Debugging Help

Abstract

Debugging is an essential part of every computer science project. The AVL is no exception, however given that it runs inside of the Unity engine this means the typical command line is not available as an option for debugging. As such, this document will explain some of the basic means to debugging your code in the AVL.

While the TA is available to help debug difficult problems that you may have, you are expected to try and solve the problem yourself first. Before you ask the TA for help, ask yourself these basic debugging questions:

1. Did I check the documentation?
2. Did I try Googling my problem?
3. Did I try experimenting with the code?
4. Did I try looking at the AVL source code I'm interacting with?

If you answered yes to all these questions and you are still having issues, then you should email the TA with your questions. Please send your C# code files in your email when you ask your question, and do not include any files from the AVL source code. If necessary, a meeting can be scheduled if the issue is particularly complex. Otherwise, the TA can usually answer your question within 24 hours via email.

Console Output

The quickest and easiest means of debugging code is by printing variable values to the console. This can be achieved using the Debug.Log(…) command, as shown below:

```
public class Example_Task
{
    public void Execute() {
        // Our variable we want to print
        float x = 10f;

        // Print the variable to the console
        Debug.Log(x);
    }
}
```

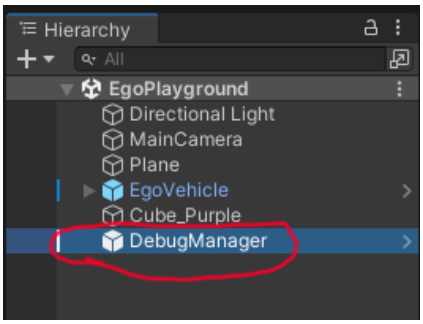
You can also embed the variable into a string to make it easier to read in the console:

```
public class Example_Task
{
    public void Execute() {
        float x = 10f;

        Debug.Log($"X value: {x}");
    }
}
```

Debug Manager

Every scene created for the AVL features the Debug Manager object, which has some functionalities that students may find helpful while they debug their code. The Debug Manager is an object within the Unity hierarchy, as shown below:

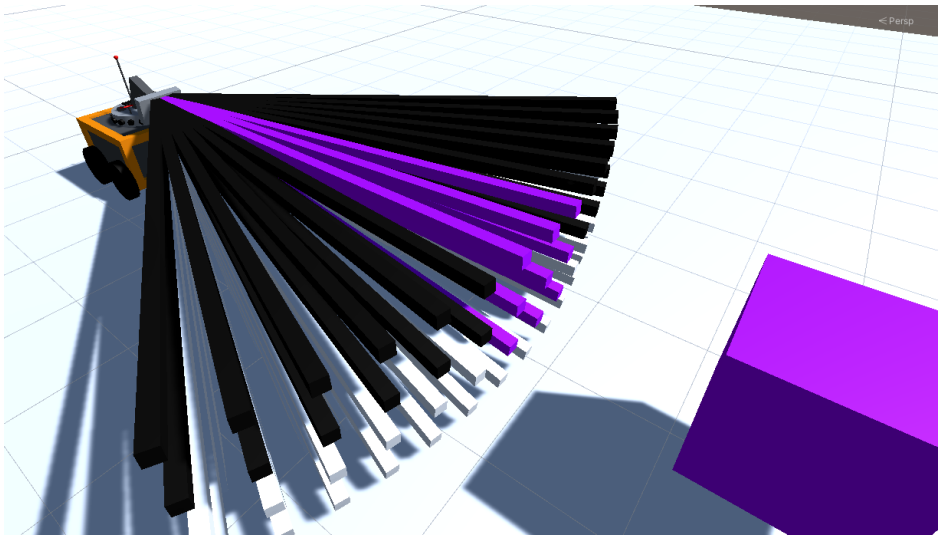


If the Debug Manager is selected via a left-click in the hierarchy, then it's properties will be displayed in the Inspector window. These properties are divided into distinct subsections, however you don't need to worry about the first section titled "References". The rest of the subsections are described below.

Camera Debugging

The Camera Debugging section features four configuration options, however the one titled "Show Camera Rays" is the main option you will be using. The option is shown below:

If you click on this checkbox before you run the simulation, then when the simulation starts the AVL will display rays from the Ego vehicle's camera. These rays show what each pixel is seeing, including the color perceived by that pixel's ray. The rays can be seen in the following image:

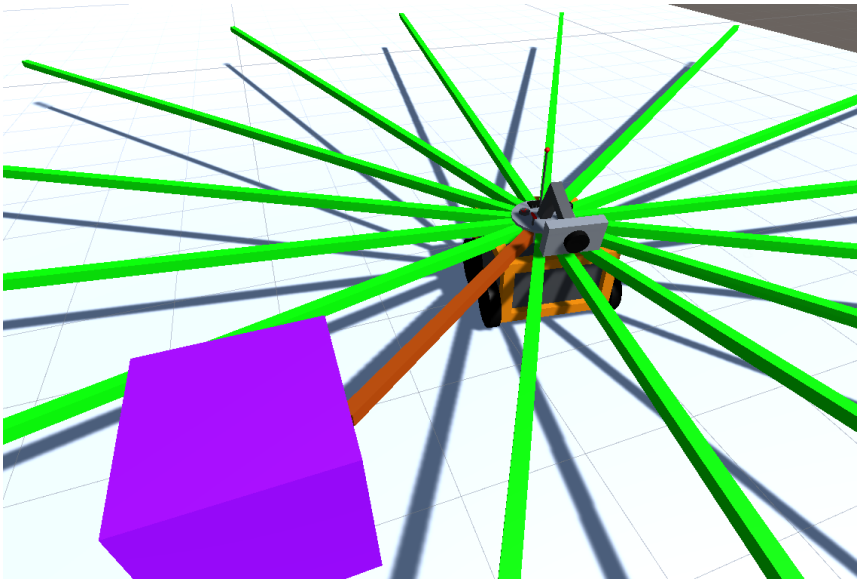


The other options in this subsection modify how these rays are rendered.

Lidar Debugging

There is only a single checkbox option for Lidar Debugging, as shown below:

When this checkbox is clicked before running the simulation, then when the simulation starts rays will be created that show what the lidar is perceiving. An example of these rays is shown below:



When a lidar does not detect an object, it's associated ray will be green. When an object is detected, the lidar ray will change it's color to reflect how close that object is detected to be. The closer an object is to the Ego vehicle, the more red it's color will be.

Important: Lidar rays will *only* be rendered if the lidar sensor is enabled. See the PDF file associated with each Assignment to find out if the lidar sensor is enabled in that scenario.

Assignment-1 Debugging

This subsection contains debugging options which are specific only to the Assignment 1 scene file. If Unity does not have the Assignment 1 scene file opened, then these options will do nothing.

There is only one option available:

- Activate Boat: When this button is pressed, the boat in Assignment 1 is immediately triggered. If the boat is currently activated, then this button does nothing.

Assignment-2 Debugging

This subsection contains debugging options which are specific only to the Assignment 2 scene file. If Unity does not have the Assignment 2 scene file opened, then these options will do nothing.

There are two options available:

- Randomize Empty Parking Spot (All Positions): When this button is pressed, a new empty parking spot is chosen at random. Any position can become the new empty parking spot, and the old position is reverted back to an occupied parking spot.
- Randomize Empty Parking Spot (First Row Only): When this button is pressed, a new empty parking spot is chosen at random, **but only from the first row**. This can be helpful when you want to test parking functionality quickly, but you don't want to wait for the ego vehicle to move through the whole parking lot.

Assignment-3 Debugging

Assignment 3 Debugging contains the exact same buttons as in Assignment 2, and they have the exact same result on the environment.