PA1, CS 81

Garret Andreine

October 2020

1 Description

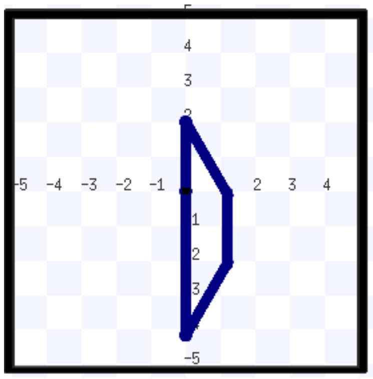
- For each task a different script was made

1.1 Question 1

- In the first question, the code was broken into 3 major functions

- Turning the robot north (pointNorth)
- Making the turns for the trapezoid (turn)
- Going straight (straight)

- Each function was tasked with the objective its name describes
- Turning the robot north just had the robot move at a 90 degree angle for a time based on its rotational velocity
- Straight had the robot go a prescribed distance based off the calculated time needed based off the length of one of it's legs
- Turn keeps track of what movement the robot is on and then turns the robot the needed degrees based off the angle provided



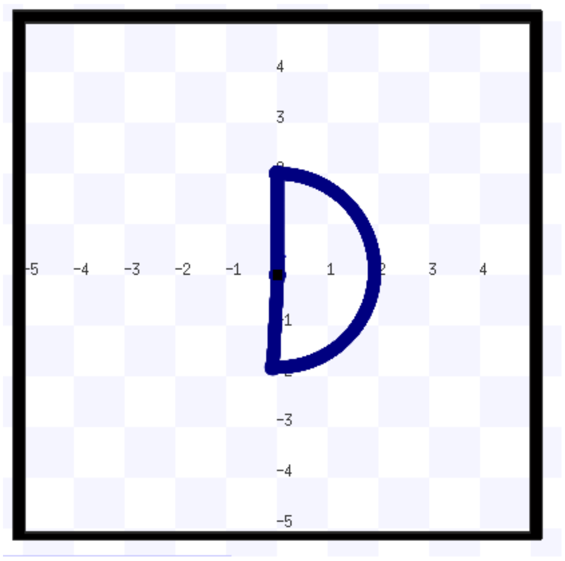
Here is a sample Trapezoid drawn by the script

1.2 Question 2

- In the second question, the code was broken into 4 major functions

- Turning the robot north (pointNorth)
- Making the turns for the beginning of the D (pivot)
- Make the arc that forms the D (half_circle)
- Going straight (straight)

- Each function was tasked with the objective its name describes
- Turning the robot north just had the robot move at a 90 degree angle for a time based on its rotational velocity
- Straight had the robot go a prescribed distance based off the calculated time needed based off the radius given
- Half Circle creates an arc based off the provided radius by creating a velocity message with both an angular z value and linear x value. The angular z is a predetermined rotational velocity and the linear velocity is based off the arc created and the rotational velocity the robot is moving at.
- Pivot keeps track of what movement the robot is on and then turns the robot the needed degrees based off the angle provided



Here is an example D drawn by the Robot

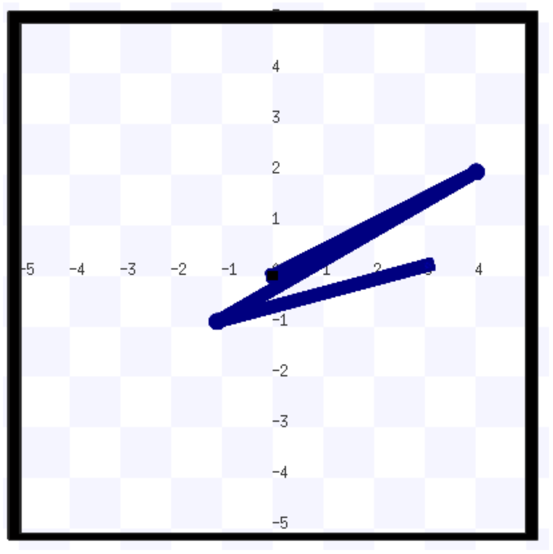
1.3 Question 3

- In the third question, the code was broken into 2 major functions

- Turning the robot (turnToDrive)

- Driving the robot (drive)

- Each function was tasked with the objective its name describes
- Turn to drive calculates where the robot needs to turn to based off the change in x and y value provided by the coordinates and then uses the math library (specifically atan2) to calculate that angle. Based off this angle it calculates the time needed for the robot to turn that direction based on its set rotational velocity.
- Drive calculates the distance the robot needs to travel based off it's initalized position and then the time it needs to go that distance based off it's velocity. Then after traveling the needed distance it continues to the next coordinate it needs to go to and calls the turnToDrive function until it has no more coordinates to travel to



Here is an example of the robot traveling to the coordinates (4,2) then (-1,-1) then (3,0)

2 Evaluation

- Since each task had a different script I will evaluate each individually

2.1 Question 1

- I think this was the roughest script for me to write.
- I was still getting used to the commands and looking at it probably could have kept track of when I needed to turn more eloquently, rather than keeping a counter variable - I think it did it's job but I limited the difficulty of the challenge by making the legs of the trapezoid be what the radius is. Then

making the trapezoid do three lengths of this leg on the base of it also let me make clean looking trapezoids.

2.2 Question 2

- This script went fairly well
- I think looking at the picture it may have gone slightly to far for the arc which could have been a rounding issue caused by the math, but it could be worth looking into
- I think the arc looks really smooth and I enjoyed using a velocity message with both an angular and linear component, it felt more realistic then just having it move in straight lines.
- I think I could have maybe combined the pivot and pointNorth functions into one but it was easy to split them up

2.3 Question 3

- This was my favorite script
- This one felt like a more traditional coding challenge and after using the commands in the other two questions I felt really comfortable doing the math to figure out how far it needed to travel/turn
- I used internal variables to track where the robot was which I thought was clever but could have been done differently using the odom associated commands
- I also enjoyed making it deal with negative angle (having to use the absolute function) and debugging that