Capstone Project Maze

April 28th, 2023

CS 1000

Presented to Prof. Gallagher

Ricardo Brutus

**Notes and Observations**

When I first looked at the Capstone maze, I was a little hesitant on if I was going to be able to do it or but upon reviewing the design requirements, I quickly identified a strategy for breaking the maze into manageable parts. I divided the maze into five sections and used these as a reference for when to begin and end each part. Despite being a self-proclaimed last-minute type of person, I completed the first section in one class session. While working on this section, I realized that incorporating math into the robot's movement would be necessary to achieve optimal results. Therefore, I calculated the number of lines in each square and determined that each line would equal one second of movement at half speed, which allowed for consistent movement throughout the maze. Each forward movement of 1 second at half speed would be equivalent to moving 6cm. By turning one wheel at 50 and the other at -50, the robot could turn 90 degrees, and two seconds of movement in the same direction would result in a 90-degree angle. With this approach, I completed the first two sections of the maze after a few hours of trial and error. It became apparent to me that continued refinement of the robot's movement strategy would be necessary to achieve the desired outcome. Here’s some things I noticed while going through the first 2 sections.

* I first thought that every movement on the first section was even, but it was not when I actually counted the number of lines there was.
* I thought I could encapsulate the zig zag parts since what I initially thought was that they were same amount all away around.
* The Zig zag parts were different.
* The length between going 45 degrees straight from the zig zag was also not the same.
* I made sure to make my movements move the same amount of how many lines were moving.
* So if the scribbler robot needed to move 8 lines. I made it move 2 times in a loop of it going straight for 3 times, and I added a block of it going straight since it was uneven.
* I later realized that in most parts in the sections I could have the robot move 3 times and call onto the subroutine and just add a second or more if I needed to get to a number that was even.
* Then I went to the point where I had to make the squares and I thought those would be easy since they are all squares, and I was wrong since the squares start off differently.
* I then realized b2 and b3 could be encapsulated and called upon but not B1
* I spent most of this trying to figure out how I could make it better and shorter.

While working on Section 3, I observed a marked improvement in my ability to break down the maze into manageable parts. Specifically, I was able to streamline the process by encapsulating B3 from B2 and incorporating the zigzag component from the beginning of the maze. This approach allowed me to more efficiently navigate the maze and make progress toward completion.

Section 4 was fairly easy to do as I just used existing code from before and finished it quickly.

Regarding Section 5, I was able to successfully navigate the circles by breaking them down into four quadrants. This approach allowed me to halt the robot at the halfway point of the circle and rotate it to the next quadrant, which facilitated smoother traversal of the maze. However, given the complexity of the coding required to execute this approach, I found it necessary to open separate S2 programs to ensure I could maintain a clear understanding of each section's code. Unfortunately, the lack of note-taking functionality in the S2 program led to some confusion, particularly when attempting to identify the source of coding errors. In retrospect, I realized that I had needlessly put all of my coding into a loop sequence, rather than simply repeating the subroutine as necessary. Ultimately, I was able to complete the maze by strategically using multiple instances of the S2 program to work on individual sections without risking data loss or confusion.