Evaluation

My final solution meets almost all the requirements in the requirements specification. The few requirements which the solution does not meet were not met deliberately, as the designs and intentions of the project were changed as development progressed. These requirements and the reasons why they have not been met are detailed later in the evaluation. The final solution also meets the technical requirements by including recursion and arrays of objects. Overall, I would deem this project a success, although ideally more time would be spent on it if possible.

Requirements Specification

The only requirements that were not met are the following:

* My original project proposal stated that I would be fulfilling the technical requirement of using a sorting algorithm. During development, it was realised that this was in fact not required, as creatures were simply added to the array of dead creatures in such a way that it was already in order. However, two technical requirements were still met as I used recursion in my word wrap algorithm, and I was still working with arrays of objects.
* User Requirement 3: Project should have a brief explanation of how neural nets work. This requirement had actually been satisfied throughout the vast majority of development, as can be seen in the high-fidelity prototype. However, this was removed after some end-user feedback in order to include explanatory text, thus further providing User Requirement 1: intuitive interface. In my opinion, sacrificing requirement 3 in order to more effectively achieve requirement 1 was the right choice to satisfy the majority of users.
* Functional Requirement 7: Creatures able to eat other creatures as well as food pellets. This function was intentionally removed during implementation, as after some research and experimentation, it was decided that the feature would actually hinder the creature’s abilities to learn.
* Functional Requirement 8: Neural nets stored in text files. This feature was also intentionally removed during implementation as it was realised it was not in fact required, and would hinder how intuitive the user interface was.

Through both end-user and on-going testing I have discovered that the program runs smoothly and is robust enough to deal with anything a user may try to do. This was achieved through a clear and uncluttered user interface which only allowed the user to do things that the program is intended to do. Additionally, many systems were setup to print error messages if anything were to go wrong, so that the errors can be quickly identified, and their sources easily found.

Testing

Component testing was carried out during the development of the solution, in order to test isolated components as they were created, and to catch bugs and errors early on in development before they became integrated with the whole program.

Integrative testing was also carried out at regular intervals during development in order to ensure that all components are able to work in tandem with each other.

Both integrative and component testing were very successful, allowing development to move on at a steady pace as I became confident that any already existing components worked correctly. It also did expose a few errors early on which were addressed.

Final testing was comprised of usability, acceptance, and functionality testing.

Functionality testing was carried out on the solution as a whole in order to test how smooth it ran, and whether it met the functional requirements. This was generally a success, ignoring the functional requirements specified above.

Usability testing was carried out in order to find out how intuitive the user interface was. Potential end-users were asked to use the program without any supervision and then asked to fill out a survey. The majority of the users surveyed were computer science students as they were the target audience, however a couple of novice users were also surveyed in order to get results from a more varied group of people. Overall, this was deemed a great success as all the users found the program interface intuitive and relatively easy to use, even those who deemed themselves novice users.

Acceptance testing was carried out to find out how interested users were in the program, and how satisfying the experience was for them overall. This was done simultaneously with usability testing using the same technique. Overall, this was mostly regarded as a success, as the majority of users reported that they enjoyed the experience, and those that didn’t also reported that they were not interested in machine learning to begin with.

End-user testing also provided a great number of potential features that are planned for future development, as detailed later in this report.

Personal Performance

During development of this project a record of progress was used to keep track of what was being worked on and note reflections about difficulties encountered and any changes to the design of the program. This was very useful, as it helped me stay on track with what I was working on, and made writing commit messages when using Git much easier.

Overall, I believe I could have been more successful with timekeeping as I did end up having to work on the project much more often towards the end of development than at the start, instead of spreading it out more evenly throughout development. As the deadline was approaching it also became clear that I would not be able to implement many of the additional features which I had designated to only be dealt with once every other aspect of the project was complete. This was fine, as it had always been intended that these features may not be implemented, however if I were to work on a project of this magnitude again I would wish to take much more care with timekeeping.

I think the choice of an object-oriented programming language to develop this program in was an excellent choice, as it massively increased the efficiency of coding the project. I believe if a non-object-oriented language had been chosen it would have taken significantly longer to develop this project.

Part of the reason I chose to undertake this project was to gain an more in-depth understanding of both neural nets and UI creation in Java, and I certainly feel I have come out with a much better understanding of both, due to both reading materials on the matters, and getting hands-on experience.

Before beginning development it was hard to know which component to develop first and how they would all fit together, however I found the creation of a Gantt chart and the writing of both pseudocode and class model diagrams really helped to breakdown what was needed and how it will logically fit together.

I believe I performed well in developing this project and ended up with a program that not only meets the requirements but that I personally am very proud of.

Further Developments

I believe the most important further development to implement would be the suggestion to implement a ‘Run for X generations’ feature, as this would make the program much more accessible for users with motor control disorders, who may find it more difficult to click buttons quickly.

Besides this, many customisation features have been suggested which would overall enhance the user’s experience, such as customised names, control over how many creatures are present, control over the speed creatures move and the speed at which food is added.

All these features are planned for further development, along with the features specified in the Gantt chart which would be implemented only if additional time was available.

It is also likely if further development were to occur that I would investigate and experiment with adding more inputs to the creature’s brains to see if any additional strategies can develop, along with experimenting with the conditions the creatures are in such as food drop rate and generation size to see how these affect their learning.

Lessons Learned

If attempting a project of similar magnitude in the future, I would make sure to spend even more time in the planning and development phase, as I believe they really helped me to stay on track with this project, and in my opinion are essential to any project.

In any future projects I would also take more time to ensure that all code I was writing was written in as modular and well-organised way as possible, as although I did mostly write modular code during this development, there were still areas in which the code was quickly written and not well designed, and so was harder to maintain when making changes or finding bugs. In future I will make sure to regularly take time to check through my code and ensure it is well designed with maintainability and efficiency of coding in mind.

Implications

The intended use of this project is for those who are interested in computer science and machine learning to play around with, either simply as a source of entertainment or to get them more interested in the topic by showing them what it’s capable of. As such, it would either be used in the context of a classroom or at home for people to research in their own time.

There are no known legal implications of this project as it does not store user details therefore cannot violate the data protection act, all work is entirely my own therefore it cannot violate the Copyrights Designs and Patents act, and it does not contain any malicious code and therefore cannot violate the Computer Misuse act. At the start of development, the chosen project name, Evolution Simulator, was discovered to already belong to an existing piece of work. For this reason, the name was changed to CreatureSim to avoid any implications of the Copyrights Designs and Patents act.

There are no ethical implications to my project as it is simply an educational and entertaining computational experiment.

There are almost no economic implications to this program as the only other software required to run it, Java, is entirely free. As with any software, however, a computer is required to run this program which does provide a slight economic implication.

As with all computer software, this project carried the environmental implication that it requires a computer to run, and manufacturing and running a computer contributes to the accumulation of greenhouse gases. However, I do not believe that the existence of this program in any way increases the demand of computers and so cannot be considered to contribute to the accumulation of greenhouse gases.

There are no social implications of this program as it does not require interaction with any other people, simply interaction between one person and the program itself.