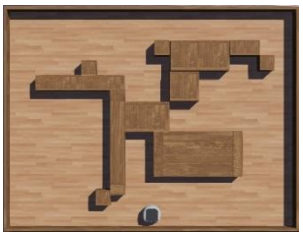
LAB 13 – Finding Obstacle Borders

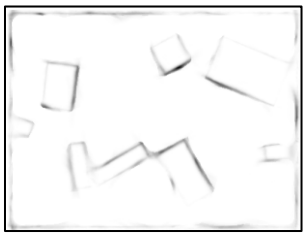
(1) Download the **Lab13_FindingObstacleBorders.zip** file and unzip it. In this lab, we will not use the Webots environment. Instead, you will compile the **MapperApp** program separately, using any Java IDE that you like. However, you must not arrange your code in packages (some IDEs do this by default). A file has been provided for you in this lab which is called "Using the IntelliJ IDE". It describes how to set up the IDE and open the lab files within it so that you can compile and run the code. We will be using pre-computed maps from the Webots environments shown here (which had a higher distance error for the sensors):

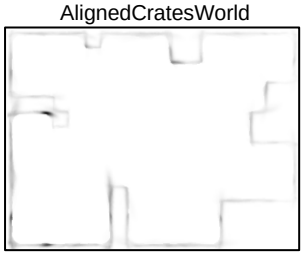

CratesWorld


AlignedCratesWorld

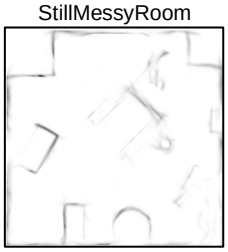

CombinedCratesWorld


StillMmessyRoom


CratesWorld.map

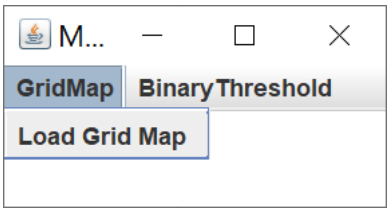

AlignedCratesWorld.map


CombinedCratesWorld.map

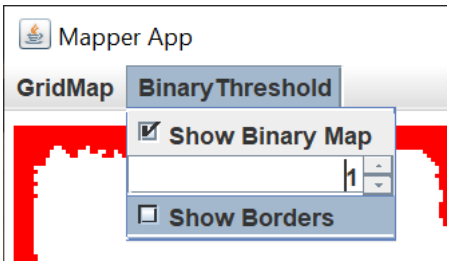

StillMessyRoom.map

The **.map** files have been provided for you in the same directory as the source code. The main application that you will run is the **MapperApp** which has two menus when you run it:

- The **GridMap** menu lets you **load** a map from a file.



- The **BinaryThreshold** menu lets you turn on/off the binary map view using the **Show Binary Map** checkbox. The spinner box lets you set the threshold of the binary map to an integer value. A value of **0** indicates to include every non-zero cell as an obstacle. Increasing it to **1** indicates that only cells whose value is above **1** will be considered an obstacle … and so on. The **Show Borders** checkbox allows you to turn on/off the display of the obstacle borders.



Of course, the **Binary Threshold** menu functions do not work yet. You have some work to do to make that happen. You will be writing ALL your code for this lab within the **Map.java** class.

(2) To begin, you will create a binary grid. The map is stored in the **occupancyGrid** array of floats that represents the probability of an object being at each cell. A **binaryGrid** array has been defined already as a **Map** attribute, as well as a **BINARY_THRESHOLD** integer. Complete the **computeBinaryGrid()** method so that it fills-in the values of the **binaryGrid** according to the values of the **occupancyGrid** (see slide 4 in the notes). Any cell whose **occupancyGrid** value is above the **BINARY_THRESHOLD** should be set to **1** in the **binaryGrid** and **0** otherwise. You should typecast your **0** and **1** literal constants to **(byte)**. The **BINARY_THRESHOLD** value is set via the spinner from the **BinaryThreshold** menu in the **MapperApp**, so you will not change it in your code.

Once you think that you have this working, run the **MapperApp**. Load up the **CratesWorld.map** file, which will take a couple of seconds before it displays. Check off the **Show Binary Map** checkbox in the **BinaryThreshold** menu. You should see the map readings turn red. Adjust the threshold by using the spinner in the menu. Notice that as the threshold is increased, the obstacles become thinner. Save three screen snapshots of the **MapperApp** window as **Snapshot1.png**, **Snapshot2.png**, **Snapshot3.png** that shows the binary map as you adjust the threshold to **0**, **1** and **10** for the three snapshots, respectively.

(3) Complete the **computeBorders()** method so that it finds all the borders of the binary map (see slides 6 to 9 in the notes). The method should assume that the binary map has already been computed. It should then apply the algorithm that you learned in class so that it sets each **binaryGrid** cell to **2** if it is a border cell. Make sure to handle the special case when **x == 0**. When checking the boundaries … **DO NOT** use the **isInsideGrid()** function because it used for a different purpose and makes use of a resolution variable. Instead, you must write your own IF statements to check for the proper boundaries of the grid. You may want to create a **TraceWholeBorder()** helper method, as was done in the notes. Once all is ready to go, you can test it by loading up a map and then turning on the **Show Borders** checkbox from the menu. You should see a blue border around the obstacles if all worked out well.

Debugging Tips:

- If your code hangs and becomes unresponsive, it is likely that you have an infinite **while** loop. It could be one of the following:
  - You did not exit the loop when reaching the start point again (i.e., (**Xc**,**Yc**) = (**Xs**,**Ys**)). Check your values for (**Xs**, **Ys)** and your IF condition for checking if you reached that point again.
  - You are not updating **d'** properly (check formula or IF statement).
  - You may be checking if the **tempX** value is **> 0** when you need to allow the case when it equals **0** as well.

- If your code produces an **ArrayIndexOutOfBoundsException**, then it is likely one of two things:
  - Check your **IF** statement condition for checking whether **tempX** and **tempY** are beyond the grid boundaries, you may have written the code wrong (see slide 9). You must also remember that an array of height 300 allows indices from 0 to 299 … not 300!
  - In the **computeBorders()** method, make sure that you handled the case where (**x == 0**) … since you cannot access the array using (**x-1**) in that case (see slide 6).

Save the following snapshots using the following maps and binary threshold values. For each snapshot, make sur that you unselect the **Show Binary Map** from the menu so that you just see the original map with the blue border outlines:

CratesWorld	Threshold = 0	Snapshot4.png
	Threshold = 1	Snapshot5.png
AlignedCratesWorld	Threshold = 1	Snapshot6.png
	Threshold = 5	Snapshot7.png
CombinedCratesWorld	Threshold = 1	Snapshot8.png
	Threshold = 3	Snapshot9.png
StillMessyRoom	Threshold = 1	Snapshot10.png
	Threshold = 2	Snapshot11.png

Submit your **java** code. It is important that there are no package statements at the top of your source files (some IDEs require you to put everything into a project or package). Just submit the source files without the package lines at the top. If you are using IntelliJ and are unsure how to do this, please see step **(10)** of the “Using the IntelliJ IDE” file provided.

Submit the **4** **.map files** as well as the **11** snapshot **.png** files.

Make sure that your name and student number is in the first comment line of your code.