1 Project: Main Phase Guard dog and GUI

In this project you will be creating a robot that drives around peacefully but when commanded acts like a guard dog. When commanded the robot will patrol the area you tell it to in a friendly manner but if an intruder enters the area they will get warned and if they do not remove themselves the robot will start shooting colorful happy balls at them. After this you will create a GUI (Graphical User Interface) to control the robot. In the end you will have a GUI that anybody can use to control the robot and perform different tasks.

The GUI should have two states "live state" and "building state". In "live state" anytime a button is pressed the robot will perform the action connected to the button directly. It will be like using a Xbox, PS controller to control the robot but in this case using the GUI on the computer instead.

In "building state" you will click a button which will add an instruction to a list or other data structure(list, dictionary etc) that you find best. When you are finished you will press a button that will run all the instructions in the list. This is the basic idea of programming, you build a program using instructions(buttons) and when you are done you press run and the program will be executed and perform all the tasks you told it to.

The GUI will also contain an area that shows how the robot has moved/move in the environment.

1.1 Milestone 1

Use the shooterbot.pdf to build the shooterbot. IMPORTANT!!! skip page 10 and 11 when building. These instructions are for the older NXT version of mindstorm but everything should be the same except for the colors of pieces.

1.2 Milestone 2

When you turn on the robot it should always start out in a playful/peaceful mode. Make the robot drive around randomly in the room and if it encounters an object it should make a sound and keep going another way.

1.3 Milestone 3

When you want your robot to start patrolling (from peaceful mode to guard mode) you will need to interact with the sensors somehow to let him know.

Find a way to make it possible for you to physically interact with the robot and fiddle with the sensor so it goes into guard mode. When it goes into guard mode it should do some kind of signature dance so you clearly can see that it is entering guard mode.

1.4 Milestone 4

After you built your robot you will notice it has more then 2 wheels or use tracks instead of wheels. Rewrite/remake the earlier functions you made in the exercises or find another way to make the new robot calculate turns correctly to be able to patrol later on.

1.5 Milestone 5

Know that the robot is in guard mode it will need to patrol and guard an area. Make it patrol a square of at least 1 * 1 meters repeatedly.

Explanation:

When entering guard mode and after it has done the signature dance it should start to patrol the given area(square) of at least 1 * 1 meter in front of it, you can assume it starts at the corner of the square. It is not enough that it runs around in a square repeatedly and only sees the edges of the square. It should patrol every space of the square at some point.

1.6 Milestone 6

If the robot spots an intruder it needs to be able to fire colorful happy balls at them. Create a function that uses the third servo motor to shoot away balls.

1.7 Milestone 7

When an intruder is spotted it should stop and give a warning sound. After the sound it should start an countdown and change the color of the light every two seconds until 6 seconds have passed, then initiate fire by using the function you created in milestone 5. If the intruder moves away so the robot cannot detect it anymore, the countdown should be stopped and the robot resume his patrolling from where it was before it detected the intruder.

1.8 Milestone 8

If the robot runs out of balls to shoot it can no longer scare away intruders. Keep an counter over how many balls you have fired and if no balls are left to shoot away, abort the current patrolling(exit guard mode) and make the robot make sounds until it finds a corner in the room to hide in then exit program.

Explanation:

The robot should be able to find his way to a corner somehow and be able to detect that it is a valid corner then move as close as possible before exiting the program.

1.9 Milestone 9

When you want your robot to stop patrolling and go back to peaceful mode, you will need to interact with the sensors somehow.

Find a way to make it possible for you to physically interact with the robot and fiddle with the sensor so it goes into peaceful mode. When it goes into peaceful mode it should do some kind of signature dance so you clearly can see that it is entering peaceful mode.

1.10 Milestone 10

Look at the tkinter wiki and do An Introduction to Tkinter to learn how to use it or try and find other examples.

1.11 Milestone 11

This is were you start creating your own custom GUI. Create the "live state", connect all the functions from the API.pdf to your buttons. Solve how to enter additional parameter information when clicking the buttons. Your goal is to be able to click a button on the GUI that in turns calls the API function with the given information, which makes the robot perform the task.

1.12 Milestone 12

Combine some functions to create a button that does something the API.pdf functions cannot do themselves. An example would be to make a button that makes the robot make a soft turn by using one wheel only.

1.13 Milestone 13

Add functionality so you can see what mode the robot is in (guard mode or peaceful mode).

1.14 Milestone 14

Create an area in the GUI showing how the robot has moved so far approximately. If you detect and intruder/object it should show up as an object on the screen to.

1.15 Milestone 15

Create the "building state" using the previous buttons plus a run button. You will now need to figure out how to add the buttons instructions to some data structure(list, dictionary), and when you press the run button all the instructions in the data structure you have chosen should be run/executed. A structure of lists containing lists might be suitable.

1.16 Milestone 16

The user wants to get an overview over what building instructions has been added to the list or chosen data structure. Add an area to the GUI showing what buttons/instructions have been added and in what order they were added. This is so the user can get an overview over how the program will run, look at Figure 11 to get an idea.

1.17 Milestone 17

The user wants to talk to robber or whatever the guarddog might encounter. Add som functionality so the user can enter some text and then press a button to make to guarddog say it.

1.18 Milestone 18

Add an if statement button/buttons and solve how to add conditions to an if statement and how to add buttons/instructions to be executed inside the if statement as long as it is true/false. Figure out a good way to display the if statement in the milestone 14 task, so the user can see how he built the if statement(the condition) and what instructions it contains.

Explanation:

The user adds a "forward" instruction to the list and then he wants to add an "if" statement with some condition and if the condition is true/false he wants the "if" statement to run an "Turn left" and a "forward" instruction then exit the "if" statement and continue with the rest of the instructions. And as said above figure out a good way to display the if statement/statements and conditions in the milestone 14 task.

1.19 Milestone 19

Add an while loop statement button/buttons and solve how to add conditions to a while loop block and how to add buttons/instructions to be executed inside the while loop as long as it is true. Figure out a good way to display the while loop in the milestone 14 task, so the user can see how he built the while loop and what instructions it contains.

1.20 Milestone 20

Add a feature so you can save the list with instructions(program) you have put together with a name, so you can restore everything as it was. Serialize your structure with the cPickle module might be a good idea.

Explanation:

User adds some instructions to the list then saves it with the name example_A, the user then creates a new list with instructions and saves it with the name example_B. After a while the user decides to load example_A which shows up and loads exactly as it looked when you saved it.

1.21 Milestone 21

Add functionality so you can see what speed the robot is traveling with.

1.22 Milestone 22

Add functionality so you can see how far the robot has traveled since the program started approximately.

1.23 Milestone 23

Remake the guard mode so that instead of patrolling 1 * 1 meters it finds a way to find identify all the corners of the room then proceeds to patrol the whole room. To make it easier you can assume there is no objects like chairs or tables in the room.

Explanation:

You turn on guard mode in which the robot finds a way to scan the room somehow so it knows how to patrol every inch of it.

1.24 Milestone 24

After each day at work guarding stuff like a champ the robot enjoys his hobby of chasing black lines on the ground.

Modify the robot so the color sensor is pointing towards the ground instead(remember to place it around 1 cm from the ground). You can ignore all the previous milestone you have done and start these features in a new file.

1.25 Milestone 25

The first thing you need to know is that when you follow a line, you are not following the middle of the line, but the edge of it. If you try to follow the middle of the line, and if the robots color sensor encounters white, you will not know which way it should turn. By following either the left or right edge, you will always know what the robot should do.

To determine what is the line and what is the background, you will need to determine what a good brightness value to use as the "cut-off" between the two states(light and dark). You can do this in two steps, first measure the center of the line (darkest) then measure the floor next to it (brightest). By saving the values you can then average them to find the cut-off.

Create a function that calculates the cut-off value and saves it for further use. You can assume that you always start with the robot on the line. You can always read more about this on the web maybe there is a better way of doing it.

If you want the robot to follow the line smooth you need some kind of control mechanism but this is above this course and completely volontary but you can check out. `http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html` for some ideas if you are interested

1.26 Milestone 26

Solve how to make the robot follow the black line on the testpad.

1.27 Milestone 27

Create a function that makes the robot turn around and keep going the other way on the testpad.

1.28 Milestone 28

Make it run as fast as it can without failing and then research online if there is a better way of doing it.

1.29 Milestone 29

Make sure you robot can handle very sharp turn. Put out duct tape or cut out a piece of paper and make a 90 degrees line turn, make sure the robot can pass this turn.

1.30 Milestone 30

Solve how to make the robot go around a object and continue if it should encounter something.