Chapter 6

Supervisor Control in Webots

Objectives:

By the end of this chapter, students will be able to:

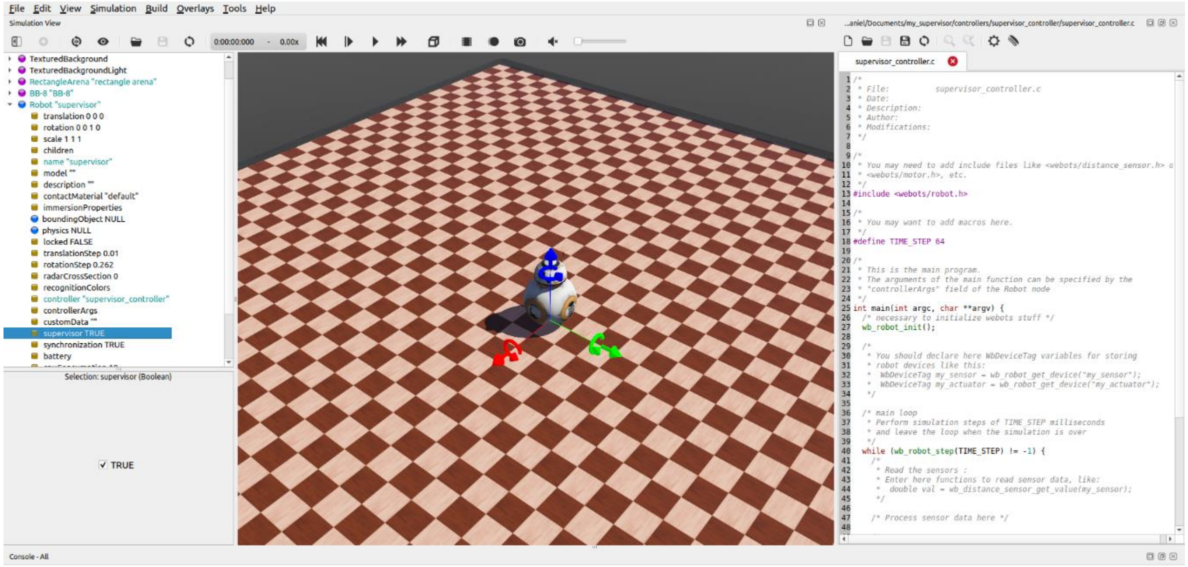
- Explain the role of a supervisor in Webots simulations.
- Differentiate between a robot controller and a supervisor controller.
- Use Supervisor API functions to monitor, reset, and modify simulations.
- Implement a supervisor to track robot behavior and control the simulation environment.

Software and Tools:

- Webots
- Computer with at least 8GB RAM, Mouse/keyboard, and Internet connection to access tutorials
- Text editor/IDE (built-in Webots editor is sufficient)
- Programming language support: C/C++/Java/Python/MATLAB

6.1 THEORY

In Webots, a Supervisor is a special type of controller that has access to the entire simulation environment, not just a single robot. Unlike a robot controller, which is limited to the sensors and actuators of its associated robot, a supervisor can manipulate all objects in the scene tree.



TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
BB-8 "BB-8"
Robot "supervisor"
translation 0 0 0
rotation 0 0 1 0
scale 1 1 1
children
name "supervisor"
model ""
description ""
contactMaterial "default"
immersionProperties
boundingObject NULL
physics NULL
locked FALSE
translationStep 0.01
rotationStep 0.262
radarCrossSection 0
recognitionColors
controller "supervisor_controller"
controllerArgs
customData ""
supervisor TRUE
synchronization TRUE
battery

Selection: supervisor (Boolean)

☑ TRUE

...aniel/Documents/my_supervisor/controllers/supervisor_controller/supervisor_controller.c

```c
/*
 * File:          supervisor_controller.c
 * Date:
 * Description:
 * Author:
 * Modifications:
 */

/*
 * You may need to add include files like <webots/distance_sensor.h> o
 * <webots/motor.h>, etc.
 */
#include <webots/robot.h>

/*
 * You may want to add macros here.
 */
#define TIME_STEP 64

/*
 * This is the main program.
 * The arguments of the main function can be specified by the
 * "controllerArgs" field of the Robot node
 */
int main(int argc, char **argv) {
  /* necessary to initialize webots stuff */
  wb_robot_init();

  /*
   * You should declare here WbDeviceTag variables for storing
   * robot devices like this:
   *  WbDeviceTag my_sensor = wb_robot_get_device("my_sensor");
   *  WbDeviceTag my_actuator = wb_robot_get_device("my_actuator");
   */

  /* main loop
   * Perform simulation steps of TIME_STEP milliseconds
   * and leave the loop when the simulation is over
   */
  while (wb_robot_step(TIME_STEP) != -1) {
    /*
     * Read the sensors :
     * Enter here functions to read sensor data, like:
     *  double val = wb_distance_sensor_get_value(my_sensor);
     */

    /* Process sensor data here */

```

Figure 6. Resulting Environment in Supervisor Webots

The Supervisor API provides functions to monitor robot states, reset simulations, add or remove objects, and record data. This makes it essential for tasks such as evaluating experiments, automating tests, or managing complex multi-robot environments.

Supervisors operate with elevated privileges, allowing them to read and modify node fields in the scene tree. For example, a supervisor can check the position of a robot, reset it to its initial pose, or even create a new robot during runtime. This capability is especially useful for reinforcement learning, evolutionary algorithms, or repeated trials where conditions must be consistent.

While powerful, supervisors should be used carefully: excessive modifications can reduce simulation realism. Typically, robot controllers are used for behaviour programming, and supervisors are reserved for global management and monitoring.

Here are the **key concepts** you need to understand:

- **Supervisor**: A controller with global access to the simulation environment.
- **Scene Tree Access**: Ability to read and modify node fields.
- **Reset/Control Functions**: Functions to reset or terminate simulations.
- **Monitoring**: Track robot positions, collisions, and performance.
- **Global Management**: Essential for experiments, automation, and multi-robot systems.

These concepts form the foundation for building physically accurate and computationally efficient robotic simulations.

6.2 PRACTICE GUIDE: SUPERVISOR IN WEBOTS

Now it is time to apply the theory in practice. In this section, students will create a supervisor to monitor and control the simulation environment.

Practice Objectives
Through this tutorial, you will learn how to:

1. Create a new controller and assign it as a supervisor in the world file.
2. Use Supervisor API functions to retrieve the position of a robot.
3. Program the supervisor to log robot coordinates during simulation.
4. Add functionality to reset the robot to its starting position when it collides with an obstacle.
5. Test the supervisor by running the simulation multiple times.

Please access and follow the steps in the tutorial at the link below:

- Webots Tutorial 8: The Supervisor or https://cyberbotics.com/doc/guide/tutorial-8-the-supervisor

Experiment

To deepen your understanding, try the following scenarios and observe the results:

1. Extend the supervisor to monitor multiple robots at once.
2. Add a condition where the supervisor ends the simulation if a robot reaches a certain point.
3. Implement a function for the supervisor to insert a new obstacle during runtime.
4. Modify the supervisor to reset the simulation after a fixed number of steps.

Exercises

Complete the following tasks to validate your skills:

1. Write a supervisor that tracks the number of collisions a robot makes.
2. Modify the supervisor to respawn the robot at a random location after each reset.
3. Extend the supervisor to record robot positions into a file for later analysis.
4. Run a simulation with at least two robots: one controlled by a robot controller and one monitored by a supervisor. Compare their roles.
5. Propose a real-world robotics experiment that would benefit from using a supervisor.

6.3 REFLECTION QUESTIONS

1) What is the main difference between a robot controller and a supervisor?
2) In what scenarios would using a supervisor be essential?
3) How does accessing the scene tree benefit a supervisor?
4) What are potential drawbacks of relying too much on supervisors?

In this chapter, students learned about the Supervisor in Webots and its role in controlling simulations at a global level. They practiced creating a supervisor to log data, reset robots, and manipulate the environment. By understanding the difference between robot controllers and supervisors, students can design experiments that combine autonomous robot behaviors with global monitoring and management. These practices prepare students for advanced simulation tasks and complex robotics projects.