Contact Information

Monday, May 15, 2023 12:58 PM

Name: Isaac Joseph

Email: isaacjoseph11602@gmail.com

Best Contact Discord: A_Thought#7770

Introduction

Monday, May 15, 2023 1:02 PM

Hello Everyone.

First of all Thank You for looking into my Pure Pursuit. This took me quite a few months to accomplish and I am happy with my final result. My goal for this is to expand young men/women's knowledge of programming with robot movement and learning different math applications you can use to get certain outputs in values to accomplish what you want.

I would like to point out that this is my original work and not copied by any team. In fact I never looked at another teams version of Pure Pursuit during/before this project. I will say that I was not the original founder of Pure Pursuit and that there are many versions out there. This version is one that I created myself. If there is another one that is out there that is just like mine but created prior to mine then I have no issue giving a shout out to whom ever also developed the same code and released it to the public. If their code was not released to the public and was used for private purposes then I will not be giving a shoutout.

I would like to give a shout out to the following people for aiding me in my research and or growth in robotics and programming.

Joshua Strickland: Mr. Strickland is my coach for robotics. He is an amazing robotics and engineering teacher out of Lovejoy Highschool and I am honored to have been a student of his.

Clark (360X/580X/Ronald McDonald): Clark is a great friend of mine who allowed me to use his supplies when available and allowed me to use his robot during the final testing phase of this code

Hasif Shaikh: Hasif was my mentor throughout Change Up and Tipping Point. I am grateful for his help in my development and growth in robotics and couldn't have made it as far as I have without him

Sai Senapathi: Sai is my main building mentor and also a member of the EX-VEXU team SPARK. Sai helped me and my team through out Change Up and Tipping Point and helped me with a lot of math stuff for this project

That is all and hope you all enjoy my Pure Pursuit!

What is Pure Pursuit

Monday, May 15, 2023 1:02 PM

Pure Pursuit uses set points ahead of you and calculates the power of the robots motors to move your robot along a given path

My version uses a point at a set distance in front of the robot to move the robot along the path. Doing this allows the robot to have near perfect precision. This may not be very necessary if you setup about 1 point per inch along the given pathway however when speaking with others it sounds like most teams just do about 10 points along the path. This maybe fine however it does introduce error that can easily eliminated by adding a point Infront of the robot at a set distance so no matter what the point will always be Infront of the robot about a inch (depending on where you set it) instead of the points possibly having 2-3 inches between them and the points constantly changing distances from the robot.

There may be many more differences however this is the only noticeable one that comes to mind at this time (Like I said. I didn't look into any other teams or Pure Pursuit documents)

Pathing

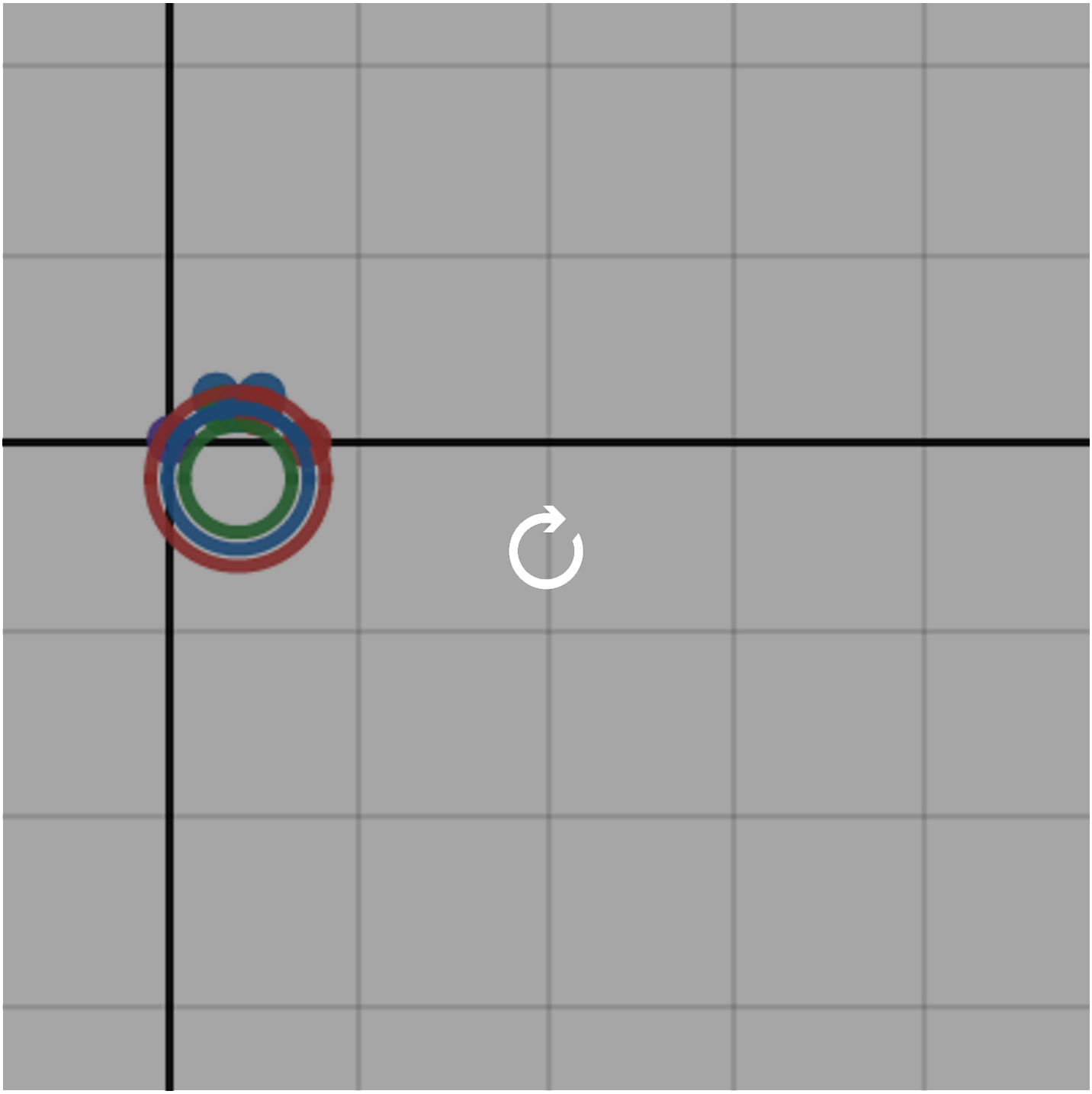
Monday, May 15, 2023 1:03 PM

My Pure Pursuit uses Beziers Curve as a pathing tool to develop the robots path from the given point of the robot to the target point on the field.

I do also use Odometry to calculate the position of the robot and position of the points that will be set. I will not be explaining how to use Odom in this however there are plenty of Odom documents out there and if you would like help then you can reach out to me in the page "Contact Information".

I mainly used Desmos to do my math/pathing. I will be attaching my Desmos via link here.

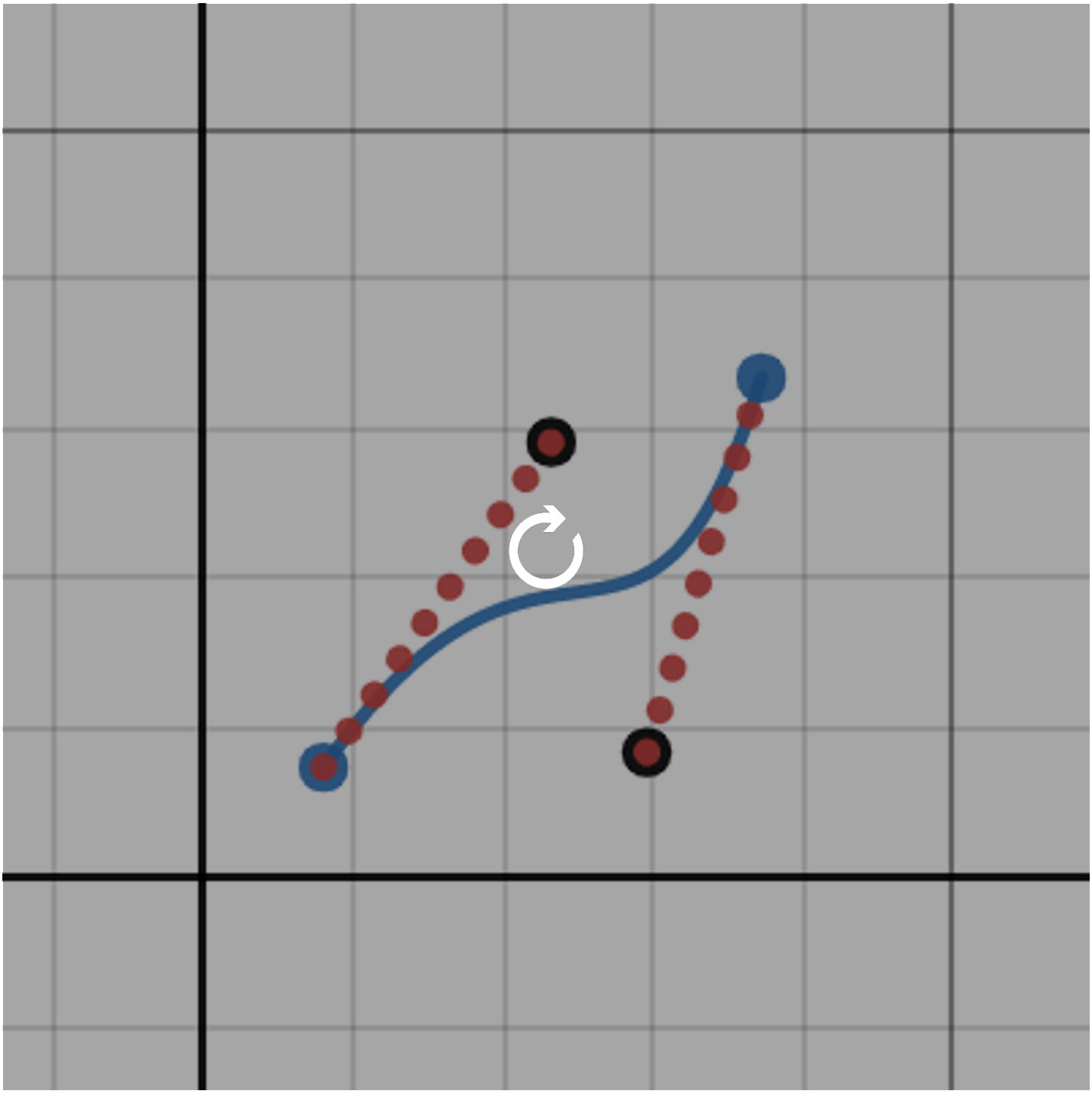
Pure Pursuit - https://www.desmos.com/calculator/iryy2jggbw



That has all the math you will need when it comes to number crunching math for the pathing and robot movement and will help you visualize what's happening.

Learning the Bezier curve path is easy. Just copy and paste the math into Desmos. There is also another Desmos Bezier curve that I will add here that helped me understand how it worked. I never did learn how the Bezier curve math works but so far in my learning I don't believe it is necessary to learn as in my code I never once plugged in the math for the pathing of the Bezier Curve as it isn't necessary. Here is the example Bezier Curve.

Cubic Bezier Curve - https://www.desmos.com/calculator/hg467yvicv



From here on out through the notebook you should be referencing the Pure Pursuit link I first added whilst reading through my notebook for easier understanding

In Desmos I have generated 3 points along the curve. The point in the back is the robots position, the point in the front is the target that the robot is chasing (this point is further forward in the Desmos than what'll actually be, I set it up like this so that its easier to visualize), and the point in the middle is just a mid point, this allows us to create a 3 point arc between the robots current position and the target. Creating a three point arc is essential and possibly the only accurate way to calculate the power that is needed for the right side wheels and the left side wheels.

You can see the three point calculation under the "Three Points Calculation" folder in Desmos. When looking in there you will first see the variables "G, O, V". The main one you will want to pay attention to is "G". "G" is the main variable and it moves from 0.25 to 1.00. The whole path from beginning to end on the Bezier curve is from 0 to 1 (I only have it set from 0.25 to 1.00 due to it being the front coordinate out of the three and if I had it move from 0 to 1 then it would go off the path). In your code you will want this to go from 0 to 1.

After this you will see 3 (x, y) point calculations. These are the points on the Beziers Curve. These points are needed to calculate the difference in power for the wheels. I did a lot of research to figure out how 3 point arcs work and how to implement it here (I don't have the citations for this as this project has been going on for a very long time. Yes I know, terrible me. If anyone has a cite that shows these calculations please feel free to email me it and I will update it here).

Using these three points we now will calculate A, B, C, D. I'm not really too sure if these have a name but they are used for calculating the power so I just copied and pasted it in and changed it accordingly.

You will then use A, B, C to calculate 1 last (x, y) coordinate which is the center point of the three point circle. Once this is calculated you use the following formula (Quadratic formula)

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

This is then used to calculate the power which you can see under the "Power Calculation" folder. This takes an equation and is multiplied by the max value of the motor. I have it multiplied by 11 as this is the max voltage for vex motors however you ca also multiply it by 100 for 100% of a motor if your using percentage or you can multiply it by 80 if you only want to use up to 80% of a motor. Also for this equation the number 9 that you are adding and subtracting in the division part is the width of your robot divided by 2.

The output value for your Power Equation you will want to set to the inside motor. For example when the robot is turning left you want this value to be changing the left motors power. When the robot is turning right the you want this value changing the right motors power. When your going straight the it doesn't matter. Keep in mind of the red and green three point circles in Desmos. These circles represent the outsides wheels paths. Notice how when the green one switches from circling left to circling right it doesn't stay on the same side and instead they both switch. Ill leave it to you from here on how to write out the if statements to counter this to keep everything correct

Math for Pathing

Monday, May 15, 2023 1:05 PM

Here is the 3 Points for the three point curve on the path

```
PP::x7 = (pow((1 - PP::o), 3) * PP::x1) + (((3 * PP::o) * pow((1 - PP::o), 2)) *
x2) + (((3 * pow(PP::o, 2)) * (1 - PP::o)) * x3) + (pow(PP::o, 3) * x4);


PP::y7 = (pow((1 - PP::o), 3) * PP::y1) + (((3 * PP::o) * pow((1 - PP::o), 2)) *
y2) + (((3 * pow(PP::o, 2)) * (1 - PP::o)) * y3) + (pow(PP::o, 3) * y4);



PP::x8 = (pow((1 - PP::v), 3) * PP::x1) + (((3 * PP::v) * pow((1 - PP::v), 2)) *
x2) + (((3 * pow(PP::v, 2)) * (1 - PP::v)) * x3) + (pow(PP::v, 3) * x4);


PP::y8 = (pow((1 - PP::v), 3) * PP::y1) + (((3 * PP::v) * pow((1 - PP::v), 2)) *
y2) + (((3 * pow(PP::v, 2)) * (1 - PP::v)) * y3) + (pow(PP::v, 3) * y4);



PP::x9 = (pow((1 - PP::g), 3) * PP::x1) + (((3 * PP::g) * pow((1 - PP::g), 2)) *
x2) + (((3 * pow(PP::g, 2)) * (1 - PP::g)) * x3) + (pow(PP::g, 3) * x4);


PP::y9 = (pow((1 - PP::g), 3) * PP::y1) + (((3 * PP::g) * pow((1 - PP::g), 2)) *
y2) + (((3 * pow(PP::g, 2)) * (1 - PP::g)) * y3) + (pow(PP::g, 3) * y4);
```

Here is the math for A, B, C, D

```
PP::A1 = PP::x7 * (PP::y8 - PP::y9) - PP::y7 * (PP::x8 - PP::x9) + (PP::x8 *
PP::y9) - (PP::x9 * PP::y8);


PP::B1 = (pow(PP::x7, 2) + pow(PP::y7, 2)) * (PP::y9 - PP::y8) + (pow(PP::x8, 2)
+ pow(PP::y8, 2)) * (PP::y7 - PP::y9) + (pow(PP::x9, 2) + pow(PP::y9, 2)) *
(PP::y8 - PP::y7);


PP::C1 = (pow(PP::x7, 2) + pow(PP::y7, 2)) * (PP::x8 - PP::x9) + (pow(PP::x8, 2)
+ pow(PP::y8, 2)) * (PP::x9 - PP::x7) + (pow(PP::x9, 2) + pow(PP::y9, 2)) *
(PP::x7 - PP::x8);


PP::D1 = (pow(PP::x7, 2) + pow(PP::y7, 2)) * ((PP::x9 * PP::y8) - (PP::x8 *
```

```
PP::y9)) + (pow(PP::x8, 2) + pow(PP::y8, 2)) * ((PP::x7 * PP::y9) - (PP::x9 *
PP::y7)) + (pow(PP::x9, 2) + pow(PP::y9, 2)) * ((PP::x8 * PP::y7) - (PP::x7 *
PP::y8));
```

That’s all I will be giving out for a copy paste as this was very annoying to type in. this was taken directly from my code so it may not work first try for you however the math works and all the variables and properly put in. If there are any issues you can refer to the equations in Desmos and or you can email me

Things to Note

Monday, May 15, 2023 1:06 PM

Here I will explain some useful/helpful tips and tricks I did when creating my Pure Pursuit code

One thing I do in my code is I set the maximum power value to equal the PID Power. This is so that it'll slow down properly when it starts getting close to the target. If this is not done then it'll do some circles.

Another thing I do was I have 3 different power values. The main power value is the power value that PID sets. The other values are for the left power and right power of the motors. These power values will be equal to or less than the main power value. Also one of these power values will be set to the value calculated from the Three Point Circles Power Value.

I created my Odom, Turn PID, Lateral PID, and Pure Pursuit all in separate folders and called them through namespaces when necessary (if you don’t know what a namespace it then I recommend researching into it. I use it very frequently when doing multiple different things such as pure pursuit, Lateral PID, Turn PID, Odom, and different functions

When I call Odom it actually only calls the turn PID and then the Lateral PID because I implemented Odom just directly into my PID so that the tracking of the robot would be tracked properly with no issues. Doing this allows me to either call PID and still keep my current coordinates or allows me to call Odom or pure pursuit and still process everything and keep track of my positioning with no issues

Last Part

Monday, May 15, 2023 1:06 PM

There is a lot that I left out such as using InchesPerTick (IPT) for the wheel to translate the ticks that is travelled with the encoders to the amount of inches using the circumference of the wheel. Also using Lateral PID and Turn PID and Odom stuff however this stuff you can do research on to find out as there is plenty of documentation behind it and also you can find your own methods of finding that information

My code is completely private and will not be given out to anyone. This is so that students are forced to learn this instead of copy and paste and move on. I do not condone giving people the full code once this has been created however if you share parts of the code such as the ones I personally already shared for ease of use in the page "Math for Pathing" then that is ok by me. It is not my position to tell you what to do with the code after you use it, I just ask that the full code is not shared and if any then only small parts of it to encourage understanding and learning.

Keep in mind, I created this OneNote to encourage students to learn something that seems hard but really isn't. If I had the Brain and Battery then I could've completed this within 2-3 weeks from start. It only took me so long because I had access to a Brain and such once every 1 to 1.5 months. I encourage anyone and everyone to try and learn PID, Odom, and Pure Pursuit as these are used plentiful in the real world and also are somewhat challenging to first learn.

If anyone has any questions please contact me via discord as I am mostly on that and if you don’t have that then I have also added my email

Thank You.