Joey Bloom

12/4/13

AI proposal

My project will be a zoo. The zoo will be some environment with houses/cages, human spectators, rocks or other obstacles, and of course, zoo animals. These entities can be placed with the mouse onto a bird’s eye view of the zoo.

Entity is the main interface of the project. All entities will have properties like smell, color, size, and location, and other entities can get the values of those properties as if by sensory input and use the values of those properties to determine appropriate action.

The zoo animals will be the focus of the project. The zoo animals will extend the abstract AbstractAnimal class, which itself implements Entity. This way, animals interact with objects like rocks and human spectators and cages in the same way that they interact with other animals.

Animals evaluate entities by “scoring” them based on their properties. Small, distant, entities with a low smell value and a low visibility value (meaning they are well camouflaged, or otherwise hidden from view) will generally be scored low, and large, close, smelly, obviously visible entities will be scored high. Low scored entities will be ignored by the animal, while high scored entities will cause the animal to take some action, whether it is running, hiding, eating, playing, or other action.

It is in this scoring system that the artificial intelligence enters the program. Animals use the scores to rank each Entity they encounter, according to the scoring algorithm for that implementation of Animal. Animals then store the scores in a TreeMap<Entity,int> and then iterate over the map taking action according to the relative scores of the Entities. My hope is that some emergent properties will come about from simple algorithms, such as predator-prey relationships or packs of animals.

|  |  |  |  |
| --- | --- | --- | --- |
| **interface Entity** | **Rock implements Entity** | **abstract class AbstractAnimal implements Entity** | **Cage implements Entity** |
|  | -int smell  -Color color  -int visibility  -int size  -int x  -int y | -int smell  -Color color  -int visibility  -int size  -int x  -int y  -TreeMap<Entity,int> | -int smell  -Color color  -int visibility  -int size  -int x  -int y  -AbstractAnimal hidden |
| +int getSmell()  +Color getColor()  +int getVisibility()  +int getSize()  +double getDistanceFrom(int,int) | /\*no methods beyond Entity interface; rocks don’t do anything\*/ | -int score(Entity)  -void move(int,int) /\*Entity interface methods…\*/ | -void hide(AbstractAnimal)  /\*Entity interface methods…\*/ |

Ideas:

* Instead of TreeMap, use a PriorityQueue<Entity> with a Comparator<Entity> that compares by looking up scores in a HashMap.