The Observer Pattern choice

There are different ways to share information with sensors. The first approach is to use the Observer Pattern in which sensors are observers and stimulus are subjects. Each stimuli will have a list of sensors in the arena. Whenever a stimuli updates its status, it will notify all sensors to the arena about its position, type, etc. Each sensor then decides what to do with the information it just received. To implement this, two new classes - Subject and Observer - will be added. The Subject class will have a list of Observer objects as its member. It also provides a method to register new observer, a method to remove an unwanted observer, and a method to notify all of its observers when a status gets updated. The Observer class will have a method to manipulate the data it has received from the subject. Then, the programmer will create a new class for the new stimuli and allow it to inherit from both the ArenaMobileEntity (or ArenaImmobileEntity depends the motion behavior of the new stimuli object) and the Subject class. The sensor class in turn will inherit from the Observer class because sensors keep waiting for the entities' signals to take the next action. One modification to the Observer Pattern is that the Sensor class now has a list of stimulus which it wants to register to in the arena. As the result, every stimuli in the arena will be able to notify all sensors when it updates its status, e.g. position, size, type, etc. One advantage of this approach is that it creates a convenience for a programmer if they want to add a new kind of sensor such as light sensor. All they need to do is to create a corresponding class for the new sensor and let it inherit from the sensor class. Since the Sensor class has a list of stimulus in the arena, the new type of sensor class can reuse this data.

Another approach is to let the arena notify all sensors at once. After advancing the time for all entities, the arena will send the information of each entity to all sensors. To implement this, besides a list of entities, the arena should also have a list of sensors to which it will notify. This can be accomplished by adding a new member, e.g. sensor_list_, in the Arena class. The Sensor class also has a reference to the arena so that it can register to get the updated information. This approach does not use the Observer Pattern so no new classes need to be added. When a programmer wants to add a new type of sensor, they only need to create a new class and make it inherit from the sensor class.

Tutorial Outline

In order to add a new stimuli to the simulator, the programmer needs to create a new class for that stimuli and let it inherit from both the ArenaMobileEntity class (if it is a mobile stimuli) or the ArenaImmobileEntity class (if it is an immobile stimuli) and the Subject class which is part of a Observer Pattern. The programmer also needs to implement all method from the Subject class because the Subject class only acts as an interface. If it is a mobile entity and it has a different motion behavior, the programmer also needs to create two new classes for the motion handler and motion behavior of the stimuli. The new stimuli class then requires to have these new motion controllers as its members.