Hephaestus

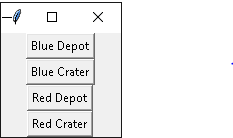
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Hephaestus in greek mythology was the blacksmith of the gods, he built all of the equipment for the gods. Our version of Hephaestus is a program we built to make autonomous programs for us, allowing us to make complementary autonomous programs with our alliance partners.

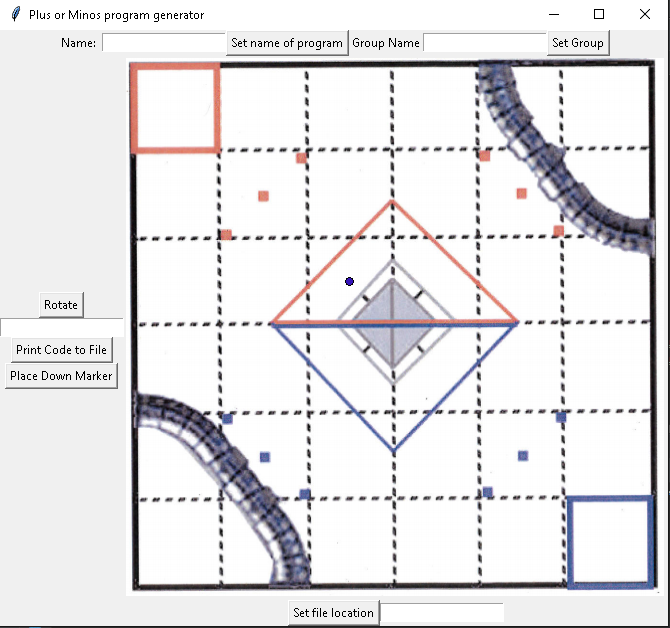
Program Creation:

The program creation process starts with a screen instructing the user to choose a starting position. This is important as it allows us to set the starting angle of the program and its starting coordinates. Without these two important pieces of information the program couldn’t accurately say which direction to drive in, as well as how far.

The starting position screen:



The program closes the screen once a position has been chosen. It then opens up a screen with a bird's eye perspective of the field, this is where the main part of the program creation takes place. Clicking any part of the picture will have the position be recorded as well as placing a small dot in the spot for reference. This position is then used in a function that uses the position in a pair of equations that gives us the angle the robot needs to drive in and the distance it needs to go; after the numbers are run they are then formatted into a encoder drive statement.



There are several buttons on the screen, all of which are explained below:

Set Program Name: Saves the name that is put into the entry box next to it, this is to name the program once it is printed.

Set Group Name: Save the group name that is placed into the entry box next to it. This is for you to place into the program under the correct group on the phone.

Set Location: This is to place the program into the correct folder once it is printed. It’ll default the location that is pulled from a text file in Hephaestus’s folder so we don’t have to add the location every time.

Print Code to File: This takes all of the instructions that have been given to the program and throws it into a Java file.

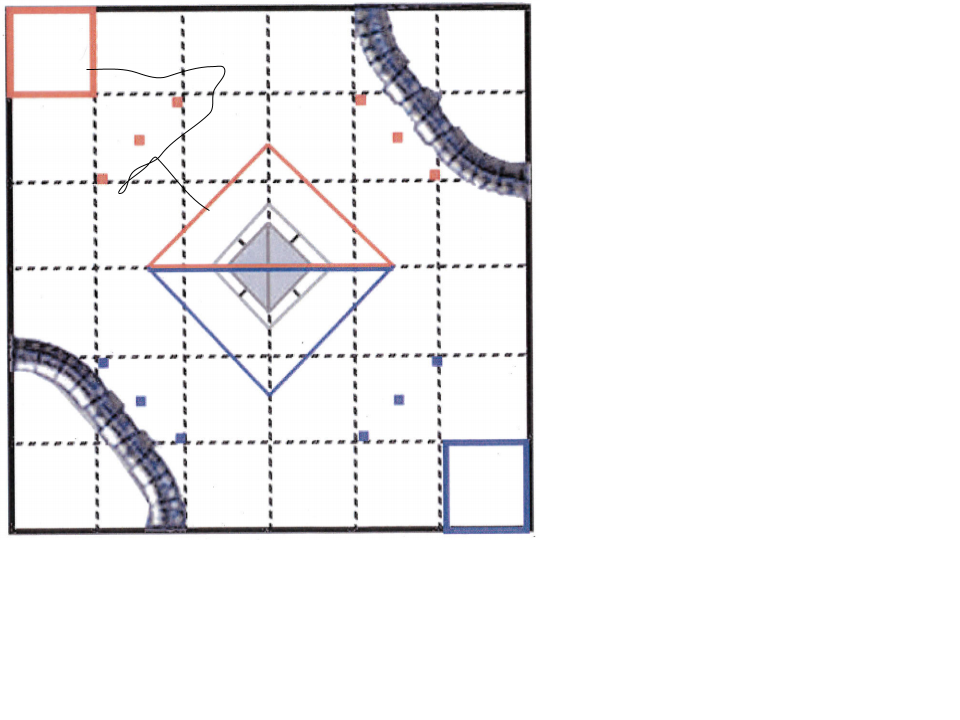
Rotate: Pulls the number from the textbox below it in order to tell robot how far to turn.

Place Marker Down: Tells the robot to drop the team marker.

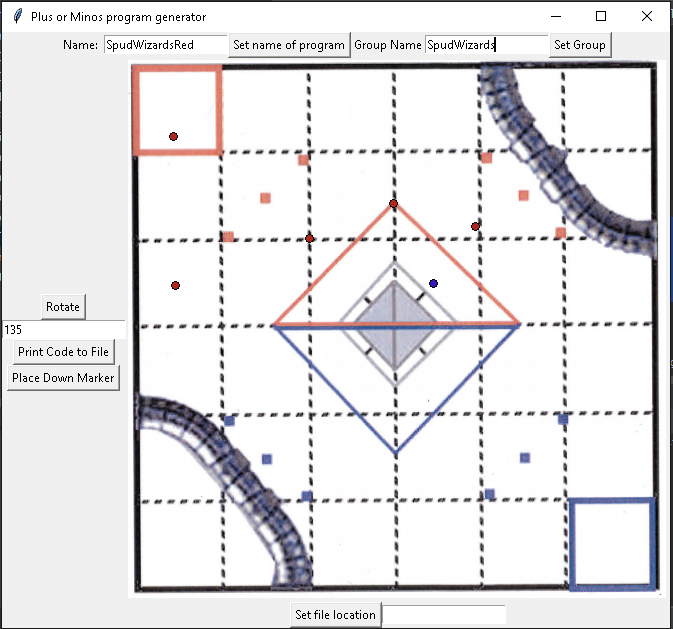
Example of use:

Let’s say we are allied with a team named the “Spud Wizards” for a round. Before the round(hopefully before the games start) we go over to them and ask them to draw their autonomous program on a printout of the field.

The Spud Wizards autonomous path



We take this drawing and make a complementary autonomous program around it



This layout will program the bot to go around the Spud Wizards bot while dropping our own marker.This program should complement the Spud Wizards program nicely and allow us to work together completely.

How it works:

Hephaestus is written in Python using the TkInter library as well as the OS and Math packages. TkInter is a object-oriented gui toolkit that we are using to create the screens and buttons that Hephaestus uses to create its programs.

The code for the position choosing screen using TkInter:

test = Tk()  
test.title("Choose starting condition")

framebox = Frame(test, width = 100, height = 100)  
framebox.pack()  
blueDepotButt = Button(framebox, text = "Blue Depot", command = test.destroy)  
blueDepotButt.bind("<Button-1>",blueDepot)  
blueDepotButt.pack(side = TOP)  
blueCraterButt = Button(framebox, text = "Blue Crater", command = test.destroy)  
blueCraterButt.bind("<Button-1>",blueCrater)  
blueCraterButt.pack(side = TOP)  
redDepotButt = Button(framebox, text = "Red Depot", command = test.destroy)  
redDepotButt.bind("<Button-1>",redDepot)  
redDepotButt.pack(side = TOP)  
redCraterButt = Button(framebox, text = "Red Crater", command = test.destroy)  
redCraterButt.bind("<Button-1>",redCrater)  
redCraterButt.pack(side = TOP)

test.mainloop()

The Math package allows us to use the equations needed to calculate the angle of driving and the distance it will travel. The OS package allows us to use accurate file locations and drop the file in the correct folder when the “Print Code to File” button is pressed.

Equations Used:

Finding The Angle:

₂ : The Y coordinate of the clicked position

₁: The Y coordinate of the previous location

₂: The X coordinate of the clicked position

₁: The X coordinate of the previous location

This equation allows us to find the angle between two points using a reverse tangent function (also known as arctangent). In order to use this function correctly we need to create an imaginary triangle. We do this by subtracting the two coordinate pairs together. We then divide these two “sides” of the triangle together, with the Y acting as the dividend and the X acting as the divisor. This gives us the tangent of C. We then plug the tangent of C into the arctangent, giving us the angle that we need.

Finding the distance:

₂ : The Y coordinate of the clicked position

₁: The Y coordinate of the previous location

₂: The X coordinate of the clicked position

₁: The X coordinate of the previous location

: The distance in ticks

The distance constant ()

In order to find the distance between two points, we need to use the pythagorean theorem. We take coordinates and subtract the two X’s and Y’s together. We take the differences and square them, after this we add the two numbers together. We then take and square root the result, this gives us the distance in pixels. In order to get the distance in the correct format we need to multiply it by the distant constant, which gives us the distance in ticks. In order for the robot to drive in the correct direction, we have to modify the angle with the angle the bot currently at.

The code to modify the angle:

if angle >= 0 and angle < 90:

angle = 360 - angle

elif angle >= 90 and angle <= 180:

angle = angle + 90

elif angle <= -90 and angle >= -180:

angle = abs(angle)

elif angle <= 0 and angle > -90:

angle = abs(angle)

angle = angle - botAngle

if angle < 0:

angle = angle + 360

angle += 90