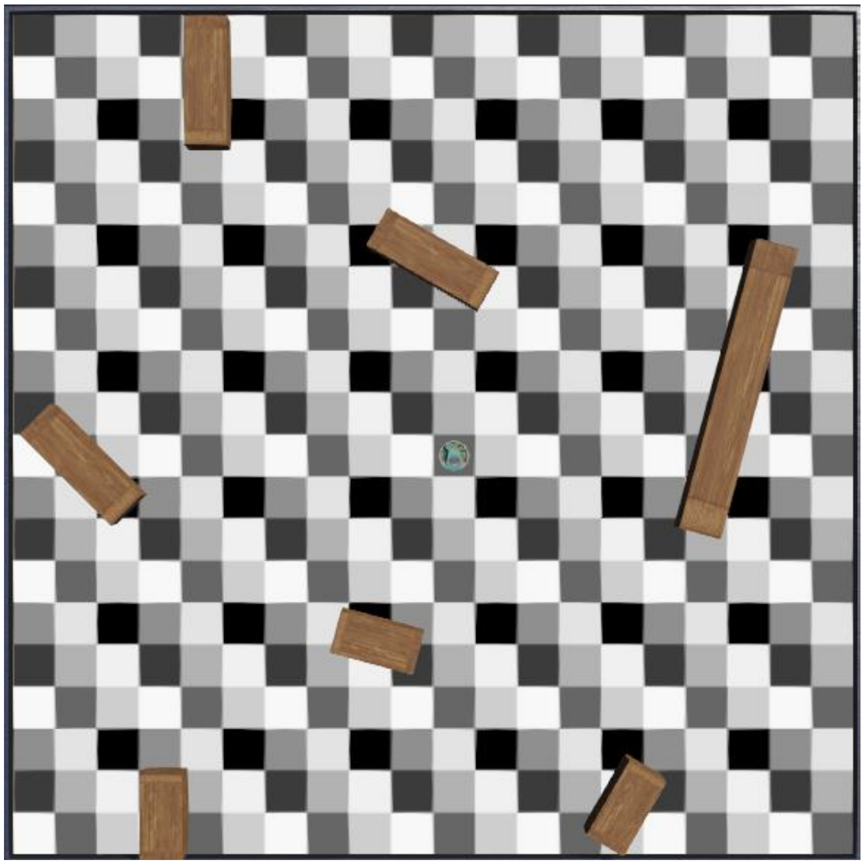
LAB 7 – Grid-Based Estimation

(1) Download the **Lab7_GridBasedEstimation.zip** file and unzip it. Load up the **PositioningGrid** world. If the lab does not load or does not show the grid pattern, may be because the location is not set properly in the world [In this case, click on the **RectangleArena** in the Scene view. Expand the **floorAppearance Appearance** and then expand the **texture ImageTexture** and expand **url**. Click on the file name specified. At the bottom of the Scene view, click the **Select…** button. Navigate to the **Lab7** folder that appeared when you unzipped and then on the **protos** subfolder and then select the **checkered_colors.png** file.] The world should appear as shown below. You may have to turn on **3D View** from the **Tools** menu.

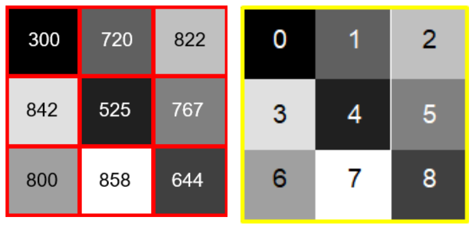
If you get an error message indicating that the ground sensors cannot download, look at the error message in the console window. You will need to copy the link that it mentions into a browser and download that file. Place it in the **protos** directory of the **Lab 7** folder and then reload the world.

The goal of this lab is to compute an estimate of the robot's location as it moves around in the environment. The estimate is based on the start location being (0,0) and is updated according to the robot's ground sensor readings as the robot moves from cell to cell in the grid.

You will only need to insert code into the WHILE loop in a couple of places. You may add whatever functions that you'd like, if it makes your coding easier. But you MUST NOT alter the manner in which the robot moves. Your code should not affect that.



(2) The **Lab7Controller** code has been started for you. The given code causes the robot to wander around as in **Lab 1**. To begin, we will make sure that we can get the proper color index for each grid cell. The **calibrationValues** array shows a typical ground sensor reading for each of the **9** grid cells as shown here in red. The order of the array, follows the ordering of the cell colors and is shown here in the yellow.



Readings will fluctuate, especially when travelling across lines. Your first step will be to make sure that you get the correct color index for each sensor reading. Set the robot speed to **0** temporarily. Print out the sensor reading in your main while loop. Then run your code and drag the robot around manually to each cell color to get a feel for the value ranges. Remember that it is not the center of the robot that detects the color, it is the front edge of the robot.

Once you understand the color ranges, write a simple function that takes the reading and then returns the proper color index for that reading. Change your print statement so that it prints the color index instead of the color reading. Run your code, manually moving your robot around to make sure that it gets the proper color index every time.

(3) Now we need to add code so that the robot displays its new estimated position each time it changes grid cells. The robot starts at position $(\mathbf{x}_0,\mathbf{y}_0, \mathbf{a}_0) = (0, 0, 90°)$ and each grid cell is **10cm x 10cm** in size. It also starts on a greyscale color index with value **8**. You should only display the location when the robot moves into a new cell. The format should be as shown in the blue below:

```
(0, 0) at 90 degrees        // robot starts with this pose
(0, 10) at 90 degrees       // assumes robot went up 1 cell
(0, 20) at 90 degrees       // assumes robot went up 1 cell
(10, 20) at 0 degrees       // assumes robot went right 1 cell
```

So that you don't make mistakes, the **X_OFFSETS**, **Y_OFFSETS** and **A_OFFSETS** arrays have been given to you. You will notice that these 3 arrays have dimensions of 10x10 instead of 9x9. That is because an extra index is used to represent the case where a color did not match … which will be index 9.

It would be good to test by manually moving the robot up/down and left/right. You will notice that there will be issues when crossing the cell boundaries … because you will get some "noisy/fluctuating" color indices. You will need to handle this by waiting until the color index settles before you are sure that you have entered another cell. Otherwise, you can assume there is no match and that the robot is transitioning from one color cell to another. You can simply count the number of times that you received a consistent set of color indices. If you have enough of the same indices consecutively (you decide how many is enough) then you can assume that you are in that color index cell. Then when you cross a border … you will get a few spurious indices … just ignore them until you get another consistent set of color indices.

Test your code with the original robot speeds now. Your code should work fairly well … but you will never get it perfect. If the robot travels long enough, it will eventually end up with a wrong estimate (because the robot may travel along a cell boundary for too long … making it impossible to track the robot's cell change accurately.

Submit your **Lab7Controller.java** code. Make sure that your name and student number is in the first comment line.

<u>Tips</u>:

- You will often likely move the environment or alter its view by mistake. Don't forget that you can use ALT-1 to adjust for a top-down view if you mess up the viewpoint. You can also restart everything from the beginning.
- You can change the default view so that it goes to that view on a restart, but just make sure that the robot is at its original start position BEFORE you save the world. To do this … make sure that the simulation is not running, press restart to get the robot back to the start, then change your view to how you want it (WITHOUT running the world). Then save the world. Then you can start the simulation with that new starting view.