

EyeSim Guide

Developed by Aditya Patil 2026

Purpose of EyeSim

EyeSim is a simulator **designed for running the same programs** that run on the physical robots, **removing the need** to use a real robot every time you want to test a program.

Note: The physics in the simulator **will not** match a real robot exactly. Motor behaviour, friction, surface differences and random error all affect a real robot. **Example:** A 90 degree turn on a real robot may not come out as exactly 90 degrees. Get your program working in the simulator first, then tweak it if the real robot behaves differently.

Installing EyeSim

MacOS

Download the **.pkg** file, open it, and follow the prompts. The same file works on both Intel and Apple Silicon Macs, so there is only one download.

Note: MacOS may warn you that the installer is from an unidentified developer. If that happens, right-click the **.pkg** and choose **Open**, then confirm.

MacOS does not ship with a C compiler, so you also need Apple's Command Line Tools to compile your programs. Open the **Terminal** app and run:

```
xcode-select --install
```

Accept the prompt and let it finish. If you already have it, or you have Xcode installed, it will just tell you so. You only ever do this once. The first time you run **gccsim** without it, MacOS will pop up the same installer by itself.

Windows

Windows comes in two versions and you need the matching download, so check which one you have first:

1. Open **Settings** (Windows key + I)
2. Go to **System** -> **About**
3. Look at **System type** under Device specifications

If it says **x64-based processor**, you have the x64 version. If it says **ARM-based processor**, you have the ARM version. Most laptops and desktops are x64. ARM devices are usually newer machines.

x64 (most devices):

- Download **EyeSim-v2.0.0-x64.zip**

- Unzip it, run the installer inside, and follow the prompts

ARM:

- Download [EyeSim-v2.0.0-arm-BETA.zip](#)
- Unzip it, run the installer inside, and follow the prompts

Note: The ARM version is the newest addition to EyeSim and is still being tuned, which is why it is marked BETA. Everything you need for the labs is there, but if you spot something that behaves differently to the x64 version, let us know so it can be fixed.

The installer sets up EyeSim along with everything needed to compile and run your C programs, so there is nothing else to install afterwards.

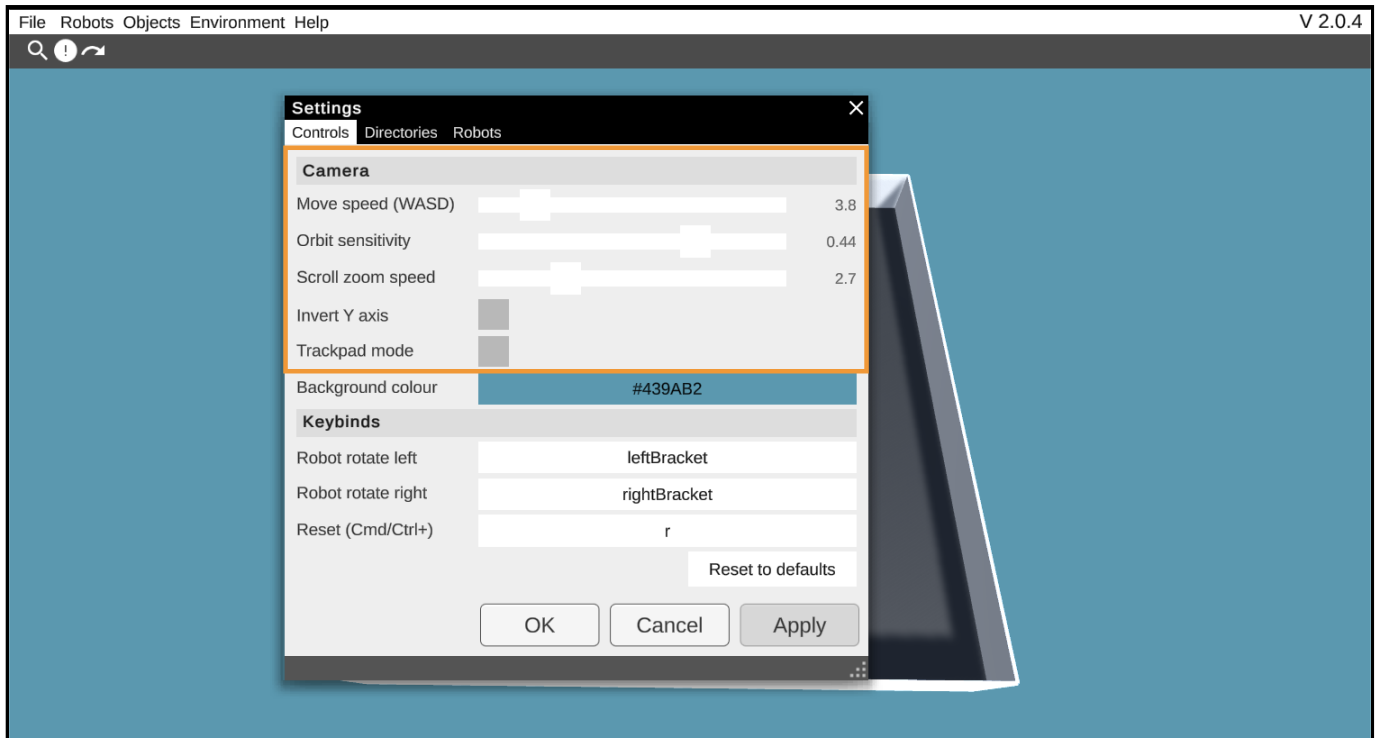
EyeSim interface

The following is what you are greeted with when EyeSim is launched. There is an Options Menu, Quick Options Menu, and the Environment view.



Navigating EyeSim

- To move the camera at its current height, use the **WASD** or **arrow** keys. You can change the Move speed in File -> Settings.
- To change the camera height, use **SHIFT** to move down, and **SPACE** to move up.
- To rotate/orbit the camera, right-click and drag your mouse around. You can change the Orbit speed in File -> Settings. On a trackpad, drag with two fingers to pan.
- You can also zoom in and out by scrolling. You can change the Zoom speed in File -> Settings.
- Press **Command/Control + R** to reset the camera to its starting position.



Options Menu:



The options menu contains **File**, **Robots**, **Objects**, **Environment** and **Help** menus.

File:

- **Settings:** Here you can change a wide range of settings such as:
 - Camera controls
 - Background color
 - Keybinds for certain commands
 - Directories: set the Home directory to your **EyeSimX-Examples** folder and the robot, object, world and simulation directories are filled in for you.
- **Load Sim:** Used to load .sim files
- **Open Terminal:** Opens a terminal in your home folder, ready to compile programs. On Windows this is the EyeSim shell (see below).
- **Exit:** Closes EyeSim.

Robots:

- **Add Robot:** Add a robot in the scene by clicking and dragging onto a floor.
- **Load Robot:** Load a .robi file to import a custom robot.

Objects:

- **Add Object:** Add an object in the scene by clicking and dragging onto a floor.

- *Load Object*: Load a .esObj file to import a custom object.

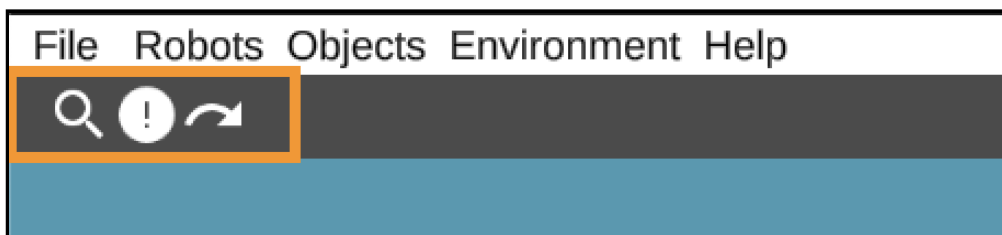
Environment:

- *Add Wall*: Allows you to add a wall. Once selected, click on a floor to place the start point, then drag to place the end point.
- *Add Shape*: Place a cube, cylinder or sphere in the scene, the same way as a wall.
- *Load Environment*: Load .wld files in scene.
- *Create World*: Creates an empty walled world at a width and length you enter (in mm).
- *Save World*: Saves the current floor and walls to a .wld file, including any walls you added, moved or deleted.
- *Reset Environment*: Reloads the current .sim file, putting everything back to its starting state.

Help:

- *Open Console*: Opens up the console to view any logs or error messages.
- *Open LCD Display*: Reopens the LCD display of the selected robot if you closed it.
- *Reset Camera*: Resets the camera to its original position.

Quick Options Menu



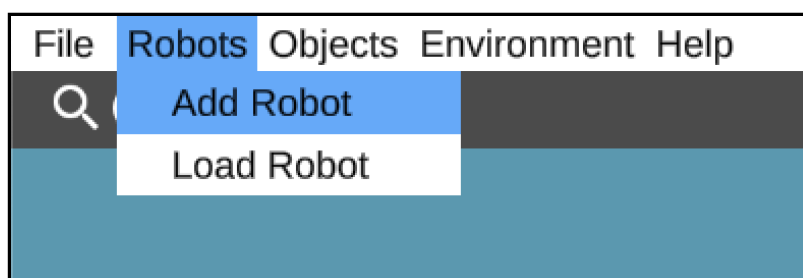
The Quick Options Menu contains the most-used functions for quick access, including pausing and resuming the simulation and a speed button that runs it at double speed. You can hover over each icon for information on what it does.

How to run programs on EyeSim

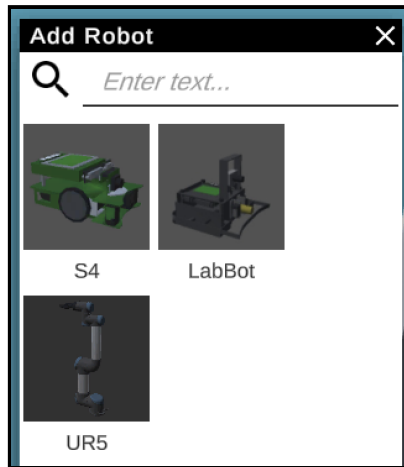
Running programs on EyeSim is meant to replicate running programs on the physical robots: you write a C program using the RoBIOS API library (which provides functions for driving, sensors, the camera, and so on), compile it, and then run it from a terminal while EyeSim is open. A set of example programs is available from the EyeSim website, and working through them is highly recommended.

Let's walk through running a sample program:

1. **Load a robot in the scene:** You can add a robot using the Robots -> Add Robot menu (if you don't already have one).

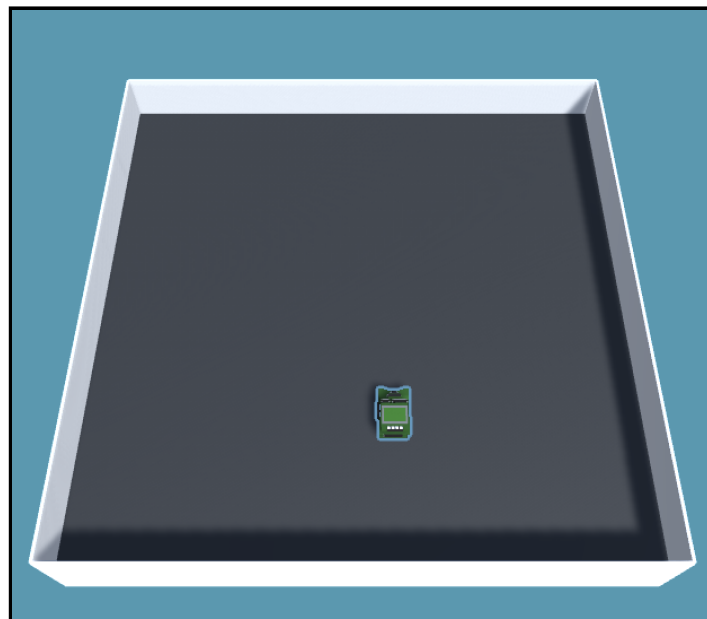


Here, you will see a handful of robots to select from. **Choose the S4** (most examples and labs are written for this robot), and click the floor to place it. You may already have an S4 in the scene, since the default world includes one.



Note: Robots **use different drive types**, so an example written for one drive type will not work on a robot with another. The drive type an example expects is usually noted in the example file itself.

2. **Ensure a robot is selected:** Click a robot to select it. The selected robot is highlighted in its own colour, and it is the robot your program will control. If several robots are in the scene, select the one you want to drive before starting your program.



3. **Compile your C program:** To execute a program, you must first compile your C program. To do so, follow the following based on your operating system:

- a. **MacOS/Linux:**
 - i: Open the **Terminal** app.
 - ii: Navigate to the folder holding your **.c** file (using the **cd** and **ls** commands). For this example, we will use **drive-and-return.c** from the **examples-c/drive** folder:

```

/Users/addy/Downloads [addy@addys-MacBook-Pro] [14:39]
> cd eyesimX-examples
(base)
/Users/addy/Downloads/eyesimX-examples [git::master *] [addy@addys-MacBook-Pro] [14:39]
(base)
/Users/addy/Downloads/eyesimX-examples/examples-c [git::master *] [addy@addys-MacBook-Pro] [14:39]
> cd drive
(base)
/Users/addy/Downloads/eyesimX-examples/examples-c/drive [git::master *] [addy@addys-MacBook-Pro] [14:39]
> ls
dead-reckoning.c  drive-and-return.c  drive-xy.c  encoder.c  speed.c
diff-drive.c      drive-cam.c         drive.c      Makefile   square.c
Diff.sim          drive-vw.c         drivedemo.c  menu-vw.c  Square.sim
(base)
/Users/addy/Downloads/eyesimX-examples/examples-c/drive [git::master *] [addy@addys-MacBook-Pro] [14:39]
>

```

- iii: Compile it with `gccsim filename.c -o program-name`. Here: `gccsim` is the EyeSim compiler command, which builds your program against the RoBIOS API `-o program-name` is whatever you would like to call the compiled program (no extension required).
- iv: Once completed, a new file will appear in the folder you ran this command in.

```

/Users/addy/Downloads/eyesimX-examples/examples-c/drive [git::master *] [addy@addys-MacBook-Pro] [14:47]
> gccsim drive-and-return.c -o drive-and-return
(base)
/Users/addy/Downloads/eyesimX-examples/examples-c/drive [git::master *] [addy@addys-MacBook-Pro] [14:47]
> ls
dead-reckoning.c  drive-and-return  drive-vw.c  drivedemo.c  menu-vw.c  Square.sim
diff-drive.c      drive-and-return.c  drive-xy.c  encoder.c     speed.c
Diff.sim          drive-cam.c        drive.c     Makefile     square.c
(base)
/Users/addy/Downloads/eyesimX-examples/examples-c/drive [git::master *] [addy@addys-MacBook-Pro] [14:47]
>

```

Then to execute, run: `./[program-name]`:

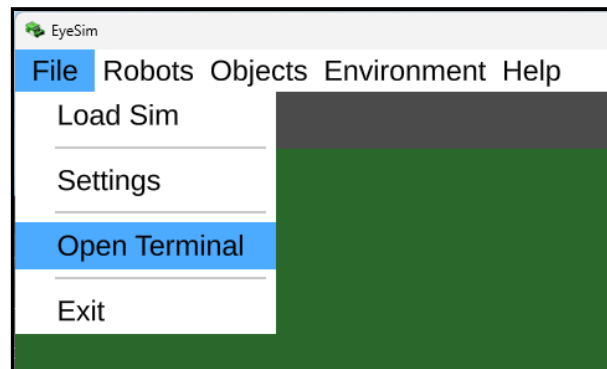
- v: If you look back at EyeSim simulator, the desired program should execute.



If your program calls any LCD or MENU function, a small window appears to simulate the LCD display on the physical robots. You can use it to show information such as sensor readings while your program runs.

b. Windows:

- i: Open the EyeSim terminal by going to File -> Open Terminal. On Windows this opens the EyeSim shell, which is where `gccsim` is installed.



- ii: Navigate to the folder holding your `.c` file (using the `cd` and `ls` commands). For this example, we will use `square.c` from the `EyeSimX/examples-c/drive` folder, which you can download from the EyeSim website. I have placed my examples folder in my Downloads folder:

```
File Robots Objects Environment Help

eyesim@DESKTOP-JV9KJIO: /mnt/c/Users/adi$ cd Downloads/
eyesim@DESKTOP-JV9KJIO: /mnt/c/Users/adi/Downloads$ cd EyeSimX/examples-c
eyesim@DESKTOP-JV9KJIO: /mnt/c/Users/adi/Downloads/EyeSimX/examples-c$ ls
All.sh      basic      carolo-sim  lc         myrobot    portlib    starman    swarm-random
Makefile    cam        drive      lidar      neural-digits  pioneer    subwaring  terrain
Ackermann   coast     formula-one manipulator neural-drive  radio      swarm-follower
audio       cam       genetic     path       arm        random     swarm-group

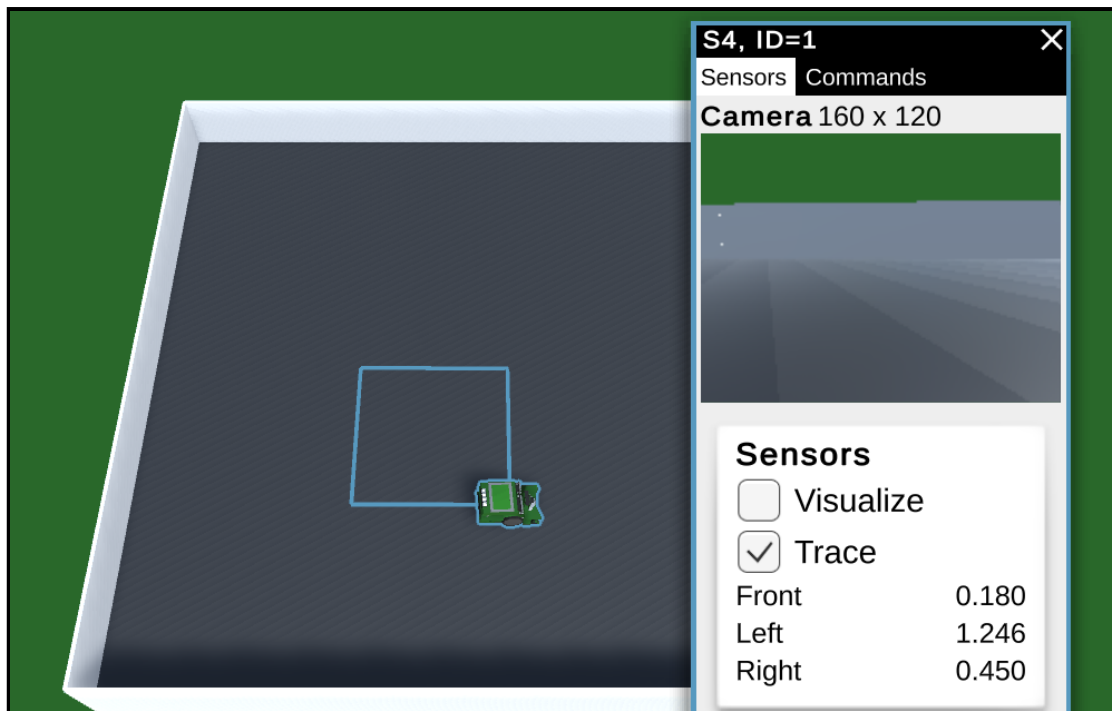
eyesim@DESKTOP-JV9KJIO: /mnt/c/Users/adi/Downloads/EyeSimX/examples-c$ cd drive/
eyesim@DESKTOP-JV9KJIO: /mnt/c/Users/adi/Downloads/EyeSimX/examples-c/drive$ ls
Diff.sim  Square.sim  dead-reckoning.c  drive-and-return.c  drive-vw.c  drive.c  encoder.c  speed.c  square.x
Makefile  a.out      diff-drive.c      drive-cam.c        drive-xy.c  drivedemo.c  menu-vw.c  square.c

eyesim@DESKTOP-JV9KJIO: /mnt/c/Users/adi/Downloads/EyeSimX/examples-c/drive$ |
```

- iii: Compile it with `gccsim filename.c -o program-name`. Here: `gccsim` is the EyeSim compiler command, which builds your program against the RoBIOS API `-o program-name` is whatever you would like to call the compiled program (no extension required).
- iv: Once completed, a new file will appear in the folder you ran this command in. Run it with `./program-name`:

```
eyesim@DESKTOP-JV9KJIO: /mnt/c/Users/adi/Downloads/EyeSimX/examples-c/drive$ gccsim square.c -o square.x
eyesim@DESKTOP-JV9KJIO: /mnt/c/Users/adi/Downloads/EyeSimX/examples-c/drive$ ./square.x
Connection established. Handshaking...
Handshake complete. Waiting for server ready...
Server ready. Beginning control.
```

- v: If you look back at EyeSim simulator, the desired program should execute.



EyeSim Files

EyeSim accepts the following file types, each with its own purpose:

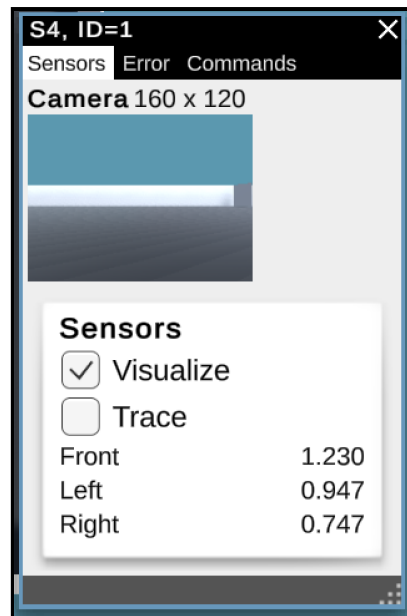
- **.robi files**: Describe a robot (its body, drive type and sensors) so it can be loaded into EyeSim.
- **.esobj files**: Describe an object that can be placed in the scene.
- **.wld files**: Contain the environment: the floor size and the walls in it.
- **.sim files**: Tie everything together, naming the world to load, which robots and objects to place, where to place them, and which program each robot should run.

You will find many examples of all of these in the EyeSimX folder.

Robot Parameters

Each robot in EyeSim has its own Inspector window and LCD display. A robot's LCD display only appears once its program calls an LCD or MENU function.

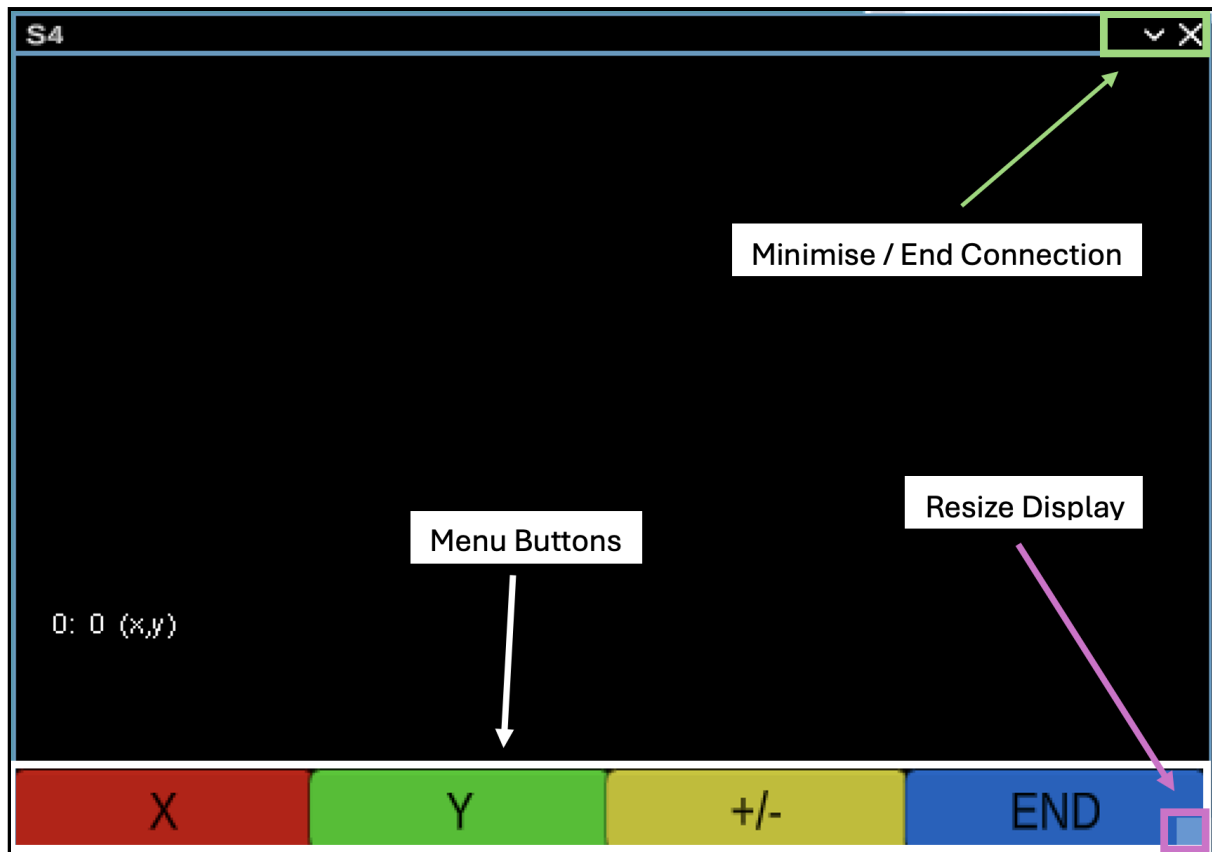
Inspector Window



Double-clicking a robot opens its Inspector panel. The **Inspector panel** contains:

- Name of Robot and ID as the title
- A mini camera view (if the robot has a camera)
- Sensor Parameters:
 - *Visualise*: When checked, each time your program reads a PSD sensor, that sensor's beam is drawn on-screen for a few seconds. Useful for debugging.
 - *Trace*: Draws a line on the floor showing where the robot has driven. Note: unchecking and rechecking it clears the previous trace!
- PSD Sensors: Lists the robot's PSD sensors with their names and current readings (in metres). Note that the RoBIOS **PSDGet** function returns millimetres.

LCD Display



Most robots have some form of display, which you can print text on or draw to. The LCD display only appears in the simulator once a program calls an LCD or MENU function from the RoBIOS library. You can use these functions to print information such as PSD sensor values, or to show the robot's camera image. The default size is 480x320 pixels.

Features:

- You can resize or minimise the display to free up space to see the environment.
- The X button (top right) ends the robot's connection to its C program.
- A coloured border shows which robot the display belongs to.