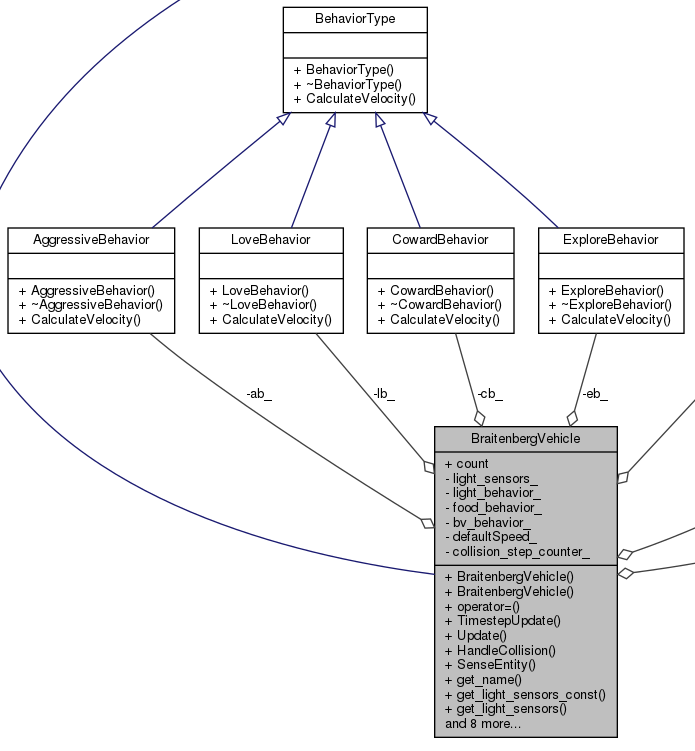
Moti Begna

CSCI 3081

4/1/2019

Iteration 2 Preliminary 1

**Doxy Generated UML for Strategy Pattern**

****

**BV method using Strategy/Behavior**

|  |  |  |
| --- | --- | --- |
| void BraitenbergVehicle::Update() {  WheelVelocity light\_wheel\_velocity = WheelVelocity(0, 0); | | |
|  |  | | | |
| int numBehaviors = 3; | |
|  | | | |
| switch (light\_behavior\_) { | | | |
| case kExplore: | | | |
| light\_wheel\_velocity = eb\_.CalculateVelocity( | | | |
| closest\_light\_entity\_, defaultSpeed\_, light\_sensors\_); | | | |
| break; | | | |
| case kAggressive: | | | |
| light\_wheel\_velocity = ab\_.CalculateVelocity( | | | |
| closest\_light\_entity\_, defaultSpeed\_, light\_sensors\_); | | | |
| break; | | | |
| case kLove: | | | |
| light\_wheel\_velocity = lb\_.CalculateVelocity( | | | |
| closest\_light\_entity\_, defaultSpeed\_, light\_sensors\_); | | | |
| break; | | | |
| case kCoward: | | | |
| light\_wheel\_velocity = cb\_.CalculateVelocity( | | | |
| closest\_light\_entity\_, defaultSpeed\_, light\_sensors\_); | | | |
| break; | | | |
| case kNone: | | | |
| default: | | | |
| numBehaviors--; | | | |
| break; | | | |
| } | | | |
|  | | | |
| WheelVelocity food\_wheel\_velocity = WheelVelocity(0, 0); | | | |
|  | | | |
| switch (food\_behavior\_) { | | | |
| case kExplore: | | | |
| food\_wheel\_velocity = eb\_.CalculateVelocity( | | | |
| closest\_food\_entity\_, defaultSpeed\_, light\_sensors\_); | | | |
| break; | | | |
| case kAggressive: | | | |
| food\_wheel\_velocity = ab\_.CalculateVelocity( | | | |
| closest\_food\_entity\_, defaultSpeed\_, light\_sensors\_); | | | |
| break; | | | |
| case kLove: | | | |
| food\_wheel\_velocity = lb\_.CalculateVelocity( | | | |
| closest\_food\_entity\_, defaultSpeed\_, light\_sensors\_); | | | |
| break; | | | |
| case kCoward: | | | |
| food\_wheel\_velocity = cb\_.CalculateVelocity( | | | |
| closest\_food\_entity\_, defaultSpeed\_, light\_sensors\_); | | | |
| break; | | | |
| case kNone: | | | |
| default: | | | |
| numBehaviors--; | | | |
| break; | | | |
| } | | | |
|  |
|  |  | | | |

…

**BV method using BV “sensor”**

|  |  |
| --- | --- |
| void BraitenbergVehicle::SenseEntity(const ArenaEntity& entity) { | |
| const ArenaEntity\*\* closest\_entity\_ = NULL; | |
| if (entity.get\_type() == kLight) { | |
| closest\_entity\_ = &closest\_light\_entity\_; | |
| } else if (entity.get\_type() == kFood) { | |
| closest\_entity\_ = &closest\_food\_entity\_; | |
| } else if (entity.get\_type() == kBraitenberg) { | |
| closest\_entity\_ = &closest\_bv\_entity\_; | |
| }  …  void BraitenbergVehicle::Update() {  … | |
|  |
| WheelVelocity bv\_wheel\_velocity = WheelVelocity(0, 0); |
|  |  | |
| switch (bv\_behavior\_) { |
| case kExplore: |
| bv\_wheel\_velocity = eb\_.CalculateVelocity( |
| closest\_bv\_entity\_, defaultSpeed\_, light\_sensors\_); |
| break; |
| case kAggressive: |
| bv\_wheel\_velocity = ab\_.CalculateVelocity( |
| closest\_bv\_entity\_, defaultSpeed\_, light\_sensors\_); |
| break; |
| case kLove: |
| bv\_wheel\_velocity = lb\_.CalculateVelocity( |
| closest\_bv\_entity\_, defaultSpeed\_, light\_sensors\_); |
| break; |
| case kCoward: |
| bv\_wheel\_velocity = cb\_.CalculateVelocity( |
| closest\_bv\_entity\_, defaultSpeed\_, light\_sensors\_); |
| break; |
| case kNone: |
| default: |
| numBehaviors--; |
| break; |
| } |
| … |
| if (numBehaviors) { |
| wheel\_velocity\_ = WheelVelocity( |
| (light\_wheel\_velocity.left + food\_wheel\_velocity.left + |
| bv\_wheel\_velocity.left)/numBehaviors, |
| (light\_wheel\_velocity.right + food\_wheel\_velocity.right + |
| bv\_wheel\_velocity.right)/numBehaviors, |
| defaultSpeed\_); |
| } else { |
| wheel\_velocity\_ = WheelVelocity(0, 0); |
| } |
| } |
|  |  | |