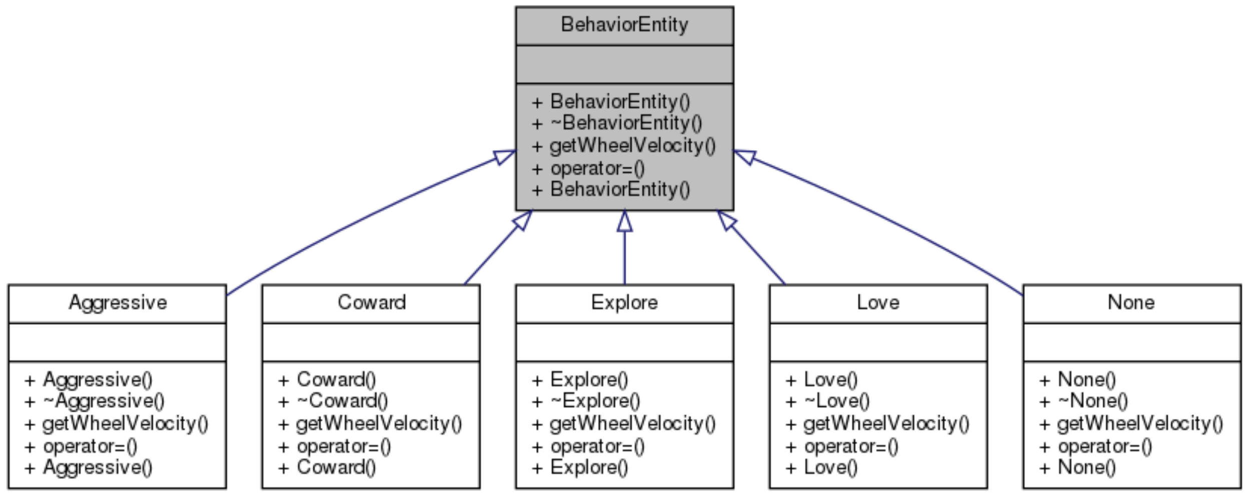
Angela Almquist | almqu071
CSCI3081W Iteration 2: 1st Deliverables

Updated UML for Strategy Pattern



Code added to support Strategy Pattern and BV sensors.

Highlighted in blue are the specific code snippets calling the behavior strategy pattern.
Highlighted in red are the specific code snippets defining sensors for other braitenberg vehicles.
Surrounding code given for context.

IN braitenberg_vehicle.h:

```
 Behavior get_light_behavior() { return light_behavior_; }

 void set_light_behavior(Behavior behavior) {
   light_behavior_ = behavior;
   delete light_behavior_ptr_;
   switch (light_behavior_) {
     case kExplore:
       light_behavior_ptr_ = new Explore();
     break;
     case kLove:
       light_behavior_ptr_ = new Love();
     break;
     case kAggressive:
       light_behavior_ptr_ = new Aggressive();
     break;
     case kNone:
       light_behavior_ptr_ = new None();
     break;
     case kCoward:
       light_behavior_ptr_ = new Coward();
     break;
     default:
       light_behavior_ptr_ = new None();
     break;
```

```
  }
}

Behavior get_food_behavior() { return food_behavior_; }

void set_food_behavior(Behavior behavior) {
  food_behavior_ = behavior;
  delete food_behavior_ptr_;
  switch (food_behavior_) {
    case kExplore:
      food_behavior_ptr_ = new Explore();
    break;
    case kLove:
      food_behavior_ptr_ = new Love();
    break;
    case kAggressive:
      food_behavior_ptr_ = new Aggressive();
    break;
    case kNone:
      food_behavior_ptr_ = new None();
    break;
    case kCoward:
      food_behavior_ptr_ = new Coward();
    break;
    default:
      food_behavior_ptr_ = new None();
    break;
  }
}

Behavior get_bv_behavior() { return bv_behavior_; }

void set_bv_behavior(Behavior behavior) {
  bv_behavior_ = behavior;
  delete bv_behavior_ptr_;
  switch (bv_behavior_) {
    case kExplore:
      bv_behavior_ptr_ = new Explore();
    break;
    case kLove:
      bv_behavior_ptr_ = new Love();
    break;
    case kAggressive:
      bv_behavior_ptr_ = new Aggressive();
    break;
    case kNone:
      bv_behavior_ptr_ = new None();
    break;
```

```
      case kCoward:
        bv_behavior_ptr_ = new Coward();
      break;
      default:
        bv_behavior_ptr_ = new None();
      break;
    }
  }
...
 private:
  std::vector<Pose> light_sensors_;
  MotionBehaviorDifferential * motion_behavior_{nullptr};
  WheelVelocity wheel_velocity_;
  Behavior light_behavior_;
  Behavior food_behavior_;
  Behavior bv_behavior_;
  BehaviorEntity* light_behavior_ptr_;  // added
  BehaviorEntity* food_behavior_ptr_;   // added
  BehaviorEntity* bv_behavior_ptr_;     // added
  const ArenaEntity* closest_light_entity_;
  const ArenaEntity* closest_food_entity_;
  const ArenaEntity* closest_bv_entity_;
  double defaultSpeed_;
```

IN braitenberg_vehicle.cc:

```
BraitenbergVehicle::BraitenbergVehicle() :
  light_sensors_(), wheel_velocity_(), light_behavior_(kNone),
  food_behavior_(kNone), bv_behavior_(kNone),
  light_behavior_ptr_{new None()}, food_behavior_ptr_{new None()},
  bv_behavior_ptr_{new None()}, closest_light_entity_(NULL),
  closest_food_entity_(NULL), closest_bv_entity_(NULL),
  defaultSpeed_(5.0) {
  set_type(kBraitenberg);
  motion_behavior_ = new MotionBehaviorDifferential(this);
  light_sensors_.push_back(Pose());
  light_sensors_.push_back(Pose());
  set_color(BRAITENBERG_COLOR);
  set_pose(ROBOT_INIT_POS);

  wheel_velocity_ = WheelVelocity(0, 0);
  // Set ID
  count++;
  set_id(count);
}
...
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) {
      if (entity.get_id() != this->get_id()) {
        closest_entity_ = &closest_bv_entity_;
      }
    }
...
void BraitenbergVehicle::Update() {
  WheelVelocity* light_wv_ptr = new WheelVelocity();
  WheelVelocity* food_wv_ptr = new WheelVelocity();
  WheelVelocity* bv_wv_ptr = new WheelVelocity();
  food_behavior_ptr_->getWheelVelocity(
    get_sensor_reading_left(closest_food_entity_),
    get_sensor_reading_right(closest_food_entity_),
    defaultSpeed_, food_wv_ptr);

  light_behavior_ptr_->getWheelVelocity(
    get_sensor_reading_left(closest_light_entity_),
    get_sensor_reading_right(closest_light_entity_),
    defaultSpeed_, light_wv_ptr);

  bv_behavior_ptr_->getWheelVelocity(
    get_sensor_reading_left(closest_bv_entity_),
    get_sensor_reading_right(closest_bv_entity_),
    defaultSpeed_, bv_wv_ptr);

  int numBehaviors = 3;    // FIGURE THIS PART OUT
  if (numBehaviors) {
    if (food_behavior_ && light_behavior_) {
      set_color(BRAITENBERG_COLOR);
    }
    wheel_velocity_ = WheelVelocity(
      (light_wv_ptr->left + food_wv_ptr->left + bv_wv_ptr->left)
       /numBehaviors,
      (light_wv_ptr->right + food_wv_ptr->right + bv_wv_ptr->right)
      /numBehaviors, defaultSpeed_); } else {
    set_color(BRAITENBERG_COLOR);
    wheel_velocity_ = WheelVelocity(0, 0);
  }
  delete food_wv_ptr;    //  added here
  delete light_wv_ptr;   //   added here
  delete bv_wv_ptr;      //   added here
}
...
void BraitenbergVehicle::LoadFromObject(json_object* entity_config_ptr) {
  json_object& entity_config = *entity_config_ptr;
  ArenaEntity::LoadFromObject(entity_config_ptr);
  if (entity_config.find("light_behavior") != entity_config.end()) {
      light_behavior_ = get_behavior_type(
        entity_config["light_behavior"].get<std::string>());
        set_light_behavior(light_behavior_);
  }
  if (entity_config.find("food_behavior") != entity_config.end()) {
      food_behavior_ = get_behavior_type(
        entity_config["food_behavior"].get<std::string>());
        set_food_behavior(food_behavior_);
  }
  if (entity_config.find("bv_behavior") != entity_config.end()) {
      bv_behavior_ = get_behavior_type(
        entity_config["bv_behavior"].get<std::string>());
        set_bv_behavior(bv_behavior_);
  }
  UpdateLightSensors();
}
```

All Unit Tests for Strategy Pattern   (comments not included)

None_unittest.cc:

```
#include <gtest/gtest.h>
#include "src/behavior.h"
#include "src/wheel_velocity.h"
#include "src/none.h"
class NoneBehaviortest : public ::testing::Test {
 protected:
  virtual void SetUp() {
    bNone = new csci3081::None();
  }
  virtual void TearDown() {
    delete bNone;
  }
  csci3081::None * bNone;
};
TEST_F(NoneBehaviortest, NoneBehavior) {
  csci3081::WheelVelocity * wv_ptr = new csci3081::WheelVelocity();
  bNone->getWheelVelocity(0, 0, 0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 0)<< "FAIL: speed always equals zero for no behavior\n";
  EXPECT_EQ(wv_ptr->right, 0)<< "FAIL: speed always equals zero for no behavior\n";
  bNone->getWheelVelocity(1.0, 4.0, 5.0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 0) << "FAIL: speed always equals zero for no behavior\n";
  EXPECT_EQ(wv_ptr->right, 0) << "FAIL: speed always equals zero for no behavior\n";
};
```

Explore_unittest.cc:

```
#include <gtest/gtest.h>
#include "src/behavior.h"
#include "src/wheel_velocity.h"
#include "src/explore.h"
class ExploreBehaviortest : public ::testing::Test {
 protected:
  virtual void SetUp() {
    bExplore = new csci3081::Explore();
  }
  virtual void TearDown() {
    delete bExplore;
  }
  csci3081::Explore * bExplore;
};
TEST_F(ExploreBehaviortest, ExploreBehavior) {
  csci3081::WheelVelocity * wv_ptr = new csci3081::WheelVelocity();
  bExplore->getWheelVelocity(0, 0, 0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 0) << "FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->right, 0) << "FAIL: speed exceeds maximum\n";
  bExplore->getWheelVelocity(0, 0, 5.0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 5.0) << "FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->right, 5.0) << "FAIL: speed exceeds maximum\n";
  bExplore->getWheelVelocity(3.0, 4.0, 5.0, wv_ptr);
  EXPECT_NEAR(wv_ptr->right, .33, .01)<< "FAIL: speed not updated correctly\n";
  EXPECT_EQ(wv_ptr->left, .25)<< "FAIL: speed not updated correctly\n";
  bExplore->getWheelVelocity(.01, 4.0, 50.0, wv_ptr);
  EXPECT_EQ(wv_ptr->right, 50.0)<< "FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->left, .25)<< "FAIL: speed not updated correctly\n";
  bExplore->getWheelVelocity(.1, .1, 5.0, wv_ptr);
  EXPECT_EQ(wv_ptr->right, 5.0) << "FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->left, 5.0) << "FAIL: speed exceeds maximum\n";
};
```

Love_unittest.cc:

```
#include <gtest/gtest.h>
#include "src/behavior.h"
#include "src/wheel_velocity.h"
#include "src/love.h"
class LoveBehaviortest : public ::testing::Test {
 protected:
  virtual void SetUp() {
    bLove = new csci3081::Love();
  }
  virtual void TearDown() {
    delete bLove;
  }
  csci3081::Love * bLove;
};
```

```
TEST_F(LoveBehaviortest, LoveBehavior) {
  csci3081::WheelVelocity * wv_ptr = new csci3081::WheelVelocity();
  bLove->getWheelVelocity(0, 0, 0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 0) << "FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->right, 0)<< "FAIL: speed exceeds maximum\n";
  bLove->getWheelVelocity(0, 0, 5.0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 5.0) << "FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->right, 5.0)<< "FAIL: speed exceeds maximum\n";
  bLove->getWheelVelocity(3.0, 4.0, 5.0, wv_ptr);
  EXPECT_NEAR(wv_ptr->left, .33, .01)<< "FAIL: speed not updated correctly\n";
  EXPECT_EQ(wv_ptr->right, .25)<< "FAIL: speed not updated correctly\n";
  bLove->getWheelVelocity(.01, 4.0, 50.0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 50.0)<<"FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->right, .25)<< "FAIL: speed not updated correctly\n";
  bLove->getWheelVelocity(.1, .1, 5.0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 5.0)<<"FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->right, 5.0)<<"FAIL: speed exceeds maximum\n";
};
```

Aggressive_unittest.cc:

```
#include <gtest/gtest.h>
#include "src/behavior.h"
#include "src/wheel_velocity.h"
#include "src/aggressive.h"
class AggressiveBehaviortest : public ::testing::Test {
 protected:
  virtual void SetUp() {
    bAggressive = new csci3081::Aggressive();
  }
  virtual void TearDown() {
    delete bAggressive;
  }
  csci3081::Aggressive * bAggressive;
};
TEST_F(AggressiveBehaviortest, AggressiveBehavior) {
  csci3081::WheelVelocity * wv_ptr = new csci3081::WheelVelocity();
  bAggressive->getWheelVelocity(0, 0, 0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 0)<< "FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->right, 0)<< "FAIL: speed exceeds maximum\n";
  bAggressive->getWheelVelocity(0, 0, 5.0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 0)<< "FAIL: should be no change with zero speed\n";
  EXPECT_EQ(wv_ptr->right, 0)<< "FAIL: should be no change with zero speed\n";
  bAggressive->getWheelVelocity(3.0, 4.0, 5.0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 4.0)<< "FAIL: velocity not updated correctly\n";
  EXPECT_EQ(wv_ptr->right, 3.0)<< "FAIL: velocity not updated correctly\n";
  bAggressive->getWheelVelocity(1.0, 50.0, 60.0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 50.0)<< "FAIL: velocity not updated correctly\n";
  EXPECT_EQ(wv_ptr->right, 1.0)<< "FAIL: velocity not updated correctly\n";
  bAggressive->getWheelVelocity(6.0, 7.0, 5.0, wv_ptr);
  EXPECT_EQ(wv_ptr->right, 5.0) << "FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->left, 5.0) << "FAIL: speed exceeds maximum\n";
};
```

Coward_unittest.cc:

```
#include <gtest/gtest.h>
#include "src/behavior.h"
#include "src/wheel_velocity.h"
#include "src/coward.h"
class CowardBehaviortest : public ::testing::Test {
 protected:
  virtual void SetUp() {
    bCoward = new csci3081::Coward();
  }
  virtual void TearDown() {
    delete bCoward;
  }
  csci3081::Coward * bCoward;
};
TEST_F(CowardBehaviortest, CowardBehavior) {
  csci3081::WheelVelocity * wv_ptr = new csci3081::WheelVelocity();
  bCoward->getWheelVelocity(0, 0, 0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 0)<< "FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->right, 0)<< "FAIL: speed exceeds maximum\n";
  bCoward->getWheelVelocity(0, 0, 5.0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 0)<< "FAIL: should be no change to zero speed\n";
  EXPECT_EQ(wv_ptr->right, 0)<< "FAIL: should be no change to zero speed\n";
  bCoward->getWheelVelocity(3.0, 4.0, 5.0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 3.0)<< "FAIL: velocities not updated correctly\n";
  EXPECT_EQ(wv_ptr->right, 4.0)<< "FAIL: velocities not updated correctly\n";
  bCoward->getWheelVelocity(40.0, 1.0, 50.0, wv_ptr);
  EXPECT_EQ(wv_ptr->left, 40.0)<< "FAIL: velocities not updated correctly\n";
  EXPECT_EQ(wv_ptr->right, 1.0)<< "FAIL: velocities not updated correctly\n";
  bCoward->getWheelVelocity(6.0, 7.0, 5.0, wv_ptr);
  EXPECT_EQ(wv_ptr->right, 5.0) << "FAIL: speed exceeds maximum\n";
  EXPECT_EQ(wv_ptr->left, 5.0) << "FAIL: speed exceeds maximum\n";
};
```