Design Documentation

In Iteraition2, I would like to consider the mobile entities as observers and almost each mobile entity has at least one sensor. The reason why I design in this way is that those entities activate on the arena; they will notice what happened via arena (just as observers since when update time steps, they accept the all kinds of event), and then they should accept those situations to make some adjustments on what they should do next. So, sensors as the part of those mobile entities is used to pass related information (e.g. angle, position, angle, and so on) to their own motion handler. And then, we can decide which sensor should accept those events separately, so entities' motion handlers will control the relative mobile entity properly.

I'll explore more on the sensors stuff; there are four sensors total in this iteration.

First of all, I want to discuss how to deal with Sensor Touch. Sensor Touch is inherited from Sensor base. I will replace Boolean value with integer when representing whether this sensor is on or off. As we know, Player, Robot, Superbot, Homebase, are mobile entity and for each of them has a sensor touch. It's similar to what we did in iteration 1, but check the position information to know if there is a collision by touch sensor itself. Those position information is from the collision event. Every mobile entity has chances to collide with walls, immobile entity (recharge station, obstacles), and each other. But they will have different reaction when having collision with different things. Therefore, in the sensor: 1. Check out of bounds for all mobile entity 2. Check the entity collision between each other. But in second checking, I would check some special case (1) player with recharge station. (2) player with robot

(3) player with superbot (4) robot with recharge station (5) robot with homebase (6) robot with robot (to unfreeze) (7) robot with player (8) superbot with player.
Those conditions should be considered when implementing the functionality.

Next, I am going to think about the Distress Sensor, which belongs to robot and superbot and inherited from Sensor base. From the description, we know that only the robot and superbot have the distress sensor. And what it does is when robot is frozen by some collision, this robot will send a distress call with in a range. So, I will set the range to that distress call and also denote whether this sensor is on or off based on the distress event from the arena. Therefore, we should set distressed call by checking its speed in the arena; if the speed is 0, then this robot tells its distress sensor to be activated with a given range.

Moreover, I think the entity type sensor is in every entity appeared in the arena and is used to let others know what the type of itself with in a limited range, since only arena know who they are, but they don't know each other. So, it's similar to distress sensor, but the only difference is that it will always be activated, and the output of this sensor is enumerated entity type with a given range, and this sensor receives the event to output the correct enumerated entity type.

In the end, the proximity sensor is a more difficult one to consider how to implement it. According to the requirements, only robot and superbot have this sensor. And the proximity computation is also the most important function in this one and I think every time it receives an proximity event, then to compare the position of the current entity and other entities to make sure whether they are close or not by proximity computation function. Notice that, if the robot or superbot in the range of this entities' emit type sensor, then emitting event will be given to

the emit type sensor to return a type; If this entity is the Player, then check if the robot or superbot is close enough via use proximity sensor. If it is, the motion handler will decrease its speed and adjust its heading angle to avoid having collisions until the proximity sensor return -1, since if the output for proximity sensor is -1, we don't need to adjust the speed and heading angle anymore. If a robot can get the closer entity type is another robot, more over if this robot still can't get the distress call from that robot, which suggested that it is not a frozen robot. Then change speed and adjust its heading angle to avoid having collisions until the proximity sensor return -1. Otherwise, if this current robot sensed the other one is frozen robot, we should keep make it go to unfreeze that frozen robot by motion handler with a fixed heading angle.

After talking about sensors, then I want to think about how I can control the numbers same kind entity. In the iteration1, we give Id to each entity, and since we will define the different motion handler for mobile entity, so I will pass the id number in the related motion handler to control the related entity just like lab 4 we did before.

At last, I want to talk about some trivial things. Initialization is extremely important thing, because you want users can see what happened at the first glance and they can't see the source code for security, so what I have done is to let users write in a file where they can put what entities (with pos, color, radius, and so on) they want. And anther thing
is that when robot collides with homebase, this robot become superbot. And if this collision happened, I will
replace the robot with the superbot at original position and keep the same id of this robot.