Test Plan

Throughout the course of this project, I will use the following methods of testing:

* Component testing
* Integrative testing
* Functionality testing
* Usability testing
* Acceptance Testing
* End-user beta testing
* (possible) regression testing

**Component/Integrative testing**

I plan to design this program in a modular way, and so will test each module or component individually, as they are implemented (contrary to my initial gantt chart, but reflected in my revised gantt chart). This is done to ensure each component works before I move on to develop the next component.

However, after each component is implemented, I will need to make sure it works in tandem with all the other implemented components. This will involve integrative testing, making sure that each component correctly works with the other components and passes the right data.

The following are the components which will be individually tested:

|  |
| --- |
|  |
| * Neural Net |
| * Main Menu |
| * Slow Generation |
| * Fast Generation |
| * Average Survival Time Graph |
| * Creature Cannibalism |
| * Creature Names Generation |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Test Method/Inputs** | **Expected Output** | **Actual Output** | **P/F** |
| Frame and Panel | Driver code telling canvas to display a circle: Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-11-22 at 14.10.56.png | A black background with a red sphere in the centre. | Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-11-22 at 14.12.23.png | **P** |
| Neural Net | Driver code to test the SigmoidInputNode class and the sigmoid transfer functionMacintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-11-26 at 09.03.54.png | Outputs very close to zero for low inputs (-ve), increasing output as input value increases, outputs very close to 1 for high input values (+ve) | Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-11-26 at 09.05.50.png | **P** |
| Neural Net | Driver code to test the SigmoidNode class and how it gets data from a SigmoidInputNode with value 0.5Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-11-29 at 14.21.50.png | h1 output equal to:  sigmoid(sigmoid(0.5) \* h1\_weight + h1\_bias) | Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-11-29 at 14.21.58.png | **P** |
| Neural Net | Driver code to test the HardlimNode class and how it gets data from a SigmoidNode, which in turn gets data from a SigmoidInputNode:  Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-11-29 at 15.18.38.png | u1 output equal to:  hardlim(h1\_output \* weight + bias) | Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-11-29 at 15.18.47.png | **P** |
| Neural Net | Driver code to set up a full neural net and test the frequency of getting 1s and 0s as the outputs  Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-12-13 at 14.21.08.png  Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-12-13 at 14.21.33.png | Ratio of 1s:0s is about 1:1 | Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-12-13 at 14.25.03.png  ratio of about 3:7 | **F1** |
| Neural Net | As above, but with the fix mentioned in F1 | As above | Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2018-12-13 at 14.44.40.png | **P** |
| Main Menu | Display the main menu | Layout consistent with UI Design.png |  | **P** |
| Main Menu | Create stub to handle button presses by printing the button name:  Press Slow Generation, then Quick Generation, then View Previous Generation, then Exit to Desktop, then Start Fresh. | “Slow Generation” “Quick Generation”  “View Previous Generation”  “Exit to Desktop”  “Start Fresh” |  |  |
| Main Menu | Implement response for the Exit to Desktop button. Press exit to desktop | Window closes and all processes are ended. | Window closed and all processes were terminated | **P** |
| Slow Generation | Create creature class which gives the creature a random position. Create stub method that displays the creature. Create driver code that makes game start in the in-game view | Run multiple times, creature should appear in multiple different places |  | **P** |
| Integration | Implement the slow generation button to switch to the in game view. | Click slow generation button, game starts showing creature instead of main menu |  | **P** |
| Slow Generation | Create runnables to tick the creature logic, and repaint the canvas. | Creature should move about. | Creature moves about | **P** |
| Slow Generation | Implement creatures ‘seeing’ food pellets. Reduce generation size to one creature, show line for direction it’s facing, and highlight food pellet it can see | Food Pellet highlighted red when the creature see’s it | Food Pellets were highlighted in red | **P** |
| Slow Generation | Implement new creatures being mutated from the previous generation, and a new generation being able to be competed after the original generation has ended. Print statements when new generation is being created. | After generation ends, statements detailing the creation of a new generation should be logged in the console. | Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2019-03-21 at 15.17.04.png | **P** |
| Slow Generation | Run multiple generations to see if any specific creature is dominant | After a few generations there should be multiple creatures that are all the same colour | 1st Generation:  Later Generation: | **P** |
| Integration | Implement stat box showing how many generations it’s been | Compete four generations. Stat box should display “Generation Number: 5” |  | **P** |
| Fast Generation | Implement code that runs a quick generation when the quick generation button is pressed. Press the button. | Generation number on stat block should increase | Generation numbers on stat block increased | **P** |
| Integration | Test whether a slow generation can still be run after multiple fast generations have been completed | Slow Generation starts and runs as normal | Slow Generation starts and runs as normal | **P** |
| Highest Food Graph | Test whether graph is drawn as generations are run | Each time either type of generation is run, the axes expand and more lines are drawn | Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2019-03-25 at 09.14.26.png | **P** |
| Highest Food Graph | Test whether latest point drawn on graph matches what is shown on the stat block | Latest point matches stat block’s Highest Survival Time statistic | Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2019-03-25 at 09.14.26.png | **P** |
| View Previous Generation | Test whether it will use the exact same creatures as the previous generation.  Run a slow generation and observe the colours. Then view previous generation. | Colours the second time around should be *exactly* the same | 1st:Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2019-03-25 at 09.46.32.png  2nd: Macintosh HD:Local:Users:AHCS:Desktop:Screen Shot 2019-03-25 at 09.46.13.png | **P** |
| Integration | Test that running new generations still works after viewing a previous generation | Running new generations works | Running new Generations works | **P** |
| Creature Names | Implement creatures being given random first names and second names based on their parents. Create stub code to print out all creatures names after creatures. Check whether the top | In a given generation other than the first, 4 creatures should have the exact same names as the top 4 from the previous generation.  4 other creatures should have random first names, and second names which are based on the first name of one of the top 4 from the previous generation.  The 2 last creatures should have a random first name and the last name ‘Godson’ |  | **P** |
| Creature Names | Check whether the best creature printed to the console matches the best creature displayed in the stat block | The name on the stat block should match the first of all the creatures with tied highest score from the previous generation |  | **P** |

F1 – The failure of this test was due to the fact that the hardlim function returned 1 when x was >0.5, rather than when x>0. This was fixed and retested

**Functionality, Usability, Acceptance, and Beta testing**

These will all be done only once the project has reached a stable releasable state.

Functionality testing will simply be carried out by myself, to check if all functional requirements have been met.

|  |  |
| --- | --- |
| **Requirement** | **P?** |
| Initialise a frame and panel |  |
| Create Neural Nets with random weight values |  |
| Create children of Neural Nets with random mutations |  |
| Store and read in neural net weights |  |
| Run a quick generation which makes a new set of neural nets |  |
| Run a slow generation which allows the user to see the ‘creatures’ in slowed down time as they compete |  |
| Record and display average survival time values |  |
| Creatures able to eat other creatures as well as pellets |  |
| Draw names for creatures from CSV file |  |

Usability and Acceptance (Beta) testing will be carried out by giving the release to end-users (such as peers in my class) to use the program, without any instruction or guidance from me. The end-users will then fill out a survey asking about how intuitive they felt the design was, and how well it met the user requirements, using direct questions.

**Regression Testing**

Regression testing will only be necessary in the following circumstances:

* If the other types of testing show errors in the code, or that requirements are not met, and changes are made to existing components. If that is the case, the component and integrative testing for the changed components will be redone.
* If additional time is left at the end of development, and the additional features are implemented. In this case, more component, integrative, and beta testing will be done for these components.