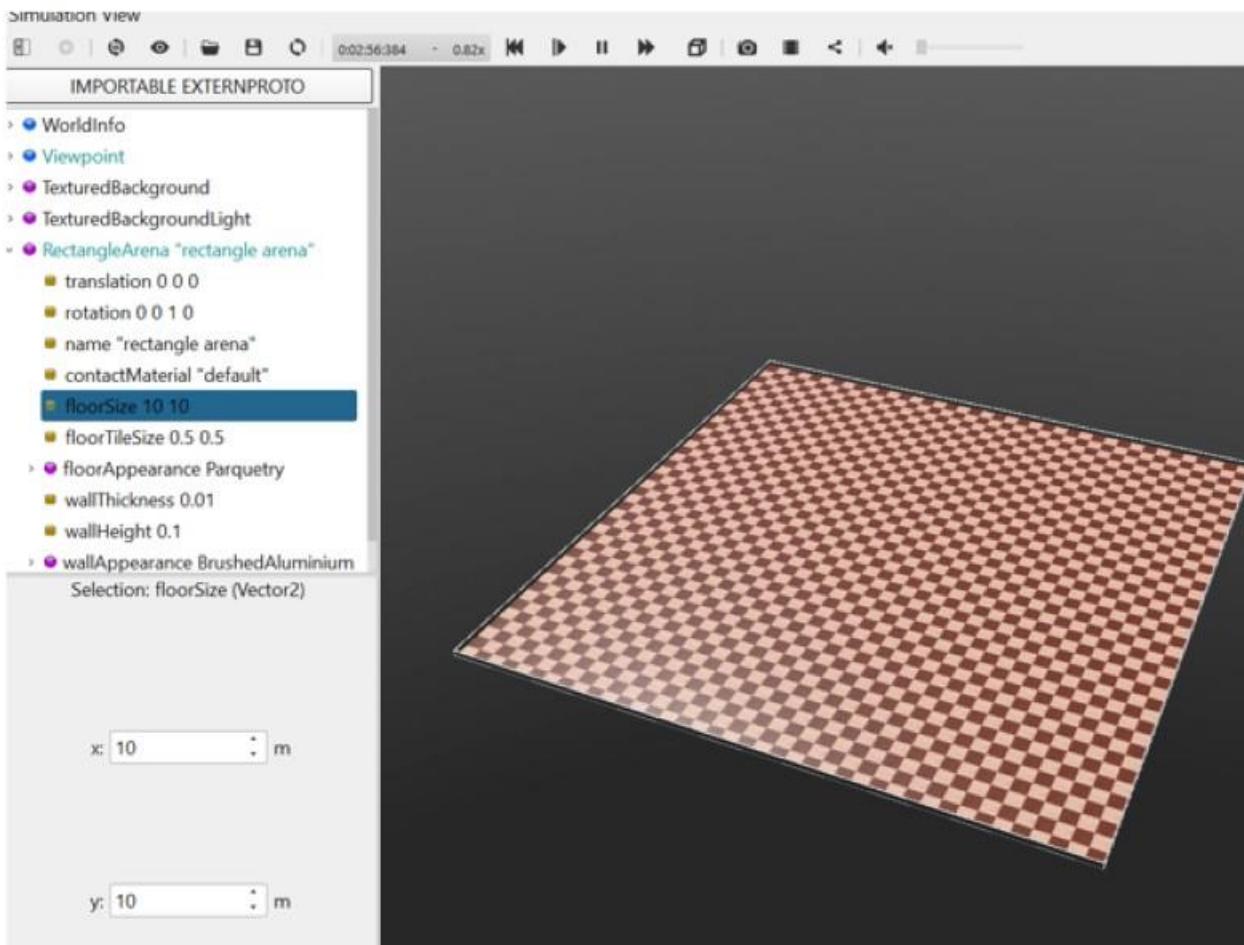
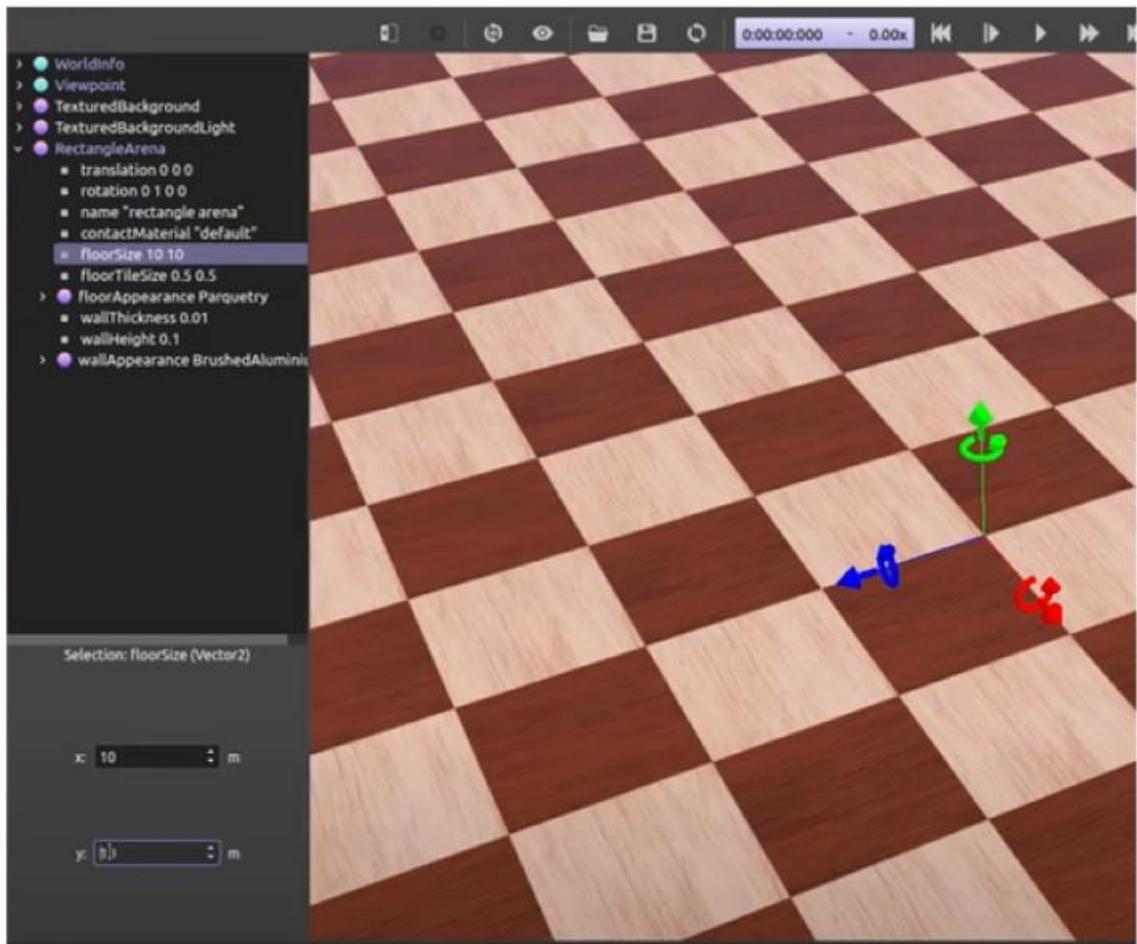
you can click on it and you can move it around to look at different views.



Day

01

Change the Floor Size



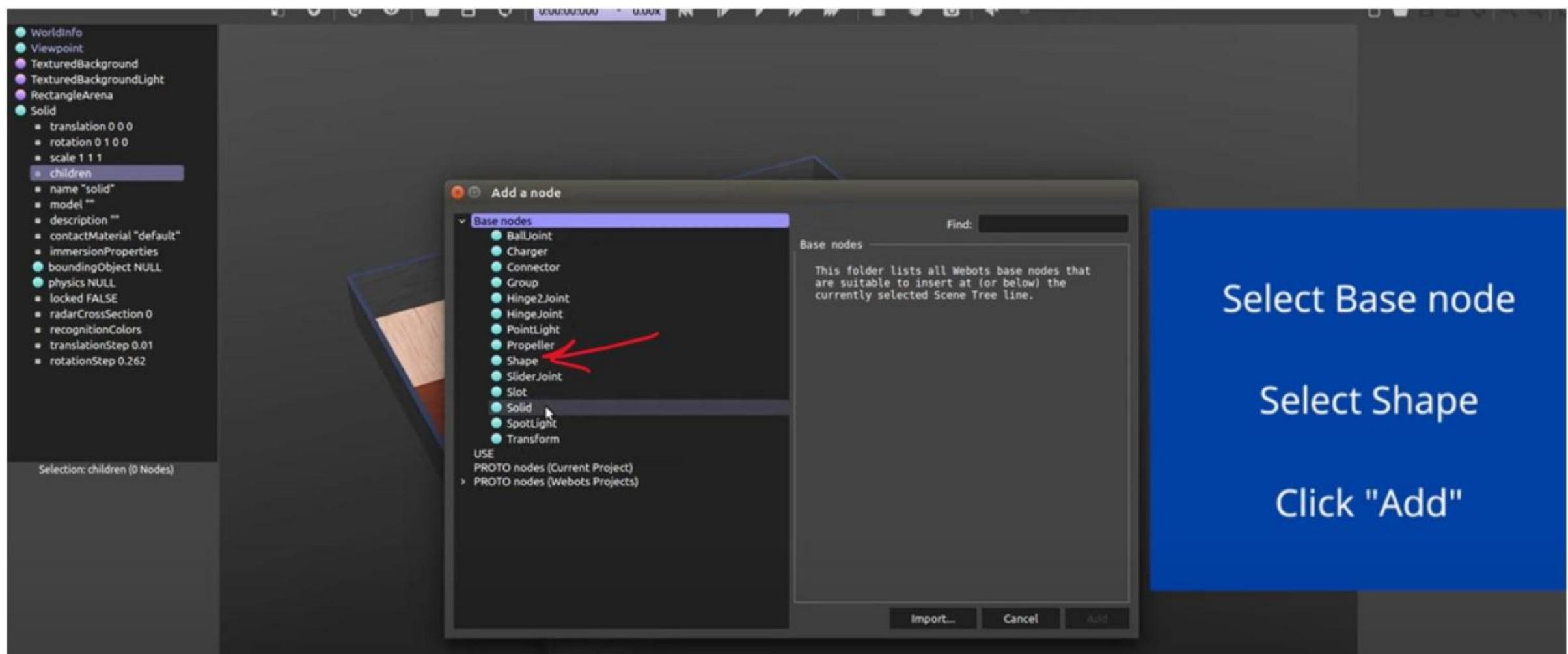
Day

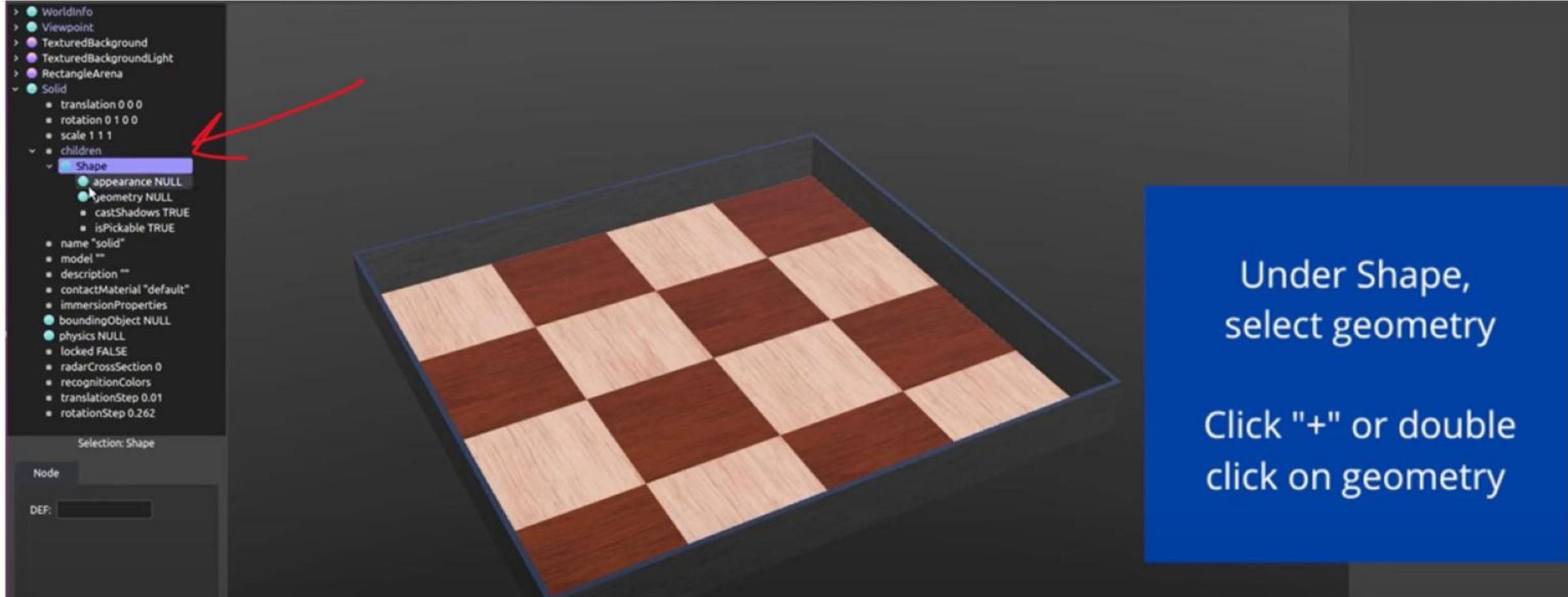
01

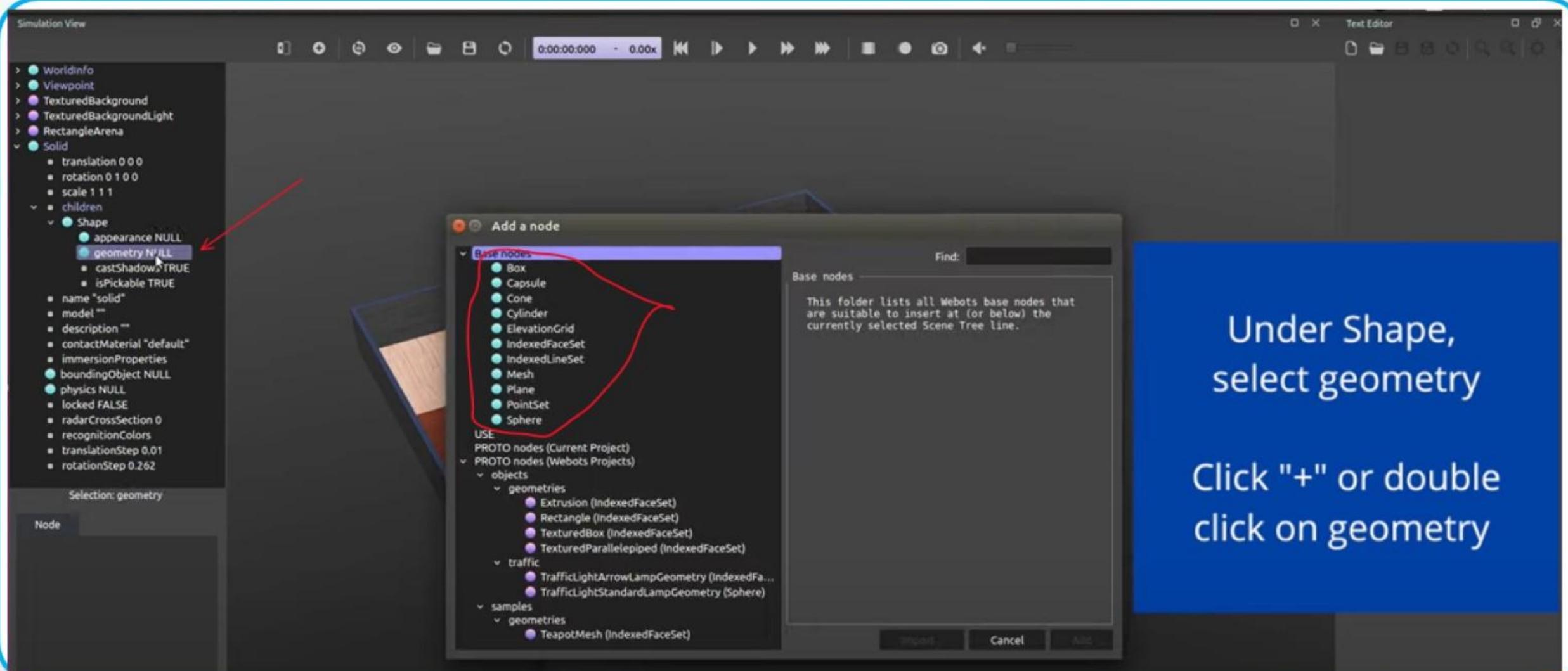
ADD Object

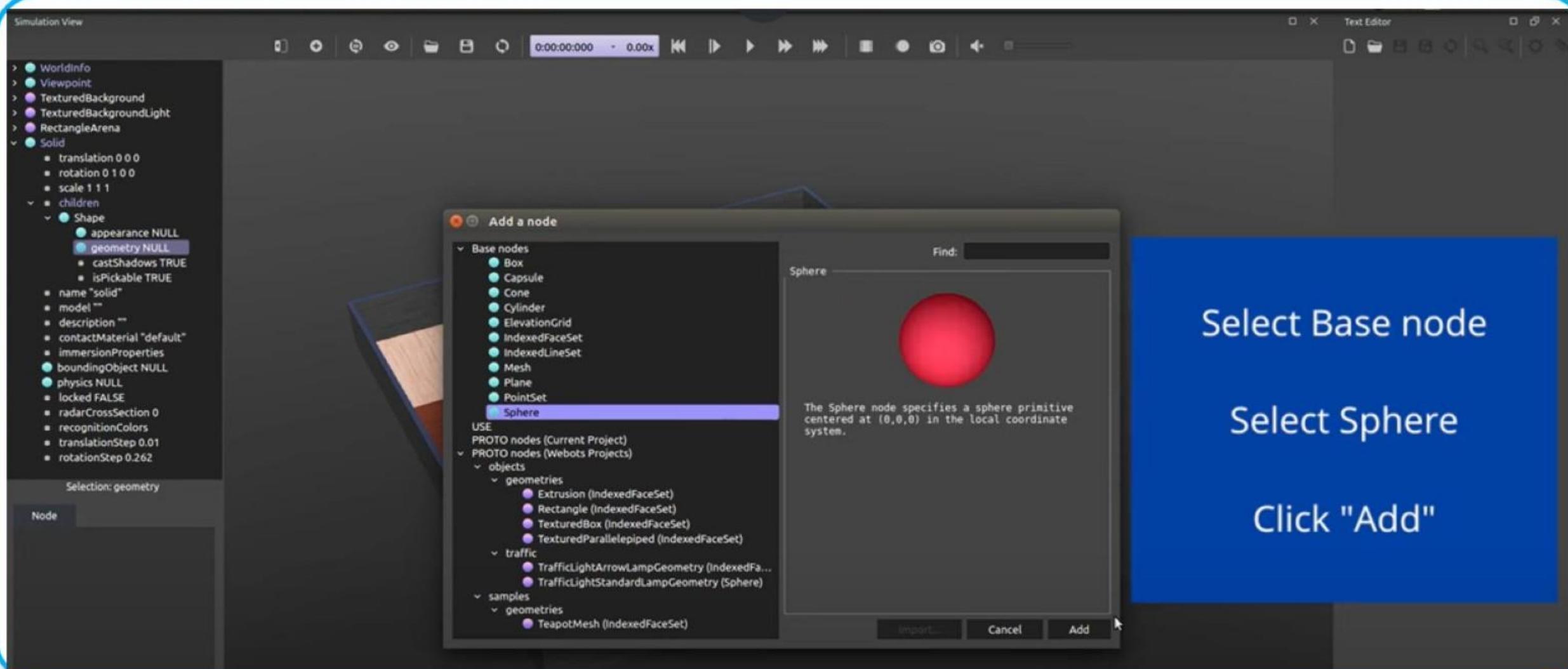


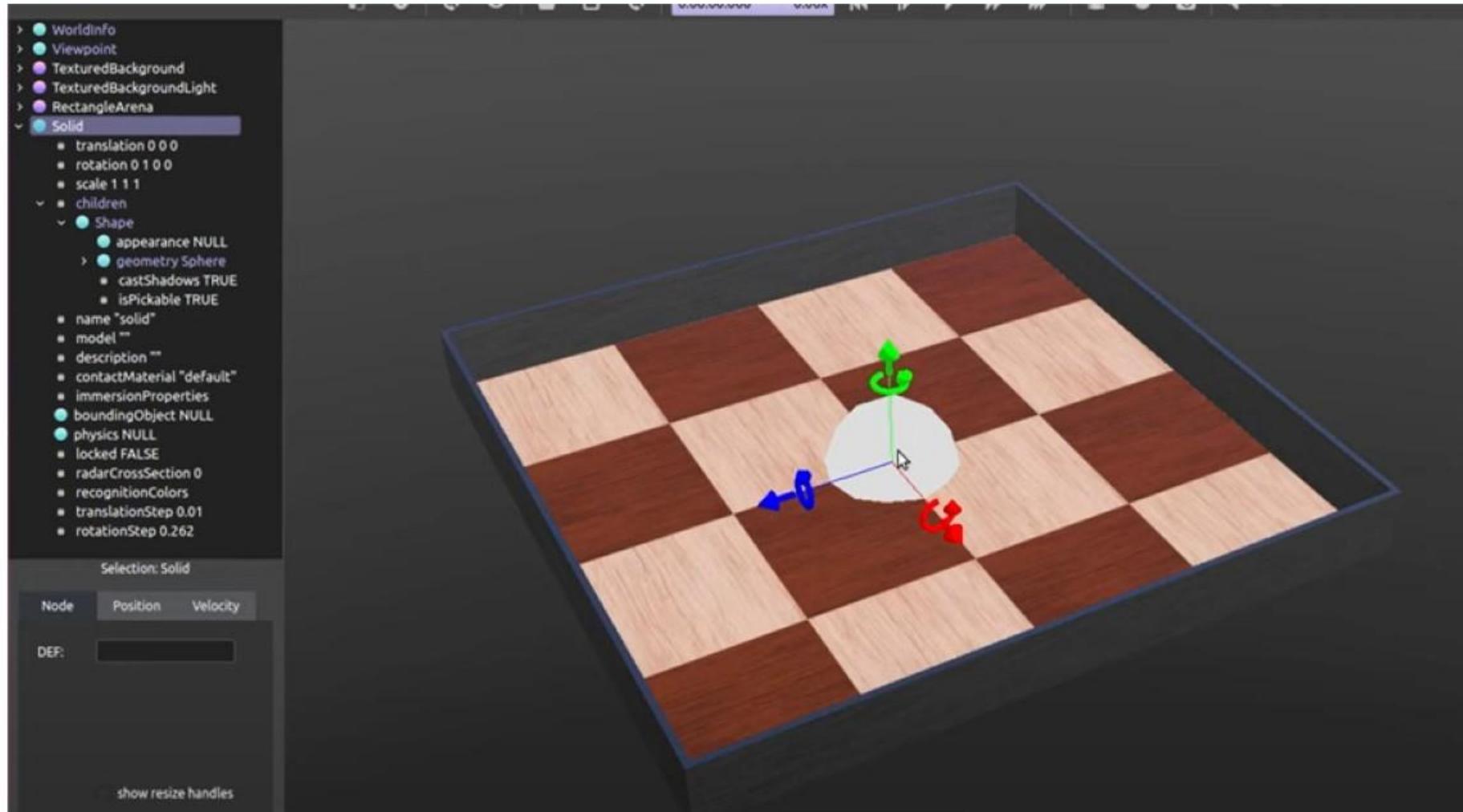
ADD NewObject.mp4









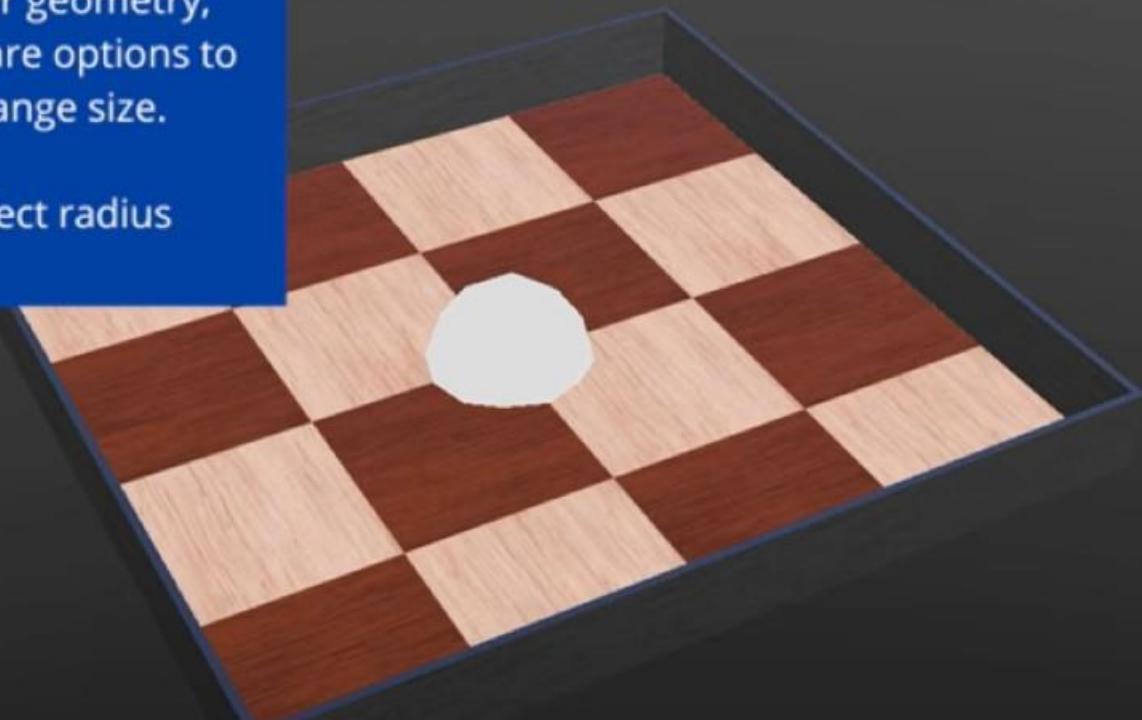


All objects added at (0,0,0).

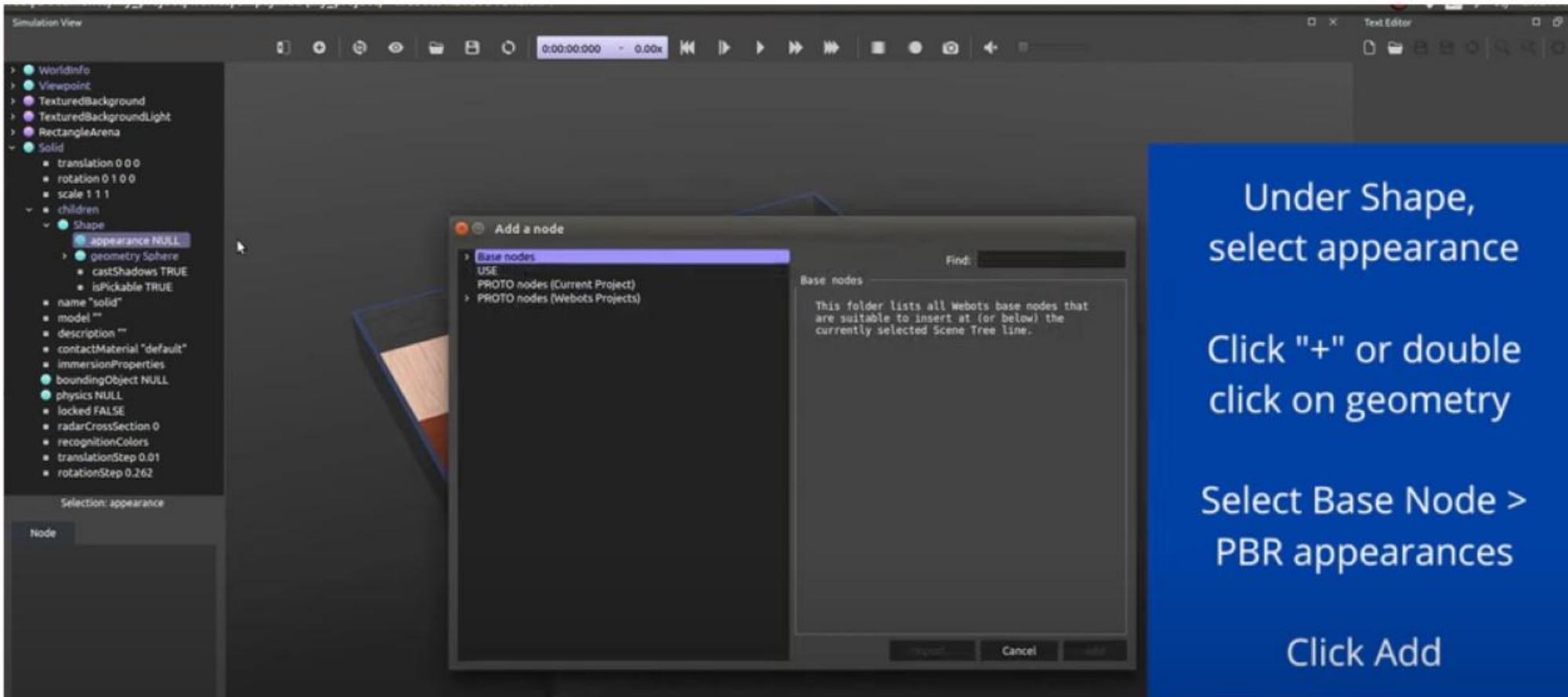
```
> ● WorldInfo
> ● Viewpoint
> ● TexturedBackground
> ● TexturedBackgroundLight
> ● RectangleArena
> ● Solid
  * translation 0 0 0
  * rotation 0 1 0
  * scale 1 1 1
  * children
    * ● Shape
      * ● appearance NULL
      * ● geometry Sphere
        * radius 0.1
          * subdivision 1
          * ico TRUE
          * castShadows TRUE
          * isPickable TRUE
        * name "solid"
        * model ""
        * description ""
        * contactMaterial "default"
        * immersionProperties
        * boundingObject NULL
      * physics NULL
      * locked FALSE
      * radarCrossSection 0
      * recognitionColors
    Selection: radius (Float)
```

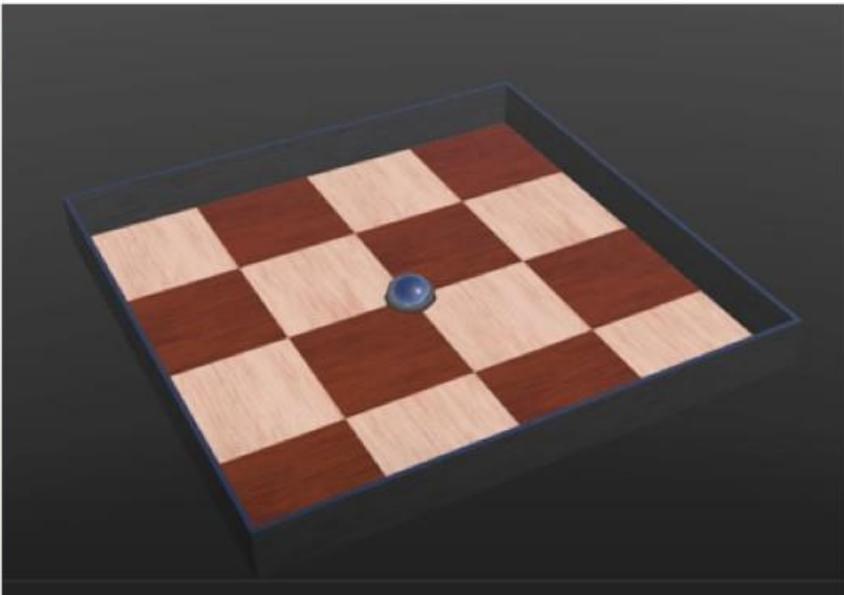
Under geometry,
there are options to
change size.

Select radius



the dimensions so we could go
from 0.1 to
0.05 and you can see the
sphere has reduced





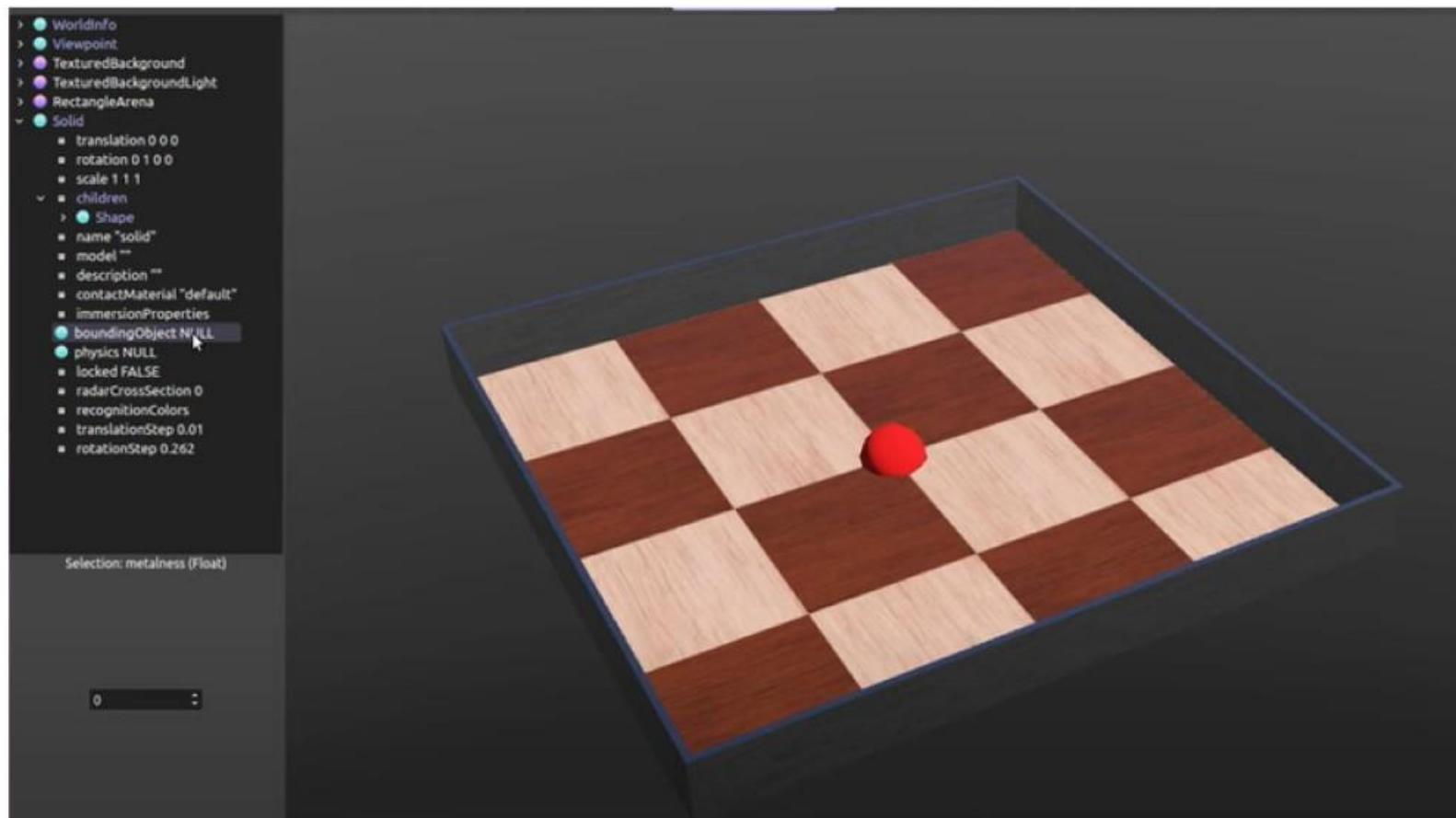
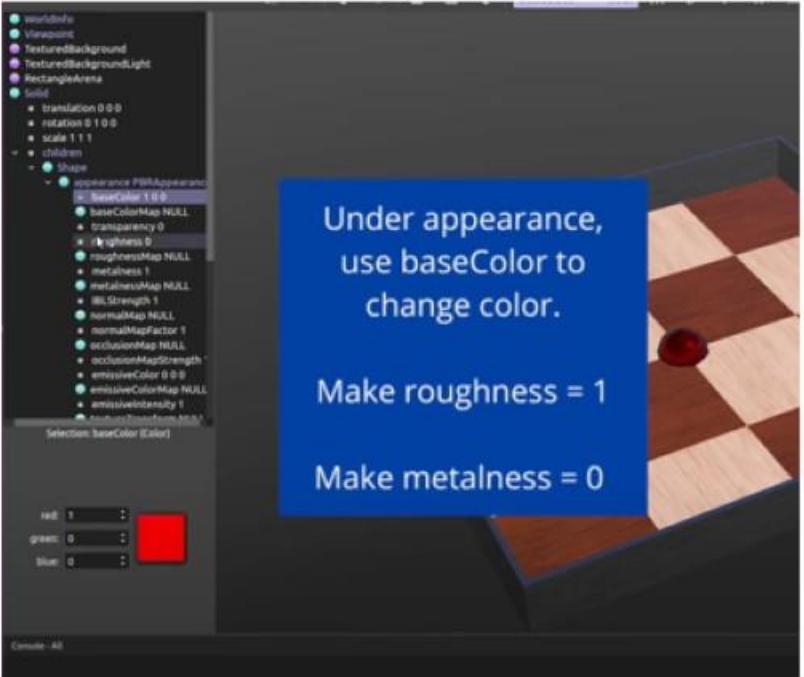
You have different options:

A screenshot of a 3D modeling software interface. On the left, there is a 3D view of the chessboard with a blue sphere. On the right, there is a material editor window. The material editor shows a tree structure with "Shape" selected, and "appearance PBRAAppearance" is expanded. Under "appearance", "baseColor 1 1 1" is highlighted. Below the tree, there is a color selection panel with "Selection: baseColor (Color)" and sliders for "red: 1", "green: 1", and "blue: 1". A blue callout box points to the "baseColor" entry in the tree with the text: "Under appearance, use baseColor to change color." Another blue callout box points to the "roughness" entry in the tree with the text: "Make roughness = 1". A third blue callout box points to the "metalness" entry in the tree with the text: "Make metalness = 0".

Under appearance,
use baseColor to
change color.

Make roughness = 1

Make metalness = 0

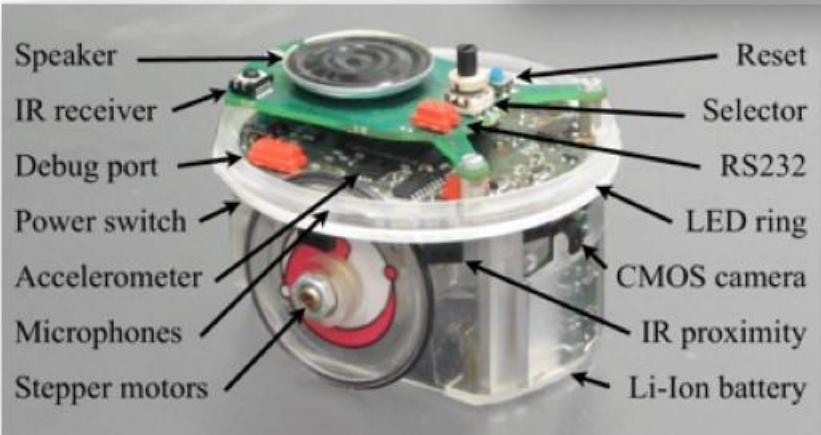


Day

01

NOW using Existing Objects and ROBOTS

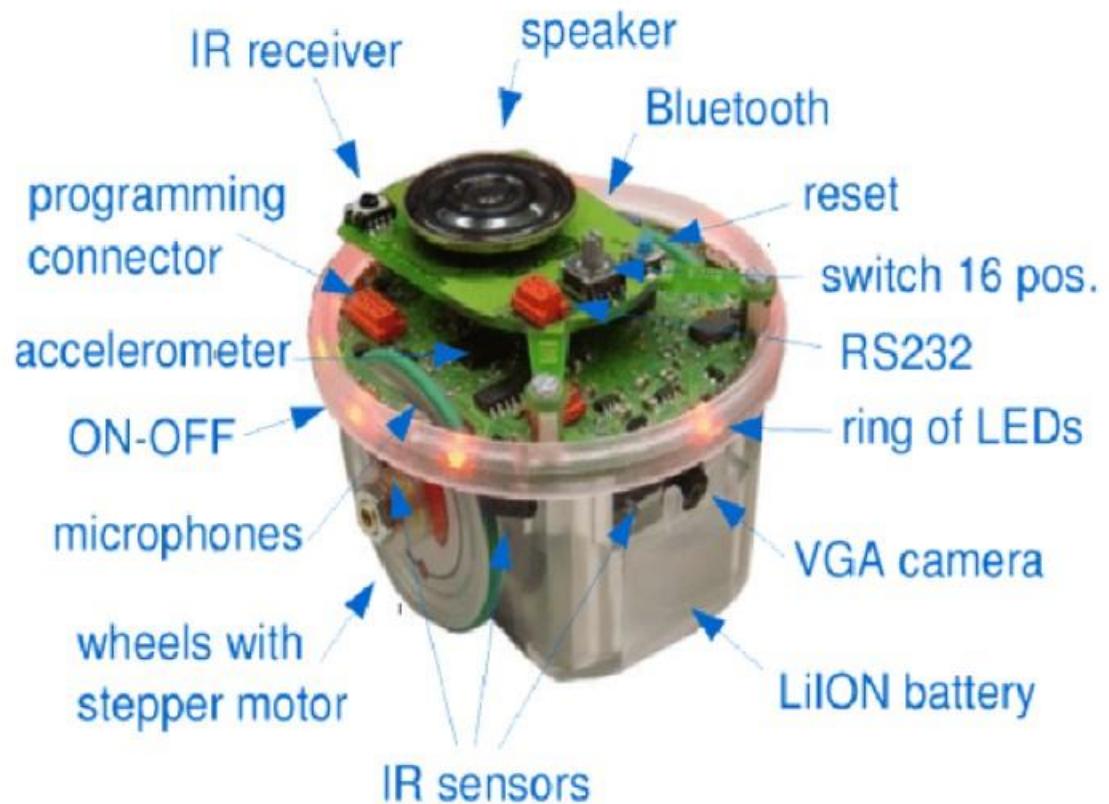
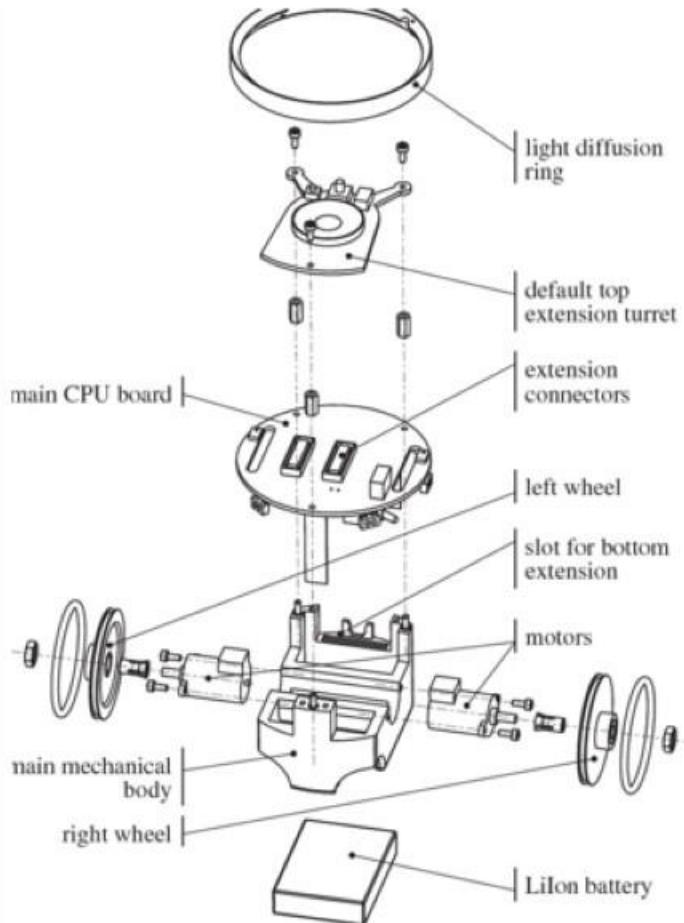
\$850

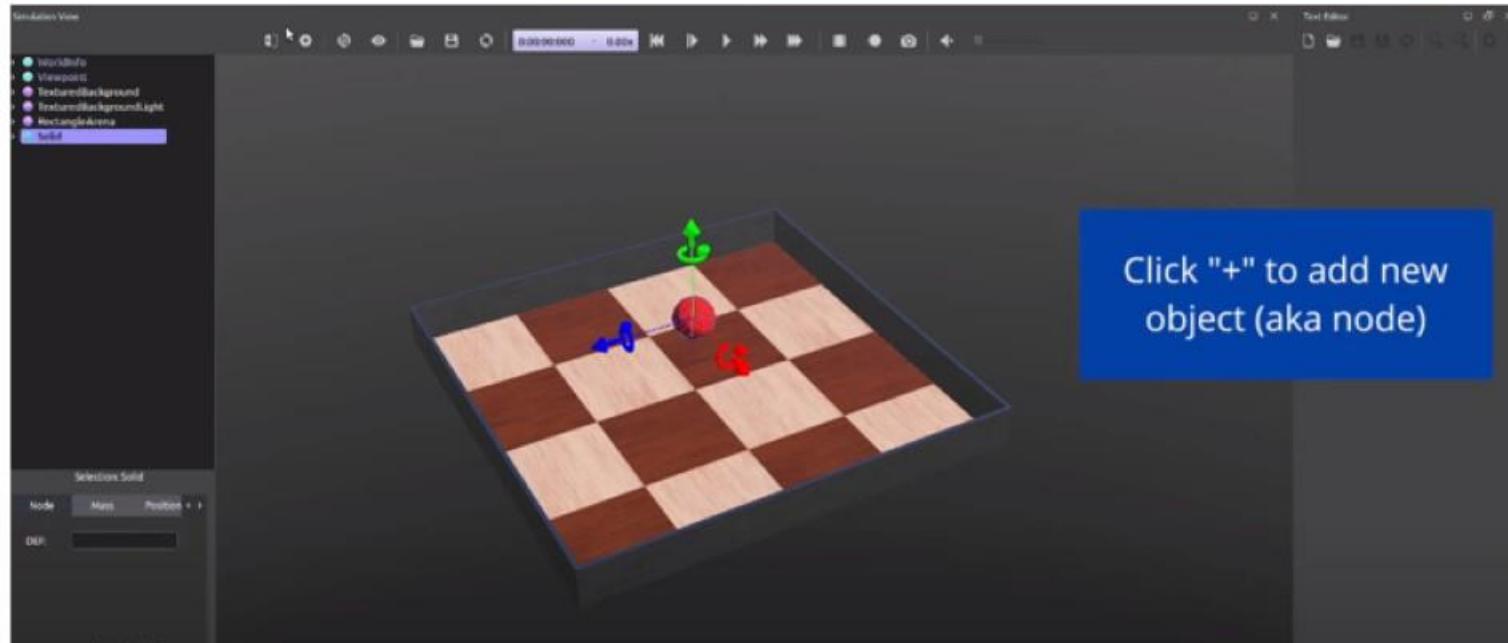


e-puck education robot

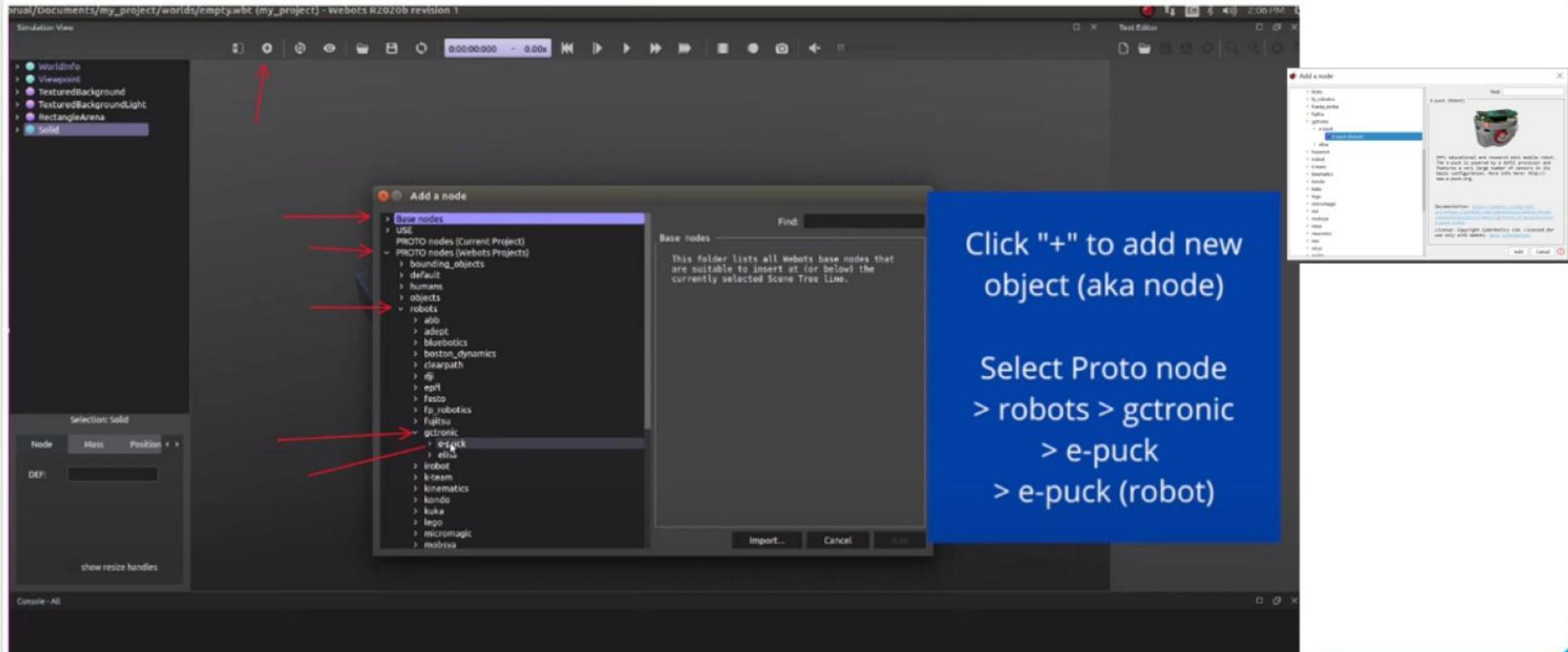
FREE

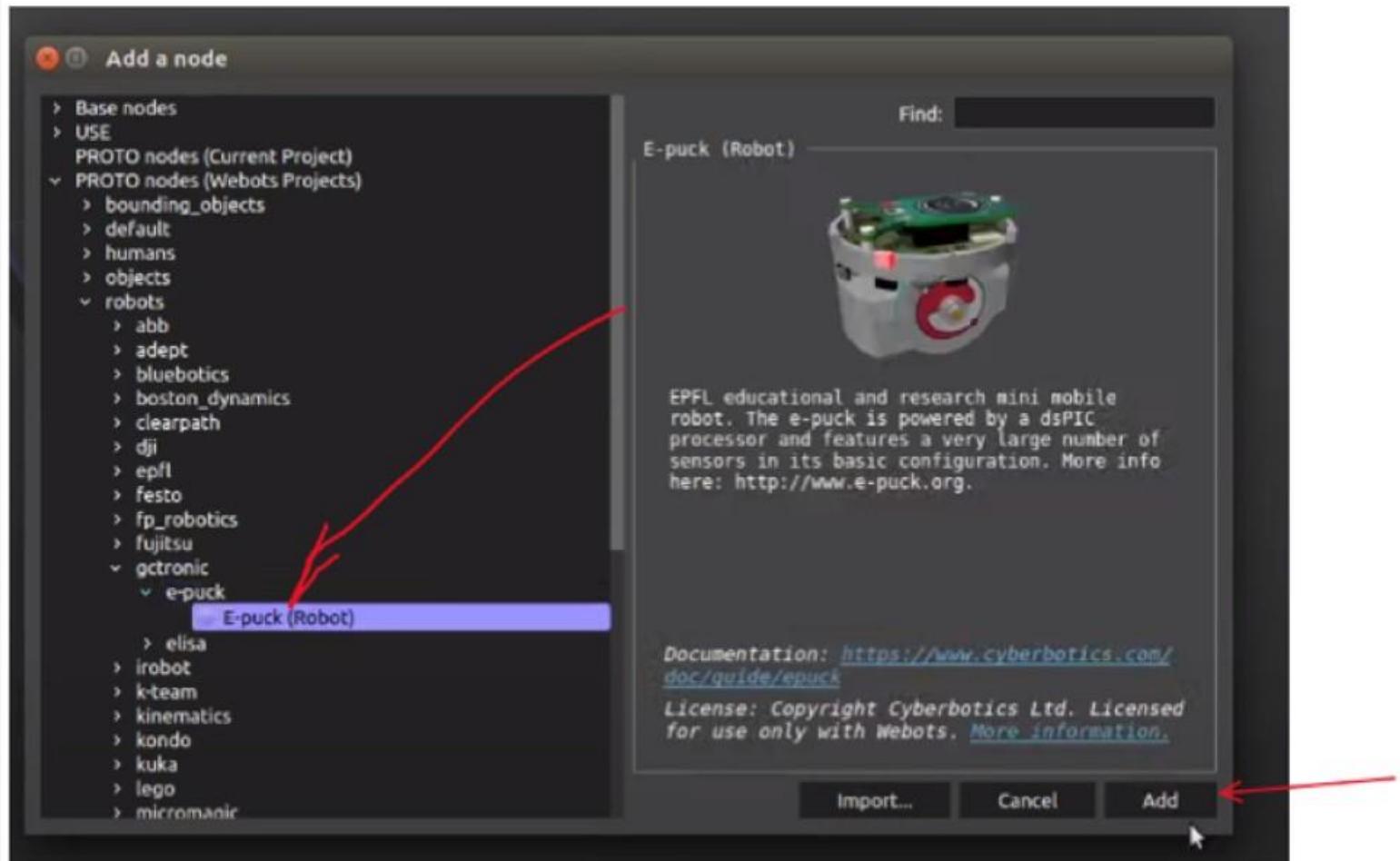






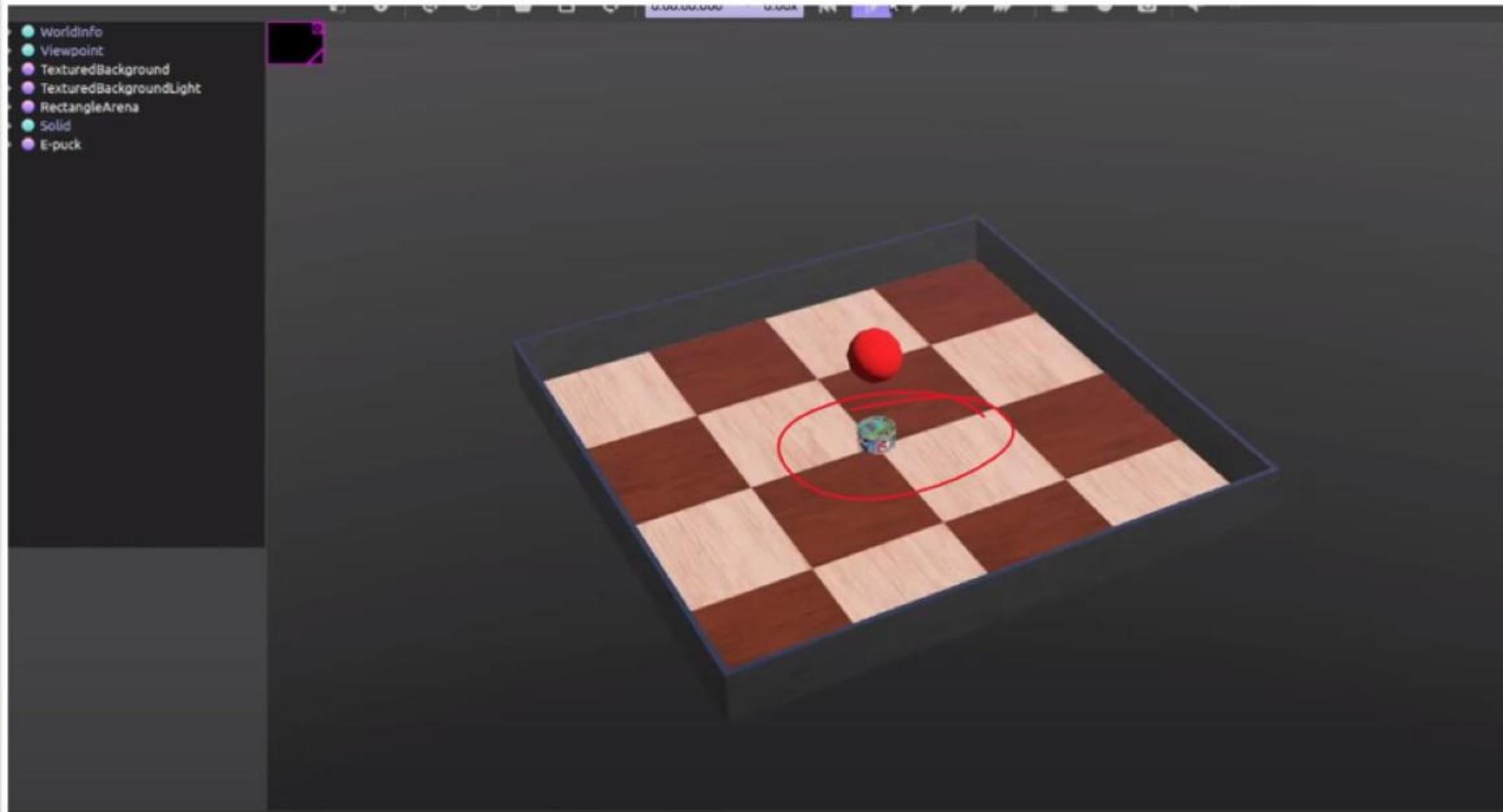
Click on the Add button at the top of the scene tree view. In the dialog box, choose PROTO nodes (Webots Projects) / robots / gctronic / e-puck / E-puck (Robot)





Day

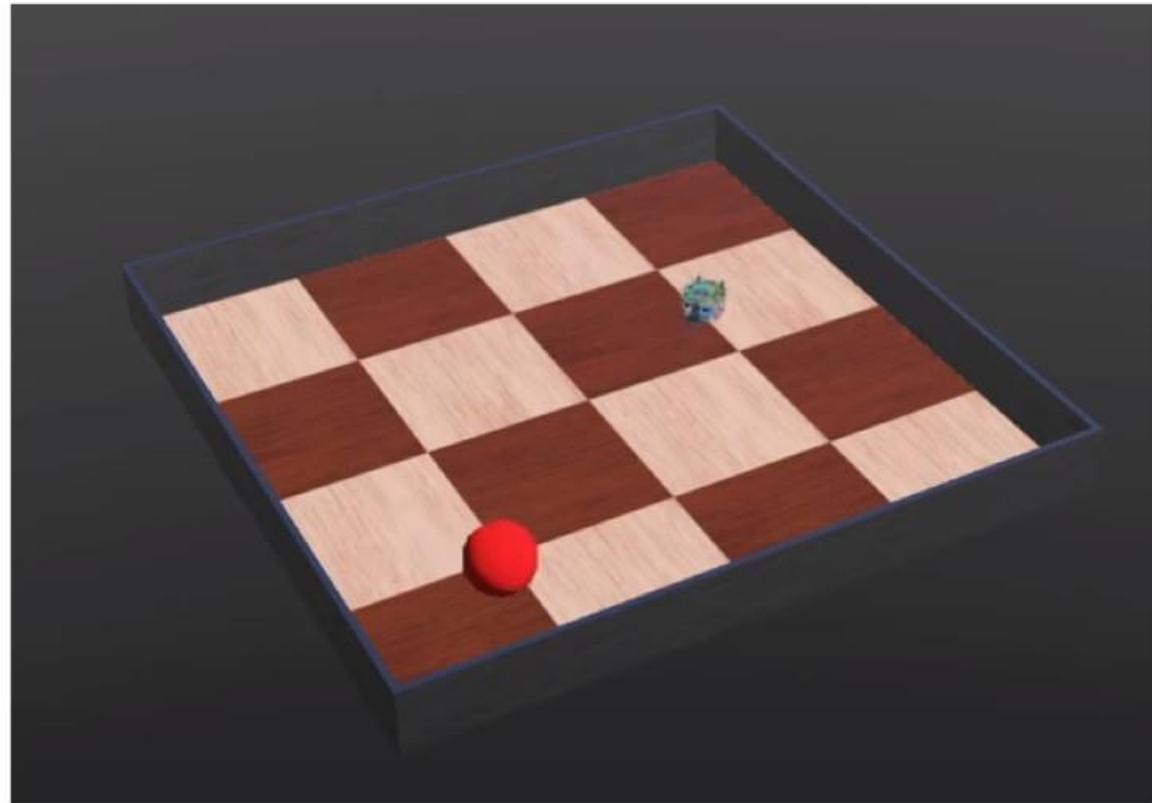
01

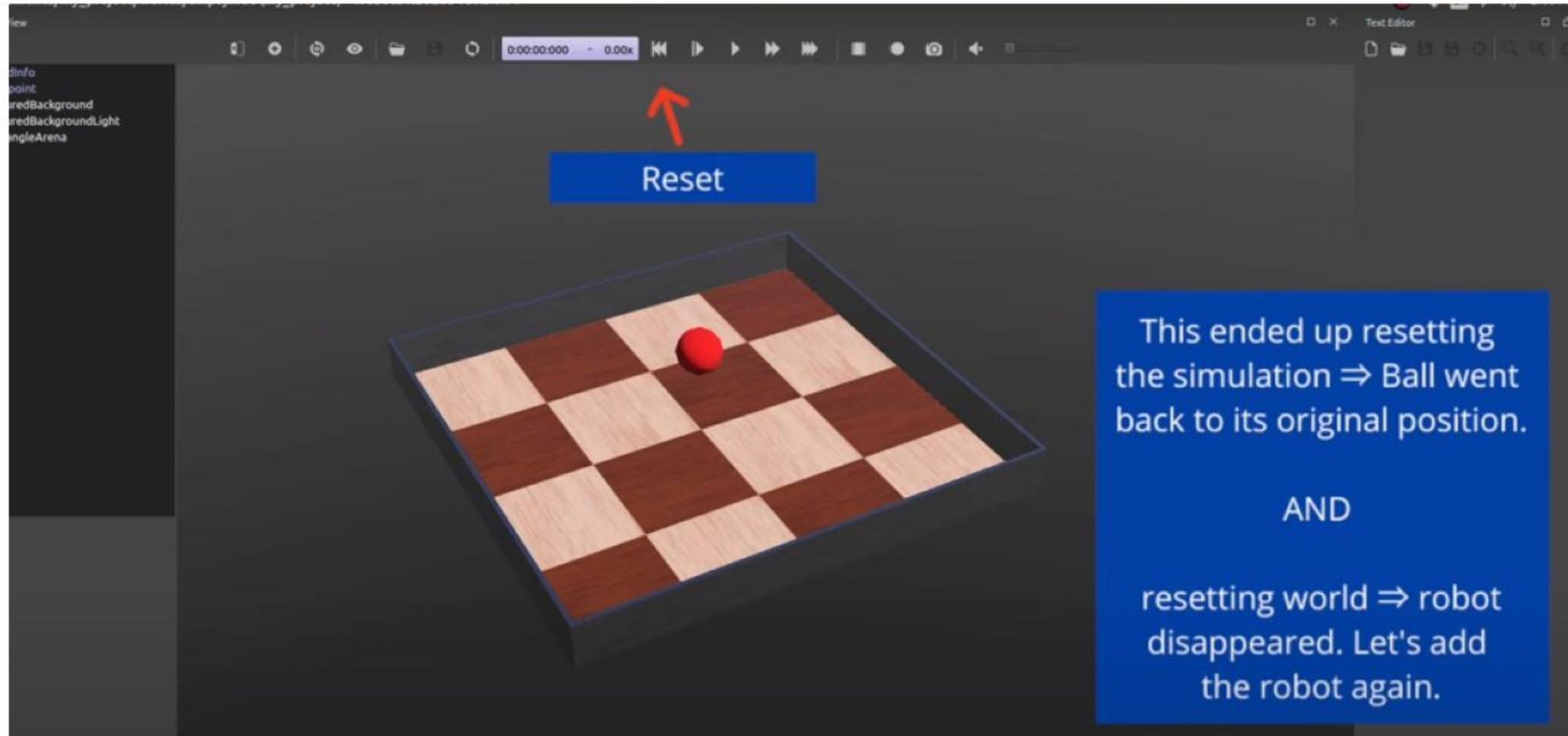


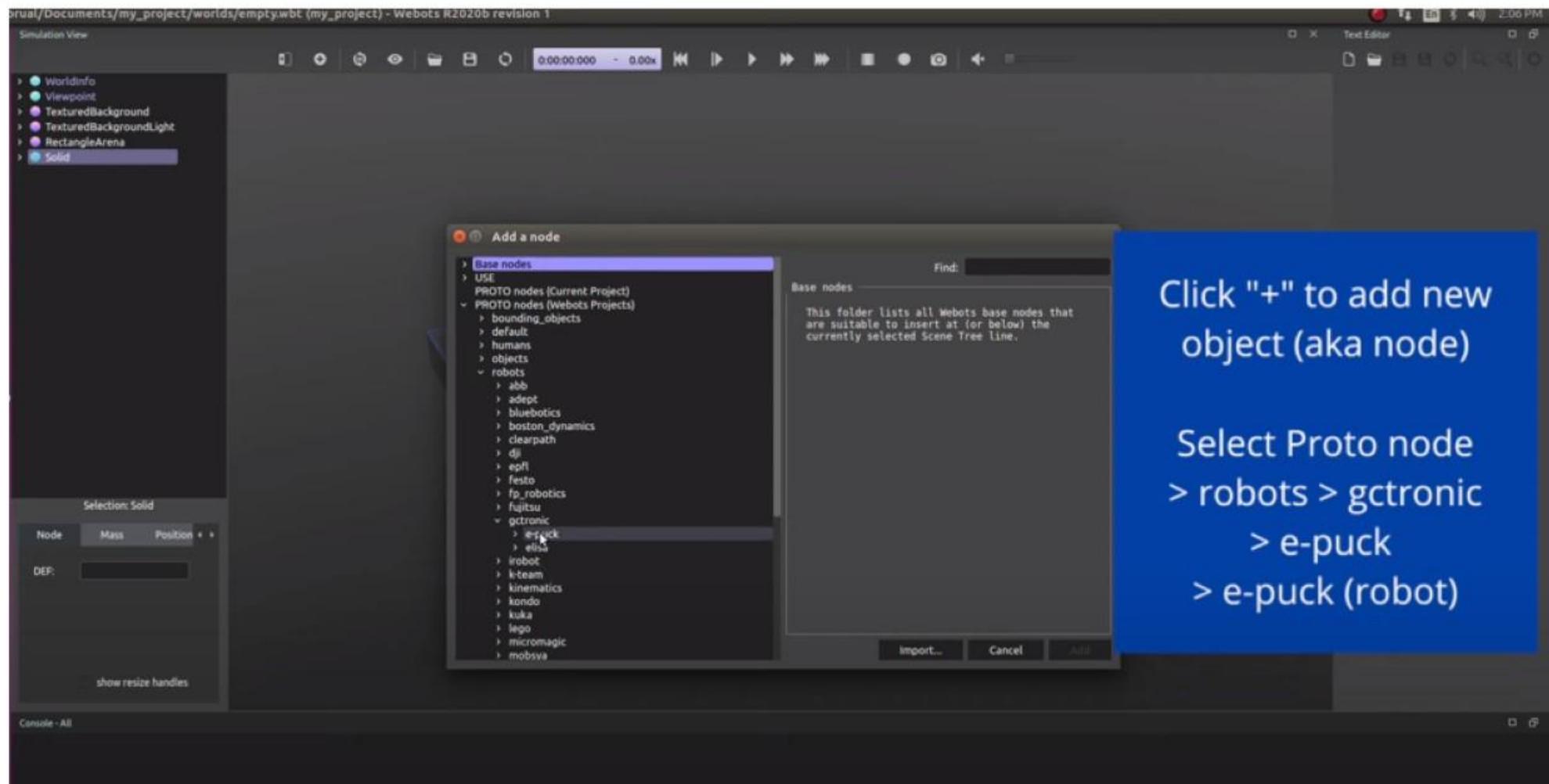
Day

01

When start the simulation (RUN) the Robot continue to drive.

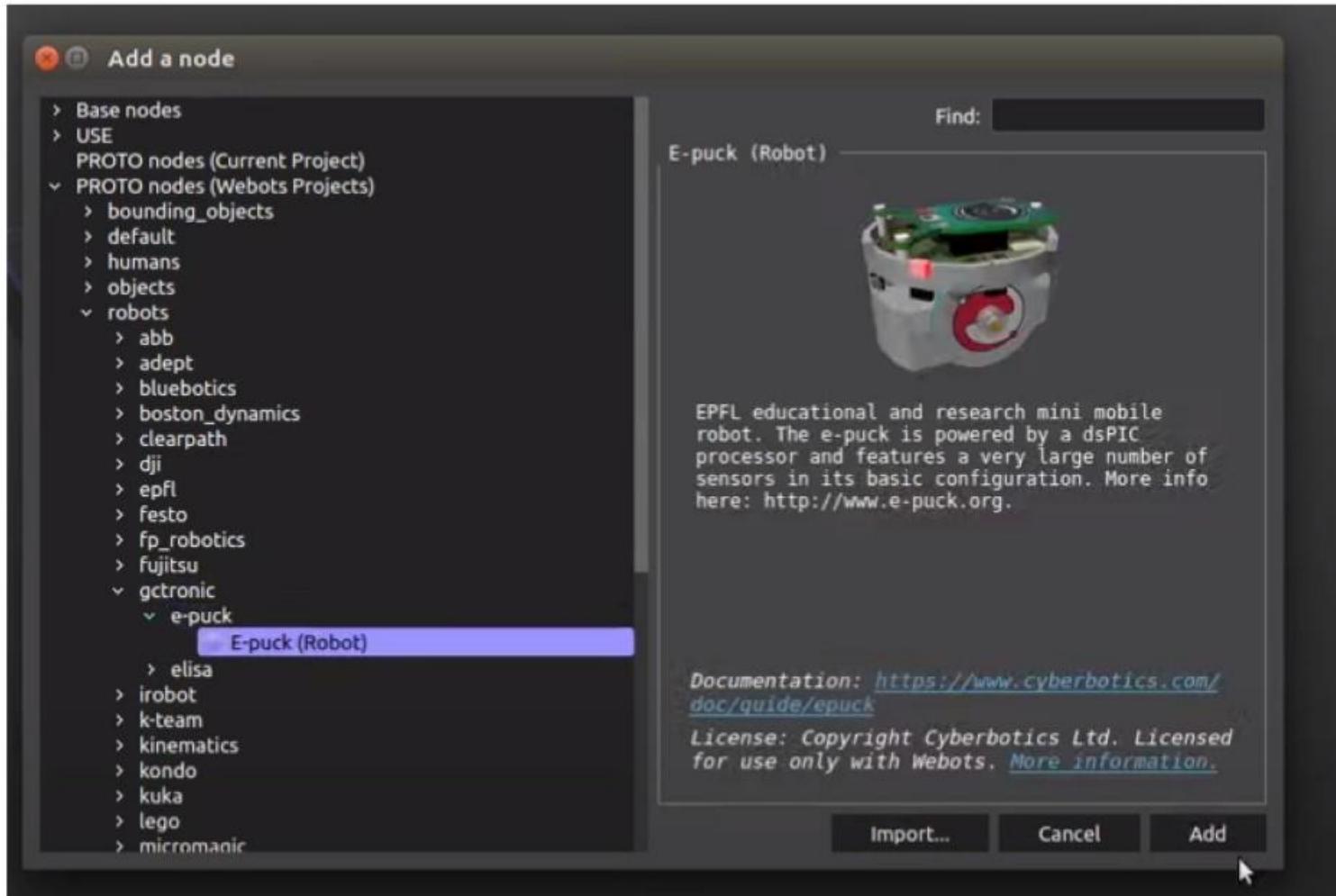






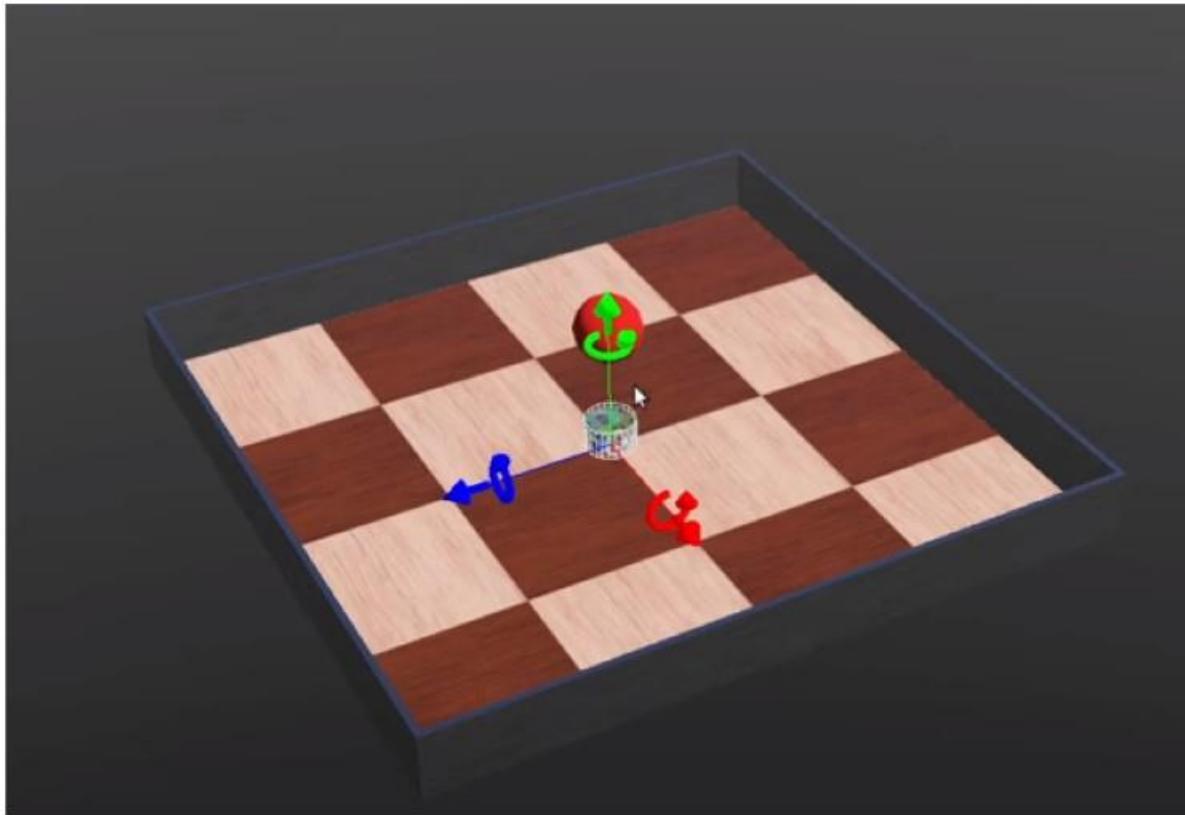
Click "+" to add new object (aka node)

Select Proto node
> robots > gtconic
 > e-puck
 > e-puck (robot)

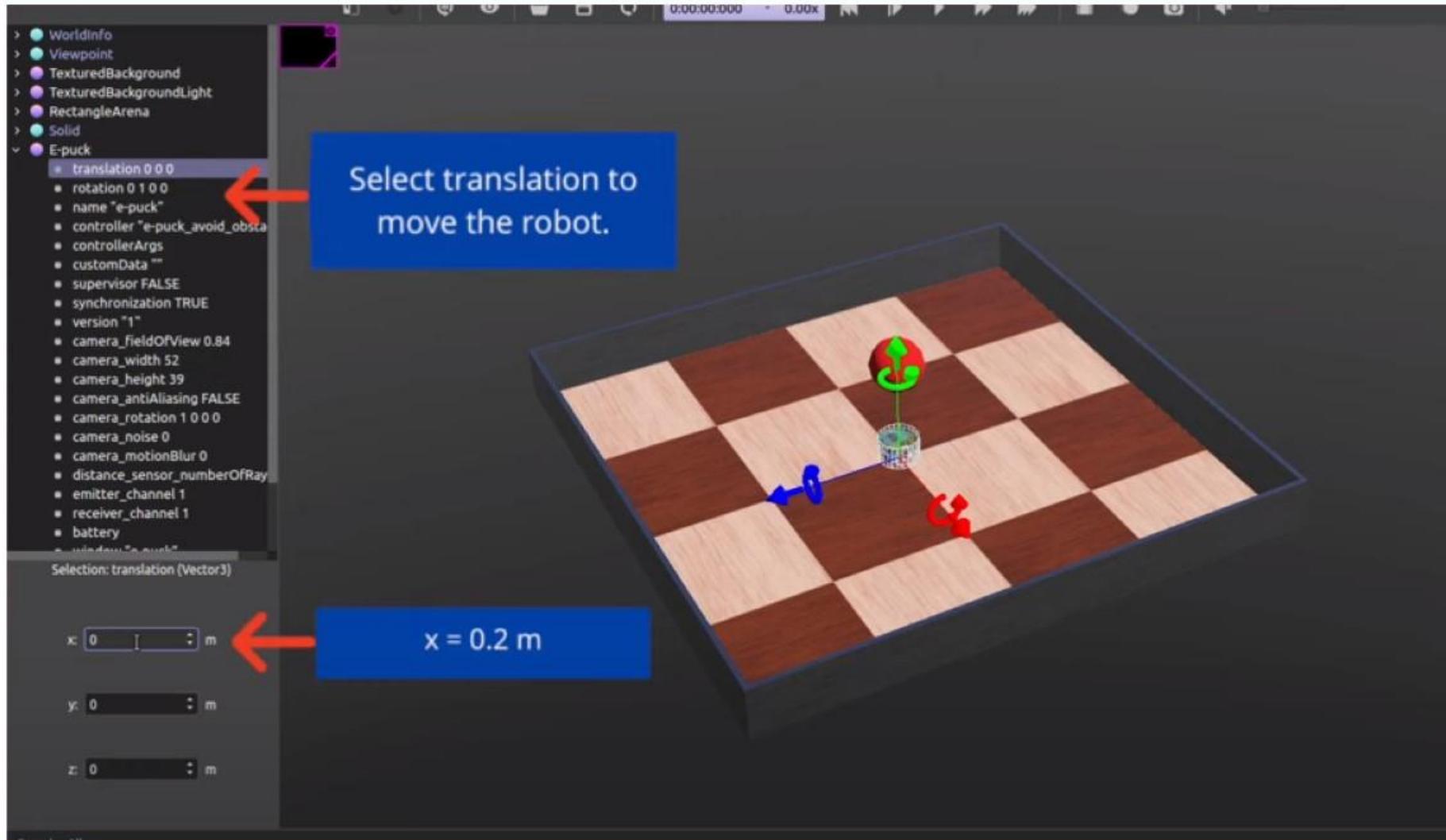


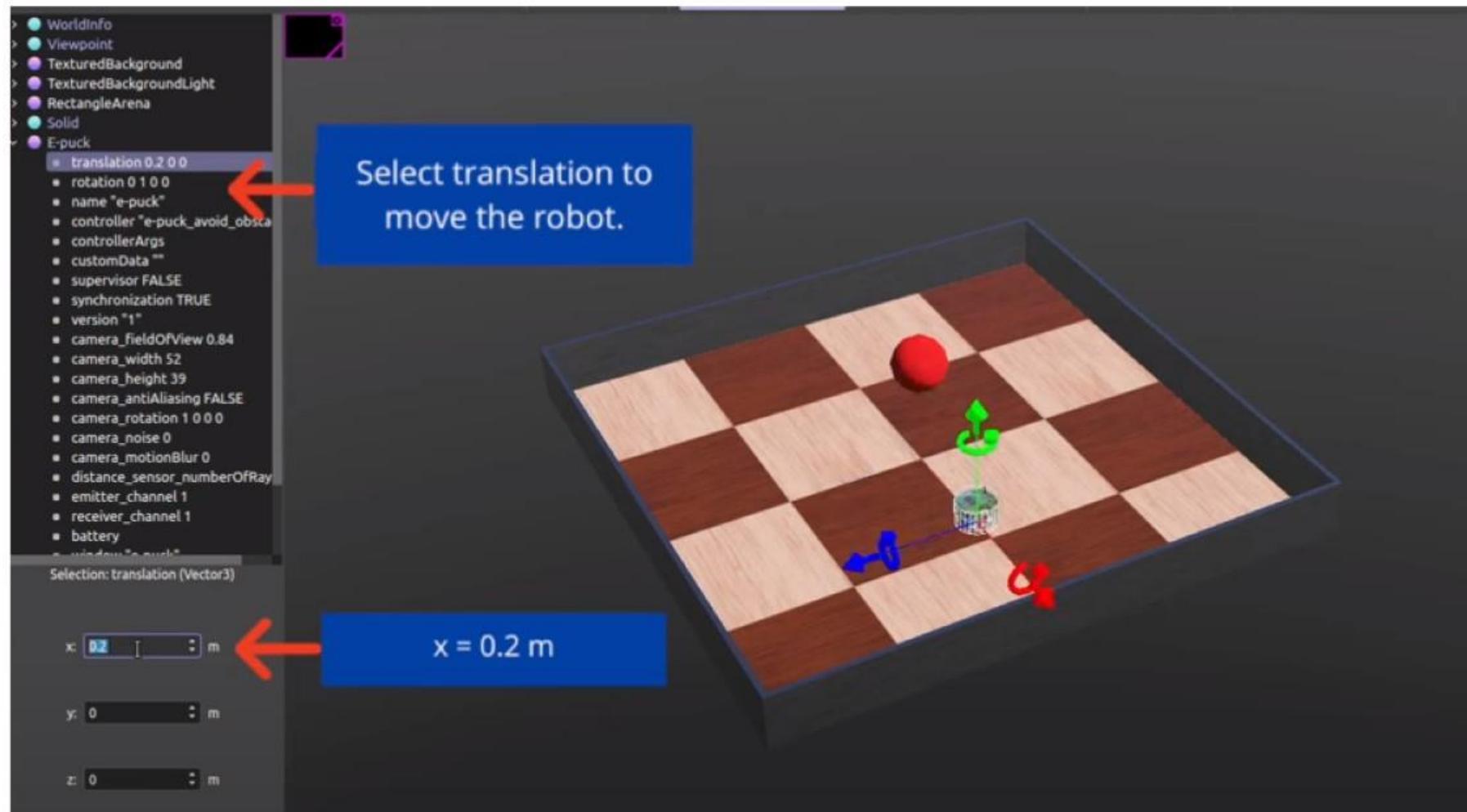
Day

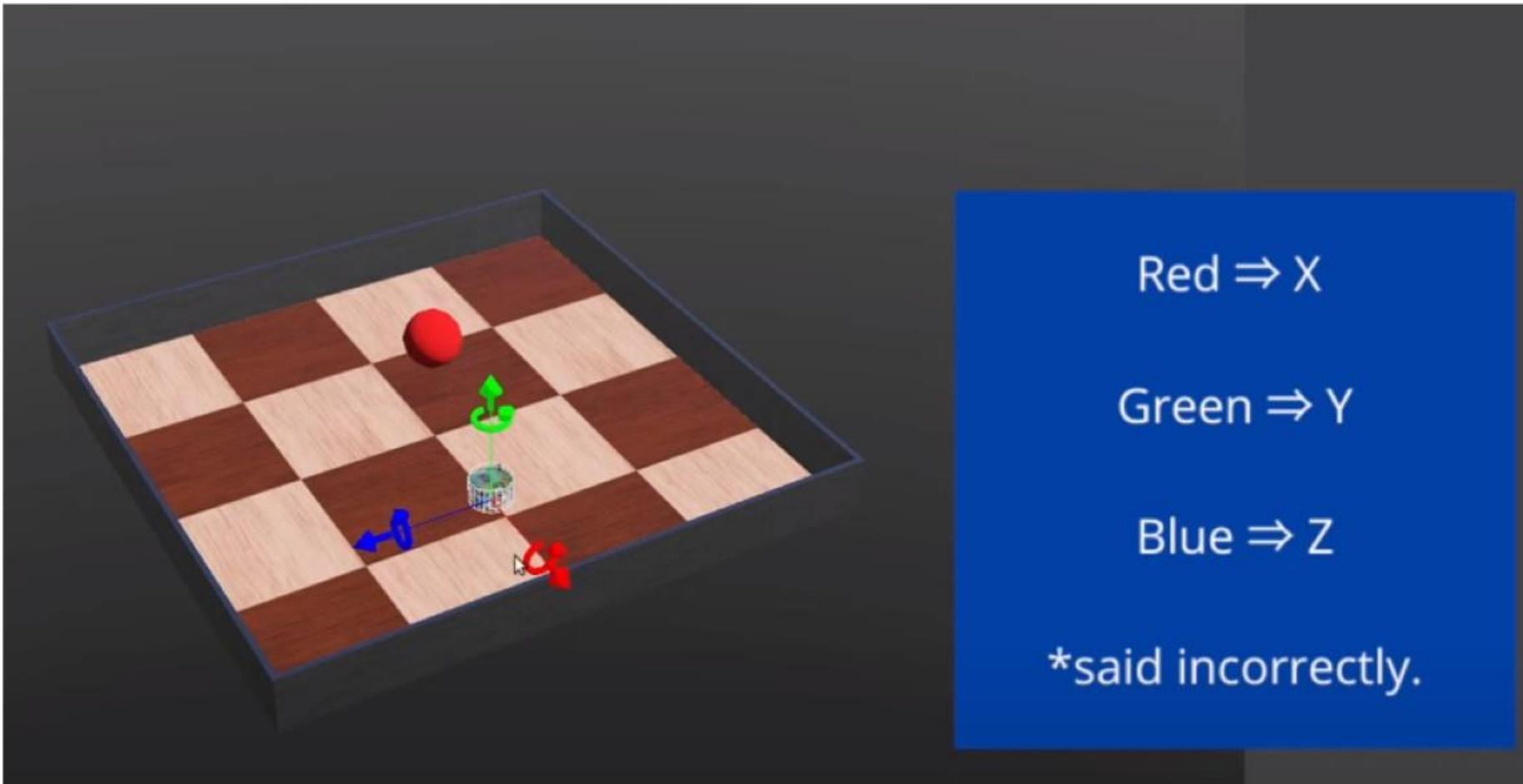
01



Lets shift robot away from the original point (0,0,0):



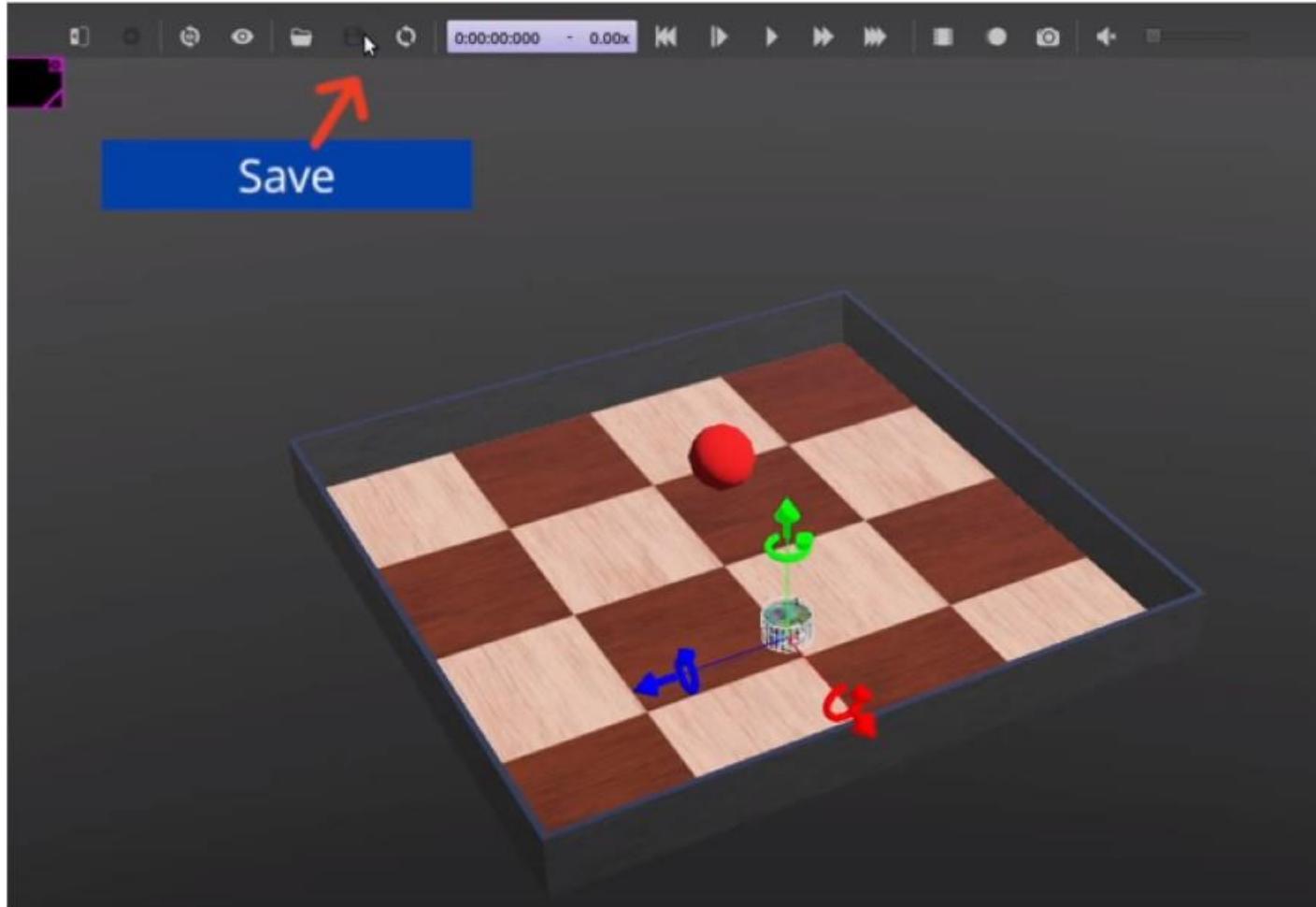




- The X-axis (Red Arrow) points forward from the robot.
- The Y-axis (Green Arrow) points to the left side of the robot.
- The Z-axis (Blue Arrow) points upwards from the robot.

Day

01



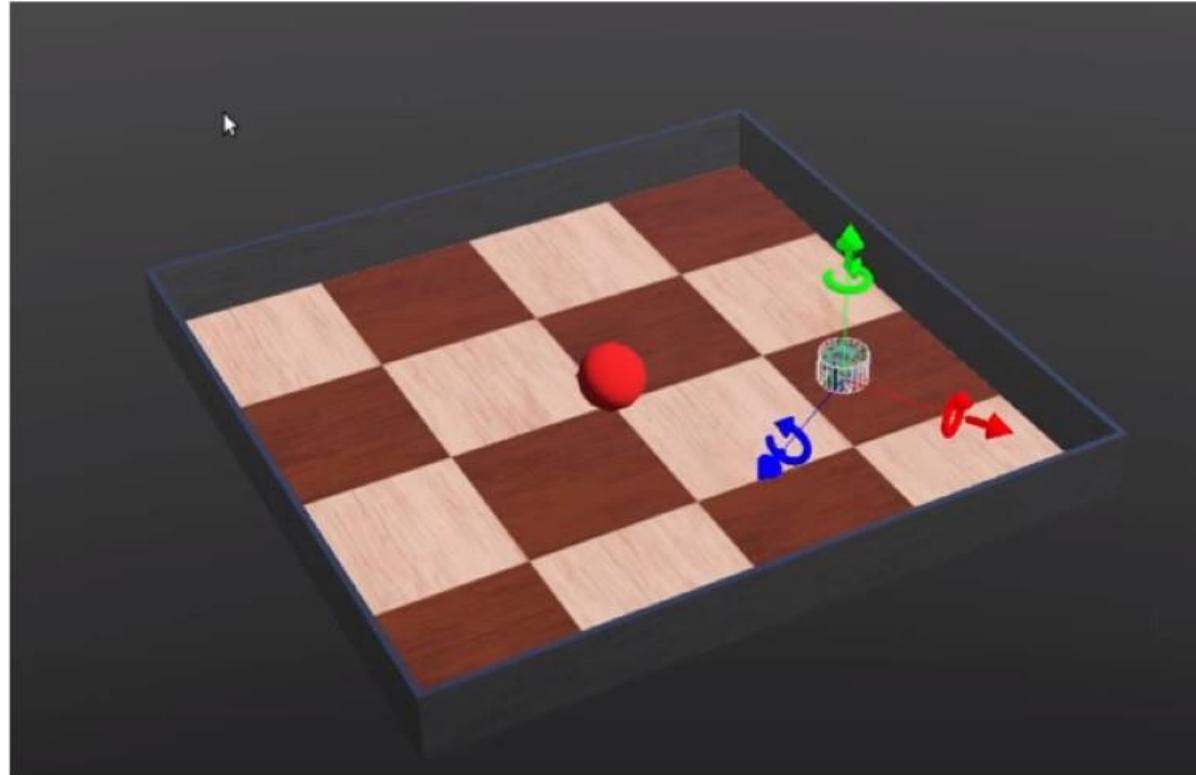
Day

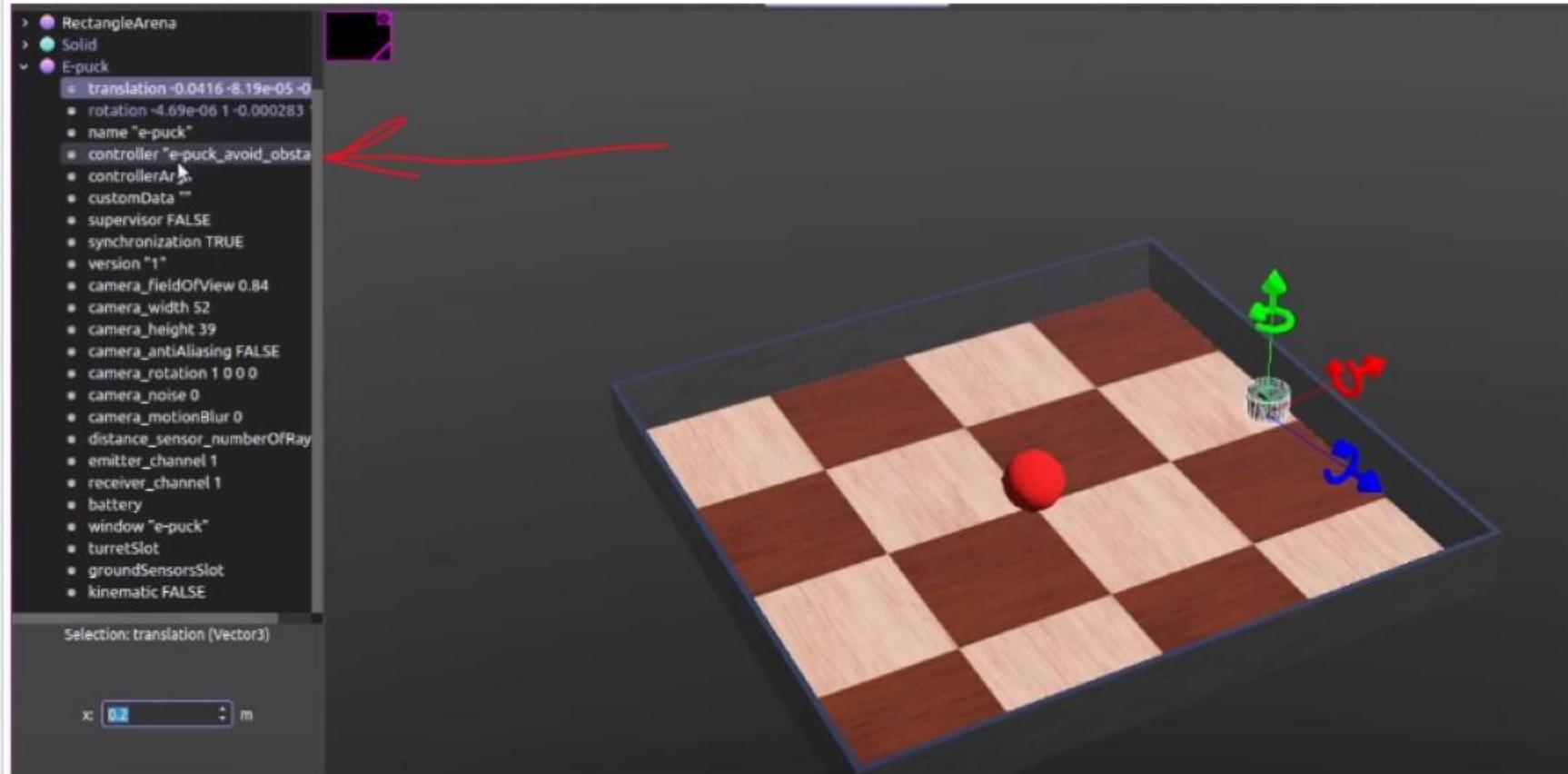
01

RUN the Simulation now: you will see the robot move ?

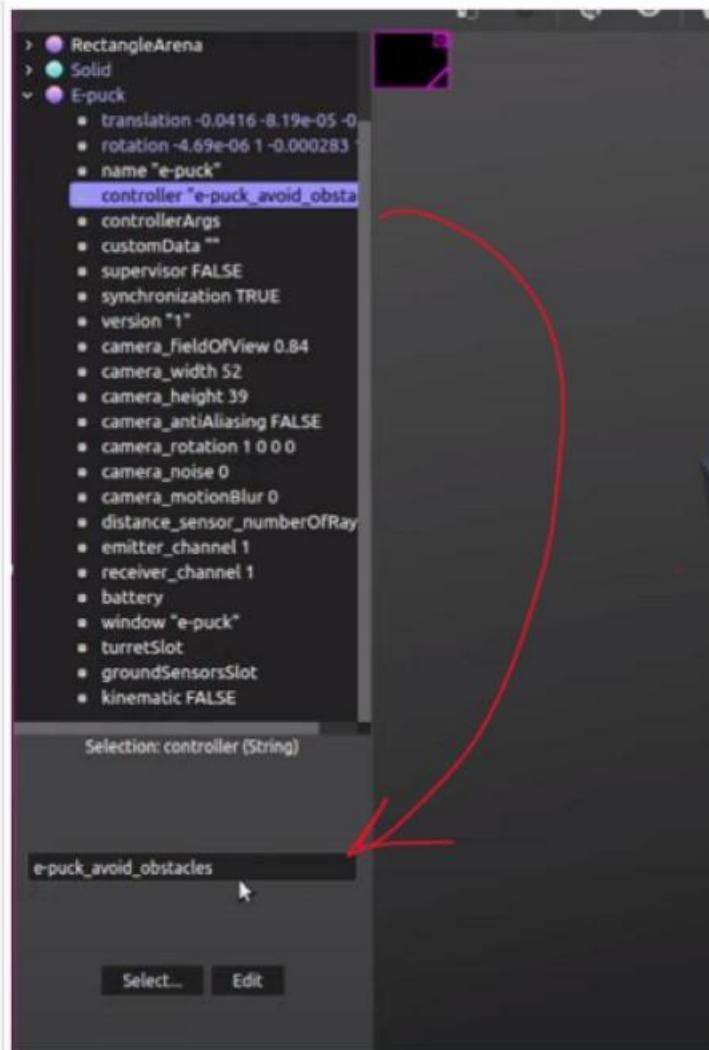
Why?

Because the robot comes with its controller.



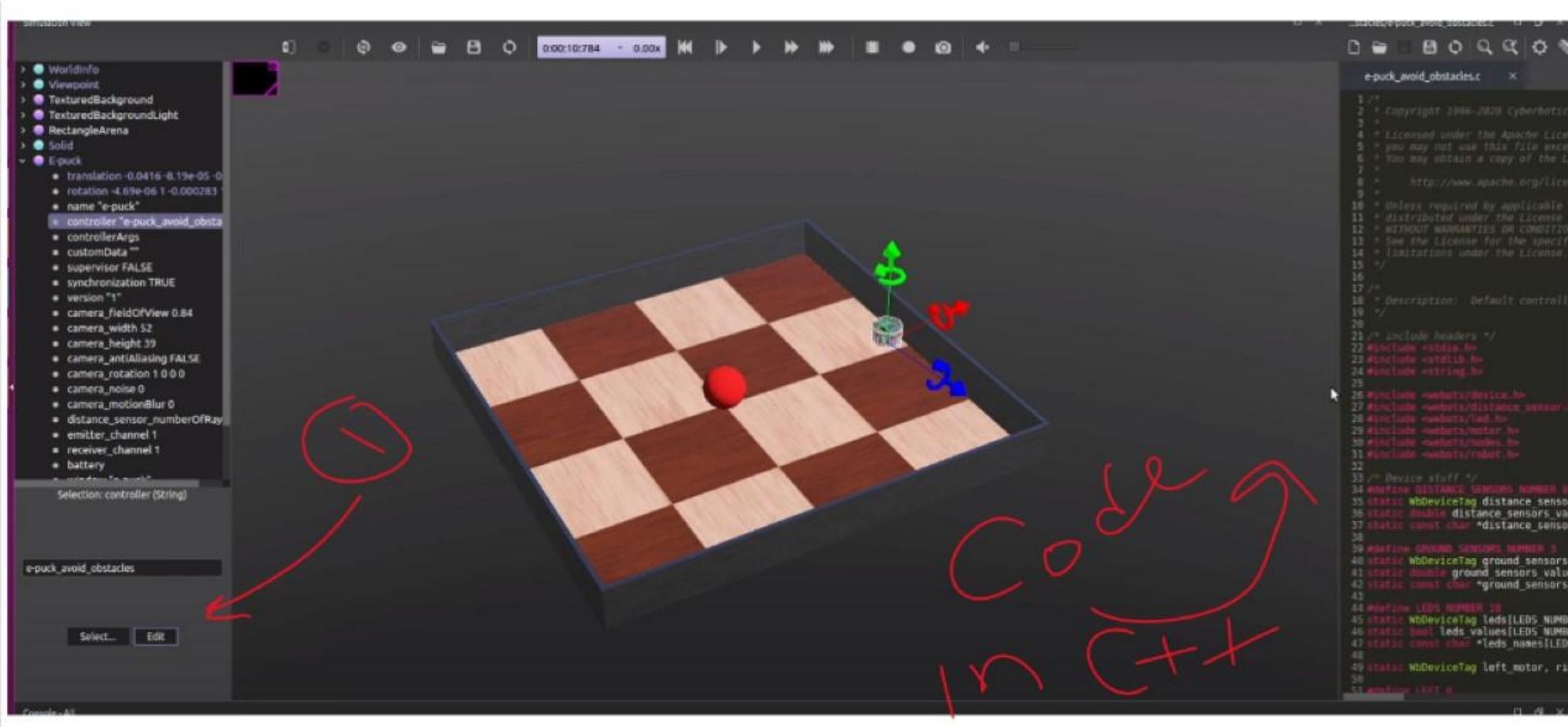


Double click on the Controller:



NOW you can add your own code , Click on the EDIT:

The code is written in C++ but you can write Python code.



Day

01

<https://github.com/keskaf-canada/MidOcean-University-Robo-Soccer-WORKSHOP2025/tree/main>

MIDOCEAN
UNIVERSITY



keskaf-canada / MidOcean-University-Robo-Soccer-WORKSHOP2025



Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings



MidOcean-University-Robo-Soccer-WORKSHOP2025

Public

Unpin

Watch 0

main

1 Branch 0 Tags

Go to file

Add file

Code



keskaf-canada Add files via upload

4816fd5 · 2 hours ago 7 Commits

Example S1

Add files via upload

2 hours ago

Example S2

Add files via upload

2 hours ago

Example S3

Add files via upload

2 hours ago

Example S4

Add files via upload

2 hours ago

ADV_ROBO_SOCCER_2025.jpeg

Add files via upload

2 hours ago

Competition Sample.mp4

Add files via upload

2 hours ago

Introduction

Create Introduction

2 hours ago

www.midocean.ae

We will now program a simple controller that will just make the robot **move forwards**.

A **controller** is a program that defines the behavior of a robot.

Webots controllers can be written in the following programming languages: C, C++, Java, Python, MATLAB, ROS, etc. C, C++ and Java controllers need to be compiled before they can be run as robot controllers.

Python and MATLAB controllers are interpreted languages so they will run without being compiled.

The controller field of a Robot node specifies which controller is currently associated to the robot. Note that the same controller can be used by several robots, but a robot can only use one controller at a time.

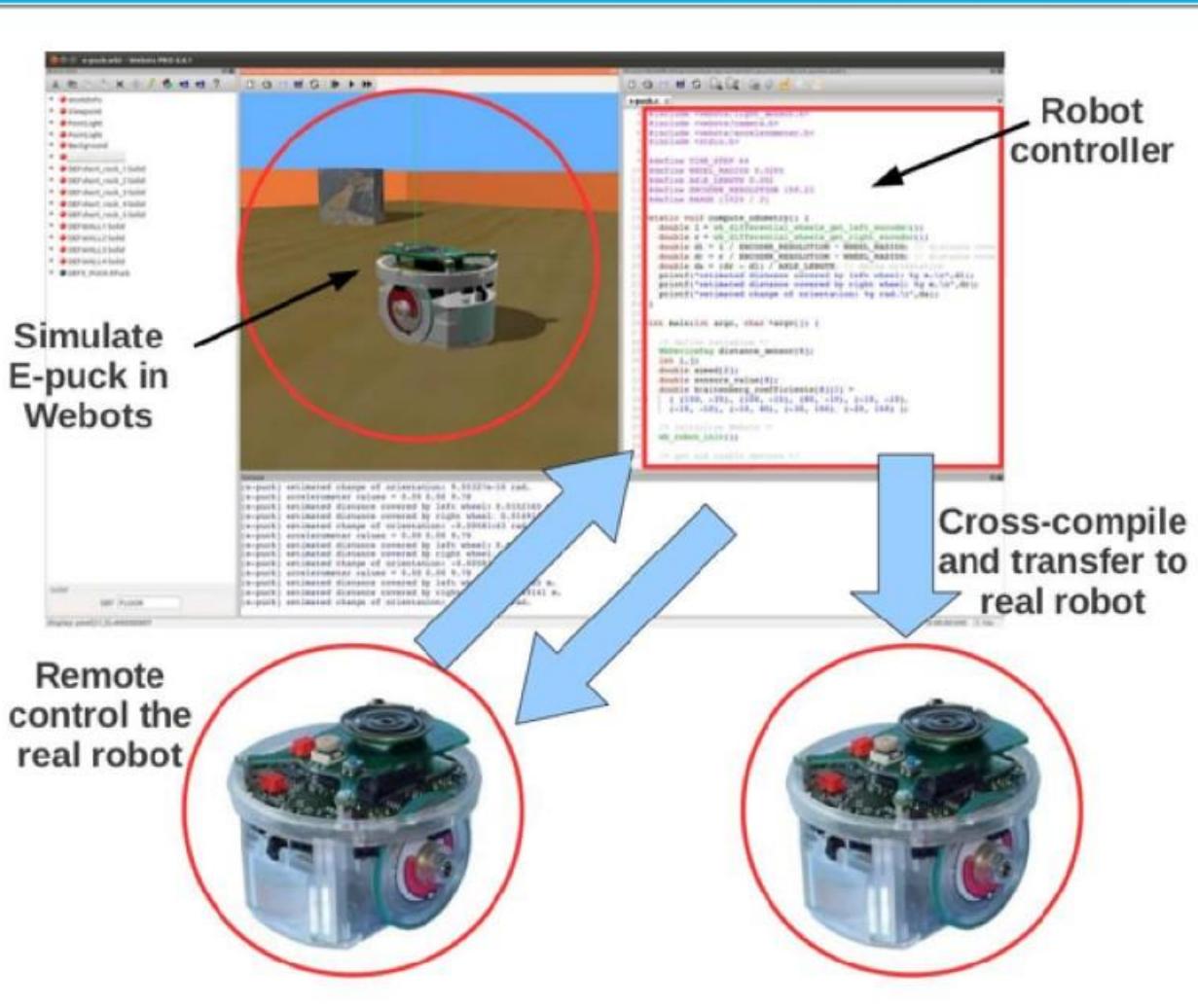
Each controller is executed in a separate child process usually spawned by Webots. Because they are independent processes, controllers don't share the same address space, and may run on different processor cores.

Day

01



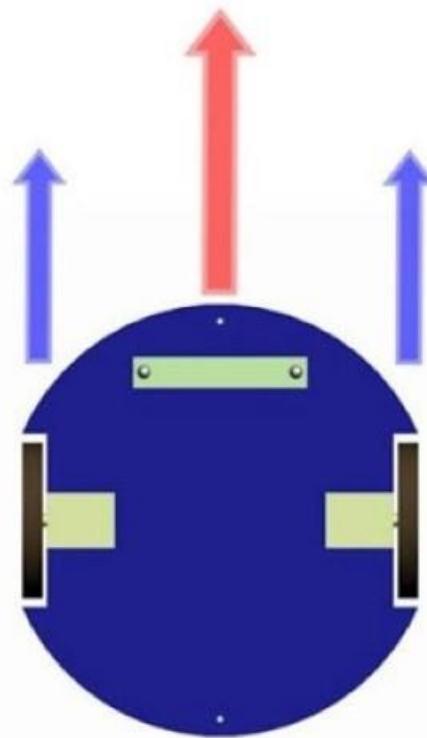
ADD ROBOT.mp4



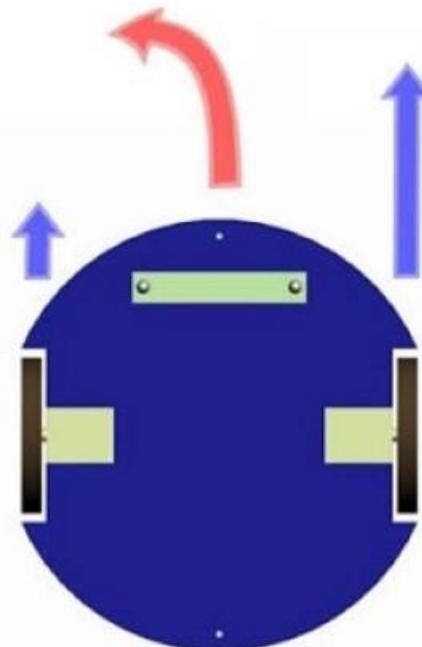
Hands-on #8: Create a new Python controller called epuck_go_forward using the **File / New / New Robot Controller...** menu item. This will create a new epuck_go_forward (or EPuckGoForward) directory in my_first_simulation/controllers. Select the option offering you to open the source file in the text editor.



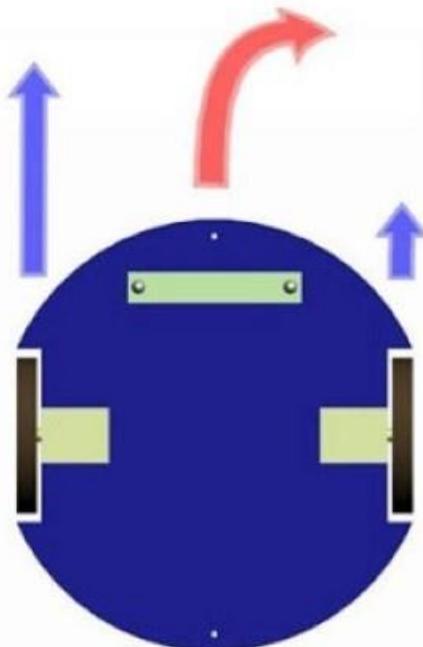
Move forward



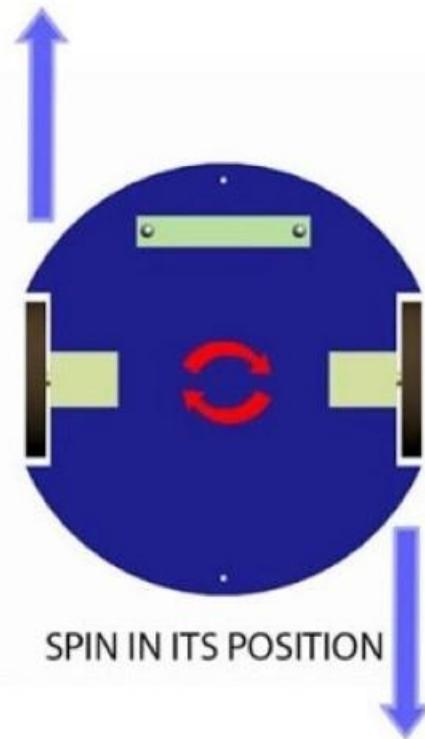
STRAIGHT



TURN LEFT



TURN RIGHT



SPIN IN ITS POSITION

Step 1: Set Up Your Webots Environment

1. Open Webots: Start Webots on your computer.
2. Create a New Project: If you don't already have a project, create a new one.
3. Add the E-puck Robot:
 - Go to the `Scene Tree` panel.
 - Right-click and select `Add...`
 - Navigate to `ROBOT` -> `E-puck` and add it to your scene.

Step 2: Associate a Controller with the Robot

1. Select the E-puck Node:
 - In the `Scene Tree` panel, find and click on the `E-puck` node.
2. Assign a Controller:
 - In the `Field` panel, find the `controller` field.
 - Click on the `Select...` button.
 - Choose `New File...`, name your controller file (e.g., `epuck_go_forward.py`), and confirm.

Step 3: Write the Controller Script

1. Open the Controller Script:
 - In your project directory, find and open the newly created Python controller script (e.g., `epuck_go_forward.py`).

Step 1: Set Up Your Webots Environment

1. Open Webots: Start Webots on your computer.

- على جهاز الكمبيوتر الخاص بك Webots ابدأ: افتح Webots.

2. Create a New Project: If you don't already have a project, create a new one.

- إنشاء مشروع جديد: إذا لم يكن لديك مشروع بالفعل، قم بإنشاء مشروع جديد.

3. Add the E-puck Robot:

• إضافة الروبوت E-puck:

- Go to the `Scene Tree` panel.

- انقل إلى لوحة `Scene Tree`.

- Right-click and select `Add...`

- انقر بزر الماوس الأيمن واحتر `Add...`.

- Navigate to `ROBOT` -> `E-puck` and add it to your scene.

- وأضفه إلى المشهد الخاص بك `E-puck` -> `ROBOT` انقل إلى.

Step 2: Associate a Controller with the Robot

1. Select the E-puck Node:

• حدد العقدة E-puck:

- In the `Scene Tree` panel, find and click on the `E-puck` node.

- وانقر عليها `E-puck`، ابحث عن العقدة `Scene Tree` في لوحة.

2. Assign a Controller:

• تعيين وحدة تحكم:

- In the `Field` panel, find the `controller` field.

- ابحث عن حقل `Field`، `controller` في لوحة.

- Click on the `Select...` button.

- انقر على الزر `Select...`.

- Choose `New File...`, name your controller file (e.g., `epuck_go_forward.py`), and confirm.

- مثلاً قم بتسمية ملف وحدة التحكم الخاص بك `New File...`، اختر `epuck_go_forward.py` وقم بالتأكيد.

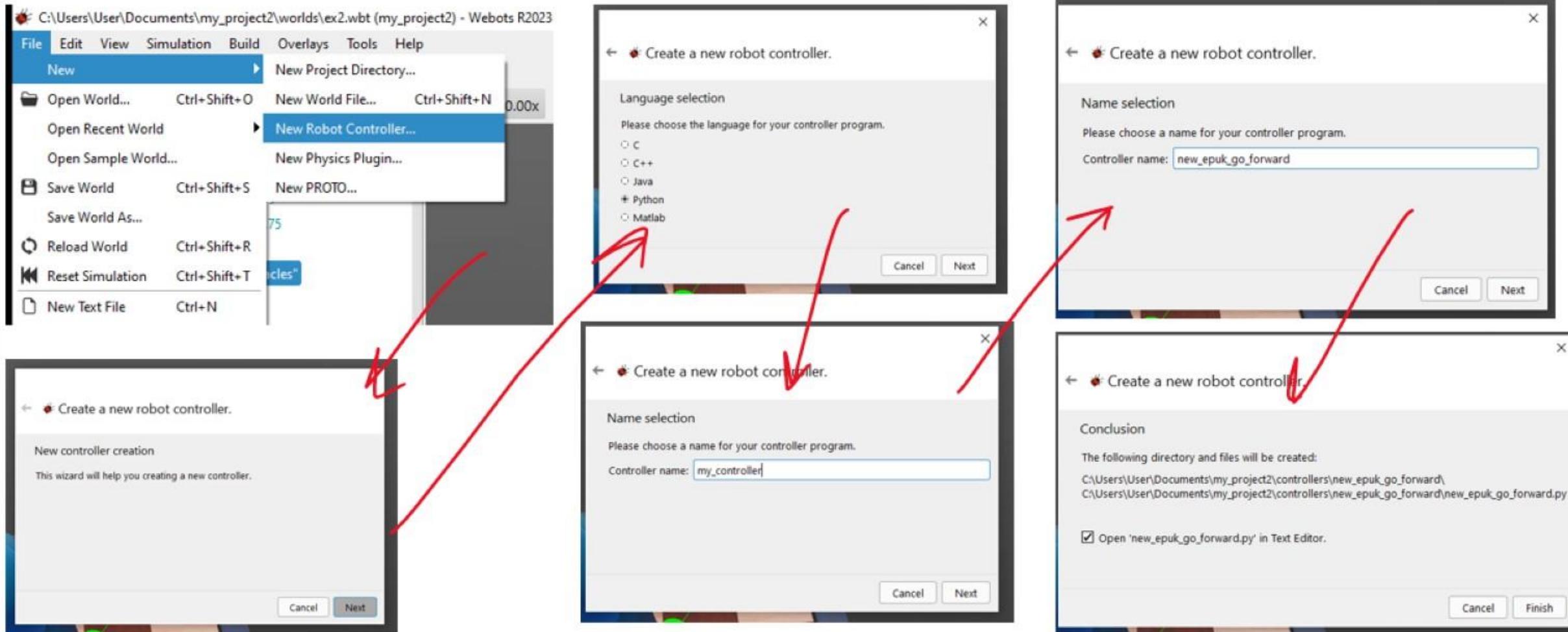
Step 3: Write the Controller Script

1. Open the Controller Script:

- افتح برنامج وحدة التحكم
- In your project directory, find and open the newly created Python controller script (e.g., `epuck_go_forward.py`).
 - مثلاً الذي تم إنشاؤه حديثاً Python في دليل مشروعك، ابحث وافتح ملف وحدة التحكم `epuck_go_forward.py`).

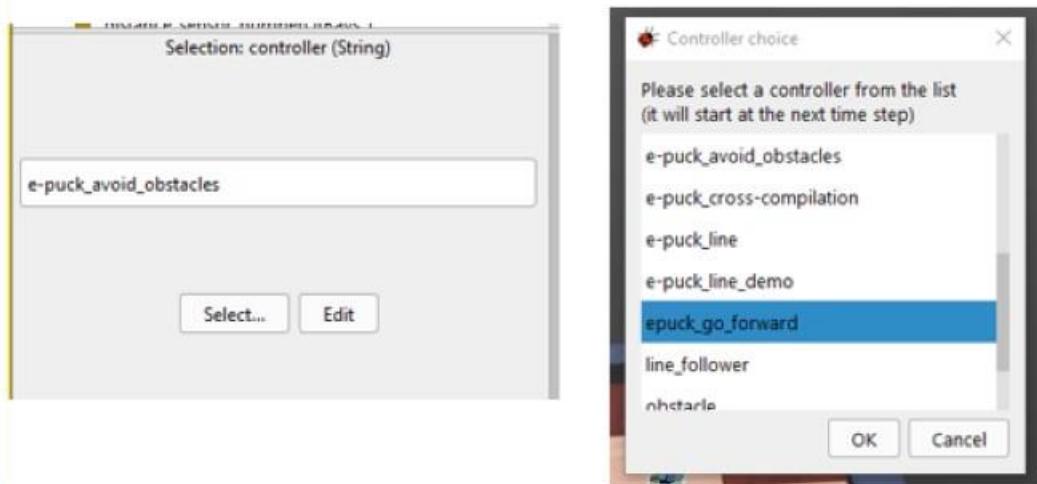
2. Write the Script:

- اكتب البرنامج
- Copy and paste the following simplified code into your controller script. Each step includes detailed comments to help you understand what each part does.
 - انسخ والصق الكود المبسط التالي في برنامج وحدة التحكم الخاص بك. يتضمن كل خطوة تعليقات مفصلة لمساعدتك على فهم كل جزء



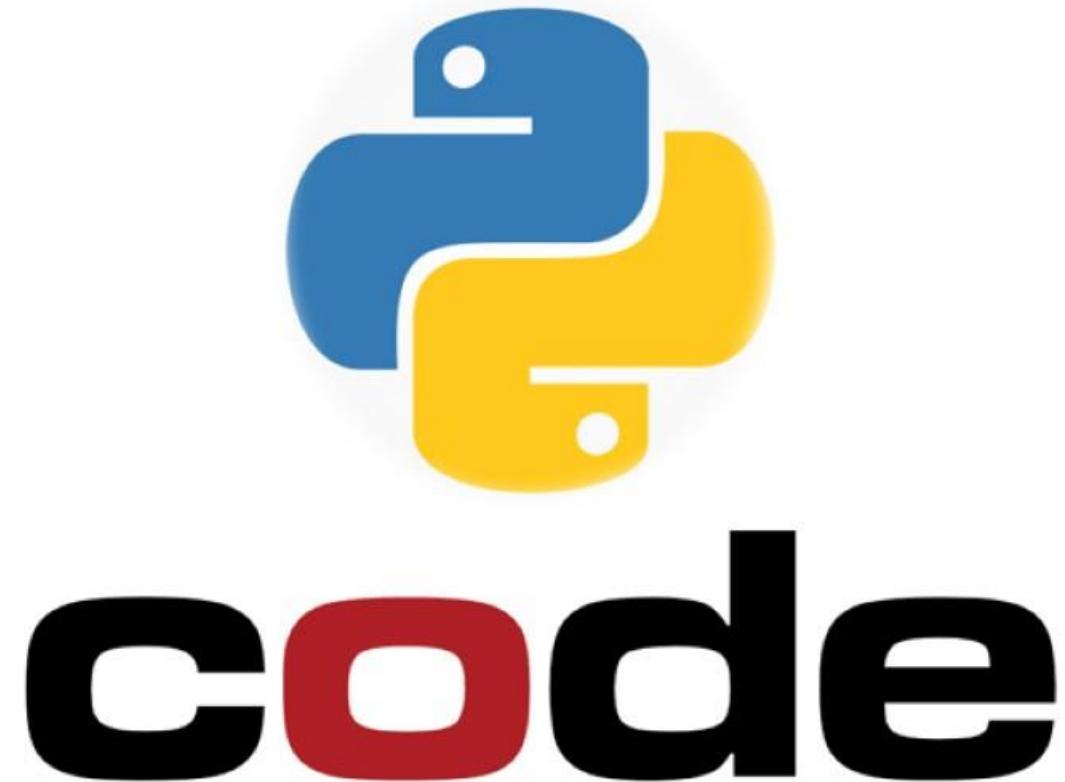
The new source file is displayed in Webots text editor window. We will now associate new epuck_go_forward controller to the E-puck node.

In the scene tree view, select the controller field of the E-puck node, then use the field editor at the bottom of the Scene Tree view: press the Select... button and then select epuck_go_forward in the list. Once the controller is associated with the robot, save the world. Modify the program by getting the motor devices (leftMotor = robot.getDevice('left wheel motor')), and by applying a motor command (leftMotor.setPosition(10.0)):



Day

01



Import the Robot and Motor classes from the Webots controller module

from controller import Robot, Motor

استيراد Robot و Motor من وحدة التحكم Webots

Define a time step for the simulation (this value should match Webots simulation time step)

TIME_STEP = 64

تعريف خطوة زمنية للمحاكاة (يجب أن تطابق هذه القيمة خطوة الوقت في Webots)

Define the maximum speed for the e-puck robot motors

MAX_SPEED = 6.28 # Note: e-puck's speed is often specified in radians per second

تحديد السرعة القصوى لمحركات الروبوت e-puck

ملاحظة: غالباً ما يتم تحديد سرعة e-puck بالراديان في الثانية

```
# Create an instance of the Robot class
```

```
robot = Robot()
```

إنشاء مثيل لفئة Robot

```
# Get the left and right wheel motors of the e-puck robot
```

```
leftMotor = robot.getDevice('left wheel motor')
```

```
rightMotor = robot.getDevice('right wheel motor')
```

الحصول على محركات العجلة اليسرى واليمنى للروبوت e-puck

```
# Set the target position of the motors to infinity (to control speed instead of position)
```

```
leftMotor.setPosition(float('inf'))
```

```
rightMotor.setPosition(float('inf'))
```

ضبط الموضع المستهدف للمحركات إلى ما لا نهاية (للحكم في السرعة بدلاً من الموضع)

```
# Set the initial speed of the motors to 10% of the maximum speed
```

```
leftMotor.setVelocity(0.1 * MAX_SPEED)
```

```
rightMotor.setVelocity(0.1 * MAX_SPEED)
```

ضبط السرعة الأولية للحركات إلى 10% من السرعة القصوى #

```
#Main loop: run until the simulation is terminated
```

```
while robot.step(TIME_STEP) != -1:
```

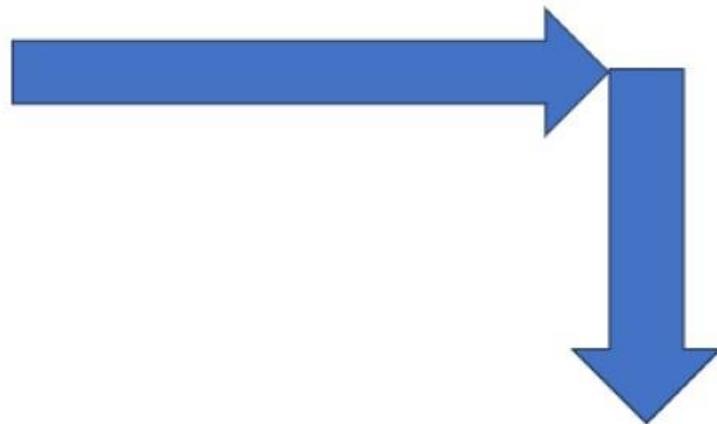
In this loop, we don't need to do anything because the robot will keep moving forward
pass

الحلقة الرئيسية: تستمر حتى يتم إنتهاء المحاكاة #

في هذه الحلقة، لا نحتاج إلى فعل أي شيء لأن الروبوت سيستمر في التحرك للأمام

Moving the e-puck robot **forward for 10 steps** and then **turning right**

epuck_move_turn.py



1. Importing Classes:

- `from controller import Robot, Motor`: Imports the `Robot` and `Motor` classes from the Webots controller module.
- `from controller import Robot, Motor` و `Robot` يستورد الفئات من وحدة التحكم `Motor` Webots.

2. Defining Constants:

- `TIME_STEP = 64`: Defines the time step for the simulation. This should match the time step set in Webots.
- `TIME_STEP = 64` يحدد خطوة الزمن للمحاكاة. يجب أن تطابق هذه القيمة خطوة الوقت في Webots.
- `MAX_SPEED = 6.28`: Defines the maximum speed for the e-puck robot's motors.
- `MAX_SPEED = 6.28` يحدد السرعة القصوى لمحركات الروبوت e-puck.

3. Creating the Robot Instance:

- `robot = Robot()`: Creates an instance of the `Robot` class, allowing you to control the e-puck robot.
- `robot = Robot()` مما يتيح لك التحكم في الروبوت ، `Robot` ينشئ مثيلاً لفئة e-puck.

4. Getting the Motors:

- `leftMotor = robot.getDevice('left wheel motor')`: Gets the left wheel motor.
- `leftMotor = robot.getDevice('left wheel motor')` يحصل على محرك العجلة اليسرى.
- `rightMotor = robot.getDevice('right wheel motor')`: Gets the right wheel motor.
- `rightMotor = robot.getDevice('right wheel motor')` يحصل على محرك العجلة اليمنى.

5. Setting Motor Positions:

- `leftMotor.setPosition(float('inf'))`: Sets the left motor position to infinity, enabling speed control.
 - `leftMotor.setPosition(float('inf'))`: يضبط موضع المحرك الأيسر إلى ما لا نهاية، مما يتيح التحكم في السرعة.
 - `rightMotor.setPosition(float('inf'))`: Sets the right motor position to infinity, enabling speed control.
 - `rightMotor.setPosition(float('inf'))`: يضبط موضع المحرك الأيمن إلى ما لا نهاية، مما يتيح التحكم في السرعة.

6. Moving Forward:

- The loop `for step in range(10)` runs for 10 iterations, moving the robot forward by setting both wheel speeds to 50% of the maximum speed.
 - تستمر لـ 10 تكرارات، تحرك الروبوت للأمام عن طريق ضبط سرعتي `for step in range(10)` . الحلقة العجلتين إلى 50% من السرعة القصوى.

7. Turning Right:

- The loop `for step in range(10)` runs for 10 iterations, turning the robot right by setting the left wheel speed to 50% of the maximum speed and the right wheel speed to 0.
- تستمر لـ 10 تكرارات، تدور الروبوت إلى اليمين عن طريق ضبط `الحلقة` سرعة العجلة اليسرى إلى 50% من السرعة القصوى وسرعة العجلة اليمنى إلى الصفر.

8. Stopping the Robot:

- The loop `while robot.step(TIME_STEP) != -1` runs indefinitely, keeping the robot stopped by setting both wheel speeds to 0.
- تستمر بلا نهاية، تحافظ على الروبوت متوقفاً عن `الحلقة` طريق ضبط سرعتي العجلتين إلى الصفر.



1



Webots

robot simulation



<https://cyberbotics.com/>

2



<https://www.spyder-ide.org/>

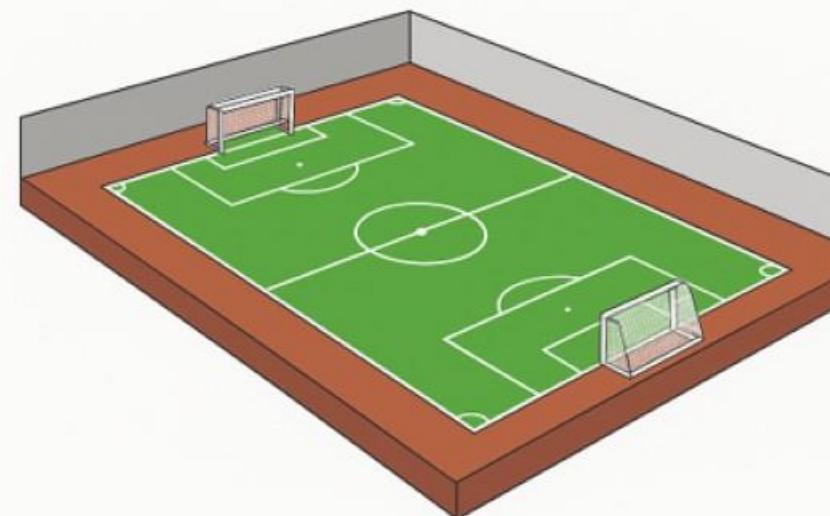
🚩 Webots Robo Soccer Workshop



RoboCup 2023, World Championship, France Bordeaux - Soccer
SSL, MSL, Humanoid, TIGERs Mannheim - 720

Prepare the Soccer Stadium

ملعب كرة القدم Example:
Soccer Field



Understanding the Soccer Ball in Robo Soccer

- In Robo Soccer, the ball is not just a shape — it's a special object that **sends a signal**.
- The robot cannot see like a human — it uses sensors to **detect signals from the ball**.
- Image of the 3D soccer ball in Webots.
- Label: Radius = 0.021m, Image: football_base_color.png



Why Infrared?

- Infrared (IR) signals are used like a flashlight — invisible to us, but the robot can detect it.
- The ball sends an IR signal. The robot listens to that signal and figures out the ball's direction.

Field	Meaning	
Emitter	Sends out IR signals	Emitter {
type	Infrared — invisible beam	name "ball emitter"
range	Signal range (e.g., 0.6 meters)	type "infra-red"
channel	Must match robot's IR receiver	range 0.6 channel 4 }

Download Folder Example S1Open Folder : **Example S1**

- 📁 controllers
- 📁 My_First_Soccer_CODE FILES
- 📁 protos
- 📁 worlds



- 📄 .SoccerWorkshop
- 📄 football_base_color
- 📄 soccer-256
- 📌 SoccerWorkshop



C:\MidOcean 2_2025\ROBOTICS WORKSHOP\WorkSop2_Soccer\Example S1\worlds\SoccerWorkshop.wbt (Example S1) - Webots R2025a

File Edit View Simulation Build Overlays Tools Help

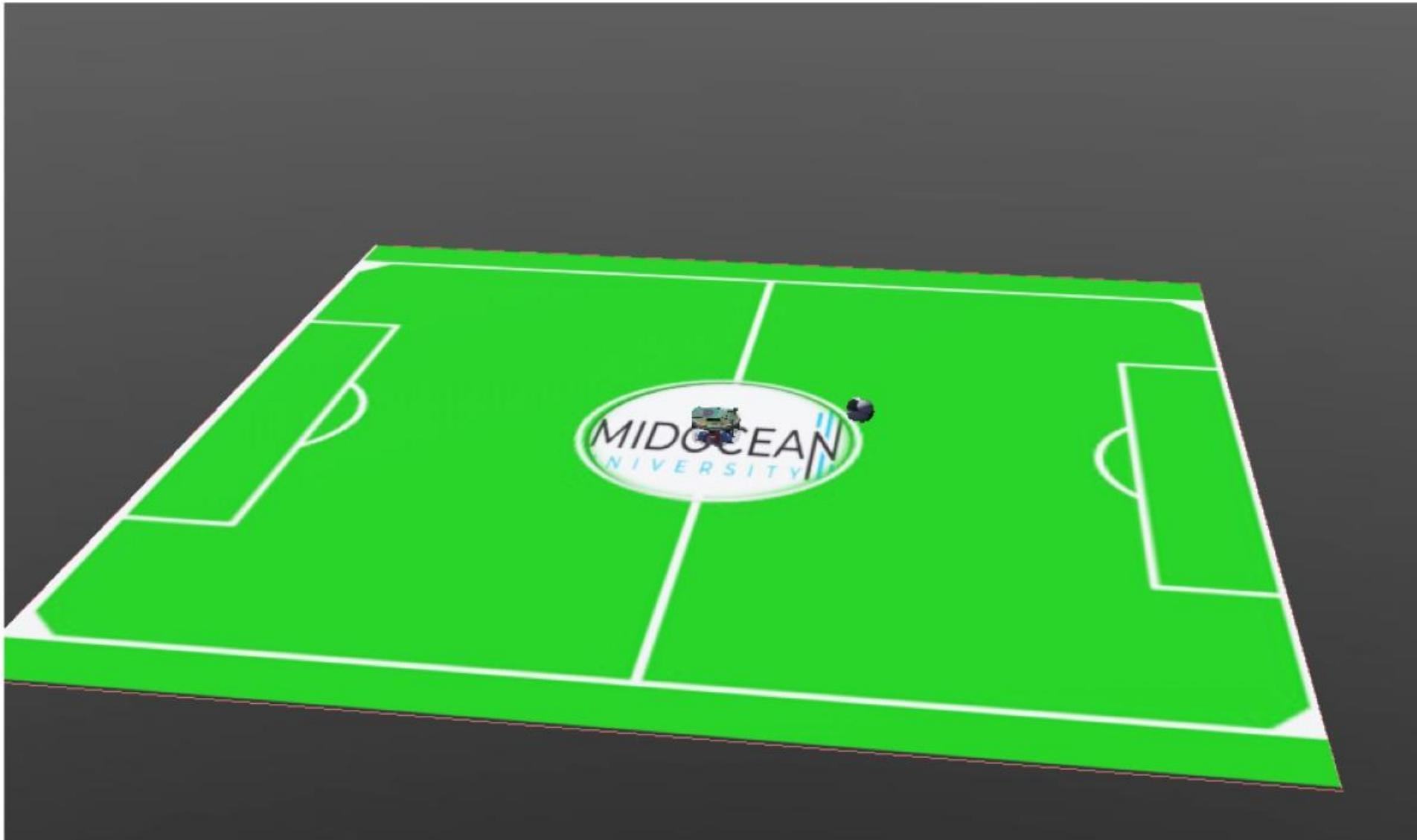
Simulation View



IMPORTABLE EXTERNPROTO

- > Worldinfo
- > Viewpoint
- > TexturedBackground
- > TexturedBackgroundLight
- > Solid "SoccerField"
- > Robot "SoccerBall"
- > E-puck "e-puck"





File Edit View Simulation Build Overlays Tools Help

Simulation View

0:00:00:896 - 0.00x

my_controller.py my_controller2.py my_controller3.py

IMPORTABLE EXTERNPROTO

- > WorldInfo
- > Viewpoint
- > TexturedBackground
- > TexturedBackgroundLight
- > Solid "SoccerField"
- > Robot "SoccerBall"
- > E-puck "e-puck"
 - translation -0.216 -2.73e-10 0.00994
 - rotation -2.74e-05 -1 1.02e-05 0.000225
 - name "e-puck"
 - controller "my_controller"
 - controllerArgs
 - window "<generic>"
 - customData ""
 - supervisor FALSE
 - synchronization TRUE
 - battery
 - version "1"

Selection: controller (String)

my_controller

Select... Edit

```

1 # -----
2 # my_controller.py
3 # A simple controller to move the e-puck forward
4 #
5
6 # Step 1: Import the Robot class from the Webots API
7 from controller import Robot
8
9 # Step 2: Create an instance of the robot
10 robot = Robot()
11
12 # Step 3: Get the time step of the current world
13 # الوقت في كل خطوة من الحالة (بالطريق ثالثة)
14 timestep = int(robot.getBasicTimeStep())
15
16 # Step 4: Get the motors for the left and right wheels
17 # الحصول على المركبات اليمنى واليسرى لجهاز المروحة
18 left_motor = robot.getDevice("left wheel motor")
19 right_motor = robot.getDevice("right wheel motor")
20
21 # Step 5: Set both motors to infinite rotation (velocity control)
22 # تحريك المركبة بجهة مسافة (ليس تحريك موقع)
23 left_motor.setPosition(float('inf'))
24 right_motor.setPosition(float('inf'))
25
26 # Step 6: Set the speed for both motors
27 # تحريك السرعة للمحركين - تجاه مواجهة شفاف للأمام
28 left_motor.setVelocity(3.0)
29 right_motor.setVelocity(3.0)
30
31 # Step 7: Main simulation loop
32 # حلقة تستمر هنا تشهد الحالة
33 while robot.step(timestep) != -1:
34     pass # لا نقم بذلك شيء آخر، ذلك نستقر في المركبة للأمام
  
```

www.midocean.ae

```
# Simple example: Move the robot forward

from controller import Robot

robot = Robot()                      # Initialize robot
timestep = int(robot.getBasicTimeStep()) # Get simulation time step

# Get access to the motors
left_motor = robot.getDevice("left wheel motor")
right_motor = robot.getDevice("right wheel motor")

# Set motors to velocity control mode
left_motor.setPosition(float('inf'))
right_motor.setPosition(float('inf'))

# Set motors speed (3 = medium speed)
left_motor.setVelocity(3.0)
right_motor.setVelocity(3.0)

# Run the simulation Loop
while robot.step(timestep) != -1:
    pass # Robot keeps moving forward
```

```

# Step 1: Import the Robot class from the Webots API
# التحكم في الروبوت Webots من مكتبة "Robot" استيراد كائن #
# هو الكائن الأساسي الذي نستخدمه للتفاعل مع أي روبوت داخل المحاكاة "
from controller import Robot
# 

# Step 2: Create an instance of the robot
# إنشاء نسخة (كائن) من الروبوت للتحكم به
# هذا السطر ينشئ الاتصال بين الكود وبين الروبوت داخل المحاكاة #
robot = Robot()
# 

# Step 3: Get the time step of the current world
# الحصول على "الخطوة الزمنية" للمحاكاة
# الوقت بين كل تحديث للمحاكاة (بوحدة المللية ثانية)
# مثال: إذا كانت القيمة 32 فهذا يعني أن المحاكاة تحدث 31 مرة في الثانية #
timestep = int(robot.getBasicTimeStep())
# 

# Step 4: Get the motors for the left and right wheels
# الحصول على المحركات اليمنى واليسرى لعجلات الروبوت
# (.wbt) نستخدم أسماء الأجهزة كما هي معرفة في ملف المحاكاة "
# e-puck "left wheel motor" و "right wheel motor"
left_motor = robot.getDevice("left wheel motor")
right_motor = robot.getDevice("right wheel motor")

```

```

# Step 5: Set both motors to infinite rotation (velocity control mode)
# ضبط المحركات بحيث تعمل بالسرعة فقط وليس بالموضع
# فهذا يعني تحكم بالسرعة setPosition(float('inf'))
# إذا وضعنا قيمة رقمية فهذا يعني تحكم بالموضع (زاوية دوران محددة)
# أما إذا وضعنا قيمة رقمية فهذا يعني تحكم بالسرعة
left_motor.setPosition(float('inf'))
right_motor.setPosition(float('inf'))

```

```

# Step 6: Set the speed for both motors
# تحديد السرعة للمحركات
# القيم الموجبة → الروبوت يتحرك للأمام
# القيم السالبة → الروبوت يتحرك للخلف
# السرعة e-puck 6.28 rad/s تقريرًا
# هنا نضع سرعة متوسطة = 3.0 rad/s
# السرعة للمotor الأيسر # السرعة للمotor الأيمن #
left_motor.setVelocity(3.0)
right_motor.setVelocity(3.0)

```

```

# Step 7: Main simulation loop
# الحلقة الرئيسية التي تعمل طول مدة تشغيل المحاكاة
# يعيد -1 إذا توقفت المحاكاة، ولا يستمر →
# هنا لا نغير السرعة داخل الحلقة، فتترك الروبوت يتحرك للأمام دائماً
while robot.step(timestep) != -1:
    # لا نقوم بأي تعليمات إضافية هنا، فقط نستمر بالحركة للأمام
    pass

```

ملاحظات تعليمية :

لتدوير الروبوت لليسار: اجعل سرعة المotor الأيسر أقل من الأيمن - #

مثال: left_motor.setVelocity(2.0); right_motor.setVelocity(4.0)

لتدويره لليمين: اجعل سرعة المotor الأيمن أقل من الأيسر - #

لإيقافه: ضع السرعة = 0.0 لكلا المحركتين -

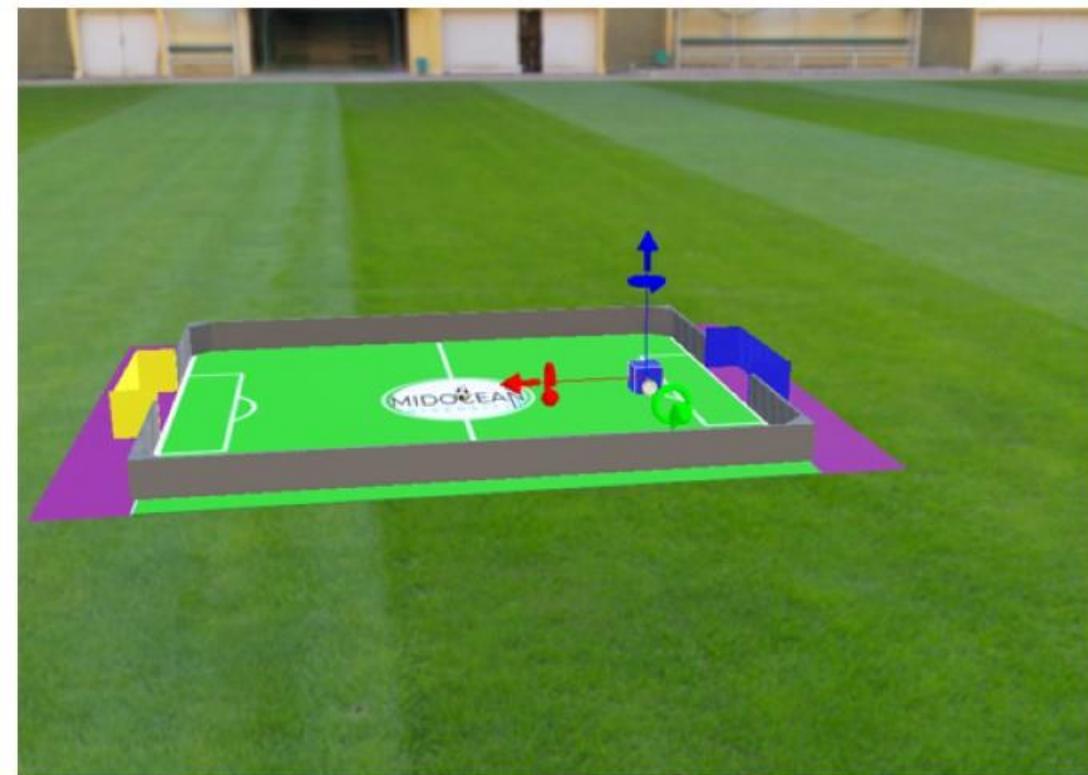
Download Folder Example S2

The main objective is to write a code allows a soccer robot (B1) in Webots to:

- Look for the ball using an infrared (IR) sensor,
- Decide if the ball is in front, to the left, or to the right,
- Move straight if the ball is in front,
- Turn if the ball is not in front,
- Stop if the ball is not detected.



Example S2



Inside folder rcj_soccer_team_blue:

📁 _pycache_	2025-07-20 12:16 PM	File folder
🐍 rcj_soccer_robot	2024-09-04 5:00 PM	Python File
rcj_soccer_team_blue	2024-09-04 5:00 PM	Python File
🐍 robot1	2024-09-04 5:00 PM	Python File
utils	2024-09-04 5:00 PM	Python File
ck access		
esktop		
ownloads		
Documents		
ictures		
seesoft Sc		
ewVideo		
ewVideo		
orlds		
:Drive - Ar		
ps		
tachment		
ptop		
otebooks		
iceMobil		

A red arrow points from the 'robot1' file in the list above to the 'robot1' file in the context menu below. Another red arrow points from the 'Add to "robot1.zip"' option in the context menu to the 'Add to archive...' option at the bottom of the menu.

- Open
- Edit with IDLE
- Add to "robot1.zip" (highlighted)
- Edit with Spyder
- Add to Share hub
- Edit with CLion
- Move to OneDrive
- Scan with Microsoft Defender...
- Share
- Open with...
- Add to archive...
- ...

```
# Step 1: Import the Webots Robot API
# This gives us access to Webots features like sensors and motors.
from controller import Robot
```

We start by importing Robot so we can connect our code to the robot in the Webots simulator.

```
# Step 2: Import a helper function to decide the direction of the ball
import utils
```

utils is a file we created that contains a simple function to help us decide:

- Should we move forward?
- Or should we turn left or right?

This is based on the position (vector) of the ball.

- utils هو ملف يحتوي على دالة بسيطة تساعد الروبوت على تحديد:
- هل يجب أن يتحرك للأمام؟
 - أم يلف لليسار أو اليمين؟
- وذلك بناء على موقع الكرة بالنسبة للروبوت.

```
# Step 3: Import the RCJSoccerRobot class
# This class connects all the sensors and motors in a simple way.
from rcj_soccer_robot import RCJSoccerRobot, TIME_STEP
```

RCJSoccerRobot is a ready-made robot class that connects:

- IR ball sensor
- Compass
- GPS
- Sonar sensors
- Motors (wheels)

TIME_STEP controls how often the robot updates (like frames in a video).

- RCJSoccerRobot هو كائن جاهز يتصل تلقائياً بـ
- حساس الكرة بالأشعة تحت الحمراء (IR),
 - البوصلة (Compass),
 - GPS جل,
 - حساسات السونار (Sonar),
 - المحركات (Motors).

أما TIME_STEP فهو يحدد كل كم ميلي ثانية يجب أن يُحدث الروبوت بيئاته (مثل إطارات الفيديو).

```
# Step 4: Create a Webots robot object
```

```
robot_api = Robot()
```

We ask Webots to give us access to the actual physical robot in the simulation.
This robot has all the hardware (sensors and motors) ready.

```
# Step 5: Create our smart robot with all sensors and motors enabled
```

```
robot = RCJSoccerRobot(robot_api)
```

Now we wrap the robot in our smart RCJSoccerRobot so we can easily:

- Read data from sensors (like IR ball, GPS),
- Control the motors (to move).

نلف كائن الروبوت داخل كلاس `RCJSoccerRobot` ليصبح من السهل علينا:

- قراءة بيانات الحساسات،
- والتحكم بالمحركات.

```
# Step 6: Start an infinite loop to control the robot
# This loop runs again and again, like a heartbeat.
while robot_api.step(TIME_STEP) != -1:
```

- This line starts a **loop that runs forever** (until you stop the simulation).
- Inside this loop, we check sensors and move the robot.
- The robot updates every TIME_STEP milliseconds.

Step 7: Check if the robot sees the ball using its IR sensor

```
if robot.is_new_ball_data():
```

نتأكد إذا كان هناك بيانات جديدة من حساس الـ IR يعني أن الروبوت اكتشف وجود الكوة.

- The robot has an IR receiver to **detect the soccer ball**.
- This line checks if there is **new information** about where the ball is.

Step 8: Get the ball's direction

```
ball_data = robot.get_new_ball_data()
```

- We read the direction of the ball from the IR receiver.
- This gives us a **3D vector** like [x, y, z] showing where the ball is.

نقرأ موقع الكرة كنقطة في الفضاء (متجه ثلاثي الأبعاد مثل: z, y, x، يحدد اتجاه الكرة بالنسبة للروبوت.

```
# Step 9: Use our helper function to decide where to go
# 0 = forward, 1 = turn left, -1 = turn right
direction = utils.get_direction(ball_data["direction"])
```

- We pass the direction vector to a helper function.
- This function returns:
 - 0 → ball is in front
 - 1 → ball is on the left
 - -1 → ball is on the right

نستخدم الدالة `get_direction` لتحديد اتجاه الكرة:

- 0 → الكرة أمام الروبوت.
- 1 → الكرة على اليسار.
- -1 → الكرة على اليمين.

Step 10: Decide how fast to spin each wheel

```
if direction == 0:  
    left_speed = 7  
    right_speed = 7
```

```
else:  
    left_speed = direction * 4  
    right_speed = direction * -4
```

- If the ball is in front, go **straight**.

- Both wheels spin forward at the same speed.

إذا كانت الكرة أمام الروبوت مباشرة:

- نحرك العجلتين بنفس السرعة إلى الأمام.

- If the ball is not in front:

- If direction = 1, turn left: left wheel slower than right.

- If direction = -1, turn right: right wheel slower than left.

- This makes the robot **rotate toward the ball**.

إذا كانت الكرة على اليمين أو اليسار:

- تبطئ عجلة وتسرع الأخرى حتى يدور الروبوت باتجاه الكرة.

Step 11: Apply the speeds to the wheels to move the robot

```
robot.left_motor.setVelocity(left_speed)  
robot.right_motor.setVelocity(right_speed)
```

We finally set the motors so the robot moves in the right direction.

نرسل الأوامر النهائية لتشغيل المحركات وتحريك الروبوت حسب السرعات التي حسبناها.

else:

Step 12: If no ball is seen, stop the robot

```
robot.left_motor.setVelocity(0)  
robot.right_motor.setVelocity(0)
```

- If the robot doesn't see the ball, it **stops moving**.
- This prevents the robot from getting lost or spinning around randomly.

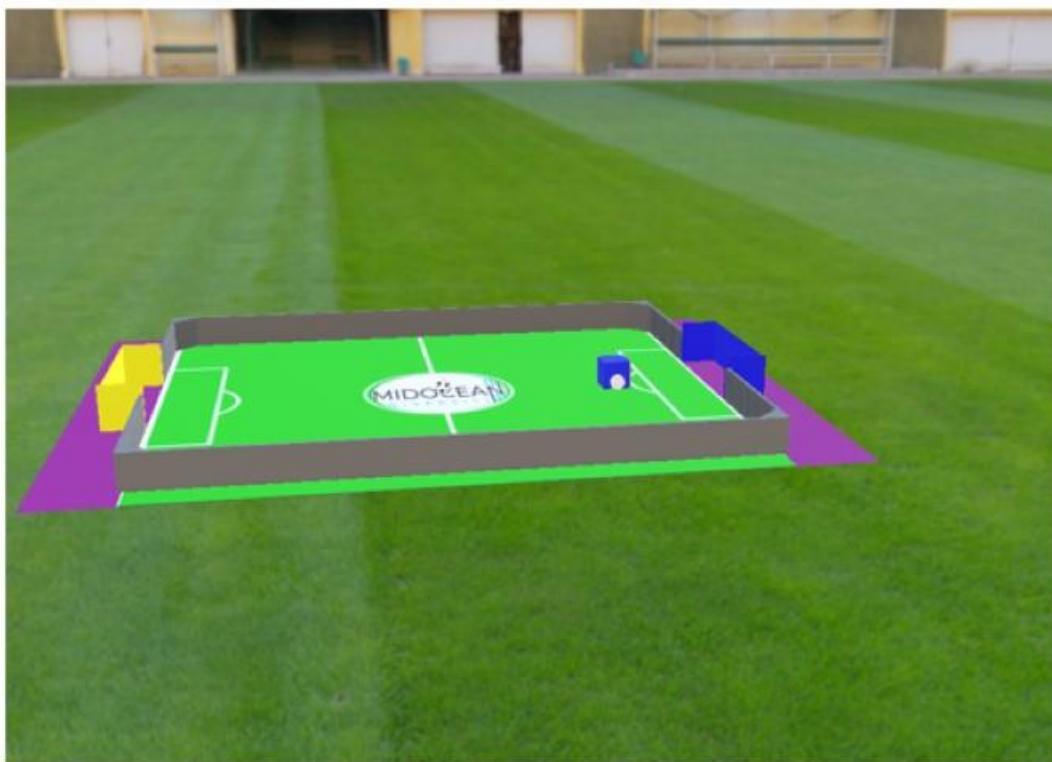
إذا لم تكن الكرة مرئية، نوقف كلا المركين حتى لا يتحرك الروبوت بشكل عشوائي.

Summary of What the Robot Does:

1. Looks for the ball using IR.
2. If it sees the ball:
 1. Moves straight if it's in front.
 2. Turns left or right to face the ball.
3. If it doesn't see the ball:
 1. Stops.

LIVE DEMO

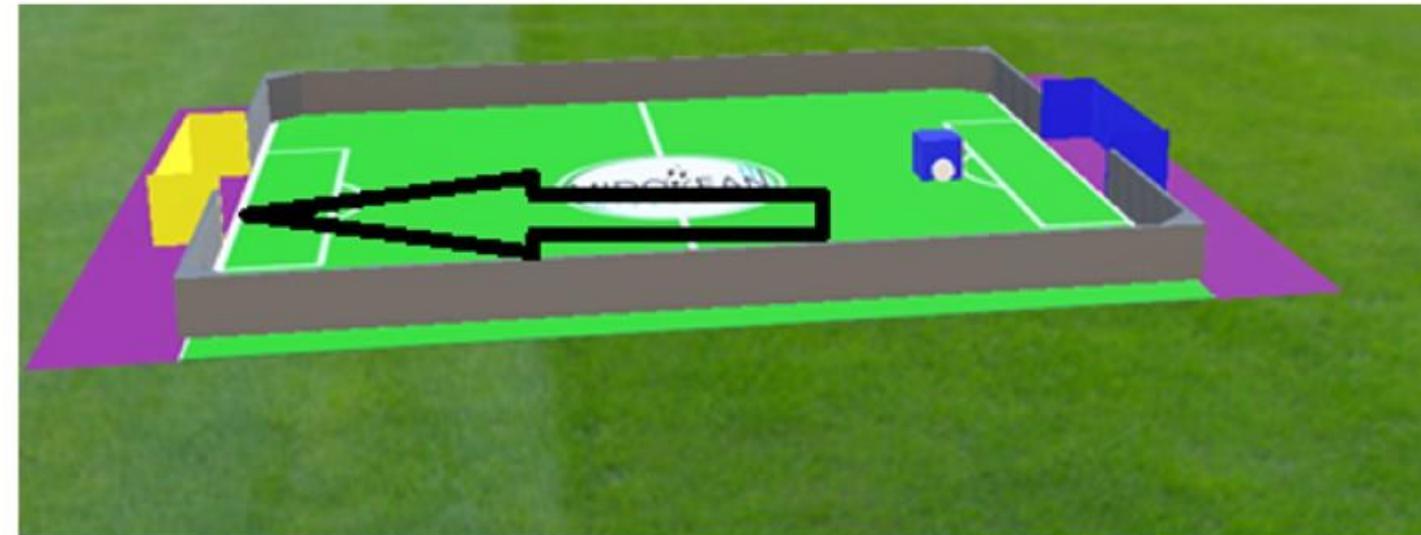
Example 2



```
robot.py x
1 # Dr.Khaled Eskaf
2 #Professor of Intelligent Systems
3 #khaled.eskaf@mail.mcgill.ca
4
5
6
7 # rcj_soccer_player controller - ROBOT 81
8 # وحدة التحكم بالروبوت 81
9
10 # -----
11 # Import the Webots Robot API
12 # استيراد واجهة برمجة التطبيقات الخاصة بـ Webots لتحكم بالروبوت
13 from controller import Robot
14
15 # -----
16 # Import helper function to decide direction
17 # استيراد مثـل utils الذي يحتوى على دالة تساعد الروبوت في تحديد اتجاه الكرة
18 import utils
19
20 # -----
21 # Import base robot configuration (sensors, motors, etc.)
22 # استيراد الكلاس RCJSoccerRobot الذي يربط بين الحساسات والمحركات
23 # واستيراد الزمن بين كل خطوة في الحركة
24 from rcj_soccer_robot import RCJSoccerRobot, TIME_STEP
25
26 # -----
27 # Create an instance of the Robot class from Webots
```

Download Folder Example S3

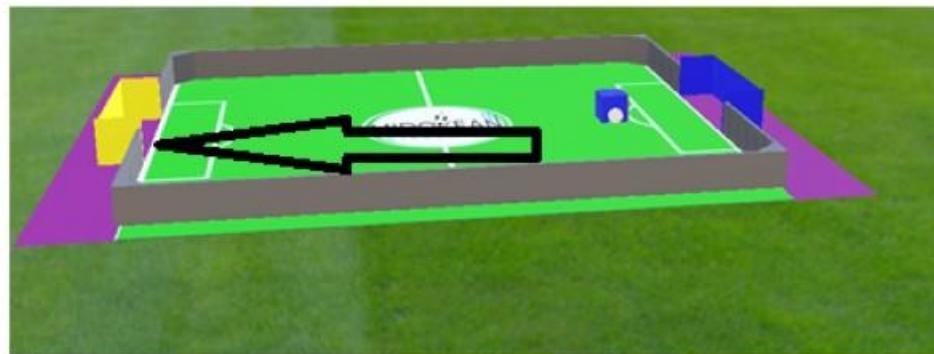
How Robot B1 Finds the Yellow Goal (Only) – No Ball



First: Understand the Field Orientation

In setup:

- **Blue team (B1)** starts on the **right side** of the field.
- **Yellow goal** is on the **left side**.
- The robot's **compass** tells it what direction it is facing.
- The **goal** never moves—it's fixed.



1. The Field Has a Fixed Layout

Team	Start Side	Goal Direction
B1 (Blue)	Right	Moves left to score into the yellow goal
Y1 (Yellow)	Left	Moves right to score into the blue goal

- The goal **does not move**.
- The robot must know where it is on the field and which **side to attack**.

Robot B1 always tries to move **left** (toward the Yellow goal on the left side of the field).

 Step 1:  What is the robot trying to do?

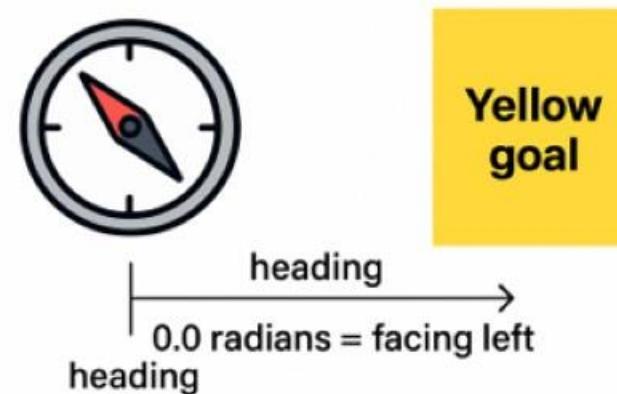
- The robot wants to score a goal  in the **yellow goal**.
- The **yellow goal** is always on the **left** side of the soccer field.
- So, the robot must **face left** before moving forward.

 Step 2: How does the robot know where it's facing?

- The robot uses a **compass sensor** (like the compass on a phone).
- When the robot reads its compass, it gets an angle called heading.

 The target heading is:

- 0.0 radians = facing **left** (toward the yellow goal).



 Step 3: How does the robot turn or move?

• The robot checks:

- If it's already looking at the yellow goal → move forward.
- If it's a little turned → rotate slowly left or right.
- It keeps checking and adjusting.

```
if abs(angle_diff) < 0.1:  
    move forward  
elif angle_diff > 0.1:  
    turn left  
else:  
    turn right
```

 Summary for Students (In Arabic + English):

English

The robot wants to go to the yellow goal.

It uses the compass to know its direction.

If it is not facing the goal, it turns.

Once it faces the goal, it moves forward.

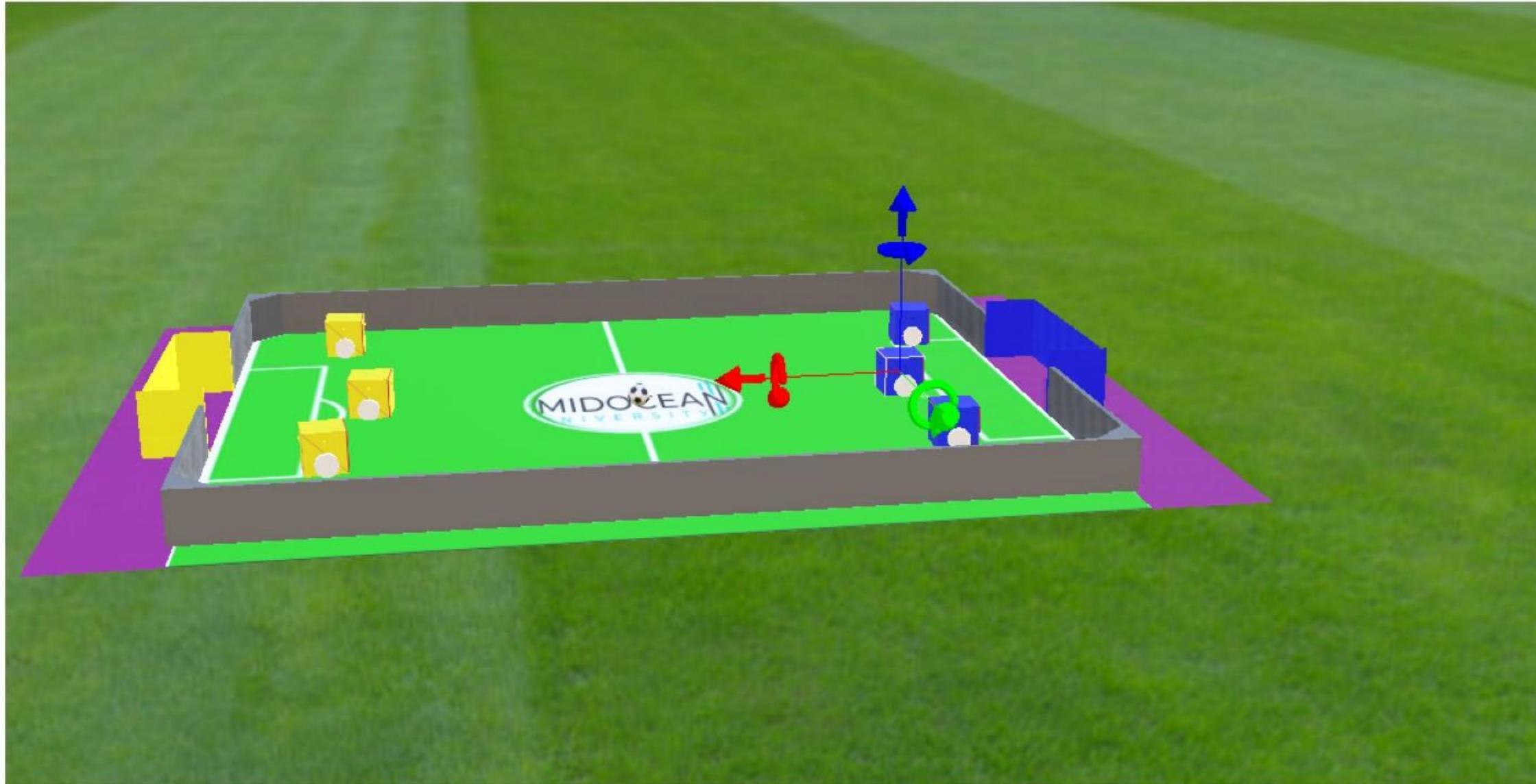
العربية

الروبوت يريد الذهاب إلى المرمى الأصفر.

يستخدم البوصلة ليعرف اتجاهه الحالي.

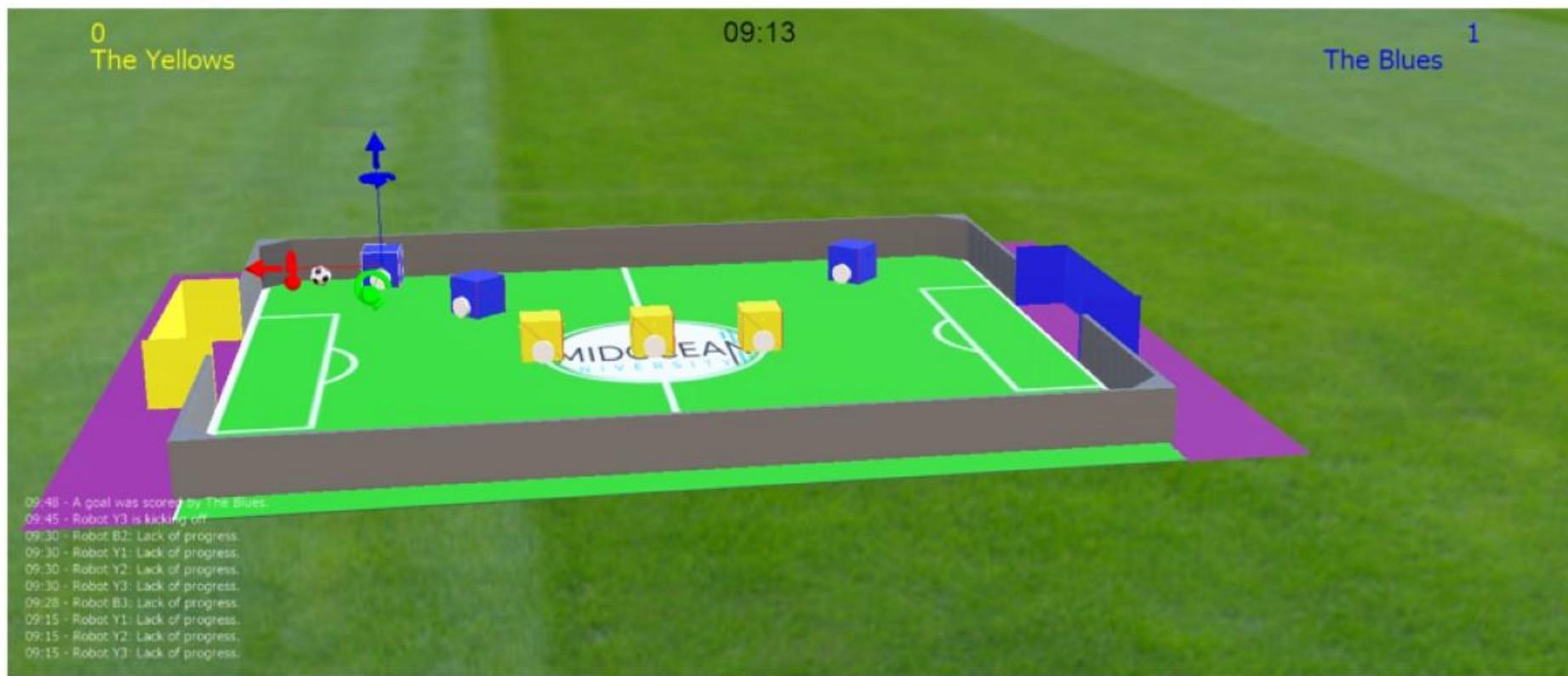
إذا لم يكن يواجه المرمى، فإنه يدور.

عندما يواجه المرمى، يتحرك للأمام.





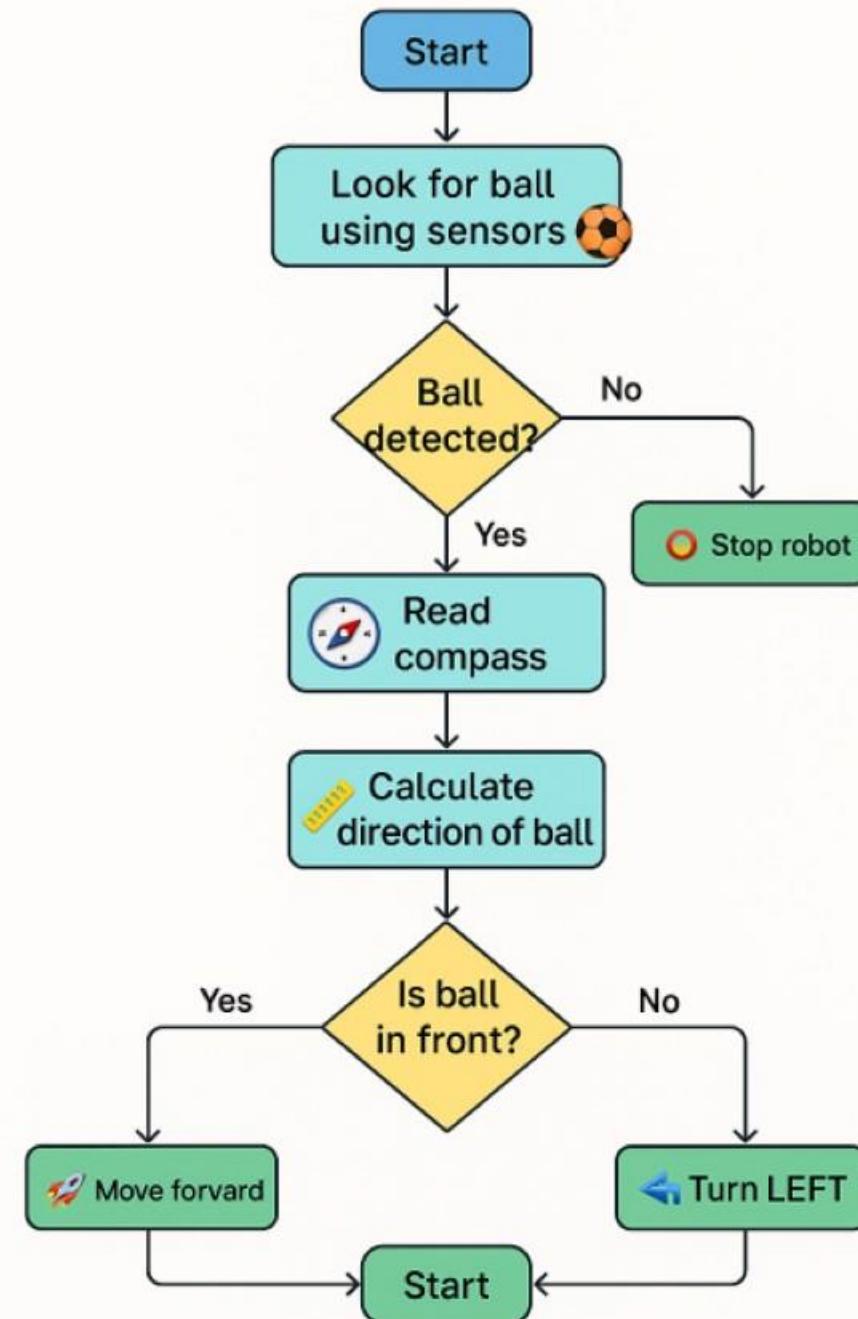
Example S4



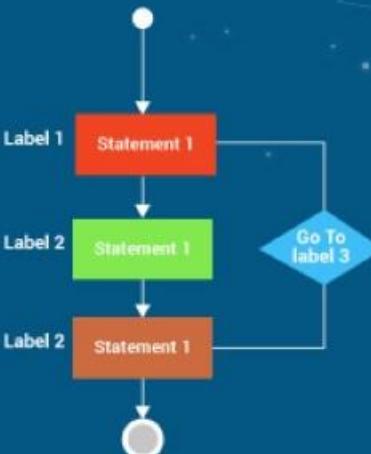
 **What are we trying to do?**

We are programming a small soccer robot to:

1. Look for the soccer ball  using its infrared sensor.
2. Decide where the ball is (front, left, or right).
3. Turn or move toward the ball to try and "catch" it.
4. Print useful info like its direction, compass heading, and location.



Goto in Python



Example 4 > controllers > rcj_soccer_team_blue

Name	Date modified	Type	Size
__pycache__	2025-08-03 1:07 AM	File folder	
rcj_soccer_robot	2025-08-02 11:03 PM	Python File	6 KB
rcj_soccer_team_blue	2025-08-02 11:04 PM	Python File	1 KB
robot1	2025-08-03 12:46 AM	Python File	5 KB
robot2	2025-08-03 12:17 AM	Python File	3 KB
robot3	2025-08-03 12:25 AM	Python File	4 KB
utils	2025-08-02 11:04 PM	Python File	1 KB

Improve the robot's behavior

GOAL: Make Robot B1 Play Smarter in Soccer

First: What does the current robot do?

- It looks for the ball using an IR sensor.
- If the ball is in front → move forward.
- If the ball is on the left or right → turn.
- It doesn't know how far it is from the goal.
- It doesn't try to avoid other robots or walls.

- يبحث عن الكرة باستخدام حساس الأشعة تحت الحمراء.
- إذا كانت الكرة أمامه → يتحرك للأمام.
- إذا كانت الكرة على اليسار أو اليمين → يلف.
- لا يعرف كم يبعد عن المرمى.
- لا يتتجنب الروبوتات الأخرى أو الجدران.





Final Assignment – Robot Soccer Competition



Deadline: [16-August-2025 at 11.30 PM]



Submission via Moodle

Final Assignment – Robot Soccer Competition.DOC

Day

02



Thank You

