Have you ever been in such darkness, with the only light to guide you being millions upon millions of beautiful, glittering lights, shining at you from every direction? Were you don’t know what is up or down, floating endlessly, with none of your troubles bothering you?

I have, and it’s fucking underwhelming. It’s one of those things that sounds good, but ends up being pretty shit. I may be confusing being in space for being blindfolded and wedgied by my crewmates while there where lights flashing in front of my eyes because they also hit my head, but it can’t be that different, can it?

Whether you’re being wedgied or not, knowing where you are can be useful, so I needed a way to do that.

First up was finding out which direction I was pointing, and I knew just the way.

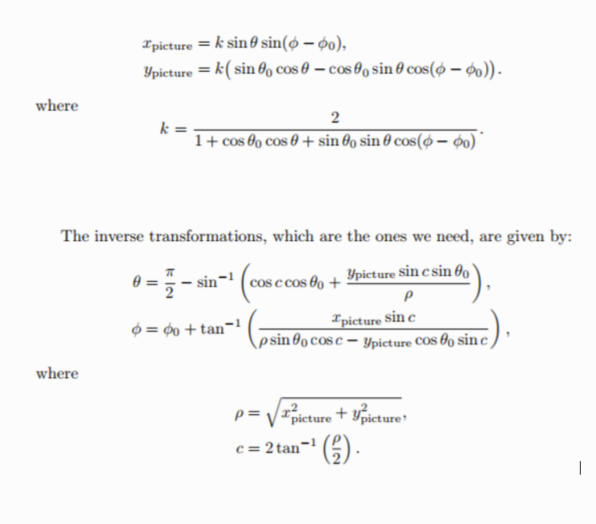
In my archives, I had a big file, containing the entire night sky, so I wanted to use that to find out which way I was pointing. The theory was simple enough; just take a picture and match it up against the picture of the sky and find out which way is which. Turns out, the coding was a bit painful, but I’ll describe that later.

Second, I wanted to find out how fast I was going. I knew I could use the doppler effect to find my velocity relative to something else, and I found an old device that could tell me how much the wavelength of light emitted from a certain star when viewed from the star system I was in. It looked pretty old, so it might be broken, but then again they made them to last in that day. I also didn’t know if it accounted for how those stars moved in relation to the system, but I could only assume that I could just plug what the machine gave me directly into my doppler equations (will show them later), since I didn’t really know how to make a machine that could do that and also fit onto my ship. But since I could then find out my speed relative to those stars, I could apply some fancy trigonometry and find my velocity relative to my star.

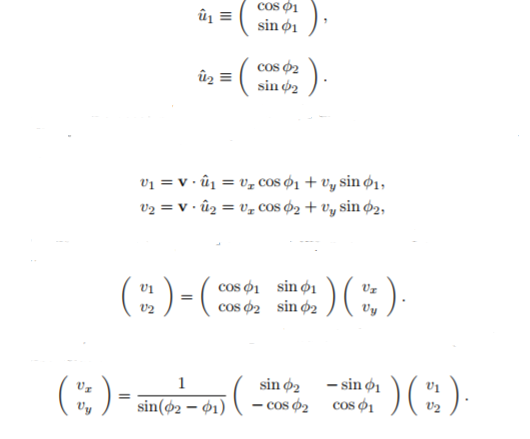
Finally, I wanted to know my position, to find out where I was in this world, or more specifically where I was in relation to the other planets. I found this device (finding a lot of devices this week, it’s like it’s my lucky week, had it not been for the fact that I was stuck on a godforsaken planet) that could tell me how far away from other celestial objects I was, but not in which direction it was in, but I could work around that, since I knew the position of all the celestial objects at any time (hah! That is the power of simulating ahead! Maybe you should think ahead, Muhammad, instead of just throwing things around in one of your temper tantrums). I’ll be honest I worked for days trying to solve it analytically, with equations, but they *just wouldn’t work even though they should (Just like fucking Greg)*, so in the end I wrote a simple and very shitty numerical approach that worked slowly, but got the job done. The numerical approach took me about half an hour. At that point I was not the most enthusiastic about maths. The analytical approach described the planets as circles, where the radius was the length from the planet to my ship, and then I tried to find an expression for that point, if by “an expression” I mean like five expression, none of which work. The numerical approach just walks along one of the circles and finds the point that is closest to all the circles intersecting. It worked.

Finding the direction, while simple enough in theory, was a bit more complicated in practice, as the pictures aren’t completely perfect, so I couldn’t just check them all and see which fits best. I couldn’t just check them by eye either, as I needed it to be accurate and fast, so I decided to use a method called the least square approximation. I made a table of 360 other tables, of size 640 by 480 with three numbers in each cell (which means in coding that I made an array of shape 360 by 640 by 480 by 3) and filled it with 360 pictures (640 by 480 pixels) and described them by their RBG code (A colour, here the colour a pixel shows, can be described using three numbers, each describing a different colour; red, green and blue, all of which can be combined to form all other colours). I then took a picture in the direction I was facing and put that in a similar table to the other 360 ones, 640 by 480 with 3 for each. In order to find which was the best fit, I took each table in the list of 360 and subtracted the RGB values so I could find the difference, then I squared that difference to remove the negative sign and make sure that bigger differences counted more. The simple solution was then that the picture with the smallest squared difference was the best fit, which is why they call it the least square fit (I know I called it the least square approximation, but fuck the system, I make my own rules)

Now, I also had to go through the hoop that I was given the picture as a flat picture, whereas the sky is actually spherical to our eyes (gasp, the world I round), and I needed the reference the sky in angles, instead of the x-y coordinates I could get, but luckily I had a handydandy formula to transform it. Someone else already did the maths for me, so I’m just going to show them to you:



For finding velocities, I needed some way to transform two angles and speeds to velocities related to my star. Luckily, someone had done that for me, too, so I’m just gonna show that to you as well:



Now, I’m going to tell you what I tried to do to find my position:



This is how you describe a circle centred in x0 and y0. You can use this formula for three different circles, with three different x0’s, y0’s and r’s. that should be three different for two unknowns (two would result in a second degree formula that gives potentially two answers), but *apparently it’s not so fucking easy. I have 9 pages of maths (not gonna show it all because it’s 9 fucking pages, and the maths doesn’t work out for me) that don’t work because why would it ever work!?* Anyways, I decided on a numerical approach instead, and because it was at that point 2am and I had spent the previous days trying to figure out why my analytical approach didn’t work, I made it janky and poor as hell, but it works. The numeric approach choose the star as primary circle (I described using the distances as a basis for making circles, didn’t I?), and two planets as secondary circle. I made a list containing all the x values from one side of the circle to the other, and used that list to make two corresponding lists of y-values, for the top and the bottom of the circle. I then went through all of them checked the distance to the other two circles. I picked a large number at the start (something like 10 to the power of 35 just to make sure it would work) and whenever the distance to the two circles was the smallest, I would replace that number with the new position, continuing until I had gone all the way around the primary circle. It was janky as all hell, and I don’t have any current plans to improve it, and will not improve it until I get really annoyed with how janky it is or I actually need to get both accuracy *and* speed.