Abdirahman Sheikh-Mohamed (sheik086)

3081W, Fall 2017

Design Document

The way I approach this iteration is that I created all the sensors and motion handlers in the requirements, and also implemented the the robot class and a superbot class that I will have inherit from robot. One design I'm thinking about changing is whether I want to have the robot inherit from the homebase currently or just have it inherit from arena mobile entity, since it has much different sensors than both the homebase and player. With the major changes from the first iteration along with the initializations which I implemented by creating a text file in the build/bin folder that you initialize everything in the area, and have the main read from that file.

The most difficult sensor to implement was the proximity sensor, which from what I seen was incorporated in lab 4 that we did earlier this semester. This helped with making the sensor class for proximity and the visual representation of the sensor. I attempted to make the proximity sensor have two sensors; a left and a right. This would make it easy to know which way to turn when an entity entered the proximity sensors range. Also had a range and distance variable, range basically being the field of view for the sensor and the distance being how long is the field of view. The event proximity checks to see if anything is within the robot’s radius + range, if so the event is activated. Then it is handled by the motion behavior to turn the robot right or left depending on which sensor was also activated of the sensor in that event that another entity entered the its proximity. The sensor class also has a on and off switch, simply being a 1/0 in variable. That is used in the arena if the entity in the proximity of the robot is another distressed robot or homebase, it will turn off the sensors in that case and will not pass an event.

The distress sensor has a range, distance, and switch variable in its sensor class. Which similar to proximity checks if an entity is in it’s range and how long its range is depending on the distance. While the switch is to check if the robot is in distress or not. So in event distress class it also checks for a event collision because that is the only times the distress event occurs depending on the entity it collides with. So the event distress changes the switch from 0 to 1 if a robot collides with a player and from 1 to 0 if collides with another robot or superbot. This mainly happens in the arena class since it is aware of everything that is going on.

The entity sensor simply takes in the enumerated types that where initialized in entity\_types file. The sensor accepts an event that once in the range and distance of the entity it will be notified what kind of entity you are through the event entity type class. Which returns the type of entity you are and sends that to all entities within your range.

The implementation of the Superbot that is made once the robot collides with the homebase was simply to delete the robot that collided with the homebase from the vector of robots and create a new instance of a superbot with the same position information as the robot. So you pull the robot out of the vector and create a brand new superbot in the same location with superbot parameters that inherit from robot. Robots were also implemented in a way that was a combination of both obstacles and a arena mobile entity as in, it was created as a vector in the arena but the difference being it is also a mobile entity. So the robot took in many of the same functions as a player minus the fact that it does not have event key presses or a battery. So the robot took much of the same functions as arena mobile entities and a few of the robots motion handlers to handle its sensors such a proximity and touch sensors. The implementation for homebase, player, and recharge station didn't change, so no design changes done to those.