

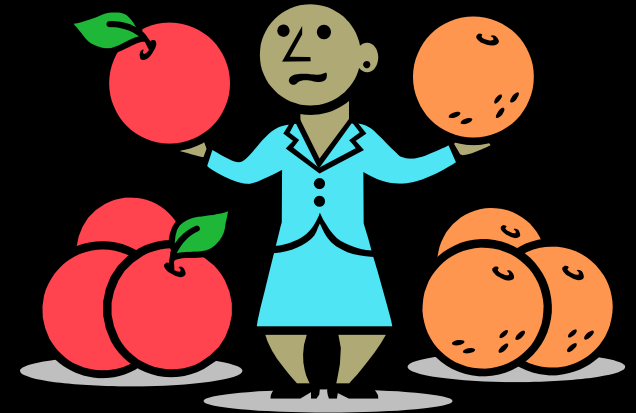
Webots Basics



Computer vs. Robotic Programs

■ Computer Programs:

- designed to **compute an answer**
- **data usually valid** when available
- **predictable** program flow
- foreseen **errors easily handled**

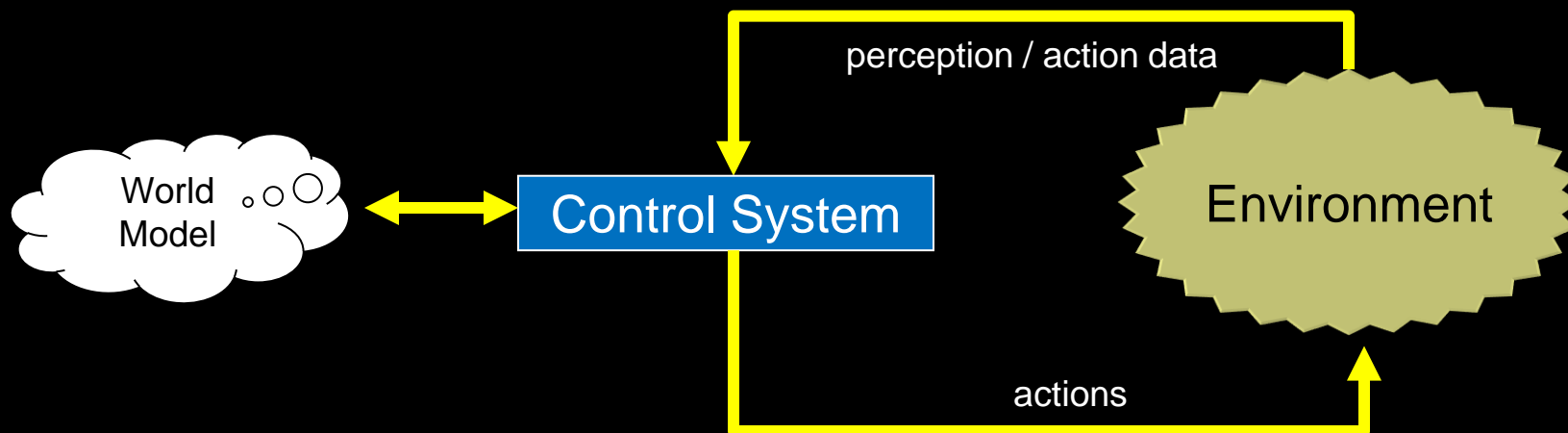
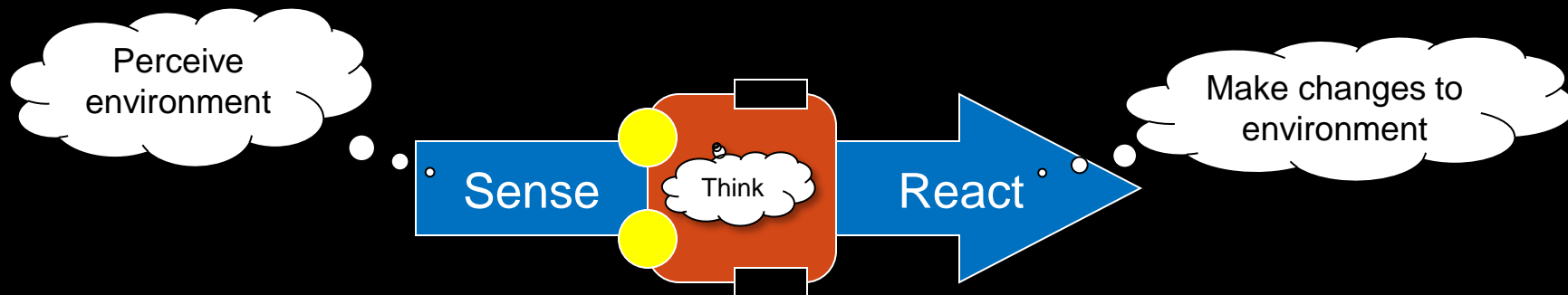


■ Robotic Programs:

- designed to **react to achieve goals**, not “an answer”
- sensors often produce **invalid data** (or data missing)
- **unpredictable situations** due to dynamic environment (e.g., unforeseen obstacles, wrong or missing sensor data, communication outages, hardware failure, etc..)
- program must degrade gracefully in **difficult situations**

Robot Processing

- Here is the basic “flow of control” commonly used:



Robot Control - Pseudocode

```
MyRobotController() {
```

```
    Set up sensors and motors
```

```
    WHILE (TRUE) {
```

```
        detectLeft = read left sensor;  
        detectRight = read right sensor;
```

```
        IF (detectLeft) THEN  
            turn = right;  
        ELSE IF (detectRight) THEN  
            turn = left;  
        ELSE  
            turn = none;
```

```
        IF (turn == left) THEN  
            turn leftMotor on backwards  
            turn rightMotor on forward  
        ELSE IF (turn == right) THEN  
            turn rightMotor on backwards  
            turn leftMotor on forward  
        ELSE  
            turn rightMotor on forward  
            turn leftMotor on forward
```

```
    }
```

```
}
```

Each robot program runs in an infinite loop.

Initialize

Set up the motors, sensors, variables etc...

Sense

Read sensors to determine if a collision is detected on the left or right sensor.

Think

Decide whether robot should go straight or turn away.

React

Turn left, turn right, or move forward accordingly.

Robot Control – JAVA code

Informs JAVA compiler where to find libraries of various functions that you will use in your program. You will add lots more **import** statements.

All classes are given a name which must match the filename ... (**LabController.java** in this case)

```
import com.cyberbotics.webots.controllers.*;
```

```
public class LabController {
```

```
    /* Declare static constants & variables here */
```

```
    public static void main(String[] args) {
```

```
        Robot robot = new Robot();           // Creates a generic Robot object
```

```
        int timeStep = (int) Math.round(robot.getBasicTimeStep());
```

```
        /* INITIALIZE ... variables, sensors, motors, etc... */
```

```
        while(robot.step(timeStep) != -1) {
```

```
            /* SENSE ... Read sensors */
```

```
            /* THINK ... Make decision as to what to do */
```

```
            /* REACT ... Move motors, head, arms, etc... */
```

```
        }
```

```
    }
```

```
}
```

main function is starting point of your program.

Simulation time-step in (ms).
You won't change this.

Loops forever (until PAUSED or STOPPED)

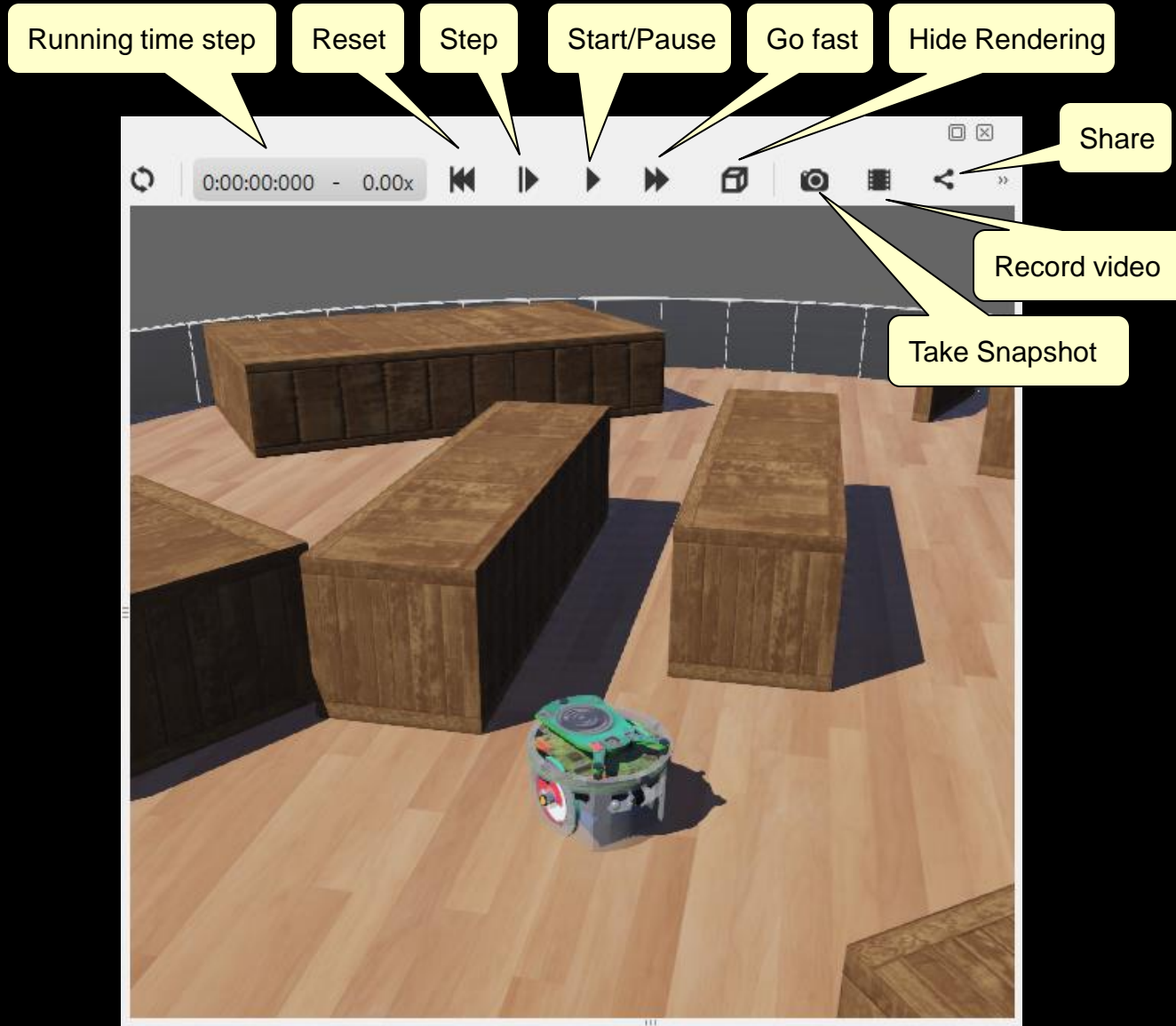
Webots Interface – main GUI

The screenshot displays the Webots R2023a interface. The top menu bar includes File, Edit, View, Simulation, Build, Overlays, Tools, and Help. The toolbar below contains icons for various functions. The main interface is divided into three panels:

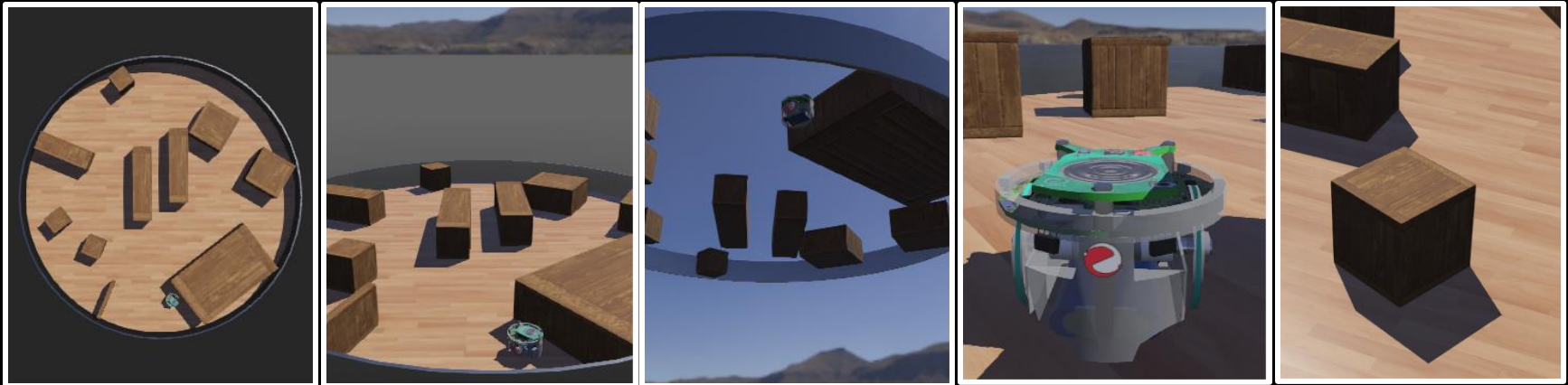
- Left Panel (IMPORTABLE EXTERNPROTO):** A list of objects available for import, including WorldInfo, Viewpoint, TexturedBackground, TexturedBackgroundLight, CircleArena "circle arena", E-puck "e-puck", and multiple instances of WoodenBox. A callout bubble says: "Edit the environment objects here."
- Central Panel (3D Simulation):** A 3D rendering of a simulated environment with a wooden floor and several wooden boxes. An E-puck robot is visible in the center. A callout bubble says: "Watch your robot being simulated Here."
- Right Panel (Lab1Controller.java):** A code editor showing the Java code for the robot controller. A callout bubble says: "Write your code In this editor."

At the bottom, there is a Console panel labeled "Console - All". A callout bubble says: "See compile errors and debug print statements here."

Webots Interface — Simulation Controls



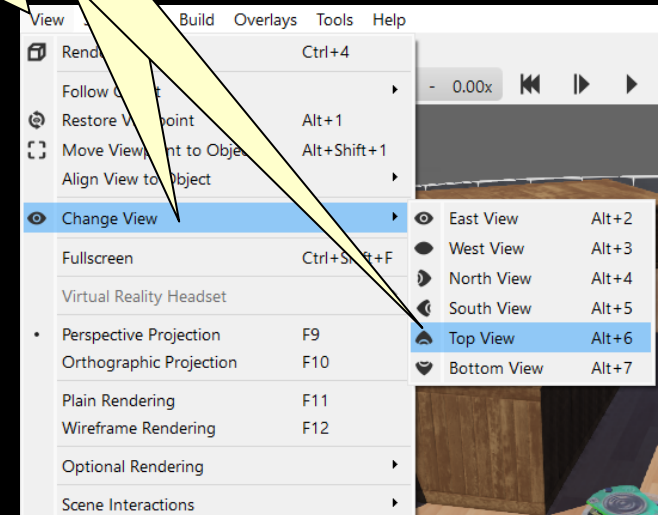
Webots Interface — Changing Viewpoint



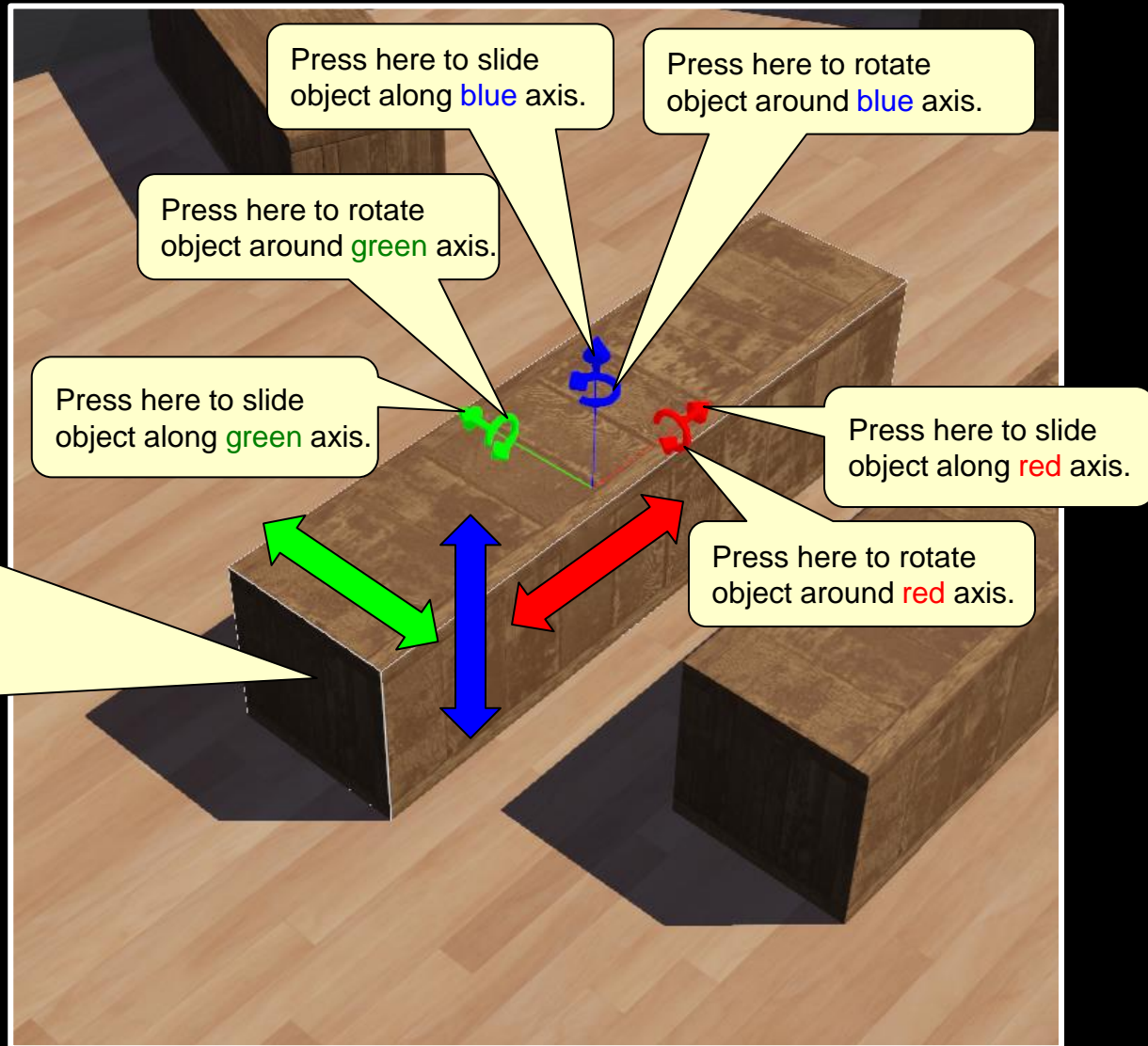
Use the mouse to change viewpoint:

Roll: left press + up/down
Spin: left press + left/right
Translate: right press + up/down/left/right
Zoom: scroll up/down
Select: left click
Menu: right click

View menu has more options



Webots Interface — Moving World Objects

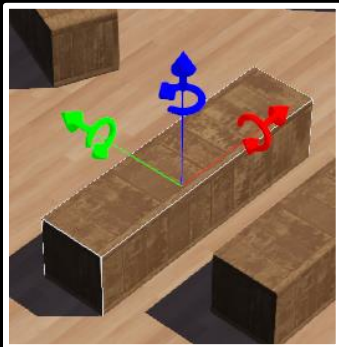
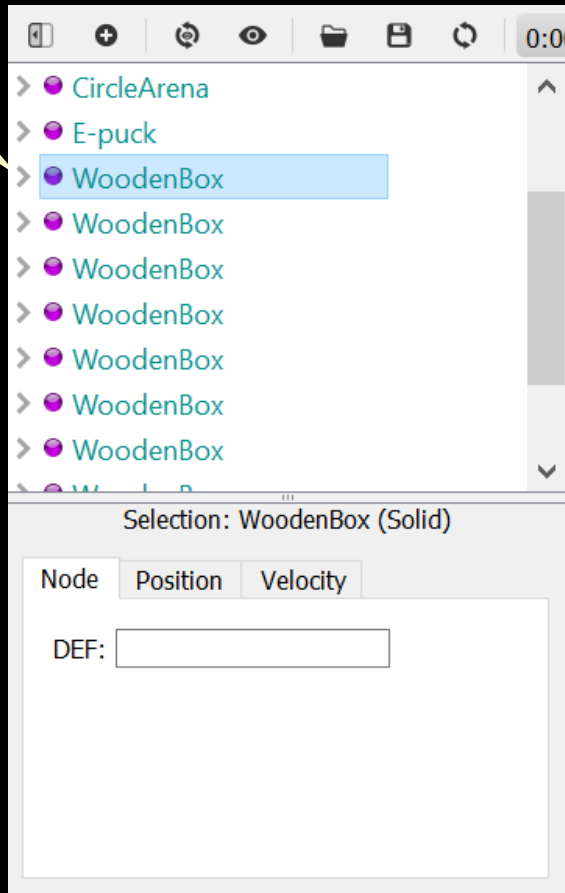


Click anywhere on object to select it, then press and hold left mouse button on one of 6 places to change object.

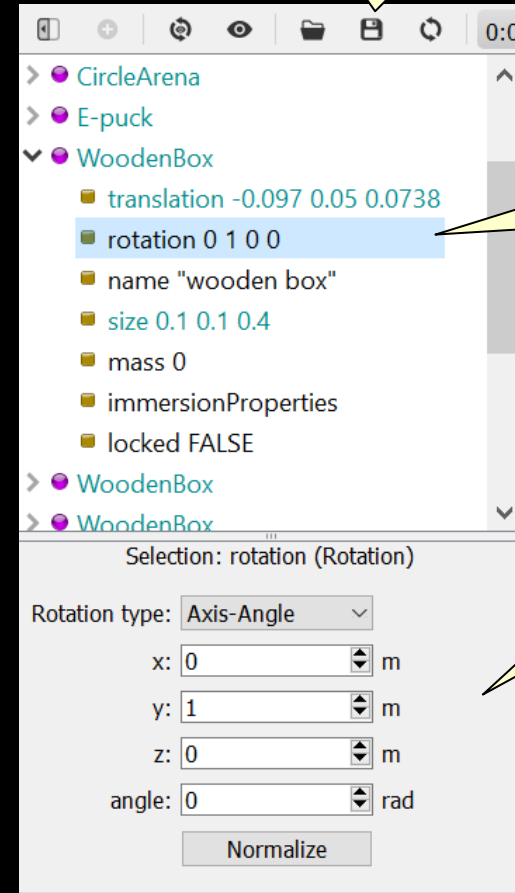
Webots Interface — Editing World Objects

1. Click on the > to expand or collapse one object's attribute view.

Objects are all listed here in the **Scene Tree**.



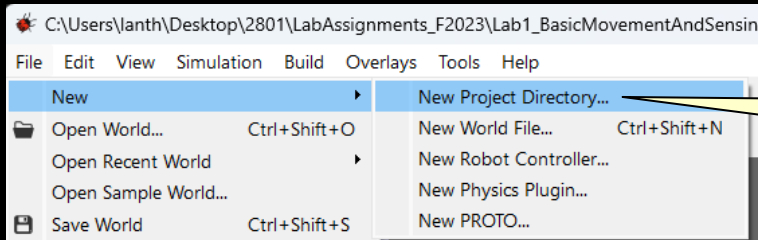
4. Save your changes.



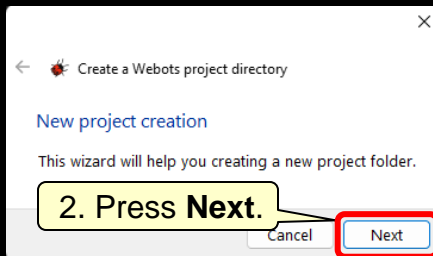
2. Select an attribute.

3. Change its values.

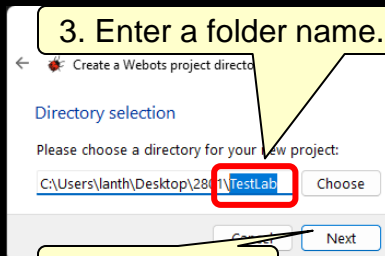
Webots Interface — Making New World



1. Select **New Project Directory...** from the **File/New** menu

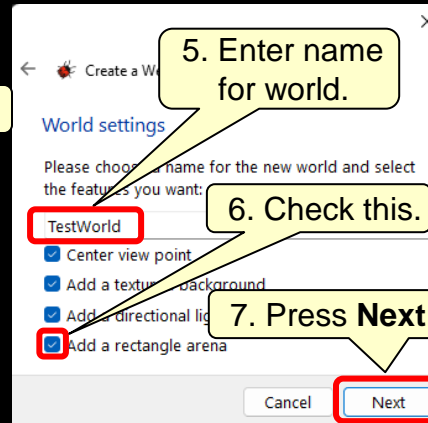


2. Press **Next**.



3. Enter a folder name.

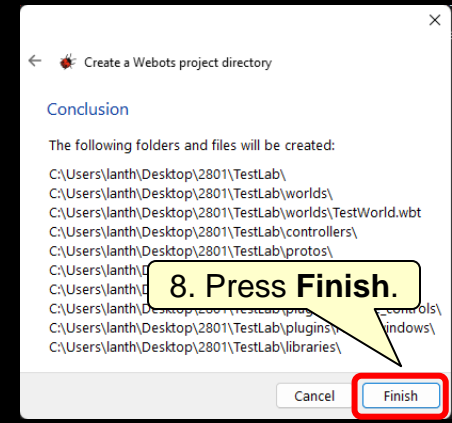
4. Press **Next**.



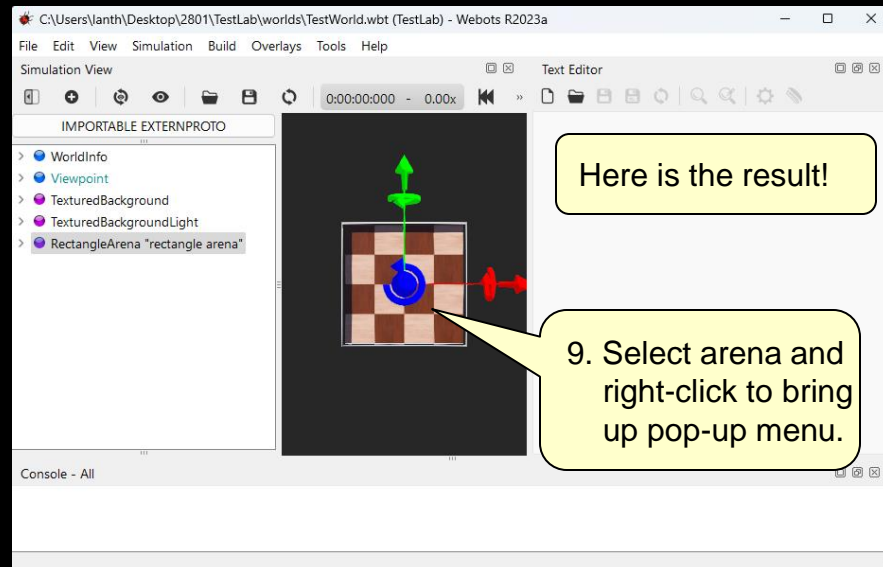
5. Enter name for world.

6. Check this.

7. Press **Next**.



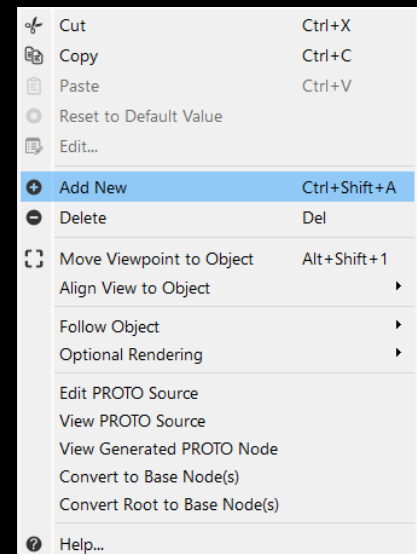
8. Press **Finish**.



Here is the result!

9. Select arena and right-click to bring up pop-up menu.

10. Select **Add New** to add objects and robots. The menu can take up to 10 seconds to appear!

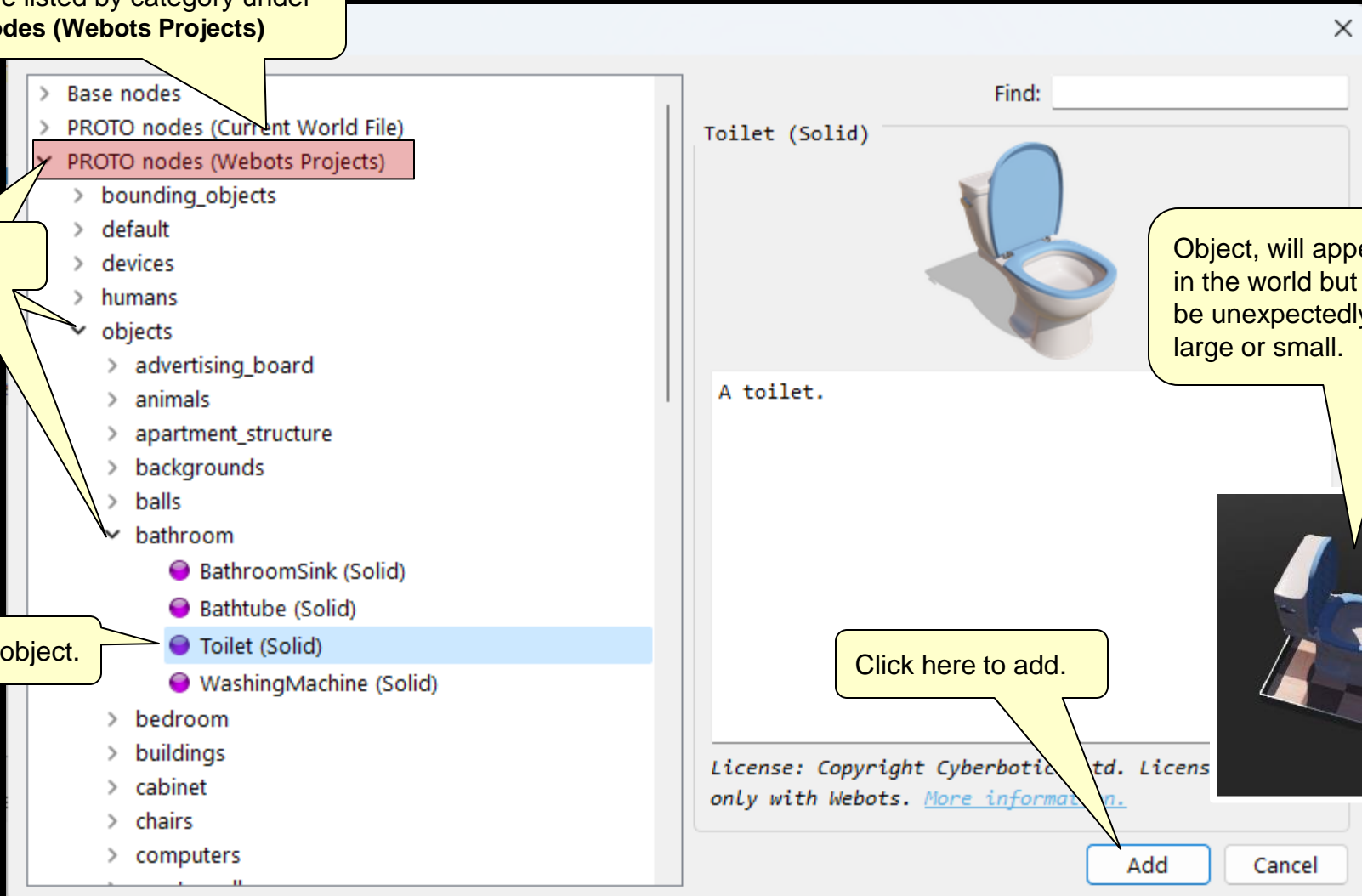


Webots Interface – Adding an object

Objects are listed by category under PROTO nodes (Webots Projects)

Press > to expand.

Select object.



Webots Interface – Resizing an Object

The screenshot displays the Webots R2023a software interface. The main window shows a 3D simulation of a room with a checkered floor, a toilet, and a robot. The left sidebar contains the 'IMPORTABLE EXTERNPROTO' list, where 'RectangleArena' is selected. Below this, the 'floorSize' property is highlighted, and its value is set to '3 5'. The bottom of the sidebar shows the 'Selection: floorSize (Vector2)' with input fields for 'x: 3' and 'y: 5'. The top of the window shows the 'Simulation View' with a toolbar and a status bar indicating '0:00:00:000 - 0.00x'. A red box at the bottom right contains the text 'Don't forget to SAVE your world after making your changes.'

2. Select **floorSize** for RectangleArena.

3. Increase or decrease **x & y** size in meters.

1. Click on floor to select it

Some objects, such as the toilet, are fixed size and cannot be grown nor shrunk.

4. Floor will grow immediately.

Don't forget to **SAVE** your world after making your changes.

Webots Interface — Adding a Robot

Add a node

- Base nodes
- USE
- PROTO nodes (Current Project)
- ▼ PROTO nodes (Webots Projects)
 - bounding_objects
 - default
 - humans
 - objects
 - ▼ robots
 - a4
 - abb
 - adept
 - bluebotics
 - ▼ boston_dynamics
 - atlas
 - ▼ spot
 - Spot (Robot)
 - clearpath

Find:

(Robot)

The "Spot" is a dog robot developed by Boston Dynamics. Original model by Greg McKechnie.

Documentation: <https://www.cylindric.com/spot>

License: Creative Commons 4.0

[More information.](#)

Add **Import...** **Cancel**

Black screen.

Sometimes robot camera displays comes up. Drag here to resize smaller.

Robot's camera display is smaller now.

Don't forget to **SAVE** your world after adding your robot.

Webots Interface – Controller Code

controller attribute specifies program that robot will run.

Robot loads with sample code and may start moving if simulation is running.

Click here to select a different program to run on the robot.

Click here to see the code in the editor.

Here is the code that the robot is running.

```
C:\Users\lanth\Desktop\2801\TestLab\worlds\TestWorld.wbt (TestLab) - Webots R2023a
File Edit View Simulation Build Overlays Tools Help
Simulation View
0:00:01:408 - 0.00x
IMPORTABLE EXTERNS
Spot "Spot"
  translation 0.0369 0.04
  rotation -0.0956 -0.989 0.10
  name "Spot"
  model "Boston Dynamics Spot"
  controller "spot_moving_demo"
Selection: controller (String)
spot_moving_demo
Select... Edit
#include <webots/camera.h>
22 #include <webots/device.h>
23 #include <webots/led.h>
24 #include <webots/motor.h>
25 #include <webots/robot.h>
26
27 #include <math.h>
28 #include <stdio.h>
29 #include <stdlib.h>
30
31 #define NUMBER_OF_LEDS 8
32 #define NUMBER_OF_JOINTS 12
33 #define NUMBER_OF_CAMERAS 5
34
35 // Initialize the robot's information
36 static WbDeviceTag motors[NUMBER_OF_JOINTS];
37 static const char *motor_names[NUMBER_OF_JOINTS] =
38   "front left shoulder abduction motor", "front le
39   "front right shoulder abduction motor", "front ri
40   "rear left shoulder abduction motor", "rear lef
41   "rear right shoulder abduction motor", "rear rig
42 static WbDeviceTag cameras[NUMBER_OF_CAMERAS];
43 static const char *camera_names[NUMBER_OF_CAMERAS]
```


Webots Interface – Your Controller

1. Select **New Robot Controller...** from the **File/New** menu

2. Press **Next.**

3. Select **Java.**

4. Press **Next.**

5. Enter program name (i.e., Java class).

6. Press **Next.**

7. Press **Finish.**

8. Select controller attribute for this robot.

9. Press **Select...** to change controller program for this robot, then choose from dialog box.

10. Choose from dialog box.

11. Press **Edit** to have program appear in text editor.

12. Editor will show a new controller template program now.

13. Close the old controller, or keep it open as a reference. It is not being used by the robot.

Webots Interface – Editing Code

The screenshot displays the Webots R2023a software interface. The main window is titled "TestWorld.wbt (TestLab) - Webots R2023a". It features a menu bar with "File", "Edit", "View", "Tools", "Help", and "Webots". Below the menu bar is a toolbar with icons for file operations (new, open, save, save as, print), search, and compilation (compile, run, reset). The central area is the code editor, showing a Java file named "TestController.java". The code includes a package declaration, imports, a class declaration, and a main method. A console window at the bottom shows the output of the compilation process, including the command "javac -Xlint -classpath C:\Program Files\Webots\lib\controller\java\Controller.jar;. TestController.java" and an error message: "TestController.java:11: error: ';' expected".

Press here (or use Cntrl-S) to save your code.

Press here to compile and load your code onto the robot.

Press here to create a new source code .java file.

Press here to open other source code .java files.

ALWAYS put your name and student number at the top of your programs.

Double-click on error line to go there in the editor.

Compile errors appear in the console window. If a compile error occurs, the code is NOT loaded onto the robot, so the robot is running its previous code. Once compiled and loaded, press **Reset** in the dialog box that appears. Press the play button in the simulation view to run it.,

```
File Edit View Tools Help Webots
C:\Users\lan\OneDrive\Desktop\2001\testlab\controller\
TestController.java x
Auth: Mark Lanthier (SN: 100100100)
import com.robotics.webots.controller.Robot;
// your controller.
// to initialize and how to run your controller.
public class TestController {
    public static void main(String[] args) {
        // create the Robot instance.
        Robot robot = new Robot()

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

        while (robot.step(timeStep) != -1) {
            // Sense
            // Think
            // React
        }
    }
}
```

Console - All

```
javac -Xlint -classpath C:\Program Files\Webots\lib\controller\java\Controller.jar;. TestController.java
TestController.java:11: error: ';' expected
    Robot robot = new Robot()
                        ^
1 error
Nothing to be done for build targets.
```



**Start the
Lab ...**