MISSION #3

Reactive/Functional Control, State

Due: Week 5 & 6

Assignment 1.2: The Smart Wanderer

Task: Write the purely reactive program called ***SmartWander()***

Description: When called, the robot will explore the environment without hitting anything. The robot should always continue to explore; it should never enter into an endless loop such as repeatedly bouncing between two obstacles.

SmartWander() should demonstrate both translational (GoTo()) and rotational (TurnTo()) movements. We will test your program not only by placing the robot in a static environment populated with walls, but also by attempting to steer your robot around. This means your robot will be faced with moving human beings. We will test this behavior by setting the robot loose in an area with an arbitrary scattering of walls and people. The robot should wander smoothly without hitting anything. We'll be looking for the fastest robot that exhibits smooth and safe behavior. ***Safety is critical; speed is desirable.*** Don't let your robot get stuck in a silly infinite loop, which will cost you.

Your code must be totally functional. This means that you may only use the current infrared sensor readings to determine your new velocity. You WILL receive bonus points if your robot plays any sounds at all.

Improve on your wandering program by adding state. Use state to ensure that the robot will never, ever become stuck in a silly situation forever. Your challenge now—and this is indeed a challenge—is to actually improve on your stateless wandering program. So ***find a fault and use state to make it better***. This is easy to do, but the real trick is doing this without accidentally ruining all the accidental good behavior of your reactive program at the same time. In class, you will be describing how you used state adjustedly to improve things rather than horribly denature things. As a special bonus, you will receive extra credit if you choose to do the advanced version, in which, rather than adding state to improve your wander, you design your wanderer so that it adjusts its parameters automatically to improve itself!

Hints: Start with something simple. The KISS (Keep It Simple, Stupid!) principle is very important for this assignment. You may want to start out by using only a small number of infrared sensors for the functional programs.

SIMBot

Assignment SSW: The SIMBot Smart Wanderer

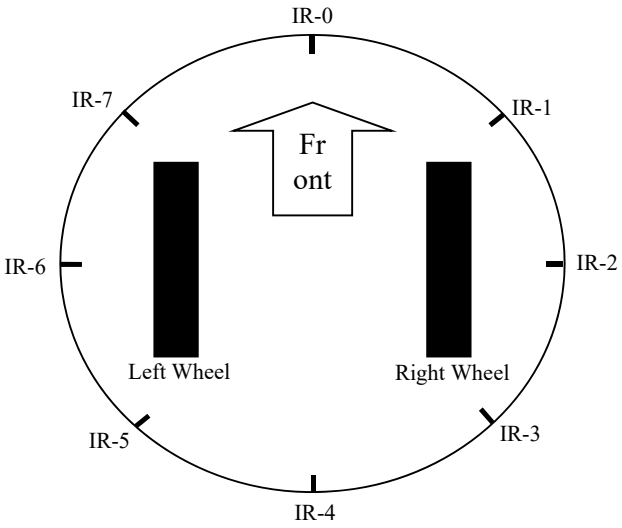
Task: On simulation Robot, write a reactive control program with limited capacities; so that the robot has to move pass the obstacles toward the food as fast as you can.

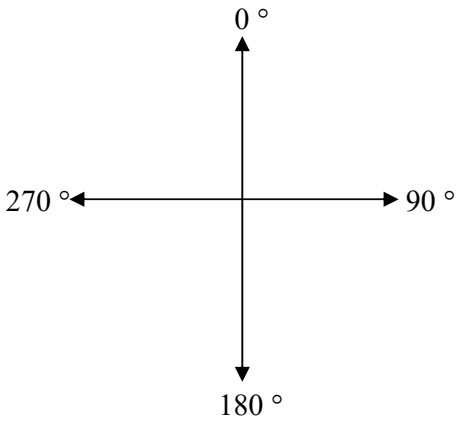
Description: Let's design a reactive control for **a very simple simulated robot** (**PySimbot**: you can download from the class website). In this assignment, we will give you simulation robot codes (in C++, of course), which you will work on it. When we are talking about a reactive control system, we are talking about the system that uses the current input information to select the actions without looking ahead or planning. When called, the robot will explore the environment without hitting anything. The robot should always try to search for the food; it should never enter into an endless loop such as repeatedly bouncing between two obstacles.

There are some limitations and specifications about the robot we are using, as follows…

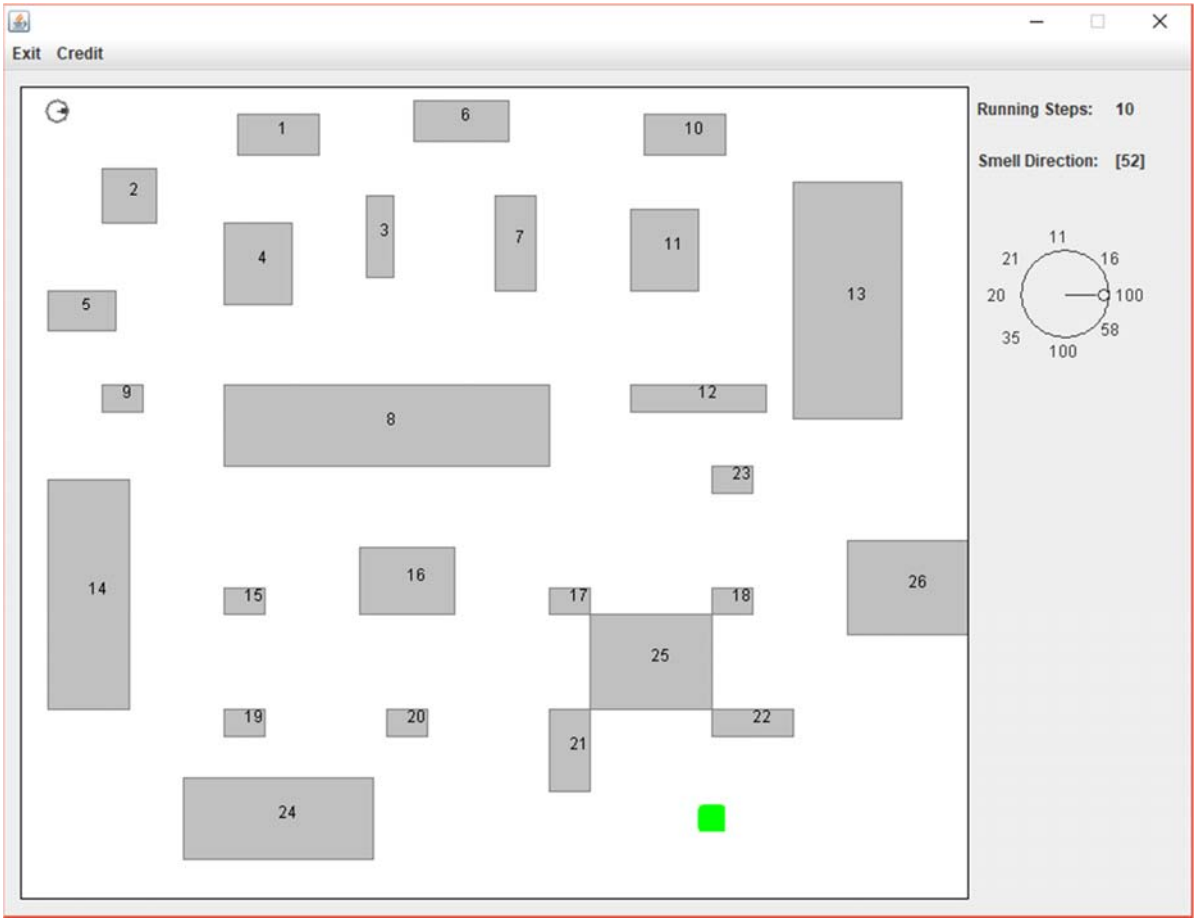
- The robot has only 8 infrared sensors (IR-0, IR-1, IR-2, …, IR-7) locating around the robot. Each sensor reads the distance to the nearest object in its direction. The possible value is 0 to 100 pixels. It means the sensors are only checked for the 100 pixels ahead in its direction.
    - The sensory information can be gathered by calling **getIRValues()**. Then, the updated sensory information are in a array variable of IR[8]. The index is used relating to sensor's position on the robot. For example, we use IR[0] keeps the information of sensor IR-0.

- The direction of the robot is referred to its head position which is referred to 0 degree. The other directions are then referred in clockwise direction.
    - In addition, the robot can smell for direction of food. By calling **smellAllFoods()**, the robot can get the direction of food referred to the heading position of the robot.
    - For the actions, there are 3 commands available to use. There are **moveForward()**, **spinLeft()**, and **spinRight()**. Please use only these commands.
    - The robot program should be written in a function called **execute()**. You can do or modify whatever you want to program it moving.





- The environment is shown below. The robot is a circle with a small circle on it. The small circle indicates the heading position of the robot. There are 26 obstacle boxes on the field. The robot cannot move pass it. If the robot is forced to move into the obstacle it will force to return the previous position. The green square on the right is the food and it can be smelled by the robot. The robot will get a direction (reference to the direction of its heading) to the food when it tries to smell.
- The environment is static. All obstacles are not moving during execution. The food is also not moving. The robot start position and head direction are the same every execution.



- Your task is

Writing a reactive control program with limited capacities; so that the robot has to as fast as it can.

- When we talking about reactive control, we are talking about the system that uses the current input information to select the actions without looking ahead or planning.

We may have ***thing is please do it by yourself***. Let's have fun with it. Don't worry about the score toward your robot performance. I love to see you efforts in solving the problem the most. At the end, you will learn some discuss about who got the best robot program. ***Please do it wisely and the most importance*** a lot. ☺