Design Document for Interfaces

Hailong Zeng

November 28, 2017

1 Introduction

1.1 Project Goal

For the iteration 2, we need to add up five more autonomous robots that will generate a random heading angle and speed at the beginning of the game. Meanwhile, they will have a cone view emanating from its center. If some entity is scanned inside the cone view, it will return the type of the entity to the autonomous robot. The autonomous robot will react base on the returned type of the entity. The autonomous robot basically is looking for homebase robot and escaping from the player robot. I will using observer pattern to make the autonomous robot to behave as they are expected.

1.2 Observer Pattern

For this project, I will use the observer patter design. It is pretty obvious that the Arena is the subject, since most of the updates happen in there. Inside the arena, there will have a vector to register the observers, which are the different sensors. Whenever a event happens, Arena will register the corresponding sensor to the register vector. At the end of update, Arena will iterates this vector to let each sensor pull out the information that they need and do the updates for themselves.

1.3 Subject

In this project, Arena is the subject, because it is the place where gathering and updating all the information. Specifically, Arena is responsible to iterate every entity to update one time step. During the updating one time step, Arena also able to check whether there will have an event. Then it is where Arena push the corresponding sensor to the register vector.

1.4 Observers

The observers in this case will be the different kinds of sensors: sensor touch, sensor proximity, sensor entity type and sensor distress. Sensor touch deal with

the collisions, such as the mobile entity collides with the boundaries and other entities. Sensor proximity shapes as cone emanated from the center of the robot, it will sense the entities' type in this cone. Sensor entity type will find out the type of the entity who sent the signal, however the robot will not know where the signal from. Sensor distress will send a signal for help when the robot is frozen. These sensors are all derived from sensor base class which will contain a Accept virtual function that will be implemented in all these sensors. Thus, we could built a sensor base vector to work as the register vector and do the updates.

2 Design picture

2.1 In Arena

In the function UpdateEntitiesTimestep, I would use for loop to iterates every mobile entity in the Arena. Thus, for every entity(inside the for loop), I would declare all possible happening events. If the event happens, I would activate the event and push up the corresponding sensor to the register vector. After iterates all the entities, the register vector should contain all different kinds of sensors. Therefor, I would go through this vector to make the sensor get the information they need and do the updates.

2.2 In Sensor Touch

The sensor touch will accept the parameter eventcollision. When the boolean value activated is true, it will send the calculated heading angle to the its related robot and reset the heading angle to finish the task and set the boolean value activated as false.

2.3 In Sensor proximity

The sensor proximity accept the parameter eventproximity. It will also check whether activated is true. Only when it detected that there is a entity appears in the cone, the activated will be set to true. If the activated is true, then it will get the type of the entity that is detected and return to its related robot. This sensor also accept distress signal. When the cone view sensors the distress signal, it will return the position of the distressed robot to its related robot.

2.4 In Sensor Distress

The sensor distress is fairly easy. The sensor accept the parameter eventdistress. When the activated is true, it will pass zero to set the robot speed as zero. Then it will send signal and its position in a certain range, which will locate itself and ask for help.

2.5 In Sensor Entity Type

The sensor entity type accept eventtype. Within a certain range of other entity, this sensor would get the type of the entity who sent the signal but don't know where the entity is.

2.6 Collisions

2.6.1 Player Robot

player/obstacle:player robot bounces to other direction.
player/recharge station:player robot bounces to other direction and battery refilled to max.
player/boundary:player robot bounces to other direction.
player/home base:player robot bounces to other direction, home base bounces to other direction.
player/autonomous robot:player robot bounces to other direction and autonomous robot frozen.
player/superbot:player robot frozen and superbot bounces to other direction.

2.6.2 Home Base

home base/player:home base bounces to other direction, player robot bounces to other direction.
home base/obstacle:home base bounces to other direction.
home base/boundary:home base bounces to other direction.
home base/recharge station:home base bounces to other direction.
home base/autonomous robot:home base bounces to other direction, autonomous robot becomes superbot.
home base/superbot:home base bounces to other direction, superbot bounces to other direction.

2.6.3 Autonomous Robot

autonomous robot/obostacle:autonomous robot bounces to other direction.
autonomous robot/boundary:autonomous robot bounces to other direction.
autonomous robot/recharge station:autonomous robot bounces to other direction.
autonomous robot/player:autonomous robot becomes frozen, player robot bounces to other direction.
autonomous robot/home base:autonomous robot becomes superbot, home base bounces to other direction.
autonomous robot/autonomous robot:
case 1:frozen autonomous robot unfreezes ,another autonomous robot bounces to other direction.

case 2:autonomous robot bounces to other direction ,another autonomous robot bounces to other direction.

2.6.4 Superbot

superbot/boundary:superbot bounces to other direction.
superbot/obstacle:superbot bounces to other direction.
superbot/recharge station:superbot bounces to other direction.
superbot/player:superbot bounces to other direction, player robot frozen for a certain time.
superbot/autonomous robot:superbot bounces to other direction, autonomous robot bounces to other robot.
superbot/superbot:superbot bounces to other direction, another superbot bounces to other direction.