Moti Begna

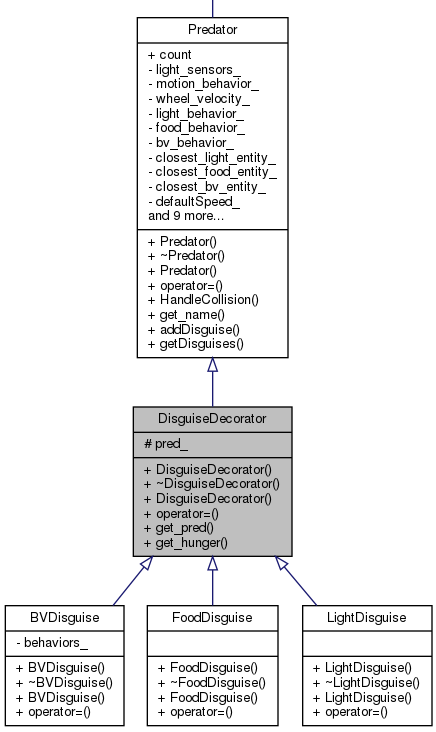
CSCI 3081

4/25/2019

Iteration 3 Preliminary 1

**Doxy Generated UML for Decorator Pattern**

My implementation of the Decorator pattern involves having a parent DisguiseDecorator class inheriting from a Predator, which derived disguises inherit from.



**Code snippet of decorator implementation and hunger implementation**

|  |
| --- |
| void Arena::UpdateEntitiesTimestep() { |
|  |
| … |
| for (auto ent : entities\_) { |
| ent->TimestepUpdate(1); |
| } |
|  |
| /\* Determine if any mobile entity is colliding with wall. |
| \* Adjust the position accordingly so it doesn't overlap. |
| \*/ |
| for (auto &ent1 : mobile\_entities\_) { |
|  |
| // initialize disguise |
| if (ent1->get\_type() == kPredator){ |
| Predator\* pred = static\_cast<Predator\*>(ent1); |
| // increment hunger |
| pred->set\_hunger(pred->get\_hunger() + 1); |
| if (pred->get\_hunger() == 150){ |
| int disguisetype = rand() % 3; |
| pred->addDisguise(disguisetype); |
| switch(disguisetype){ |
| case 0: |
| ent1 = new FoodDisguise(pred, pred->get\_hunger()); |
| break; |
| case 1: |
| ent1 = new LightDisguise(pred, pred->get\_hunger()); |
| break; |
| case 2: |
| ent1 = new BVDisguise(pred, pred->get\_hunger()); |
| break; |
| default: |
| break; |
| } |
| } |
| } |
| if (ent1->get\_is\_disguised() == true) { |
| Predator\* pred = static\_cast<DisguiseDecorator\*>(ent1)->get\_pred(); |
| // increment hunger |
| pred->set\_hunger(pred->get\_hunger() + 1); |
|  |
| if (pred->get\_hunger() == 300) { |
| int disguisetype = rand() % 3; |
| std::vector<int> vec = pred->getDisguises(); |
| // don't select a previous disguise |
| while (std::count(vec.begin(), vec.end(), disguisetype)) { |
| disguisetype = rand() % 3; |
| } |
| pred->addDisguise(disguisetype); |
| // delete old disguise |
| ArenaEntity\* container = ent1; |
| ent1 = static\_cast<DisguiseDecorator\*>(ent1)->get\_pred(); |
| delete container; |
|  |
| switch(disguisetype){ |
| case 0: |
| ent1 = new FoodDisguise(pred, pred->get\_hunger()); |
| break; |
| case 1: |
| ent1 = new LightDisguise(pred, pred->get\_hunger()); |
| break; |
| case 2: |
| ent1 = new BVDisguise(pred, pred->get\_hunger()); |
| break; |
| default: |
| break; |
| } |
| } |
|  |
| if (pred->get\_hunger() == 450) { |
| int disguisetype = rand() % 3; |
| std::vector<int> vec = pred->getDisguises(); |
| // don't select a previous disguise |
| while (std::count(vec.begin(), vec.end(), disguisetype)) { |
| disguisetype = rand() % 3; |
| } |
| pred->addDisguise(disguisetype); |
| // delete old disguise |
| ArenaEntity\* container = ent1; |
| ent1 = static\_cast<DisguiseDecorator\*>(ent1)->get\_pred(); |
| delete container; |
|  |
| switch(disguisetype){ |
| case 0: |
| ent1 = new FoodDisguise(pred, pred->get\_hunger()); |
| break; |
| case 1: |
| ent1 = new LightDisguise(pred, pred->get\_hunger()); |
| break; |
| case 2: |
| ent1 = new BVDisguise(pred, pred->get\_hunger()); |
| break; |
| default: |
| break; |
| } |
| } |
|  |
| // kill predator aftet 600 iterations |
| if (pred->get\_hunger() == 600) { |
| pred->set\_type(kPredator); |
| pred->set\_is\_alive(false); |
| } |
| } |

…