DesignDocument

SRIRAM MADDIRALA

March 2018

1 Observer Pattern Design

One implementation of the observer pattern would be where the subject is arena and the robot is the observer. In this case the robot would (in its handle collision) check the entities it is colliding with and then update the sensors according to the status of the entity it is colliding with, by passing the other entity to the sensor, and its own profile information like behavior pattern and position and then update the motionhandler through the sensor. The information about the other entities would have to be sent to Robot and the Robots would have to check each pair for collisions or proximities and direct the sensors behavior to update motion handler. The sensors would be passed the particular motion or event that has occurred i.e. CollisionWithLight, CollisionWithRobot, CollisionWithWall and CollisionWithFood. Upon updatetimestep the position of the entity will be modified by sensor through sensor acting on motionhandler which acts on motion behavior which updates the position of the entity. Another implementation of the observer pattern is where the subject is Arena because it has all the data about all the entities, while the sensors are the observers. This would work by having the individual sensors of the entities receive communication from the arena on what type of collision has occurred which could vary like CollisionWithLight, CollisionWithRobot, CollisionWithWall and CollisionWithFood. The sensors could be stored in the robot which could be accessed by the arena and passed information to through the arena. When two entities are colliding through the communication class the Arena would have them adjust accordingly. This means that if the entity is a Robot and has fear, the reading from the right sensor impacts the velocity of the right wheel and the left sensor impacts the left wheel with respect to the lights. If the entity is a Robot and has exploratory behavior then the reading from the right sensor impacts the velocity of the left wheel, and the left sensor impacts the right wheel and the left sensor impacts the right wheel with respect to the lights. If the entity is a robot and senses food it should be aggressive i.e. the sensors have a positive correlation between the right sensor and left wheel, and between the left sensor and right wheel. If the entity is a Robot and is really hungry then it should ignore and go through the lights. If the robot isn't hungry then it would not go towards the food even when it senses it. If the Robot hits food or light it should be able to pass through it but if the robot hits the wall or another robot it should move

back in an arc which would be implemented in sensor through calling functions of the motion-handler at different timesteps i.e. a state machine. After the required action is taken and are updated to the new state the arena would call timestepupdate on all the entities which would update position depending on the motionhandler which had been updated by their sensors after the sensors reached their new state. This would require iterating through every entity for every mobile entity and check for collisions and then send the relevant communication depending on the entities involved. I prefer the pattern where the subject is the Arena and the observers are the sensors. This is because it would allow division of the information that robots would hold i.e. information about other entities would not exist in a given entity and would put the burden on the arena which already holds information about other entities in it. It would also allow the Arena to work on updating the sensors on collisions rather than the entities which makes sense as sensors are supposed to be handlers of collision instead of the Robot and this cuts the Robot out entirely. One issue I do have with the chosen implementation is that the preprocessing and decision-making of iterating through all entities and happens in arena which might be not be in line with what the observer pattern necessitates as it might be more in line if all the information about all the entities are passed to the sensors for them to parse through them and conduct all the processing. I prefer doing it in arena, however, as each sensor would then have to conduct preprocessing so it would be more efficient to do it in Arena.

2 Tutorial

To detail how to add a new stimuli and sensor I would have an introductory logic section to explain how to do it on a conceptual level and why, then I would have a section that would detail the changes to be done in the arena, ArenaEntity and then a section to detail the changes to be done in the sensor and then changes to be done in the entity. Simply put the stimuli would inherit from ArenaMobileEntity or ArenaImmobileEntity depending on whether it's mobile or not. The stimuli would be made in entityfactory following the model of the other entities and then a sensorStimuli would be added to the mobile entities which would inherit from the sensor class and would be responsible for implementing the changes necessary when a collision occurs in its handlecollison/handleproximity function which would be communicated to it from arena when arena checks for collisions through iterating through every pair of entities and checking for collisions with all stimuli and once the new stimuli is detected that information would have to be communicated to the sensorstimuli through a new communication that would have to be defined in communication and then sensorstimuli would have to be implemented as above detailed.