So we all know what it’s like to be abandoned in a less-than-pleasant environment and then have your only source of entertainment be building a spaceship so you can stick it to the Almighty. At least I hope so. Or maybe I don’t, it feels quite shitty. Ok, I do wish it upon Jerry. Fuck Jerry.

Maybe I was just having that kind of a week

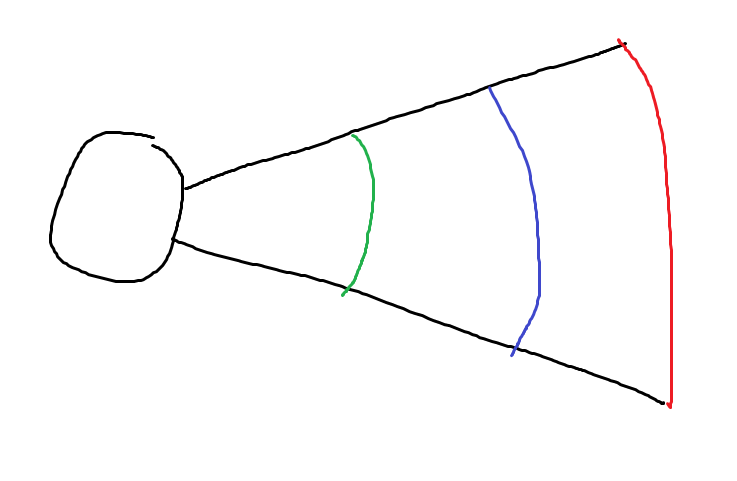
Whatever kind of week I was having, I needed to know some data on the planets I wanted to visit, because no one wants to be stranded on such a shitty world twice (I don’t care how kinky you are, you don’t want that). I wanted to know how big my solar panels would have to be for my mech-suit to work (I have one, don’t ask why), so I had to drag forth my trusty helper: A fuckton of math! As a shorthand, I used the temperature of the star and the distance between the planets and the stars to find out how much energy the star would put out over a certain amount of time, how much that would spread out as it travelled to a planet, and described that with a formula! Then, since I know how much energy a solar panel absorbs in relation to its size, I could just plug in some numbers and I got the size of the solar panels! Now, I had to account for the fact that some energy would get lost in the atmosphere, and the final size of the solar panel size I would need is 2.5m2 to power my 40w mechsuit (it is that powerefficient, just trust me, I’m amazing)

Next I would like to know if I would get boiled alive or something, so I made a very simple formula that would describe the temperature of a planet given the distance from and temperature of the star. I would just use my previous formula to see how much the star heated up the planet instead of my solar panel, and I then found how much heat the planet itself would radiate off into space. It was very simple, but I figured it got the job done. The problem was that it said that a planet was 155 kelvin, while it was described in the journals as a warm and life sustaining planet (even if it also said that it had 155 kelvin, it had pictures of nice forests and stuff, so I’ll just assume it’s a nice planet and travel there to research). In the case that it is a cold planet that has life, I think it’s very likely that that is where God actually lives, so I’mma go there anyways, just because fuck that guy. The name of the planet was Fræfdal; a nice name, although it is a bit odd.

As more of a personal thing, I also wanted to know where planets had to be related to the star they orbited in order to have comfy temperatures when they *weren’t* inhabited by some asshole deity. All I had to do was just to reorganise my previous formula for finding temperature to find the radius instead and plug in the minimum temperature I wanted and then plug in the maximum temperature I wanted.

Now, as a complete theme change, I wanted to send out my satellite! Simulating a satellite *should* be very simple; you just send it out, calculate how much each planet affects it, and move it in accordance to that. But that is how coding works, isn’t it; “This thing should be easy, so of course I’m going to spend hours more than necessary making it”. So I spent hours more than necessary making it. The problems were mostly (actually only) code telling me that I’m dumb and that code doesn’t work like that, even if I work with Python which is one of the simpler languages to code in. I’ll explain the details later, so I keep the detailed part of the log in the same order as the not-so-detailed section

So, for the section with more details: I first mentioned a formula for how much energy a star puts out, and using that for the size of my solar panels. I do this by use of black-body radiation, which is the radiation that occurs in certain materials because they’re warm. You can see this when you heat up metal a lot (it becomes redhot and actually glows), or if you use infrared cameras to see radiation that is too low-energy for us to see. Basically, you assume it gets spread out over a metaphorical ball, so that if you see how much energy there in a ball close to the star and compare it to a ball further away, you will see that they have the same amount of energy:



From the star (the ball-thingy on the left) there is energy spread out in a circle outwards. If you placed a rope along the green line, it would absorb as much energy as if you put a rope along the blue or red line. This works no matter how far around the star the line goes. Since the star outputs the total amounts of energy found by (where sigma is a constant, R is the radius of the star and T is the temperature), you can find out how much energy is put out at a distance r from the star by dividing by the surface of a sphere of radius r: as (Watt per meter squared). If we then plug in the numbers for a planet we want to check and divide the amount of wattage X (here, 40W) we need by the energy E we get which gives you the size y of the solar panels you need on a certain planet.

For finding the temperature of a planet, we just use the formula earlier, reshuffle it a bit and use E and y as the energy absorbed by the surface of the planet facing the star; . Then find the blackbody radiation of the planet the same way we did for the star. Take one minus the other and equal it to zero, and you get the steady state temperature (the temperature where it stays unchanging)

Now, I didn’t use this formula exactly, because earlier in the program I had previous calculations that meant that the code had to perform fewer calculations in total, but the formula works.

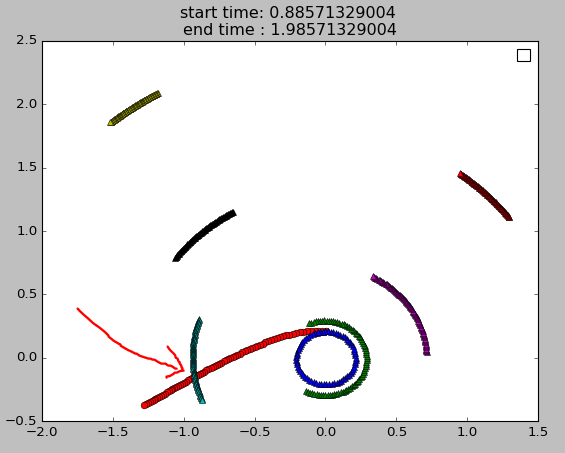
For finding what we call the goldilocks zone (the radius from the star where liquid water will form) you rearrange the equation and plug in the minimum temperature (260) and the maximum (390) and you get the range for the goldilocks zone:

This gives, for a star of temperature 4000k, 9.5\*107km and 4.5\*107km

So, for the detailed part of how sending out my sattelilte, the simple thing that took much longer than is should have! The simulation itself is rather easy, I just used the leapfrog method I described when I simulated the planets themselves with sending the satellite out from the position of the planet I’m on with the velocity of the planet added to the launch velocity I got from my launch simulations. I ignored the gravitational effect of the satellite on the planets because to be honest it is very small, and ain’t nobody got time for that.

After my initial launch I had some fuel to spare, and while I could wait and use it at the most opportune moment to get the most out of it, I decided instead to use as much as I needed at once. Therefore I blasted the satellite as soon as it was outside the gravity well of my planet (or as far away that I didn’t care about it any more. It’s less accurate but it makes it simpler so shhhh) and blasted my way to the orbital height of Frædal (I found the amount of speed I needed by just trying and it ended up being. This wouldn’t land me on the planet exactly, but if I tried this approach at lots of different times, because the planets have different orbital periods (can’t remember exactly what they are right now, but because it worked it must have worked), eventually I’ll find a launch time that will let me hit Fræfdal pretty damn close (and I did, with a distance of 0.005 AU’s (Astronomical Units).

In order to speed up the simulation I worked with inclines, meaning I checked 150 different times, by running a simulation for a certain amount of time, between a maximum time and a minimum launchtime, found the best one, then checked 150 times again between new max and min launchtimes as the time I checked before and after the best time I got. I repeated this process about 8 times, so I basically had the accuracy of checking 150^8 times, while only having to check 150\*8 times.



The red line is the satellite and the rest are the planets, the axes refer to the position in AU’s and the picture adds new points as time goes by. The picture was taken a bit after they hit eachother, but they were very close

I am planning to just give it another boost so it starts orbiting when it is next to Fræfdal, but I haven’t quite done that yet

To be quite honest, the issues were simple things such as not getting the dimensions of my arrays correct, and not really worth bringing up here, just know that all of it was really dumb.