Constructive AI – Using Webots

Matthew Lewis
m.lewis4@herts.ac.uk

General hint: The *Webots User Guide* and *Webots Reference Manual* are available in Webots under the Help menu.

Part 1: Practical Introduction to Webots (v7.4.3)

1. Open Webots in Windows from the Start menu.
   (Note: the Start menu should give the correct link. If you open Webots another way, e.g. command line, make sure you run the .bat file – not the .exe – otherwise the environment will not be configured correctly and you may get errors when it tries to use the wrong version of Python).

2. When you first use Webots, you need to make a directory to save your files. This should be somewhere you can edit them, e.g. on your U: drive.

   In the menu, go to Wizards → New Project Directory. Enter the directory name when prompted, e.g. U:\webots2017

3. A new World called new.wbt is created in the projects directory (in Webots a "World" includes both the environment and any robots) this includes a simple walled 1m × 1m arena and lighting.
   If you wish, you can save a copy of the world the with File → Save World As… or create a new world with File → New World.

4. Practice manipulating your view of the world by dragging with the mouse on the view area, and zooming with the mouse wheel.
   Change the size of the RectangleArena by changing its floorSize property in the Scene Tree on the left.
   You can change the position of the arena by changing its translation in the Scene Tree on the left, or by pressing *shift and dragging* with the mouse; *shift and mouse wheel* changes the y coordinate (height) of the selected object.

   Tip: After changing a property of an object you can reset it back to the default value by clicking the ❋ icon.

5. Try running the simulation: click on the "Play" icon. The timer at the top of the window should count up, and the simulation speed indicator next to it should say 1.00× (or something close to 1). Of course, since the world contains only a static arena, nothing will change in the world view.

Now, stop the simulation by clicking the “Pause” icon. Then reset the world to its initial state by clicking the Revert icon – the timer will reset to zero and the world file will be reloaded, returning to its initial state.

6. A newly created World includes a floor and a light by default. We delete the default directional light (similar to light from a distant source like the Sun) and replace it with a more point source (similar to light from a nearby source like a lamp).

   Click on the DirectionalLight in the Scene Tree in the left sidebar to select it. With the light selected, click on the delete icon in the icon bar or press the “Delete” button on the keyboard. The arena should go black – because there is no light.

7. Objects in the World are organised in a hierarchical tree-like structure (like the DOM of a webpage) in which elements are nested inside a “parent” element (which can then be nested inside a grand-parent element). So to insert a new object, you need to specify the parent element.

   The light should have the “Background” element as its parent. Select the Background by clicking on the “Background” element in the Scene Tree on the left. Now click on the Add New Object icon in the icon bar . A pop-up window will appear showing a variety of objects that con be added to the World. Expand the tree to New Node → PointLight, and click “Add”. The arena should become visible again, with a different lighting effect

   You can now change the properties of the new light (e.g. colour, position, brightness) by changing the values in the Scene Tree on the left.

8. Now you should **save your World**.
   It is very important that you do this regularly, or you risk losing your recent work. You can either do this in the menu, from the icon bar, or with the shortcut combination Ctrl+Shift+S.

9. Now insert a robot. We use the e-puck robot, which is a real physical robot with a high-quality model available for Webots. Insert this from PROTO (Webots) → robots → e-puck → Epuck (Differential Wheels). It will appear at position (0,0).

10. Save your world.

11. You will probably need the epuck **ground sensor**. Install this in the correct slot by expanding the EPuck entry in the Scene Tree. Select the groundSensorSlot (at the bottom), and click on the Add New Object icon . Insert PROTO (Webots) → robots → e-puck → Epuck_GroundSensorsModule.

12. Save your World. (We will stop repeating this now, but you should keep saving as you work.)

13. In Webots, robots are run by **controllers**. You will need to create a controller and then assign it to your robot.

    To create a controller, go to Wizards → New Robot Controller.

    Choose “Python” or “C” as the language for the controller according to your preference.

    You will need to provide a filename, for example “test_controller”. The file will be saved in the area created in step 2, i.e. in U:\webots2017\controllers. On the final step of the wizard, make sure the option to Open the file in the text editor is selected.

    Controller code and Worlds are saved independently: when you save your code, you are not saving your world. Be careful not to lose your work by saving one and not the other.

    (Note that languages other than Python and C are theoretically available, however there are configuration issues with these. For this project we strongly recommend that you use Python or C, and we will not be able to support you if you choose an alternative language.)

    The controller as created includes minimal framework code. This source code will be displayed in the right-hand panel. Have a look at this to get an idea of how the code works.

14. **If you are using Python:** Webots 7.4 uses Python version 2.7. If you are used to Python 3 there are some small differences in the syntax.

15. **If you are using C:** Because C is a compiled language, before assigning a controller, you will need to compile the code (i.e. create a binary executable from the source code) otherwise the controller won’t appear as an option when you come to add it. Compile the code by clicking on the cog icon ⚙ above the source code in the right panel. Confirm that you want to revert the simulation (note that reverting the simulation throws away any changes you made to the World since the last save – this is why it is important to save regularly).

    The compilation output appears at the bottom of the window.

16. You can now assign the controller to the robot.

    Expand the EPuck in the Scene Tree. Select “controller”, and the option editor will appear at the bottom to change this. Click the Select… button. In the pop-up window select your controller (test_controller) and click OK.

17. (Save your World...)

18. If you run your simulation (click on the “play” icon in the icon bar) you will notice that the robot doesn’t move. The timer should start ticking up though. This is because the default controller does nothing. You should, however, see a message in the console at the bottom of the window saying “INFO: my_controller: Starting ...” (if not, check to see if you have done the previous steps correctly). Stop and reset the simulation (click the Pause icon, then the Reload icon).

19. To create a controller that does something. Follow the instructions in **CAI – Using Python in Webots** or **CAI – Using C in Webots**.
20. Run the code and watch the robot move. Change the starting angle of the robot, so it moves in a different direction.
21. It might be useful to “follow” the robot with the camera to see how it behaves. To do this, stop and reset the simulation. Now zoom in on the robot. Now, in the *Viewpoint* object in the scene tree, select the *follow* option and give it the value equal to the name of the robot. The name of the EPuck is given by its “name” property – the default is “e-puck”, and it might be useful to change this if you have more than one robot.

    Save and run the simulation. The camera should follow the robot.

    Try inserting different objects in front of the robot. What happens when the robot hits them? Can you change any of their properties to change what happens when the robot hits them?

Troubleshooting

If you get errors when you try to run Python or compile your C source, it may be because you started Webots without setting up a suitable environment. This might happen if started from the command line or by double-clicking on a Webots world file in the File Manager.

Instead, open Webots from the Start Menu, or Webots.bat from the command line (NOT webots.exe).