Design Document

the use of the Observer Pattern

One way to implement observer pattern is to take sensors as the observer and Arena itself as the subject. All information is measured or got in Arena;The distance between vehicle and stimulus will be calculated in the class Arena. Arena will get he position of stimulus and measure the state of vehicle. Every time something changes in Arena, it would give notice to the sensor, to be specific, notifiy() calls the update() method in the class Sensor(). I will pass along the position of entities to an sensor, which is an interface, and then let the light sensors get these information.

an alternative implementation

Another way to implement observer pattern is to take all entities as subjects, and all sensors as observers. The abstract class Subject() contains a list of all sensors being observed; It is an interface used to register objects as observers and also remove objects from being observers. When something changes in our Subjects that observers may be interested in, notifiyObservers() calls the update() method in the class Sensor(). Sensor of light will received infomation from the Light object which inherit from the Subject. Robots, foods, and lights are all subjects. These classes inherit from both the Arena entity and the Subject. They are concrete subjects that implements the Subject interface. Besides register and remove functions, it also implement notifyObservers() that is used to update all the observers whenever the state changes. They have setters and getters to get the infomation. For example, when a robot gets really hungry. It will notify the Sensor to ignore the light and just sense food.

I will use the data in Arena for sensors.Because the data and stimulus are convenient to get in Arena. We just need to register and delete the sensors properly. Every object can implement the Sensor and get updates from the Arena.

outline of the tutorial

- refactor the light, food class.
- add the sensor class
- in the sensor class, add update() to update the position of entities
- add notifiy() method in sensor interface
- initialize sensor in Arena
- add register and remove method of sensors in Arena
- set the status
- modify the handle_collision method in Arena, so that light though can pass robot and food.
- add the parameters to: 1. record status of robot if it is hungry. 2. exhibit different behaviors