# Project4


## Josh Benton


## The project

The task was to create a grid mapper that utilized the laser scanner. Where the distance between the robot and the laser strike is assumed to be empty space. To handle this, I decided it was better to assume something was an obstacle, and the robot was working in a static environment.

The reason I made these assumptions, was to allow the utilization of a simpler algorithm that prioritizes a block as being "obstacle space."

## LASER

So the breakdown of the laser code is fairly simple, get the laser readings, reduce them to a set of integer blocks or strikes (no duplicate values) [this is toggle-able], then mark that block as a "hit" block, and if the blocks between the robot and the hit block are unknown, set them to occupied.

more complicated versions of this can be added, but the computational complexity of the current system is a bit lower (~2000 laser strikes reduces to a set of ~40 strikes), however currently all laser strikes must be processed once to put them in the set.

## Strange coordinates

I, like many other groups had to do a bit of a transform to make the grid mapper work correctly. In the IUtils method for processing laser data, I had to change it so that x = -sin() and y = cos().

## The open CV plunge

so one major thing I tried was using open cv to avoid needing to write a custom class, I made a simple method to return what value was at a specific x,y coordinate. I ended up figuring out that I didn't need the canvas lock to check a value.

## inhaler_gui

The inhaler gui now has functional "persistent" text and lines, so you can draw lines and text, then update them with the same messages. they use an ID system.

to update the lines and text, all future gui components will be “persistent” which unfortunately means there will be a slightly higher network overhead. To deal with that, I am setting up client side (the ros node you would be working on, so grid mapper in this case) optimizations, such as don’t send new information, if information has not actually changed. No major complix gui objects have been tested, but a square of debug text has been created with that functionality. (it looks a bit like this:)

(debug text 11) (debug text 12)...

(debug text 21) (debug text 22)...

## instructions for running my code

in the event that you want to run my code, there are now requirements for the inhaler_gui, the easiest thing is just to grab it from github, and put it in the src folder of your workspace. you may also need to write a custom compiling script or modify the cmakelists file so it actually puts the executable in the correct spot (I just have a script for this). or I can transfer the entire workspace.

# pictures