Contents

[1. Project Testing Plan 2](#_Toc431929288)

[1.1 Testing Scope 2](#_Toc431929289)

[1.2 Unit Test Plan 2](#_Toc431929290)

[1.3 Use Case Examples 3](#_Toc431929291)

[1.3.1 Environment Use Cases 3](#_Toc431929292)

[1.3.2 Brain Use Cases 4](#_Toc431929293)

# 1. Project Testing Plan

## 1.1 Testing Scope

The tests used in this project follow a traditional top-down approach. The program was initially built using an iterative methodology prior to the creation of a test plan; however the procedures used can be detailed and written down. The program consists of a number of high level modules, which are in turn composed of smaller units. The test plan will extend to the top-down overview of the primary systems and key sub-systems required to get the program operational. Since the project has already gone through a number of iterations, this document will cover the strategies already used.

## 1.2 Unit Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test ID | Description | Expected Results | Actual Results |
| 1 | 2.6.2.1.1 | Member should save data to database | New record | PASS |
| 2 | 2.6.2.1.2 | Recorded closest target to member | Smallest Distance | PASS |
| 3 | 2.6.2.1.3 | Calculate distance to reference point | [0,100][100,200] => 141.421 | PASS |
| 4 | 2.6.2.1.4 | Calculate relative angle to reference point | [0,100,90][100,200] => 45.3 | PASS |
| 5 | 2.6.2.1.5 | Given speed and angle, move member to new position | [.41,6.1][288.3,151] => [295,284] | PASS |
| 6 | 2.6.2.1.6 | Given track speeds, calculate rotational change | 0 + 0.1689 = 0.1689 | PASS |
| 7 | 2.6.2.1.7 | Collision increment counter, add energy, mark target for removal | [ 1, 1, 1] | PASS |
| 8 | 2.6.2.1.8 | Member initializes instance of Brain | Brain != None | PASS |
| 9 | 2.6.2.2.1 | A population of N members should be created when none exist | len(members) = N | PASS |
| 10 | 2.6.2.2.2 | The sum of all member scores should be 1 | 1 | PASS |
| 11 | 2.6.2.2.3 | Should select N parents leaving 0 members | [ N, 0 ] | PASS |
| 12 | 2.6.2.2.4 | Should produce N members leaving 0 parents | [ N, 0 ] | PASS |
| 13 | 2.6.2.2.5 | Should run functions successfully to mutate | Mutations >= 0 | PASS |
| 14 | 2.6.2.2.6 | Target list is updated to remove or add targets | +/- len(targets) | PASS |
| 15 | 2.6.2.3.1 | Network should initialize with shape [ 4, 1, 2 ] | Shape = [ 4, 1, 2 ] | PASS |
| 16 | 2.6.2.3.2 | Should add random neuron | Initial Layers + 1 | PASS |
| 17 | 2.6.2.3.3 | Should add random connection | Initial Connections + 1 | PASS |
| 18 | 2.6.2.3.4 | Should accept parameters and return 2 floats between -1 and 1 | [ -1/1, -1/1 ] | PASS |
| 19 | 2.6.2.3.5 | Should randomize random connection | NW does not equal OW | PASS |
| 20 | 2.6.2.4.1 | Target initializes within bounds of view | R <= X <= W || R <= Y <= H | PASS |
| 21 | 2.6.2.5.1 | Key is generated for a member using Timestamp – Lifespan – Member – Generation | K = time-life-member-generation | PASS |
| 22 | 2.6.2.5.2 | Value is saved to database by key | After save: key, value = input | PASS |

## 1.3 Use Case Examples

### 1.3.1 Environment Use Cases

#### 1.3.1.1 Score Members

|  |  |
| --- | --- |
| Use Case ID: | 2.6.2.2.2 |
| Use Case Name: | Score Members |
| Actors: | Environment, Member |
| Description: | The sum of all member scores should be 1 |
| Trigger: | Perform scoring function |
| Preconditions: | All members dead  Perform Scoring function called |
| Postconditions: | All members have scores that when summed equal 1 |
| Normal Flow: | For each member  Calculate score using lifespan plus food \* 1000  For each member  Divide member score by total score  Set member score to step 4  Add normalized scores |
| Exceptions: | Score is less than 0, set score to 0 |
| Includes: | None |
| Frequency of Use: | Once per generation |
| Special Requirements: | 2.6.2.2.1 |
| Assumptions: | Lifespan and food consumption equate to higher survivability |
| Notes and Issues: | None |

#### 1.3.1.2 Select Parents

|  |  |
| --- | --- |
| Use Case ID: | 2.6.2.2.3 |
| Use Case Name: | Select Parents |
| Actors: | Environment, Member |
| Description: | Should select a number of parents are determined by the elitist parameter, and clear the remaining members from the population. |
| Trigger: | Perform crossover function |
| Preconditions: | All members dead  All members scored  Perform crossover function called |
| Postconditions: | Parents attributes is populated with the selected members |
| Normal Flow: | Sort members by fitness values in descending order  Calculate the number of parents to select  Splice members attribute  Set members attribute to NULL |
| Exceptions: | Rounded value selection may produce a number of members smaller than population size.  Finds remaining and creates new random members |
| Includes: | None |
| Frequency of Use: | Once per generation |
| Special Requirements: | 2.6.2.2.2 |
| Assumptions: | None |
| Notes and Issues: | None |

#### 1.3.1.3 Perform Mutation

|  |  |
| --- | --- |
| Use Case ID: | 2.6.2.2.5 |
| Use Case Name: | Perform Mutation |
| Actors: | Environment, Brain |
| Description: | This function should successfully perform 3 units resulting in mutation |
| Trigger: | Perform mutation function |
| Preconditions: | All members dead  Members scored  Parents selected  New members created from crossover |
| Postconditions: | New members have gone through mutation to produce new values |
| Normal Flow: | For each member  Iterate over connection  Random chance to randomize connection weight  Random chance to add new logic neuron  Random chance to add new connection |
| Exceptions: | Refer to related use cases:  4. 2.6.2.3.3  5. 2.6.2.3.2 |
| Includes: | 2.6.2.3.3, 2.6.2.3.2, 2.6.2.3.5 |
| Frequency of Use: | Once per generation |
| Special Requirements: | None |
| Assumptions: | None |
| Notes and Issues: | None |

### 1.3.2 Brain Use Cases

#### 1.3.2.1 Add New Neuron

|  |  |
| --- | --- |
| **Use Case ID:** | 2.6.2.3.2 |
| **Use Case Name:** | Add New Neuron |
| **Actors:** | Brain |
| **Description:** | This function should successfully add a new logic neuron and randomly connect it to another layer. |
| **Trigger:** | Perform mutation function, add random neuron function |
| **Preconditions:** | Brain is initialized  Mutation is performed  Random number is less than or equal to Mutation Rate |
| **Postconditions:** | New neuron is added to existing brain |
| **Normal Flow:** | Create new layer with random type ( Sigmoid, Linear, Tanh, Gaussian )  Randomly pick layer from existing network  Add new layer as logic neuron  Add connection between two layers  Sort Modules |
| **Exceptions:** | Connection creates a logic loop  Remove connection  Leave new neuron unconnected |
| **Includes:** | Remove\_connection(), add\_logic(), add\_connection() |
| **Frequency of Use:** | .8% chance every member every generation |
| **Special Requirements:** | None |
| **Assumptions:** | None |
| **Notes and Issues:** | Unconnected neuron may spontaneously be connected at some point in the future allowing for the creation of new neural pathways. |

#### 1.3.2.2 Add New Connection

|  |  |
| --- | --- |
| **Use Case ID:** | 2.6.2.3.3 |
| **Use Case Name:** | Add New Connection |
| **Actors:** | Brain |
| **Description:** | This function should successfully add a new connection between all pre-existing layers |
| **Trigger:** | Perform mutation function, add random connection function |
| **Preconditions:** | Brain is initialized  Mutation is performed  Random number is less than or equal to Mutation Rate |
| **Postconditions:** | New connection is added to the brain |
| **Normal Flow:** | Select two random layers  Create Full Connection  If two layers are not the same, add connection  Sort network modules |
| **Exceptions:** | Connection creates a logic loop  Remove connection and continue |
| **Includes:** | Remove\_connection(), add\_connection() |
| **Frequency of Use:** | .8% chance every member every generation |
| **Special Requirements:** | None |
| **Assumptions:** | None |
| **Notes and Issues:** | Connections that create a loop between layers produce fatal errors in Pybrain. A try / catch block is used to handle this exception and remove the connection after it has been added. |

#### 1.3.2.3 Randomize Connection Weight

|  |  |
| --- | --- |
| **Use Case ID:** | 2.6.2.3.5 |
| **Use Case Name:** | Randomize Connection Weight |
| **Actors:** | Brain |
| **Description:** | This function should successfully randomize the weight of a pre-existing connection. |
| **Trigger:** | Perform mutation function, random chance is less than or equal to Mutation Rate. |
| **Preconditions:** | Brain is initialized  Mutation is performed  For each module in network  Iterate over connections in module  Random number is less than or equal to Mutation Rate |
| **Postconditions:** | Connection is randomized, but network remains otherwise unchanged |
| **Normal Flow:** | For each member  Iterate over modules  Iterate over connections for module  If random number is less than or equal to Mutation Rate  Call randomize function |
| **Exceptions:** | None |
| **Includes:** | Pybrain.structure.connection.randomize |
| **Frequency of Use:** | .8% chance every connection every module every member every generation |
| **Special Requirements:** | None |
| **Assumptions:** | None |
| **Notes and Issues:** | This loop may double the chance of randomizing a connection weight due to multiple modules containing the same connection. |