Overview of Changes:

Sensor:

-Sensor becomes polymorphic, with different sensors keeping track of different types of things.
-Each sensor maintains a state which reflects its proximity to what it is supposed to be keeping track of.
-Sensors report data back to the controller, which acts as a brain, translating sensor output into commands to send to robot.

Light:

Food:

Base:

Arena:

<u>Sensor Implementation:</u>

This Section is devoted to the implementation of the sensor class. What follows is a discussion of different options considered for the design and which design was ultimately chosen. Factors in the decision include: ease of implementation, clarity of design to users and future programmers, and efficiency.

Option A: Sensor as Polymorphic Eyes and Nose

The sensor class will be polymorphic, and act as a base class for two other, more specific classes of sensor. One sensor class is a “nose” sensor, which keeps track of the proximity of food sources and lets the controller/robot know how close the nearest food source is. The other sensor class is an “eye” sensor, which keeps track of the magnitude of light shining on it (or the proximity of each light) and reports that back to the controller/robot. In all cases, the robot acts as a subject, changing state based on stimulus from the arena, with the controller/robot acting as an observer for those changes and responding by changing its wheel velocities accordingly. This approach makes it so that sensors are keeping track of objects, which seems like the proper way to organize things, but a lot of information is kept track of by the sensors in this implementation, which could be a potential point of confusion in the future.

Option B: Sensor as Inbox for Messages from Other Arena Entities

The sensor class receives messages from obstacles and food (acting as subjects) containing the proximity of that object to the sensor and the type of object. Each time an arena_entity updates, it broadcasts out a message to all sensors in the arena, the sensors act as observers for this message and then decode it and relay the information to the robot/controller. This approach is advantageous because many objects can easily be created and deleted in the arena implies that every single entity needs to know about every sensor, a potential problem with coupling.

Decision:

Ultimately, the developers decided to go with Option A, because it follows more closely the way a robot should behave in the real world. In addition, the developers thought it inappropriate that all entities in the arena should know about the location of the sensor, as they don’t do anything with the information other than send messages back to the sensor. The robot will be modeled on an actual organism, being possessed of sensing organs (sensors) which monitor the world around them, report their findings back to the brain (controller), which in turn issues commands to the body (robot) so that the organism can respond to the stimulus. This is also advantageous because it almost completely isolates work towards revamping sensor to the sensor.h and sensor.cc.