|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Description | Expected Results | Actual Results |
| 2.6.2.1.1 | Member data should save to database. | Dictionary entry |  |
| 2.6.2.1.2 | Member should store closest target from list of targets. | Closest target is recorded |  |
| 2.6.2.1.3 | Calculate distance should return pixel count between two reference points. | Float value indicating correct distance |  |
| 2.6.2.1.4 | Calculate relative angle between member and target. | +/- float as degrees between direction and target |  |

|  |  |
| --- | --- |
| **Use Case ID:** | 2.6.2.1.1 |
| **Use Case Name:** | Save Member to Database |
| **Actors:** | Environment, Database |
| **Description:** | When a generation is finished, all members should save to the database. |
| **Trigger:** | All members set to alive = False |
| **Preconditions:** | 1. Members initialized 2. Members energy <= 0 3. Members alive = False 4. Members scored |
| **Postconditions:** | Database file contains entries for each member in current generation |
| **Normal Flow:** | 1. Members are created 2. Members lose energy over time 3. Members lose all energy 4. Members are scored based on lifespan 5. Member data is saved to the database |
| **Exceptions:** | Use Case should not produce errors |
| **Includes:** | 2.6.2.5.1, 2.6.2.5.2 |
| **Frequency of Use:** | Once per generation |
| **Special Requirements:** | None |
| **Assumptions:** | None |
| **Notes and Issues:** | None |

|  |  |
| --- | --- |
| **Use Case ID:** | 2.6.2.1.2 |
| **Use Case Name:** | Identify Object |
| **Actors:** | Member, Target |
| **Description:** | During collision detection, a member should iteratively find the closest member and save that object as a member specific value. |
| **Trigger:** | Check Collision Function is called |
| **Preconditions:** | 1. Member is initialized 2. Targets are initialized 3. Member is updated 4. Member runs check collision function |
| **Postconditions:** | Member’s attribute contains pointer to target nearest to member |
| **Normal Flow:** | 1. Member is initialized 2. Targets are initialized 3. Member runs update state 4. Member runs update position 5. Member runs check collisions 6. Member iterates over each target 7. Targets are tested for existing within range 8. If target is in range, target distance is compared to current minimum 9. If target is closer, target is recorded |
| **Exceptions:** | 9A. Targets are same distance   1. Last target iterated over is recorded as closest |
| **Includes:** | 2.6.2.1.3 |
| **Frequency of Use:** | 60 \* ( Member Count \* Target Count ) per second |
| **Special Requirements:** | 2.6.2.4.1, 2.6.2.2.6 |
| **Assumptions:** | None |
| **Notes and Issues:** | This use case is included in the check collision function as we are already iterative over targets for each member to check collision in iterative passes. |

|  |  |
| --- | --- |
| **Use Case ID:** | 2.6.2.1.3 |
| **Use Case Name:** | Calculate Distance |
| **Actors:** | Helper, Member |
| **Description:** | When called, function should calculate the distance between two points as the hypotenuse of a right triangle. |
| **Trigger:** | Get Distance function is called from Helper class |
| **Preconditions:** | 1. Reference coordinates are provided 2. Target coordinates are provided |
| **Postconditions:** | Float is returned to caller |
| **Normal Flow:** | 1. Reference coordinates are recorded 2. Target coordinates are recorded 3. Function is called providing coordinates as list 4. Sides are calculated as differentials between two points 5. Hypotenuse is returned as float |
| **Exceptions:** | 3A. Function is called with no reference or target point   1. Return None type |
| **Includes:** | None |
| **Frequency of Use:** | 60 \* ( Member Count \* Target Count ) \* 4 per second |
| **Special Requirements:** | 2.6.2.1.5, 2.6.2.4.1 |
| **Assumptions:** | Distance can be calculated as the hypotenuse of a right angle between all coordinate points. |
| **Notes and Issues:** | Due to frequency of use, this function is a good optimization point. |

|  |  |
| --- | --- |
| **Use Case ID:** | 2.6.2.1.4 |
| **Use Case Name:** | Calculate Relation |
| **Actors:** | Helper, Member |
| **Description:** | When called |
| **Trigger:** | Get Distance function is called from Helper class |
| **Preconditions:** | 1. Reference coordinates are provided 2. Target coordinates are provided |
| **Postconditions:** | Float is returned to caller |
| **Normal Flow:** | 1. Reference coordinates are recorded 2. Target coordinates are recorded 3. Function is called providing coordinates as list 4. Sides are calculated as differentials between two points 5. Hypotenuse is returned as float |
| **Exceptions:** | 3A. Function is called with no reference or target point   1. Return None type |
| **Includes:** | None |
| **Frequency of Use:** | 60 \* ( Member Count \* Target Count ) \* 4 per second |
| **Special Requirements:** | 2.6.2.1.5, 2.6.2.4.1 |
| **Assumptions:** | Distance can be calculated as the hypotenuse of a right angle between all coordinate points. |
| **Notes and Issues:** | Due to frequency of use, this function is a good optimization point. |