

```

/*****
2   * Includes
3   *****/
4   #include <gtest/gtest.h>
5   #include <fstream>
6   #include <iostream>
7   #include <string>
8   #include <vector>
9   #include <streambuf>
10
11  #include "src/Aggressive.h"
12  #include "src/LightFactory.h"
13  #include "src/pose.h"
14
15  /*****
16  * TEST FEATURE SetUp
17  *****/
18  class AggressiveBehaviorTest : public ::testing::Test {
19  public:
20      virtual void SetUp() {
21          light = factory.Create();
22      }
23  protected:
24      csci3081::AggressiveBehavior ab;
25      csci3081::LightFactory factory;
26      csci3081::Light * light;
27      double speed = 5.0;
28  };
29
30  /*****
31  * Test Cases
32  *****/
33
34  TEST_F(AggressiveBehaviorTest, NoEntity) {
35      std::vector<csci3081::Pose> light_sensors;
36      light_sensors.push_back(csci3081::Pose());
37      light_sensors.push_back(csci3081::Pose());
38
39      light = NULL;
40      csci3081::WheelVelocity wv = ab.CalculateVelocity(light, speed,
41          light_sensors);

```

```

42     csci3081::WheelVelocity expected = csci3081::WheelVelocity(0.0001, 0.0001, speed);
43
44     EXPECT_EQ(wv.left, expected.left) << "FAIL: Default left wheel velocity incorrectly calculated";
45     EXPECT_EQ(wv.right, expected.right) << "FAIL: Default right wheel velocity incorrectly calculated";
46 }
47
48 TEST_F(AggressiveBehaviorTest, CloseDistance) {
49     std::vector<csci3081::Pose> light_sensors;
50     light_sensors.push_back(csci3081::Pose(200, 190));
51     light_sensors.push_back(csci3081::Pose(200, 190));
52
53     csci3081::WheelVelocity wv = ab.CalculateVelocity(light, speed,
54         light_sensors);
55
56     double reading_left = 1800.0/std::pow(
57         1.08, (light->get_pose()-light_sensors[0]).Length());
58     double reading_right = 1800.0/std::pow(
59         1.08, (light->get_pose()-light_sensors[1]).Length());
60
61     csci3081::WheelVelocity expected = csci3081::WheelVelocity(reading_right, reading_left, speed);
62
63     EXPECT_EQ(wv.left, expected.left) << "FAIL: Left wheel velocity for close distance incorrectly calculated";
64     EXPECT_EQ(wv.right, expected.right) << "FAIL: Right wheel velocity for close distance incorrectly calculated";
65 }
66
67 TEST_F(AggressiveBehaviorTest, MediumDistance) {
68     std::vector<csci3081::Pose> light_sensors;
69     light_sensors.push_back(csci3081::Pose(200, 150));
70     light_sensors.push_back(csci3081::Pose(200, 150));
71
72     csci3081::WheelVelocity wv = ab.CalculateVelocity(light, speed,
73         light_sensors);
74
75     double reading_left = 1800.0/std::pow(
76         1.08, (light->get_pose()-light_sensors[0]).Length());
77     double reading_right = 1800.0/std::pow(
78         1.08, (light->get_pose()-light_sensors[1]).Length());
79
80     csci3081::WheelVelocity expected = csci3081::WheelVelocity(reading_right, reading_left, speed);
81
82     EXPECT_EQ(wv.left, expected.left) << "FAIL: Left wheel velocity for medium distance incorrectly calculated";
83     EXPECT_EQ(wv.right, expected.right) << "FAIL: Right wheel velocity for medium distance incorrectly calculated";

```

```

84 }
85
86 TEST_F(AggressiveBehaviorTest, FarDistance) {
87     std::vector<csci3081::Pose> light_sensors;
88     light_sensors.push_back(csci3081::Pose(200, 100));
89     light_sensors.push_back(csci3081::Pose(200, 100));
90
91     csci3081::WheelVelocity wv = ab.CalculateVelocity(light, speed,
92         light_sensors);
93
94     double reading_left = 1800.0/std::pow(
95         1.08, (light->get_pose()-light_sensors[0]).Length());
96     double reading_right = 1800.0/std::pow(
97         1.08, (light->get_pose()-light_sensors[1]).Length());
98
99     csci3081::WheelVelocity expected = csci3081::WheelVelocity(reading_right, reading_left, speed);
100
101     EXPECT_EQ(wv.left, expected.left) << "FAIL: Left wheel velocity for far distance incorrectly calculated";
102     EXPECT_EQ(wv.right, expected.right) << "FAIL: Right wheel velocity for far distance incorrectly calculated";
103 }

```