

University of Plymouth

**SCHOOL OF ENGINEERING,
COMPUTING, AND MATHEMATICS**

**COMP3000
COMPUTING PROJECT
2023/2024**

AI SEARCH COMPETITION

Samuel Alcock

10689471

BSc (Hons) Computer Science

Contents

Acknowledgements.....	3
Abstract	3
1. Introduction	3
2. Background	4
3. Literature Review.....	6
4. Method.....	7
4.1. Risk Assessment.....	7
4.2. Agile	7
4.3. Sprints	8
4.4. Product Backlog – Microsoft Planner	8
4.5. Gantt Chart.....	8
4.6. Version Control – GitHub	9
4.7. Supervisor Meetings	9
4.8. Architecture	9
5. Legal, social, ethical, and professional issues	10
6. Requirements	11
6.1. Functional Requirements	11
6.2. Non-functional Requirements.....	12
7. Design	12
7.1. Initial Design.....	13
7.2. UML Diagrams	13
8. Project Stages	17
8.1. Sprint 0 – Project Initiation	17
8.2. Sprint 1 – Requirements Analysis	17
8.3. Sprint 2 – Q-Learning.....	18
8.4. Sprint 3 – SARSA.....	19
8.5. Sprint 4 – Procedural Generation.....	19
8.6. Sprint 5 – Hill-climber	20
8.7. Sprint 6 – Procedural Generation Improvements	20
8.8. Sprint 7 – Outputting Performance Data	20
8.9. Sprint 8 – Further visualisation.....	21
9. Testing	21

9.1. User Testing	21
9.2. Unit Testing	25
10. Evaluation	25
11. End Project Report.....	27
11.1. Summary	27
11.2. Requirements Review	27
12. Post-Mortem	29
12.1. Project Successes.....	29
12.2. Project Challenges	30
12.3. Improvements	30
12.4. Future Implementation	31
13. Conclusion.....	31
14. References	32
15. Appendices	35
15.1. User Guide	35
15.2. Risk Table.....	35
15.3. Product Backlog	36
15.4. Gantt Chart.....	38
15.5. Visual changes to agents	39
15.6 Visual changes to environment	40
15.7. Unit Tests	41
15.8. Initial wireframes	42

Word Count: 9991

Code: <https://github.com/SamAlcock/COMP3000-Sam-Alcock>

Product Backlog:

<https://tasks.office.com/live.plymouth.ac.uk/Home/PlanViews/uxwuV8faA0i2-8bCvgh27ZYABWy-?Type=PlanLink&Channel=Link&CreatedTime=638488592385150000>

Acknowledgements

I'd like to thank my project supervisor, Lauren Ansell for her guidance and support throughout the duration of this project.

Abstract

This report will communicate the development of the AI Search Competition, which is a project aimed at introducing A-Level and GCSE students to different types of Artificial Intelligence Algorithms. The goal of the project is to create an engaging educational experience, and to hopefully get students interested in the field.

To begin with, the report starts by introducing the three different AI algorithms. These are Q-Learning, SARSA and a hill-climber. Further knowledge is collected about these three AI algorithms with a literature review.

Following this, the agile project management approach, and project management tools used are discussed. Furthermore, the requirements of this project are stated, and potential issues that may arise from the AI Search Competition are mentioned, along with what has been done to mitigate or remove the risk. The sections after this are a narrative of the two-week development sprints, with the goals and objectives achieved from each, and then both user and unit testing.

The evaluation section concludes that SARSA is the best algorithm, due to its slightly faster convergence to the optimal solution than Q-Learning. Both are very far ahead of the hill-climber, which fails to find the search target at any point during testing.

The final sections reflect on the AI Search Competition as a whole, by discussing the successes, challenges and improvements that can be made. After this, ideas for future implementation are given to further improve the project. The report then ends with concluding statements about the AI Search Competition.

References, and an appendix can be found at the very end of this report.

1. Introduction

The aim of this project is to introduce GCSE and A-Level students to different types of Artificial Intelligence (AI) algorithms in a visual and engaging way, and to hopefully inspire people to further inform themselves about the field. It has been found that AI is underutilised in education, which suggests that there are missed opportunities to improve quality of students' learning (Luckin and Cukurova, 2019). Luckin and Cukurova (2019) also found that "AI is here to stay" and "will have an increasing impact", so it is important for students to get used to the presence of AI and to learn more about it. Ultimately, if the general public are more informed and educated about AI then its potential risks and opportunities can be raised and discussed more effectively. Alongside educating, this project could also promote the use of AI educational tools to enhance student learning.

In this project, three algorithms have been developed, these are Q-Learning, State Action Reward State Action (SARSA), and Hill Climbing. Each algorithm controls an agent in real time, where the agent is looking for a search target in a procedurally generated environment. An agent is an entity that is able to interact with an environment, which is a space where agents exist and operate (Mohanty *et al.*, 2017). Moreover, each agent is in its own separate, identical environment. Thus, a comparison will be made between the algorithms to evaluate which is best for the given scenario.

This report will start by exploring the different algorithms by describing how they work. This will then be followed by reviewing relevant literature surrounding the project's topic to further gain a detailed understanding. Furthermore, the legal, social, ethical, and professional issues of the project will also be addressed, including the ways that they have been mitigated in the duration of development. Thereafter, the project requirements will be discussed, along with technologies used and a justification as to why, designs created and a narrative on each sprint throughout implementation. Finally, this will then be followed by a post-mortem and conclusion, where any successes, improvements needed, and future development ideas will be discussed.

2. Background

In this section, the algorithms used in this project will be mentioned and explored in depth, whilst providing relevant literature to support it. A brief overview of how they function will also be provided.

Q-Learning and SARSA are both examples of reinforcement learning algorithms. Reinforcement learning is a machine learning paradigm where an agent learns optimal behaviour through a process of trial and error, where it can either gain positive or negative feedback from the actions it takes (Kaelbling, Littman and Moore, 1996). Moreover, reinforcement learning has a wide range of applications. Some examples include healthcare where it has been used for medical imaging and personalised medicine, and in social robotics, where a robot is tasked with interacting with humans such as in elderly care (Akalın and Loutfi, 2021; Coronato *et al.*, 2020).

Q-Learning is a very popular reinforcement learning algorithm, that uses off-policy behaviour to find the optimal solution (Jang *et al.*, 2019). Off-policy behaviour is where the target policy and behaviour policy are not the same. A target policy is a strategy that an agent aims to learn, and a behaviour policy is the strategy employed by an agent for selecting its next action. Q-Learning takes actions based on a Q-value, which represents the estimated effectiveness of taking that action. Based on the reward received from the state reached from said action, the agent will then update the algorithm's Q-table. A Q-table holds all the Q-values for every action in the environment. Thus, the more the agent interacts with the environment, the more

accurate the values in the Q-table will be (Kantasewi *et al.*, 2019). The equation for the Q-Learning update function is as follows:

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$$

For this equation, $Q(s, a)$ represents the Q-value of the state s , which is where the agent has moved from and the action a that has just been taken by the agent. α (alpha) is the learning rate, this is a constant value between zero and one, lower being a slower learning rate. The reward received from moving into the state is represented by r . γ is the discount factor which represents how much the algorithm values future rewards, the higher the value of the discount factor, the more weight is given to the value of future actions. For example, if the value of γ is 1 then future values are just as important as current rewards. $\max_{a'} Q(s', a')$ is the highest Q-value of all possible actions in the next state. As mentioned previously, the amount this contributes to the overall updated Q-value depends on the value of γ .

SARSA is another reinforcement learning algorithm which unlike Q-Learning, is an on-policy algorithm. This is where the agent learns from actions that it is actually taking. Similar to Q-Learning, SARSA also takes actions based on a Q-value and updates a Q-table based on rewards gained from interacting with its environment. The equation for the SARSA update function is:

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma Q(s', a') - Q(s, a))$$

The update function for SARSA is very similar to Q-Learning, with the only difference being that SARSA does not include the maximum action when updating the Q-value. This is because SARSA uses the Q-Value of the action it is actually going to take.

The third algorithm used was an optimisation algorithm called hill-climbing. A hill-climber is a set of random instructions that performs a sequence of mutations (Sebag and Schoenauer, 2002) to find the optimal solution. Mutations include changing an instruction to a different instruction, and calculating a new fitness to see if this mutation is an improvement over the parent. If so, this new mutation will become the new parent. Hill-climbing algorithms can have varying mutation rates, but most normally prefer to have a smaller mutation rate (Sebag and Schoenauer, 2002). Real world examples of hill-climbing algorithm use are in Application Servers (Xi *et al.*, 2004), for playing Mastermind (Temporel and Kovacs, 2003) and for timetabling (Rjoub, 2020).

3. Literature Review

In this section literature relevant to the project will be reviewed. There will be a focus on research on applications of hill-climbing, Q-Learning and SARSA. The aim of this literature review is to gain a deeper understanding of each algorithm, and to see how these findings can be related to the AI Search Competition. Relevant literature will be explored and summarised to begin with. After this, applications of each algorithm will be discussed. This will then be followed by a conclusion to state key findings.

A paper that involves a hill-climber is 'An experimental comparison of a genetic algorithm and a hill-climber for term selection' by MacFarlane *et al*, 2010. The paper aims to compare a hill-climber with a genetic algorithm at their performance with term selection. The authors state that hill-climbers are prone to getting caught in a local optima. This is the most extreme point of a function, within a specific area of input space. However, their findings suggest that even though this is a problem for hill-climbers, it does not hold it back from achieving effective results. It outperforms the genetic algorithm for term selection in this paper due to a more consistent performance.

In relation to this project, even though evidence suggested that being stuck in a local optima is not a problem for term selection, in a broader context it is still a major issue for finding optimal solutions. Therefore, it is worth making sure that the hill-climber implemented is not getting stuck in a local optima when searching. Furthermore, this paper shows the usefulness of comparing different machine learning approaches for specific tasks. Even though the proposed genetic algorithm was outperformed, it further proved the effectiveness of hill-climbers for term selection. For the AI Search Competition, a hill-climber is an unorthodox approach to a search task. However, at best it could prove an innovative approach to search tasks, and at worst it can further prove the strengths of traditional approaches such as Q-Learning and SARSA.

A paper exploring Q-Learning is 'Application of Deep Q-Learning for Wheel Mobile Robot Navigation' by Mohanty *et al*, 2017. In this paper, the authors incorporate Q-Learning with a neural network. The combination of these two machine learning approaches is known as deep Q-Learning. This is applied to get a robot to navigate around obstacles in both a physical and a simulated virtual environment. In this paper, a log-based reward function was used. This is said to have caused problems with training times due to logarithms being non-linear. What also may have increased training time is the laptop specifications being "4GB RAM and i5 processors". The potential lack of a GPU would mean less parallelisation, and therefore slower training times. However, lack of hardware power is not mentioned, consequently, it is up for debate whether this was an actual problem or not.

In the context of the AI Search Competition, this paper shows that it is important to get the weighting of rewards correct, as having a relatively fast training time is crucial. This is especially relevant for this project, as training is done in real time so training time must be quick to keep users engaged. This paper also shows that Q-Learning is an

established approach to search and navigation tasks, which suggests that it is a suitable choice to use in the AI Search Competition.

“A Reinforcement Learning Approach to the Shepherding Task Using SARSA” by Kendrick *et al*, 2016 is a paper that employs SARSA to act as a dog that herds a flock of sheep to a target position in a virtual environment. A unique feature of this task is that the agent has multiple subgoals. This makes for a more complex optimal solution that the agent has to learn, which it finds consistently after roughly 350 episodes. This shows how SARSA can perform well on more complex tasks and proves that it can be a very competent algorithm when implemented correctly.

This piece of literature shows the capabilities of SARSA. For the AI Search Competition, SARSA should be expected to be at minimum very competent at searching and potentially the strongest algorithm for the task being created. This may not be the case, but if it is far off this prediction then it is likely that there is an issue with implementation.

4. Method

In this section, the main methodology involved in planning and developing the project will be discussed. This involves the project management approach employed, sprints, product backlog and the version control used.

4.1. Risk Assessment

In the early stages of the project, a risk assessment took place to identify potential issues during various stages of the project. Risks are a part of all projects, and encountering issues is inevitable. However, thinking about what risks are involved, and ways to reduce the impact will ensure that the project will run more smoothly. A risk table was created to outline potential risks that could occur during development (see appendix 15.2). They are rated in terms of likeliness of happening, and how severe the risk would be if it were to arise. Alongside this, there is an overall risk rating to easily see which risks are of the most or least concern.

Whilst many of these risks did not occur throughout the duration of this project, their impact would have been mitigated by this plan. Some risks such as illness or having slower progress than anticipated did occur and having a plan in place definitely helped reduce the impact of their disruption.

4.2. Agile

The project management approach chosen for the AI Search Competition was agile. Agile development involves iterative development that delivers a product incrementally (Barlow *et al.*, 2011). Agile is beneficial for this project as it enables more tolerance in the event of a change in requirements (A B M Moniruzzaman and Syed Akhter Hossain, 2013). The flexibility provided by agile makes it a very good approach for this project. Speaking from experiences of using agile in this project and past projects, the

adaptiveness of this project management approach helps create a higher quality product, as changing needs can be more easily addressed.

A limitation of agile would be the difficulty of understanding how different functional aspects of a project interact with each other (Seyfi and Patel, 2010). This is especially relevant as some larger functions may become fragmented due to being split up into small, separate sprints. To mitigate this disadvantage, it is important to keep up to date diagrams of architecture, and regularly reviewing sprints. This way, the project is kept organised and therefore, less likely to make this drawback a large problem.

4.3. Sprints

Each sprint during development was a length of two weeks. Two-week sprints were beneficial to this project because it would allow each larger task to be broken down into smaller, more achievable goals. This in turn allowed for the more important aspects of the AI Search Competition to receive a suitable amount of time allocated to them. Moreover, it would force steady progress to be made throughout the duration of implementation. In addition to this, a sprint length of two weeks has led to steady implementation in past projects. Therefore, it was expected that this would be a suitable sprint length. Shorter sprint lengths such as this also compliment an agile project management approach, due to the ease of being able to change priorities without having a larger impact on the rest of the project.

4.4. Product Backlog – Microsoft Planner

Sprints were managed using Microsoft Planner as a product backlog. Each bucket would represent a sprint, which would then be broken down into multiple tasks needed to be finished to complete the sprint. These tasks could then be split into 'incomplete', 'in progress' and 'completed'. Using a product backlog is advantageous as it provides an easy way to decompose a task into multiple smaller ones and to abstract the unnecessary aspects of a larger task. Using Microsoft Planner is also a good visual representation of work needed to be completed during development, which in turn provides a clear outline of what needs to be worked on. Similarly to sprint length, Microsoft Planner also complimented the agile approach as it allows for changes to sprint plans to be made and visualised quickly and easily. See appendix 15.3 for screenshots of the product backlog.

4.5. Gantt Chart

To further visualise the tasks required for the project, a Gantt chart was used. Gantt charts are useful to view an overall timeline of the project, which can then be used to see if overall project development is on track or not. Furthermore, Gantt charts give a good idea of how much time can be allocated to different parts of the project. This can then be used to see which tasks are of higher importance or risk than others. Similarly to the product backlog, it helps keep the project more organised, which leads to a better implementation and smoother development overall. See appendix 15.4 for screenshots of the Gantt chart.

4.6. Version Control – GitHub

GitHub is a popular version control platform for open-source projects (Cosentino *et al.*, 2017). According to GitHub, it is used by over 100 million developers, and 4 million organisations (GitHub, Inc., 2019). Along with being familiar with GitHub due to using it as version control for other projects, these reasons show that it is a good choice for this project.

Version control is important to keep track of progress made in development, and it also provides a version history of the project. Whilst not the main reason for using version control, it is useful to see progress of the project overall to use for reflection on sprint progress. Furthermore, version control keeps an up to date and working version of the project as a backup in case of running into issues that cause the local copy to break or to be lost. This is much more effective than keeping local backups, as if the local machine breaks, then no progress will be lost.

4.7. Supervisor Meetings

Biweekly supervisor meetings acted as sprint reviews and were in place to review project progress. These were done in groups where each participant discussed their current sprint and future sprint plans. Some meetings were a ‘show and tell’ where the projects would be shown to the rest of the group. These meetings were invaluable for receiving feedback and provided an opportunity to adjust requirements if needed. Additionally, the feedback received from meetings provided further clarity on what would need to be done in future sprints.

4.8. Architecture

4.8.1. Unity

To visualise the algorithms and their agents, Unity was chosen. Whilst Unity is commonly used in game development, it is an ideal choice for the AI Search Competition as it provides an effective way to create a 3D environment. Familiarity with the software also increases ease of use, and ultimately makes implementation smoother. Another useful aspect of visualisation in Unity is spotting errors in the AI algorithms much easier, as any unintended errors will likely be visualised when agents are searching.

4.8.2. C#

C# was chosen as the programming language for this project primarily due to it being natively supported by Unity. Because of this, built-in Unity libraries can also be used. This will allow for a smoother implementation, as developing and debugging will be much easier. Whilst Python is arguably a more convenient choice for developing AI algorithms, C# allows for the algorithms to be used in real-time instead of only visualising a replay of the learning due to the native support mentioned previously.

5. Legal, social, ethical, and professional issues

In this section, legal, social, ethical, and professional issues will be mentioned. Actions taken to address or alleviate these concerns will also be discussed.

Obeying the rules laid out by the Data Protection Act 2018 (GOV.UK, 2018) is a legal issue that needs to be addressed for every project. In a project such as this one, as the target audience is mainly younger it is even more crucial to make sure that there are no breaches of data protection. For the AI Search Competition, no personal data is stored however data is collected about the performance of each algorithm. This is the absolute minimum data that needs to be collected so that the project can properly function. To make sure that the project is not in breach of Data Protection laws, everything is stored locally and not shared. To ensure transparency when data is stored, the program will output a message saying that this has happened, and where the data is located if the user wants to find it.

A social issue would be that this project might be able to put teachers out of jobs by teaching a topic on its own. Whilst this is a large concern for many applications of AI, it should not be a major concern for this project as it is a visualisation of a process that would need to be explained by a teacher. In the context of this project in its current form, the project development was more focused on the actual algorithms themselves and the environment they search in. Therefore, it is arguably no more of a threat to a teacher's job than showing a video. Overall, this project is designed to work as a tool to enhance education in the field of AI and aimed at giving teachers a new method to teach students. Although it is important that this issue take is be taken into account for future development, especially if development is focused on increasing interactability.

An ethical and professional issue with the AI Search Competition would be potentially miseducating, or misleading students. As this is an educational tool, effort needs to be made to ensure that all AI algorithms are as high quality as possible and are fully functional to avoid these issues. In addition to this, algorithms need to not have any environmental advantages over the others. This has been addressed by giving the algorithms the same number of agents, having identical environments, and by keeping the rewards the same. For example, the reward for an empty state is -1 for all three algorithms instead of having reward increase the closer the agent gets to the reward. Also, by collecting data on the algorithm performance, any comparisons made can be backed up by statistics gathered.

Another ethical issue is the risk of dual use. Whilst dual use is a potentially beneficial use of AI, it also has the potential to be dangerous (Cherney *et al.*, 2023). This project runs the risk of the algorithms being used for a dangerous or unethical use, especially due to it having a generalised use of searching in unique environments. To avoid this, source code needs to be inaccessible when distributed to end users. However, completely hiding the source code from users may also hinder beneficial adaptations of the AI algorithms, which then creates a professional issue as the project is not open

source. Therefore, future development should allow the source code to be shared as long as users are made aware of the dangers of AI misuse beforehand.

6. Requirements

Before starting development, the functional and non-functional requirements of the project were decided on. Both of these requirement types were split up into three subcategories of 'Must', 'Should' and 'Could' depending on the importance of the individual requirement. 'Must' are the requirements that have to be in the final product, and where the quality or functionality of the project will be much worse without it. 'Should' are requirements that are important, but the project can still be of high quality without it. Finally, 'Could' are the requirements that not very important and the project quality would not be greatly impacted without them. However, they may be a good addition to the project. It is important to do this to begin to break down a project into smaller tasks, and to gain an understanding of what the project truly requires.

6.1. Functional Requirements

Functional requirements are requirements that relate to the practical elements of a project (Kurtanovic and Maalej, 2017). These would be features that are to be programmed.

- **Must**
 - Have a minimum of three Artificial Intelligence algorithms.
 - Be visually appealing so that users are engaged with project.
 - Be user friendly so that it can be operated and utilised by different types of users (e.g. students, teachers of different technical abilities).
 - Every agent is in an identical environment, so that the search area complexity and difficulty is the same for each algorithm.
- **Should**
 - Have four reinforcement learning algorithms.
 - Generate obstacles virtual environment so searching is more unique, complex, and interesting.
 - Have procedurally generated search areas to make each iteration unique, and to ensure that agents are not used to a specific environment layout.
 - Explain what is occurring during each iteration and why it is happening.
- **Could**
 - Test in real life environments by using robots as the agents.
 - Have more than four reinforcement learning algorithms.
 - Increase interactivity for users by being able to modify parameters of task (e.g., more search time, agents move faster, environment more complex etc).

6.2. Non-functional Requirements

Non-functional requirements are requirements about the properties that the project contains (Kurtanovic and Maalej, 2017). These requirements are used to increase the overall quality of a project.

- **Must**
 - Algorithms must be able to learn in an appropriate amount of time – becoming competent searchers must not take days of iterating.
 - Ensure that algorithms are properly implemented.
 - Have an appropriate level of environment complexity.
- **Should**
 - Be visualised in Unity.
 - Have agents making decisions in real-time instead of training before visualisation.
 - Implement in C# for compatibility with Unity.
- **Could**
 - Have implementation of algorithms in real-time in C# and trained beforehand in Python to compare search performance.

User stories are an expansion of the functional and non-functional requirements and put them in the perspective of the end users. This also shows the different types of end users that would be using the project. In the context of the AI Search Competition, the end users are students and teachers. User stories are important to be able to understand the needs of the end users, doing this can help validate which requirements are the most crucial in the project. Prioritising requirements based off user stories helps for making sure that the features most needed will have more focus, this will lead to better user satisfaction and a higher quality product overall.

- As a student, I would like to have an interesting and engaging way to introduce me to reinforcement learning.
- As a student, I would like to be educated in a unique way to make learning interesting.
- As a student, I would like to have a way to introduce me to topics that I may not learn about otherwise.
- As a teacher, I would like to be able to educate students in a new and unique way.
- As a teacher, I would like an easy-to-use system so that the project can be used to its full potential.
- As a teacher, I would like the project to provide educational value by explaining to students what is occurring.

7. Design

In this section, diagrams used for designing the AI Search Competition will be shown along with a description of each diagram.

7.1. Initial Design

Very early on in the project, an initial wireframe design of what the search environment and layout could look like. A wireframe is the designing of software which gives an idea on how it will look and function (Faranello, 2012). It is a highly simplified design of the final product (Chen *et al.*, 2020). Wireframes can be made as a drawing or designed in software. In this case, the wireframe was designed in a software called draw.io. Wireframes are useful to get an early idea of what the final product might look like, and a good foundation to start thinking about what is required to achieve the functionality needed. The diagrams mentioned can be found in appendix 15.8.

7.2. UML Diagrams

Unified Modelling Language (UML) is a notation representing a large number of diagrams which are able to effectively model any aspect of a project (Reggio *et al.*, 2013). For this project, UML diagrams used are a class diagram, use case diagram, and an activity diagram. UML diagrams are useful as they can be adapted as project needs change, this means that there can be an up to date and visual representation of the functionality of the project. Similarly to the wireframe design, the UML diagrams shown were created with draw.io.

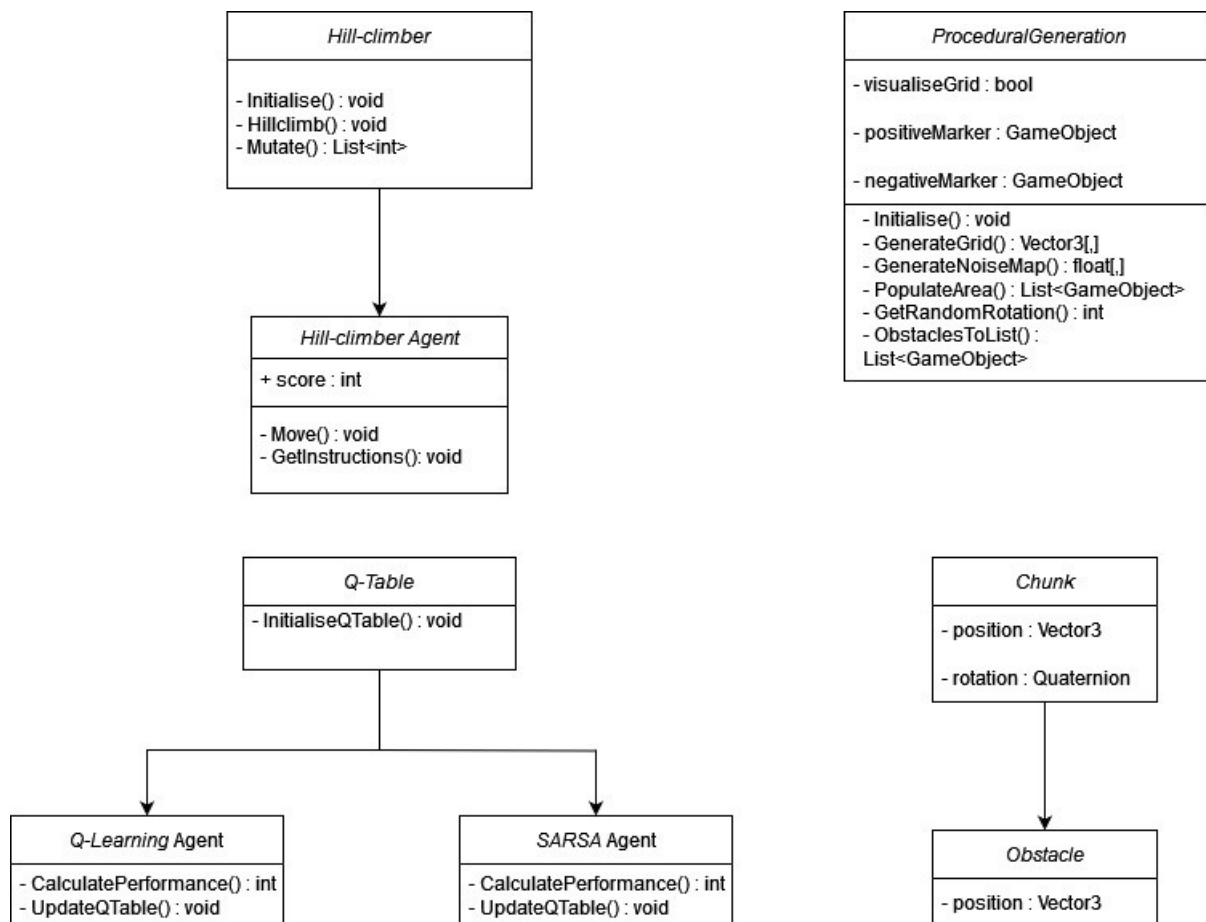


Figure 1 Class Diagram

The UML diagram shown in the figure above is a class diagram. This diagram represents the different classes within the project, and their relation to one another. Classes are linked by arrows, which represent an association (Evans, 1998). Whilst the initial diagram is not likely to be a fully accurate representation of the actual implementation, future adaptations can give a more accurate representation of the final product. Furthermore, a class diagram is important to use to get a good idea of how to structure code, so that development can be a smoother process.

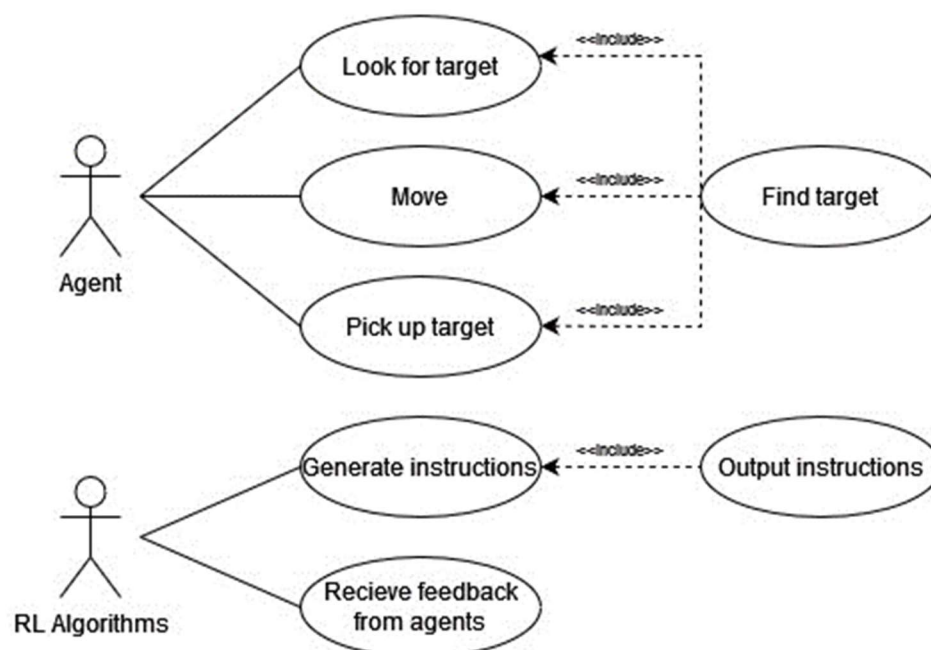
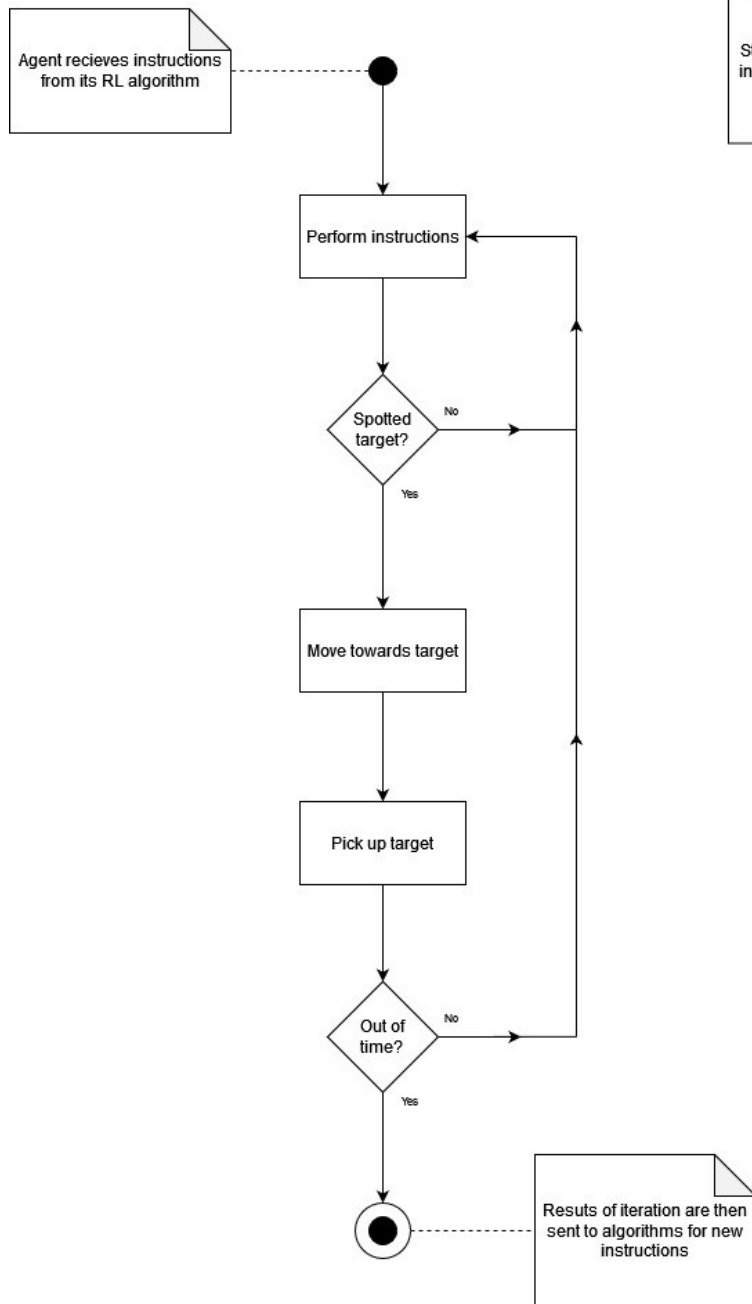


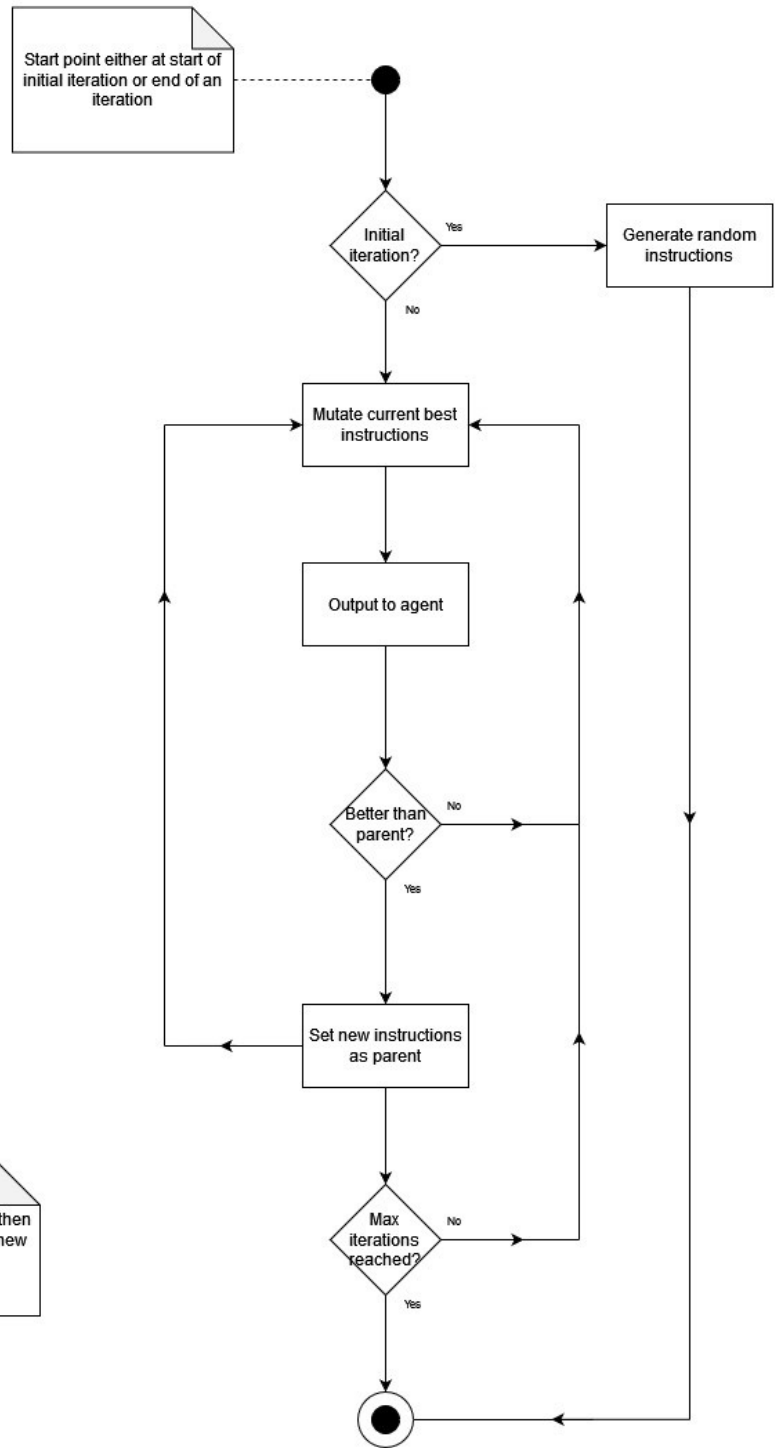
Figure 2 Use Case Diagram

The figure shown is a use case diagram. Use case diagrams consist of actors, which is an entity that plays a role in the program, use cases, which are a function performed to achieve a specific objective, and associations, which represent a link between an actor and a use case. A use case diagram is an effective way of outlining how agents will interact with their environment, and a general overview of how the AI algorithms will function. Similar to the class diagram and UML diagrams as a whole, they evolve alongside the evolving needs of a project. Therefore, there will always be an up-to-date version of the diagram that remains as an accurate representation of the project.

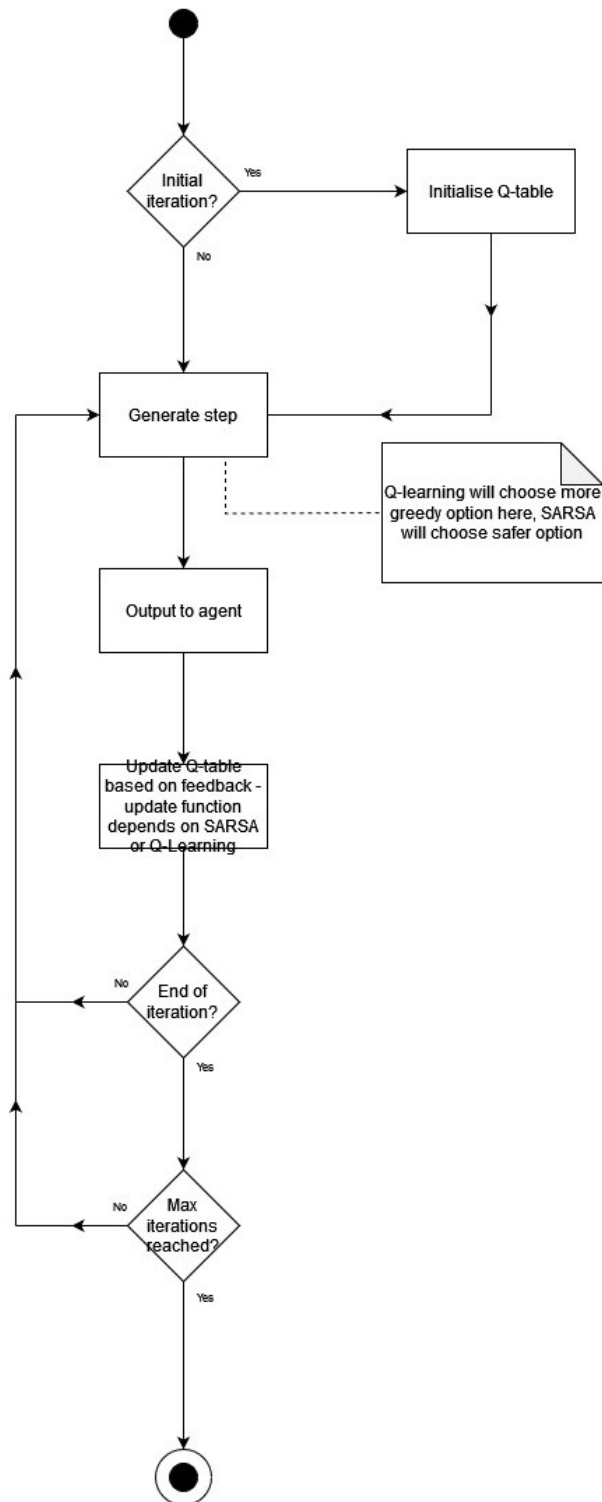
Activity Diagram for agents each search iteration



Activity Diagram for Hillclimber



Activity Diagram for Q-Learning and SARSA



Activity Diagram for Environment Procedural Generation

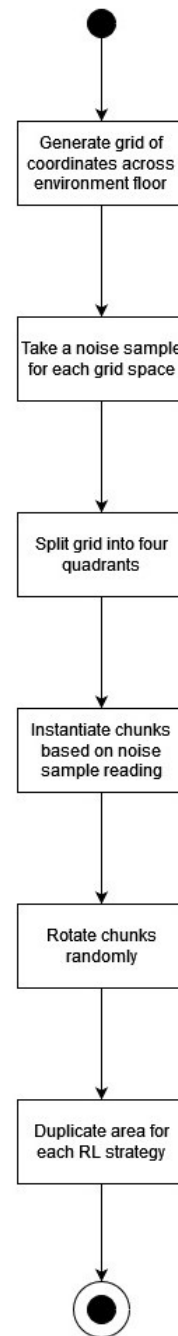


Figure 3 Activity Diagrams

Similarly to flowcharts, activity diagrams represent a sequence of instructions and executions (Eshuis and Wieringa, 2004). Activity diagrams consist of activities, which

represent an action to be taken, connectors, which are arrows that order each activity, decisions, which have branching connectors based on the output of a decision and start and end symbols to show the start and end of an activity diagram. In the context of this project, activity diagrams are beneficial to break down each of the major features needed to be implemented to get a better understanding of how to implement them.

8. Project Stages

This section will describe each sprint that took place throughout the project. Each sprint will be summarised, as well as providing the outcomes.

8.1. Sprint 0 – Project Initiation

Sprint 0 required planning out what the project would be about, and identifying the tools needed to achieve this. This involved setting up the project's GitHub repository, deciding on what technologies to use, creating a risk assessment, initial project vision and title, and to create a product backlog.

Deciding which AI algorithms to use meant the need for initial research into different methods. This was when Q-Learning and SARSA were chosen originally as the best algorithms to implement. Hill-climber was later chosen at a later stage. The project title and vision were also made during this sprint, however some changes have been made since. The project was originally titled 'RL Search Competition'. However, once it was decided to implement a Hill-climber the name was changed to better reflect the algorithms included in the project.

The outcome of this sprint was that the project was decided to be an educational tool that compares multiple AI algorithms and that this would be coded in C# so that it can be visualised in Unity. This sprint served as a platform to gather the project requirements in the next sprint.

8.2. Sprint 1 – Requirements Analysis

Sprint 1 involved setting out the initial requirements and designs of the AI Search Competition. This meant creating UML diagrams, user stories, and then defining functional and non-functional requirements from this.

Due to this project using an agile project management approach, the UML diagrams created during this sprint were not final. However, the diagrams created provided a solid foundation to start initial development in future sprints.

The requirements and user stories also gave a valuable insight into the order of development for future sprints. As mentioned in the methods section of this report, they were a useful tool to prioritise certain tasks over others.

Overall, this sprint clarified exactly what needs to be done in the next sprints and an outline of the steps needed to be taken to implement the requirements.

8.3. Sprint 2 – Q-Learning

This sprint was focused on implementing the Q-Learning Algorithm. The aim of this sprint was to have a Q-Learning agent to be able to competently navigate an environment. It was expected that another sprint may be needed to fully complete this however.

To achieve this, an environment for the agent to navigate had to be created. For now, this could be an area with obstacles that the agent cannot walk through. But future development would allow for environments to be procedurally generated. The environment was split into a grid of squares, the agent could move to squares adjacent to the square it is currently on.

As rewards are an integral part of reinforcement learning algorithms, therefore rewards within the environment needed to be created (Li, 2017). Initially the rewards an agent could receive were as follows:

Object encountered	Reward
Start state	0
Empty state	-1
Obstacle	-500
Reward state	50

This seemed to be a relatively good balance of rewards, as to achieve the maximum total reward the agent would have to find the most efficient path to the reward state. This is because moving on more empty states than needed would lead to a lower total reward. The main issue with this set of rewards was the start state giving 0. This meant that the agent would constantly go back and forth between the start state and an empty state. Although, once the start state reward was changed to -1 this was no longer a problem.

To implement Q-Learning, a Q-table was created where each state in the environment had four Q-values for each potential direction. Using the update function mentioned earlier, the agent then updated the Q-table based on the information it gathered in the environment.

A minor issue encountered was getting exploration for the agent functioning. This was because when the agent receives values from the Q-table, it gets four values for each state. This includes edge states which have invalid moves outside of the search environment. This is not a problem for the normal process of choosing a Q-value, as the best value is chosen. However, when it is random it can lead to an invalid move. This made implementation of exploration more difficult than it should have been, and caused a small amount of lost time.

This aims of this sprint were achieved. The Q-Learning algorithm was functional and controlled an agent able to navigate an environment and find the reward state. Progress on this sprint was quicker than originally expected due to no large issues

being encountered. The agent was initially finding the optimal solution slower than expected, this was due to an incorrect calculation of rewards in the update function. Once this was fixed, the algorithm was much faster and effective at searching.

8.4. Sprint 3 – SARSA

In this sprint, the goal was to implement SARSA. Similarly to the implementation of Q-Learning, the aim was to have SARSA fully functioning and competently searching for the reward state.

For developing this, SARSA had the same requirements as Q-Learning did. It needed an environment to search, a reward system, and a Q-table. This was already created in the previous sprint and could be used for SARSA as well. This allowed for more development time to be focused on the actual algorithm. It would not make sense time wise to redevelop this and would create potential imbalances when comparing both algorithms due to them using different reward systems, or different initial Q-values. It is worth mentioning that whilst both algorithms used the same Q-table code, they both had their own instantiations of the Q-table.

The aims of this sprint were achieved as SARSA was functioning as intended, and the agent was effective at searching for the reward state. However, there were problems with creating an identical environment using the original environment used for Q-Learning when placing the reward state and start position. This slowed development and but led to developing a more efficient method to manage multiple environments, which helped when developing the Hill-climber.

8.5. Sprint 4 – Procedural Generation

Implementing a procedurally generating environment was the aim of this sprint. Whilst ideally having a wide range of diverse search environments to be generated, this sprint mainly aimed to have procedural generation to be working so it can be improved later on.

The way this was implemented was to use Unity's built-in Perlin noise function to sample different noise values. Four chunks would then be instantiated onto the environment based on the value of the noise sample. The chunks are a layout of pre-made obstacles, which each cover one quarter of the search area. These chunks are then randomly rotated. Consideration needed to be made when making these chunks is to make sure that it doesn't block an agent into a corner of the environment, or that it does not overlap with another chunk as this could create unintended side effects.

This sprint was a success overall as procedural generation was implemented, and at this stage of development it was ready to be built upon with more different chunks in a future sprint.

8.6. Sprint 5 – Hill-climber

A hill-climber was the third and final algorithm needed to be implemented. Similarly to the other two AI algorithms in past sprints the aim was to have the algorithm fully functioning by the end of the sprint.

As the foundational features of the search environment were fully complete, the hill-climber agent had a working area to search in already. Also, as mentioned in the SARSA sprint a reward structure was already in place. Therefore, this sprint could be fully focused on implementing the hill-climbing algorithm.

Overall, the sprint was satisfactory. The hill-climbing was implemented and was functional. However, the agent does not perform well. This could be a sign of either poor implementation, or that hill-climbing algorithms perform poorly in this type of task. More development time had to be allocated to this problem to rectify this issue.

8.7. Sprint 6 – Procedural Generation Improvements

The aim of this sprint was to improve the chunk designs, and to create more chunks so that search environments will be more complex, interesting and varied. By the end of the sprint, the procedural generation aspect of this project should be fully complete.

The idea for chunk redesigns was to make chunks that are more similar to a maze. To do this, chunks were made with more dead ends and pathways. An issue with this is that chunks need to be able to piece together so that agents will not be blocked off from areas in the search space but leaving large gaps in between chunks will greatly decrease complexity. To solve this, chunks were carefully designed so that they could link together but did not link together so much that it would block agents off.

Progress in this sprint progressed as planned, and the environment procedural generation was fully complete which meant that the focus for the remainder of the project could be on other features.

8.8. Sprint 7 – Outputting Performance Data

Whilst it can be relatively easy to see which AI algorithm performs the best by watching them search, there needed to be statistical evidence to prove which is best. This can also be used to see which algorithm is best on average, as performance can vary depending on environment.

To do this, it was decided to output each AI algorithms' total reward per generation so that it can be put on a graph and easily compared. A 'FileStream' was used to implement .csv management. Each algorithm had its own .csv file.

This sprint was quite small, so development was completed without any issues.

8.9. Sprint 8 – Further visualisation

The visual appeal of a project is important when it the end user perceiving the quality and usability of a product (Lindgaard *et al.*, 2011). This means that a sprint needed to be dedicated to making the AI Search Competition look better.

Based on feedback received from user testing, it was decided that the AI Search Competition firstly needed more colour as walls and obstacles were grey. To improve this, the walls surrounding each agents search environment were made to match the colour given to the agent. This reduced the amount of grey, fulfilling improvements suggested in user testing.

Another improvement suggested was to make agents more unique, as they were currently different coloured capsules based on what AI algorithm they were related to. To rectify this, agents were each given faces. This led to an unforeseen issue of agents now having to be rotated depending on what direction they were moving in, as without this their faces would be facing in a different direction than the direction that they were moving.

To see screenshots of these changes, see appendix 15.5.

9. Testing

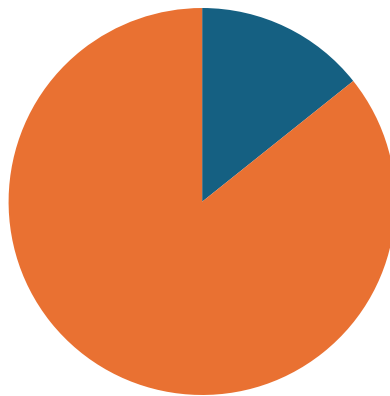
9.1. User Testing

User testing was undertaken to get a clearer idea of how the visual aspects of this project would be perceived by end users. To do this, the AI Search Competition was shown to participants. They then would fill out a Google Form which would ask them questions about their opinions on how the project looked. Users were asked three Likert scale questions, and then two longer answer questions to get opinions on what is good or bad about the AI Search Competition.

The feedback received was overall positive, and the feedback received was useful to learn the positives and negatives about the project. As mentioned previously, the improvements suggested could then be implemented in the final sprint. After they were implemented, the same questions were sent to the same participants to see if opinions have improved.

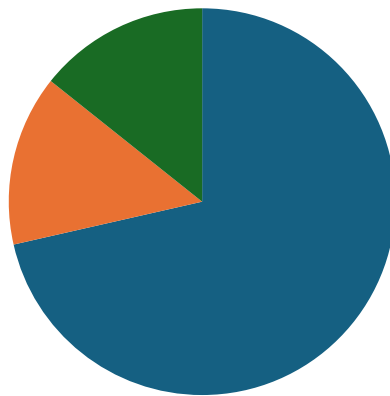
9.1.1. First round of questions

The Unity implementation is visually appealing



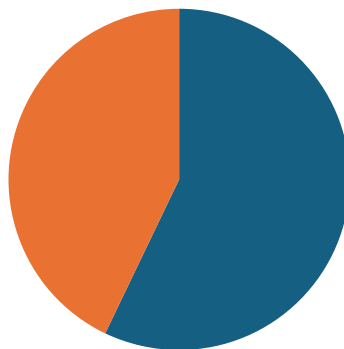
Strongly agree Agree Neutral Disagree Strongly disagree

It is easy to tell which AI algorithm is which



Strongly agree Agree Neutral Disagree Strongly disagree

This would be an interesting way to learn about AI



Strongly agree Agree Neutral Disagree Strongly disagree

9.1.2. Improvements Suggested

What is good visually about this project?

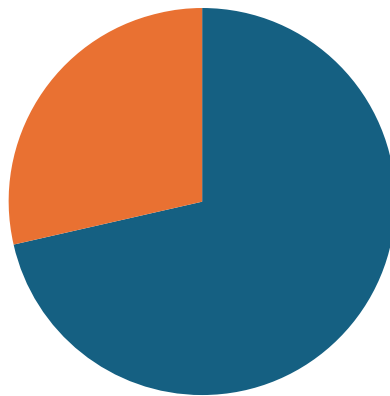
- “Simplicity helps the user focus on the important elements, improving understanding.”
- “The way the AI bots moved around the screen.”
- “Consistent endpoints across all 3 models, and a clear starting point (blue box).”
- “The visual simplicity creates clarity, in understanding a topic that is not well understood by a vast majority.”
- “Colourful.”
- “Each of the 3 algorithms are clearly defined and you can compare all 3 at the same time as the programme is running.”
- “Simple and easy.”

What could be improved visually about this project?

- “Make the contrast of the agents stronger, so they stand out a little more, or make them slightly bigger.”
- “Bigger images.”
- “Making the agents and reward more visually distinct could perhaps remove any misunderstandings as to their function.”
- “Could be more stylised.”
- “Maybe the use of more contrasting colour scheme.”
- “Use a different colour for the obstacles in each of the 3 algorithm sections.”

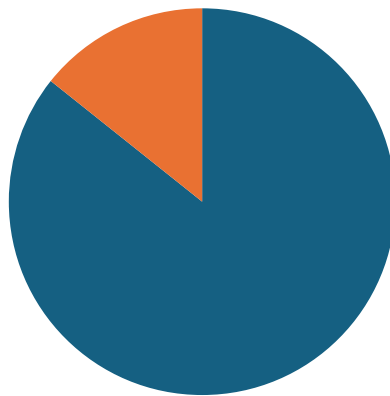
9.1.3. Second round of questioning after implementing improvements

The Unity implementation is visually appealing



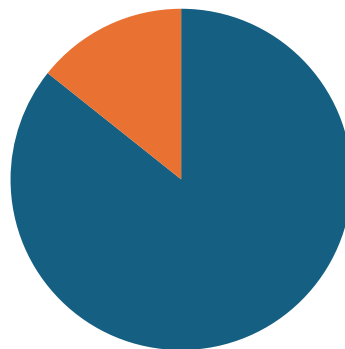
Strongly agree Agree Neutral Disagree Strongly disagree

It is easy to tell which AI algorithm is which



Strongly agree Agree Neutral Disagree Strongly disagree

This would be an interesting way to learn about AI



Strongly agree Agree Neutral Disagree Strongly disagree

The second round of questions show that the visual quality of the AI Search Competition has made an improvement, with all participants either agreeing or strongly agreeing with the statements shown.

9.2. Unit Testing

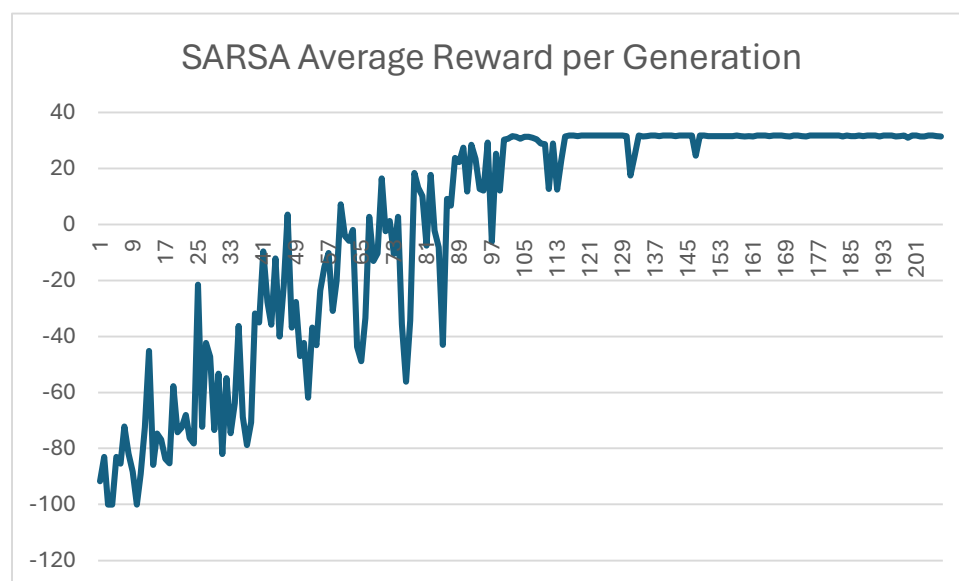
Unit testing is an important part of software development (Daka and Fraser, 2014). It is the isolation and testing of small parts of code, to make sure that it is producing the desired result (Runeson, 2006). Unit tests are very useful as they can save a lot of development time trying to fix a problem (Ahmad Dalalah, 2014).

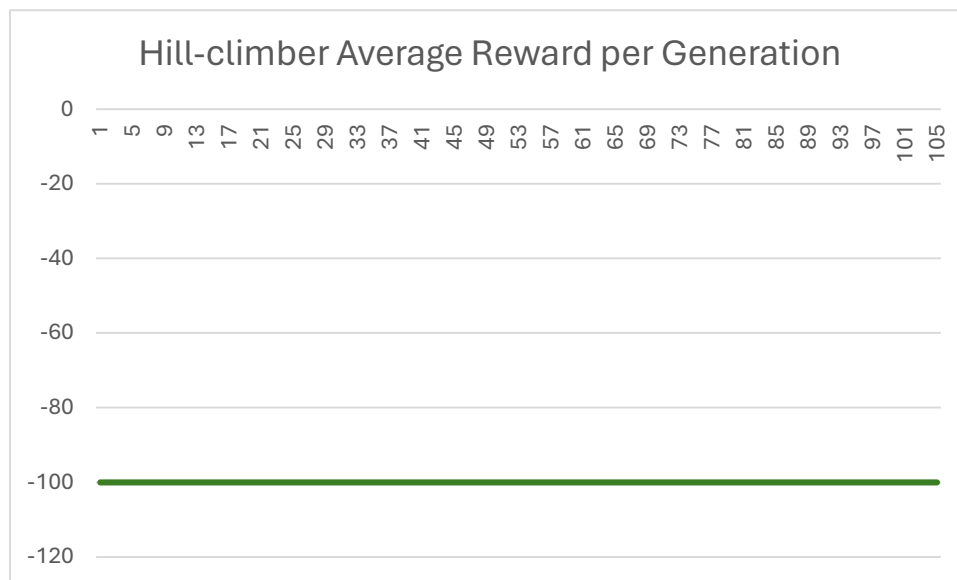
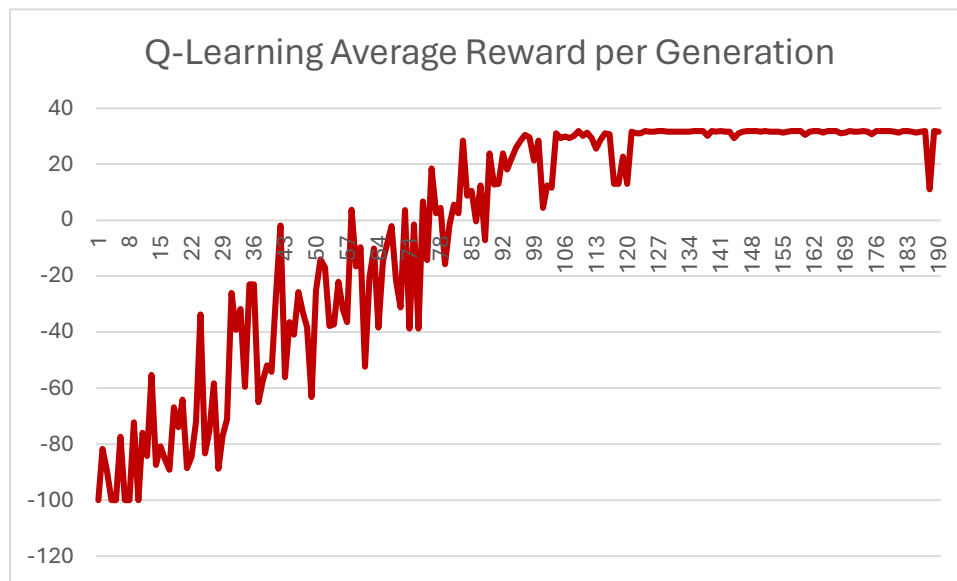
Unit tests were successful (appendix 15.7), with all tests passing. This shows that the core functionality of the AI Search Competition is working as intended.

10. Evaluation

In this section, a small evaluation will be carried out on what AI algorithm is the best at the task, using data gathered from the agents searching. The purpose of this is to have statistical evidence as to which is best, so end users are not being misled.

To gather this data, the AI Search Competition was ran seven times, for ten minutes each. The reward of each generation was collected, and then an average reward of each generation was calculated per algorithm.





The graphs show that SARSA performs slightly better than Q-Learning, and that the Hill-climber is far behind as it was unable to find the reward state at any point during data collection. Besides a few dips in average reward around generation 110, most likely due to bad feedback gained from random exploration. SARSA converges to an optimal solution at generation 100. This is slightly quicker than Q-Learning, which converges to an optimal solution at generation 105. Another reason why the data suggests that SARSA is the best AI algorithm is because on average it runs for more generations than the hill-climber and Q-Learning. In the AI Search Competition, a new generation will start when either 100 moves have been made, or if the agent finds the reward. This means that the SARSA agent was finding the reward quicker than the other agents, therefore it was able to have more generations. A reason why SARSA is a better algorithm for this task is likely because of it being an on-policy algorithm, so it is learning from the actions it actually ends up taking. It is also worth noting however

that due to the environment layout being different every time the program is run, other environments may favour Q-Learning over SARSA due to how they are generated. Therefore, even though the data shows SARSA is the best algorithm, collecting more data may potentially show otherwise depending on environmental factors.

Hill-climber's poor performance is most likely down to two factors. The first is that the instructions given to the hill-climber need to be validated first, so that they do not make the agent walk inside a wall. This makes it quite difficult for the agent to fully explore the search area, as it normally gets stuck with a set of instructions that keep it nearby the start state. The other reason is getting stuck in a local optima. In the rare event that the hill-climber agent is able to get to the reward state, it would be difficult to mutate this set of instructions to improve on it. This in turn means that at best, the hill-climber will have a set of instructions that lead to the reward state, but not an optimal path.

To improve the hill-climber, mutation rates could be increased. For this evaluation, 2% of the instructions were changed each generation. This could potentially be increased to find good behaviour faster, but it is likely not enough to make the hill-climber competitive with SARSA and Q-Learning.

11. End Project Report

11.1. Summary

Overall, the project was largely successful. The main goal was to create AI algorithms that could search for a target in an environment, and this was fulfilled. Whilst there are future additions that could be made, when looking at the requirements, all of the 'Must' and the majority of the 'Should' functional and non-functional requirements have been met. This shows that the project is in a state appropriate for the end user.

11.2. Requirements Review

In this subsection, each requirement will be listed and evaluated on whether it has been met or not.

11.2.1. Have a minimum of three AI algorithms.

This requirement was achieved, the minimum number of algorithms required for an effective final product was met with the implementation of Q-Learning, SARSA and hill-climber. Each algorithm had its own separate sprint dedicated to its development. Allocating time to be able to fully focus on each algorithm definitely helped to meet this objective in an appropriate amount of time. Identifying this as a critical requirement aided in recognising that each algorithm needed its own separate sprint, which led to all three being implemented properly.

11.2.2. Be visually appealing so that users are engaged with project.

Whilst being visually appealing is quite subjective, after gaining feedback and implementing suggested improvements from user testing it can be said that this

requirement was met. However, whilst it is hard to objectively say that a project is visually appealing, to gain further certainty on the completion of this requirement it would be beneficial to have gathered more feedback by getting more participants for user testing.

11.2.3. Be user friendly so that it can be operated and utilised by different types of users.

This is another requirement where whether it has been met is subjective, although user testing also indicates that this requirement has been met. However, there is not as much user interactability as originally planned, so the bar for this is lower than initially thought. There is less user interaction than planned due to some 'Could' requirements not being met. Whilst this requirement has been met overall, future implementation will need to revisit this if deciding to add more user interaction.

11.2.4. Every agent is in an identical environment, so that the search area complexity and difficulty is the same for each algorithm.

This requirement has been achieved by only procedurally generating one environment, and then duplicating it for all agents. Whilst this is not a requirement that needed a lot of time dedicated to fulfilling, it was crucial to meet it so that the integrity of the comparisons made between algorithms is upheld and to mitigate the risk of misleading and miseducating students.

11.2.5. Algorithms must be able to learn in an appropriate amount of time.

This has been mostly met, due to Q-Learning and SARSA learning in an appropriate amount of time, however the hill-climber is not achieving this. In all environments generated when testing, Q-Learning and SARSA will find the search target within 10 minutes. This is a suitable amount of time. Whereas the hill-climber can only find the search target in a suitable amount of time when the environment has generated the start point and search target relatively close together with no obstacles in the way. Therefore, it is up for debate however whether a hill-climber could realistically do this reliably in a similar time to the other algorithms. As a result, this requirement may have been impossible to fully meet in the first place.

11.2.6. Ensure that algorithms are properly implemented.

After testing, algorithms were shown to be properly implemented. Therefore, this requirement was met. This is another requirement that does not necessarily require anything major work to be done, and at most required minor fixes, but it is very important to make sure that this is met. As similarly to the identical environment requirement, without this it would damage the accuracy of comparisons made between the algorithms.

11.2.7. Have an appropriate level of environment complexity.

This requirement goes hand in hand with the requirement for algorithms learning in an appropriate amount of time. Algorithms finding the search target reliably in an appropriate amount of time would suggest a suitable level of environment complexity.

This requirement has been met, as it is a good level of complexity for both Q-Learning and SARSA. However, it is likely too complex for the hill-climber. The reason this requirement is said to be fully met is that there is not a level of environment complexity that would be perfect for all three algorithms. The environments could be made less complex for the hill-climber, but it would then become too easy for SARSA and Q-Learning. Therefore, the level of complexity created is the best-case scenario.

12. Post-Mortem

12.1. Project Successes

A success in this project was implementing the AI algorithms in the time allocated for them. They were identified as the most challenging parts of the project, and also the most important to have. Without any one of them included, the quality of the project would have greatly suffered. Due to this, a lot of time was spent researching and planning the design of each algorithm so implementation could be as unproblematic as possible. This is a success because it shows the effectiveness of good project planning, and the fact that these were implemented fully in the sprints planned meant that more focus could be on other features during the rest of the project.

Another successful aspect of this project is choosing Unity to create a 3D visualisation of searching. The idea for using Unity was that it would create a more engaging and interesting project. Whilst using it creates new considerations of how the project is designed, it is undoubtedly a better visual representation of the AI algorithms than visualising using a command line would be. This was a very good decision, as the project could more easily portray the end product in mind. It also shows the importance of choosing the correct technologies, and weighing up what different advantages each technology can offer.

The risk assessment was another success of this project. All risks that occurred during the course of the project were planned for, and the action plan developed was successful in mitigating the damage to the project. Having a risk score assigned to each risk was also helpful, as it helped focus on which risks to put the most focus into trying to lower the chances of them occurring. This shows the importance of planning prior to implementation. If there was no planning and only development took place, then running into these issues would have a huge negative impact on the project as there would be no plan in place. It would likely lead to the project progress being slower, and the final product being of lower quality.

Agile development has been a good choice for this project. Employing this software development methodology allowed for changes to be made to designs if the need arose. It also when unforeseen challenges in sprints occurred, instead of derailing the whole project these issues could be put into a new or future sprint. Allowing adaptability in the project led to being able to deal with issues in the project more easily than if a more rigid methodology were to be used such as a waterfall model. Moreover,

as this is also an educational opportunity for the author, agile has been useful to be able to reflect on the project after each sprint and to learn what can be improved for future sprints and future projects beyond university.

12.2. Project Challenges

Getting a good number of participants for user testing was a challenge, and even though the feedback was mainly positive it was not enough responses to be able to confidently say that visual appeal requirements have definitely been met. This in turn is more likely to lead to improvements suggested being implemented that are not something that is seen as a visual improvement by end users. Furthermore, the questionnaire could have also been longer and have a larger focus on functionality as well as visualisation, but this may have run the risk of having less participation due to there being more questions and taking more time to fill out the whole questionnaire.

Making sure that the procedurally generating environment generated looking like one cohesive area, and not four separate chunks whilst allowing the agents to access the whole search area was a challenge. It was difficult to strike a balance between the two, the inability to find a suitable solution has reduced the quality of the search area which in turn affects the overall project.

12.3. Improvements

To improve this project, a different algorithm would be chosen instead of a hill-climber. Whilst the hill-climber is good to show how some algorithms are unsuitable for this task, and it further proves how Q-Learning and SARSA are good choices for a task such as this one, it would be a good idea to have use development time to implement a more competent algorithm to have a more interesting comparison to Q-Learning and SARSA.

In addition to this, user testing would take place earlier and would have been sent to more people. This way, as mentioned earlier results would be more accurate with more participants, and if taken place earlier there would be more time to implement changes if they required a major change. Moreover, this would be more aligned with the adaptability of agile project management. To achieve this outcome, a more final visual design of the project would have to be created earlier on in the development lifecycle to be able to show to user testing participants.

Another type of testing that could have been improved was unit testing. In hindsight, unit testing should have started earlier and been done alongside the implementation of the code being tested. This is because it was all done at the end of development, so there was not as much time available for unit testing to occur and therefore less unit tests than desired took place. This is a failure of planning, and in this area Agile was not taken advantage of fully, but in future projects unit this will be taken into account and planned for more effectively.

12.4. Future Implementation

Future implementation should focus on adding another AI algorithm. This would lead to more comparisons to be made, and more algorithms to educate students about. This would make for a more interesting 'competition' between them as well. This could also be expanded to having algorithms have multiple agents. However, as project is scaled up, there should be greater focus on optimisations such as parallelisation to make sure that the project is still able to run quickly and efficiently.

Another feature to add would be multiple search targets in the environment. This would allow for more advanced searching to occur, and a larger range of solutions to the searching. For example, which search target to find first or what path is most efficient. A consideration that would have to be made for this is the amount of time taken to find multiple search targets, as this would make the agents' task much more complicated.

Adding more opportunities for users to interact with the AI Search Competition would be another focus of future implementation. Examples of what could be added for this include user set levels of environmental complexity, where higher levels mean more obstacles getting generated, an option to regenerate the environment, being able to input a seed for a specific environment, or for users to change agent movement speed, or edit Q-table values to see how that changes Q-Learning or SARSA agent behaviour.

The AI Search Competition could also be transferred a physical environment for future implementation. This could be by training the agents in the current virtual environment, and then recreating the search area physically and using robots to act as the agents. This would require a lot of new considerations to take into account, such as how the environment can be recreated physically. However, doing this would add another level of interactivity and engagement for students, and undoubtedly increase the quality of the learning experience.

13. Conclusion

In conclusion, the AI Search Competition has been a success. The requirements for this project have mostly been met, and there is a solid foundation of plans for future development to build from. The AI algorithms function as intended, and largely perform well for the task created. Whilst there are a few problems with the procedurally generating search environment, it still functions well and is able to create an interesting and unique environment. The overall aim of this project was to provide an engaging educational experience for students, and this aim has been met.

The AI Search Competition has also provided an opportunity to build software development skills, familiarity with both C# and Unity, to build knowledge of the field of AI as a whole, and to gain further experience with project management. Reflection on the project has a whole has also given a sense of what needs to be improved and worked on in future projects. Therefore, it has been a very good educational experience for the author.

14. References

A B M Moniruzzaman and Syed Akhter Hossain (2013). Comparative Study on Agile software development methodologies. *arXiv (Cornell University)*.

Ahmad Dalalah (2014). Extreme Programming: Strengths and Weaknesses. *Computer Technology and Application*, 5(1). doi:<https://doi.org/10.17265/1934-7332/2014.01.003>.

Akalin, N. and Loutfi, A. (2021). Reinforcement Learning Approaches in Social Robotics. *Sensors*, 21(4), p.1292. doi:<https://doi.org/10.3390/s21041292>.

Barlow, J., Giboney, J., Keith, M.J., Wilson, D., Schuetzler, R., Lowry, P.B. and Vance, A. (2011). Overview and Guidance on Agile Development in Large Organizations. *SSRN Electronic Journal*. doi:<https://doi.org/10.2139/ssrn.1909431>.

Chen, J., Chen, C., Xing, Z., Xia, X., Zhu, L., Grundy, J. and Wang, J. (2020). Wireframe-based UI Design Search through Image Autoencoder. *ACM Transactions on Software Engineering and Methodology*, 29(3), pp.1–31. doi:<https://doi.org/10.1145/3391613>.

Cherney, R., Major, R. and Fitzpatrick, T. (2023). Qualify AI Drug Discovery Tools through FDA ISTAND Program to Model Responsible Drug Discovery AI and Mitigate Dual Use Concerns. *The journal of science policy & governance*, 22(03). doi:<https://doi.org/10.38126/jspg220302>.

Clark Kendrick Go, Lao, B., Yoshimoto, J. and Ikeda, K. (2016). A reinforcement learning approach to the shepherding task using SARSA. doi:<https://doi.org/10.1109/ijcnn.2016.7727694>.

Coronato, A., Naeem, M., Pietro, G.D. and Paragliola, G. (2020). Reinforcement Learning for Intelligent Healthcare Applications: A Survey. *Artificial Intelligence in Medicine*, p.101964. doi:<https://doi.org/10.1016/j.artmed.2020.101964>.

Cosentino, V., Canovas Izquierdo, J.L. and Cabot, J. (2017). A Systematic Mapping Study of Software Development With GitHub. *IEEE Access*, 5, pp.7173–7192. doi:<https://doi.org/10.1109/access.2017.2682323>.

- Daka, E. and Fraser, G. (2014). *A Survey on Unit Testing Practices and Problems*. [online] IEEE Xplore. doi:<https://doi.org/10.1109/ISSRE.2014.11>.
- Eshuis, R. and Wieringa, R. (2004). Tool support for verifying UML activity diagrams. *IEEE Transactions on Software Engineering*, 30(7), pp.437–447. doi:<https://doi.org/10.1109/tse.2004.33>.
- Evans, A.S. (1998). *Reasoning with UML class diagrams*. [online] IEEE Xplore. doi:<https://doi.org/10.1109/WIFT.1998.766304>.
- Faranello, S. (2012). *Balsamiq Wireframes Quickstart Guide*. Packt Pub Limited.
- GitHub, Inc. (2019). *Build software better, together*. [online] GitHub. Available at: <https://github.com/about>.
- GOV.UK (2018). *Data Protection Act*. [online] Gov.uk. Available at: <https://www.gov.uk/data-protection>.
- Jang, B., Kim, M., Harerimana, G. and Kim, J.W. (2019). Q-Learning Algorithms: A Comprehensive Classification and Applications. *IEEE Access*, 7, pp.133653–133667. doi:<https://doi.org/10.1109/access.2019.2941229>.
- Kaelbling, L.P., Littman, M.L. and Moore, A.W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4, pp.237–285. doi:<https://doi.org/10.1613/jair.301>.
- Kurtanovic, Z. and Maalej, W. (2017). Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. *2017 IEEE 25th International Requirements Engineering Conference (RE)*. doi:<https://doi.org/10.1109/re.2017.82>.
- Li, Y. (2017). Deep Reinforcement Learning: An Overview. *arXiv (Cornell University)*. doi:<https://doi.org/10.48550/arxiv.1701.07274>.
- Lindgaard, G., Dudek, C., Sen, D., Sumegi, L. and Noonan, P. (2011). An exploration of relations between visual appeal, trustworthiness and perceived usability of homepages. *ACM Transactions on Computer-Human Interaction*, 18(1), pp.1–30. doi:<https://doi.org/10.1145/1959022.1959023>.

- Luckin, R. and Cukurova, M. (2019). Designing educational technologies in the age of AI: A learning sciences-driven approach. *British Journal of Educational Technology*, 50(6), pp.2824–2838. doi:<https://doi.org/10.1111/bjet.12861>.
- MacFarlane, A., Secker, A., May, P. and Timmis, J. (2010). An experimental comparison of a genetic algorithm and a hill-climber for term selection. *Journal of Documentation*, 66(4), pp.513–531. doi:<https://doi.org/10.1108/00220411011052939>.
- Mohanty, P.K., Arun Kumar Sah, Kumar, V. and Kundu, S. (2017). Application of Deep Q-Learning for Wheel Mobile Robot Navigation. doi:<https://doi.org/10.1109/cine.2017.11>.
- Nitchakun Kantasewi, Sanparith Marukatat, Somying Thainimit and Okumura Manabu (2019). Multi Q-Table Q-Learning. doi:<https://doi.org/10.1109/ictmsys.2019.8695963>.
- Reggio, G., Leotta, M., Ricca, F. and Clerissi, D. (2013). What are the used UML diagrams? A Preliminary Survey. pp.3–12.
- Rjoub, A. (2020). Courses timetabling based on hill climbing algorithm. *International Journal of Electrical and Computer Engineering (IJECE)*, 10(6), p.6558. doi:<https://doi.org/10.11591/ijece.v10i6.pp6558-6573>.
- Runeson, P. (2006). A survey of unit testing practices. *IEEE Software*, 23(4), pp.22–29. doi:<https://doi.org/10.1109/ms.2006.91>.
- Sebag, M. and Schoenauer, M. (2002). A society of hill-climbers. *HAL (Le Centre pour la Communication Scientifique Directe)*. doi:<https://doi.org/10.1109/icec.1997.592329>.
- Seyfi, A. and Patel, A. (2010). Briefly introduced and comparatively analysed: Agile SD, component-based SE, aspect-oriented SD and Mashups. doi:<https://doi.org/10.1109/itsim.2010.5561582>.
- Temporel, A. and Kovacs, T. (2003). *A heuristic hill climbing algorithm for Mastermind*.

Xi, B., Liu, Z., Raghavachari, M., Xia, C.H. and Zhang, L. (2004). A smart hill-climbing algorithm for application server configuration. *Proceedings of the 13th conference on World Wide Web - WWW '04*.
doi:<https://doi.org/10.1145/988672.988711>.

15. Appendices

15.1. User Guide

15.1.1. Minimum System Requirements

OS - Windows 7 (SP1+), Windows 10 and Windows 11

CPU – AMD Ryzen 3 4100 equivalent or better

GPU – Nvidia GeForce GTX 1650 equivalent or better

RAM – 4GB or better

15.1.2. Installation

- Open the GitHub link provided in this report.
- Download a .zip file of the repository.

15.2. Risk Table


Risk	Likelihood of occurring (1 = unlikely, 5 = very likely)	Severity (1 = not severe, 5 = very severe)	Risk rating (Likelihood * Severity)	Risk reduction techniques
Development of Reinforcement Learning algorithms is too complex, leading to more time needed than estimated	3	2	6	Focus on developing algorithms at the start of implementation phase to have as much time as possible. If needed, reduce number of algorithms from four to three
Project is deemed not complex enough	3	4	12	Add more RL algorithms, allow more complex environments to generate, compare algorithms in a physical environment e.g., controlling robots
Time for algorithms to learn how to effectively navigate/search is too long to get meaningful data	4	4	16	Simplify search environment by reducing size of area, reduce/remove obstacles, reduce number of search targets. Also reduce time for agents to search for targets per iteration

All algorithms struggle to adapt to different search environments	4	3	12	Firstly, evaluate code to see if there are any issues causing the problem. If not, reduce search environment complexity, make them more similar to each other
Lose track of what needs to be developed/worked on at what time	2	3	6	Work on project consistently, follow product backlog, and keep it up to date. Always be aware of current and future sprints
General progress of project is slower than anticipated	3	5	15	Make sure an appropriate amount of time is invested into project. Be aware of challenges faced to be able to communicate this in presentations. Break down tasks into smaller more achievable sprints e.g., focus on key features for interim presentation, then build from that for final presentation
Illness leads to being unable to progress with project for some time	5	2	10	Maintain a healthy work-life balance to reduce chances of becoming ill. As catching colds is quite likely to occur at some point, focus on recovering quickly when it does happen.
Technical issues leading to loss of work	1	5	5	Ensure progress is regularly committed to GitHub, keep all documents backed up to OneDrive

15.3. Product Backlog

The screenshot displays a Jira Product Backlog for the project COMP3000. The interface includes a top navigation bar with tabs for Grid, Board, Charts, and Schedule. The main area is divided into five columns representing different sprints, each with an 'Add task' button and a 'Completed tasks' count.

- Sprint 0 - Project Initiation** (4 completed tasks):
 - Select-Supervisor (Completed by (s) Samuel Alcock ...)
 - Project-Vision (Completed by (s) Samuel Alcock ...)
 - Set-up-GitHub-repo (Completed by (s) Samuel Alcock ...)
 - Gantt-Chart (Completed by (s) Samuel Alcock ...)
- Sprint 1 - Requirements Analysis** (5 completed tasks):
 - Activity-Diagram (Completed by (s) Samuel Alcock ...)
 - Class-Diagram (Completed by (s) Samuel Alcock ...)
 - Use-Case-Diagram (Completed by (s) Samuel Alcock ...)
 - MOSCOW-Planning (Completed by (s) Samuel Alcock ...)
- Sprint 2 - Q-Learning** (2 completed tasks):
 - Create-search-environment (Completed by (s) Samuel Alcock ...)
 - Q-Learning-Algorithm (Completed by (s) Samuel Alcock ...)
- Sprint 3 - SARSA** (1 completed task):
 - SARSA-Algorithm (Completed by (s) Samuel Alcock ...)
- Sprint 4 - Procedural ge** (0 completed tasks):
 - Create-different-el (Completed by (s) Samuel Alcock ...)
 - Procedural-Gener (Completed by (s) Samuel Alcock ...)



COMP3000

COMP3000

Grid
Board
Charts
Schedule
...

SA (s) Samuel Alcock

Members

Filter (0)

Sprint 5 - Hill-climber

+ Add task

Completed tasks 1

Hill-climbing Algorithm

Completed by (s) Samuel Alcock ...

Sprint 6 - Procedural Generation improv

+ Add task

Completed tasks 2

Increase environment complexity

Completed by (s) Samuel Alcock ...

Create more chunks

Completed by (s) Samuel Alcock ...

Sprint 7 - Outputting performance

+ Add task

Completed tasks 1

Output data to .csv

Completed by (s) Samuel Alcock ...

Sprint 8 - Further visualisation

+ Add task

Completed tasks 2

Implement improvements suggested

Completed by (s) Samuel Alcock ...

User testing

Completed by (s) Samuel Alcock ...

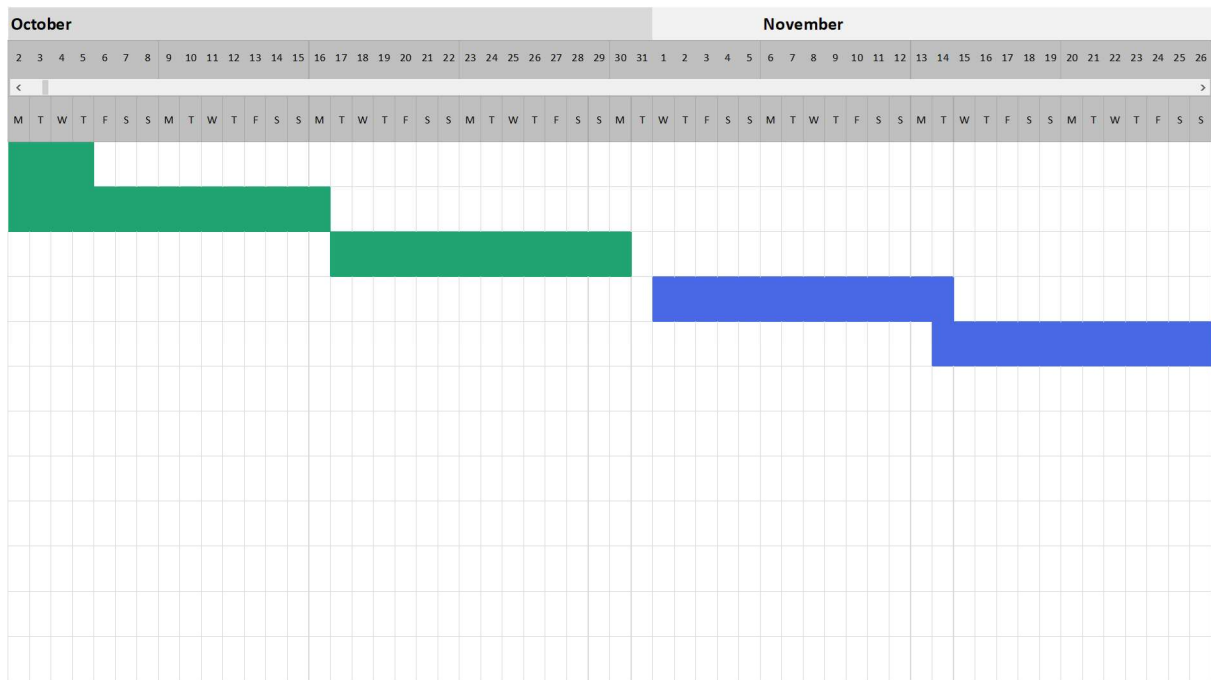
37

15.4. Gantt Chart

Project Start Date: 27/09/2023

Scrolling Increment: 5

Milestone description	Category	Assigned to	Progress	Start	Days
Supervisor Selection	On Track		100%	27/09/2023	9
Sprint 0 - Project Initiation	On Track		100%	29/09/2023	18
Sprint 1 - Requirements Analysis	On Track		100%	17/10/2023	14
Sprint 2 - Q-Learning	Med Risk		100%	01/11/2023	14
Spint 3 - SARSA	Med Risk		100%	14/11/2023	14
Sprint 4 - Procedural Generation	Low Risk		100%	28/11/2023	14
Sprint 5 - Hillclimber	Med Risk		100%	05/01/2024	14
Sprint 6 - Procedural Generation improvements	On Track		100%	19/01/2024	14
Poster and Description	On Track		100%	01/03/2024	20
Project Portfolio	On Track		100%	01/04/2024	22
Viva	On Track		100%	01/04/2024	28
Sprint 7 - Outputting Performance	On Track		100%	02/02/2024	14
Sprint 8 - Further Visualisation	On Track		100%	16/02/2024	14



15.5. Visual changes to agents

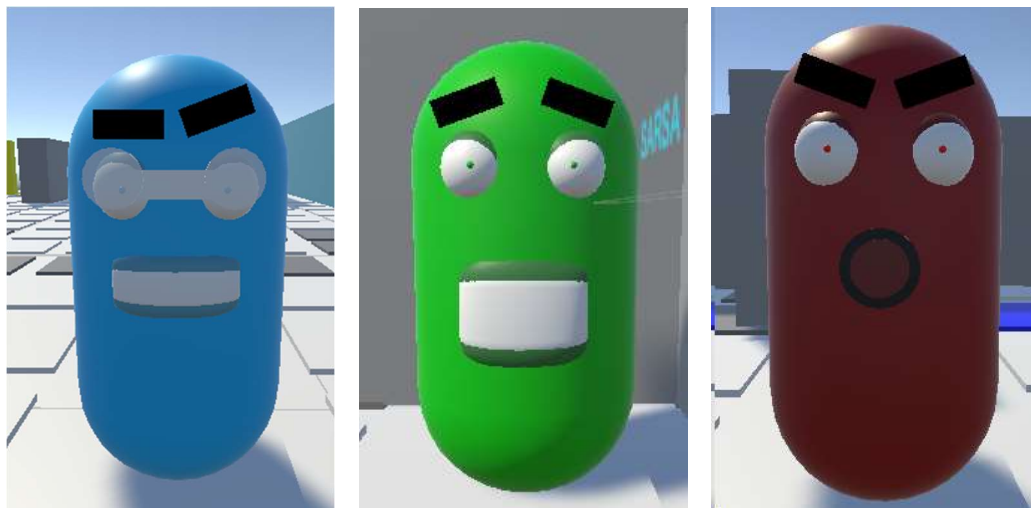
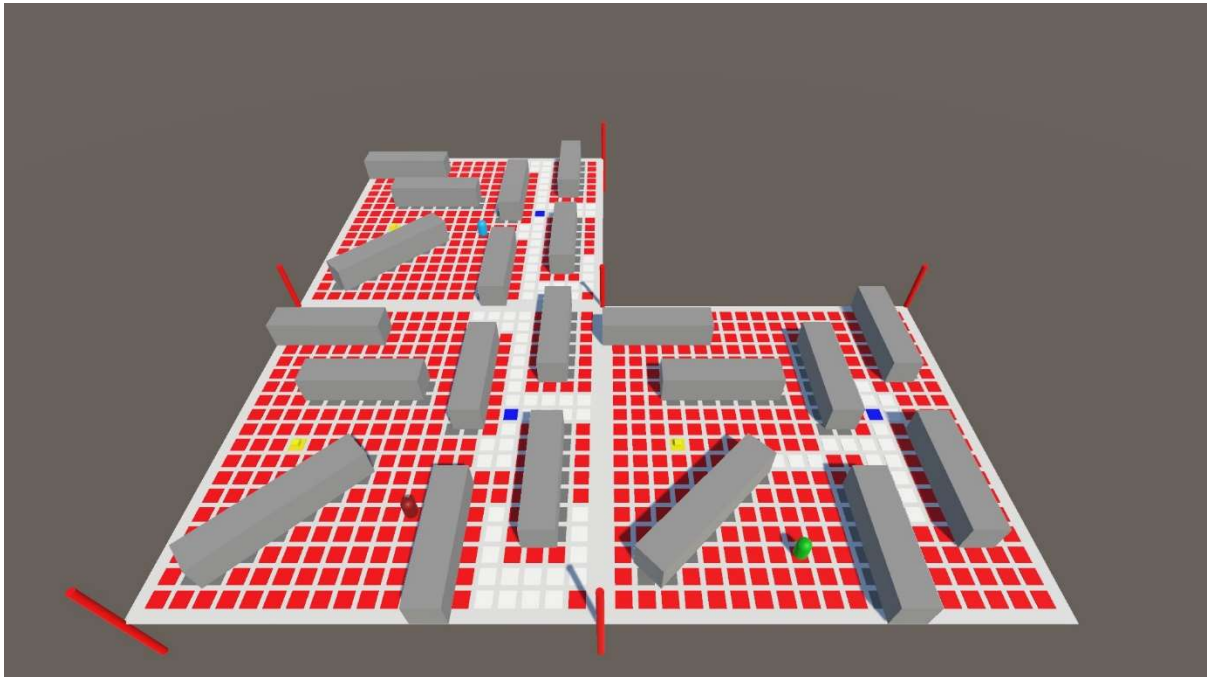


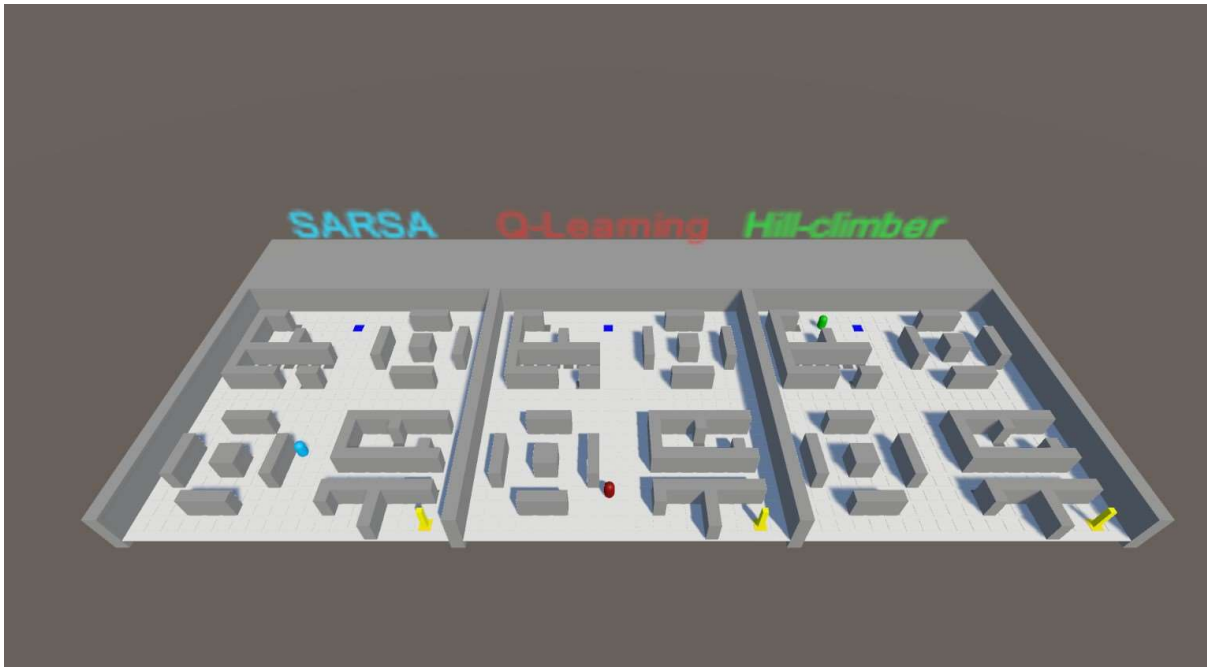
Figure 4 Left to right, SARSA, Hill-climber, Q-Learning

15.6 Visual changes to environment

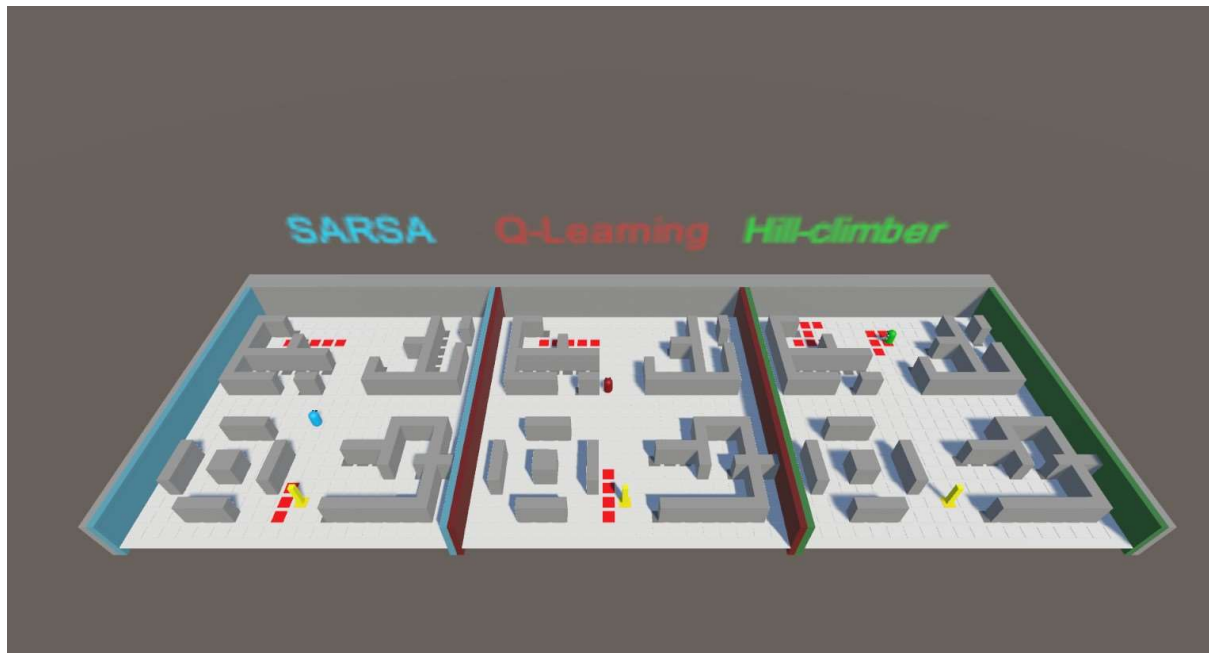
15.6.1. *Original – no visualisation*



15.6.2. *Visualisation before user testing*



15.6.3. Visualisation after user testing



15.7. Unit Tests

Function name	Unit test	Result	Expected result	Actual result
MakeMove	Q-Learning update function working	Pass	1.524	1.524
InitialiseQTable	Check for working Q-table	Pass	Output of 2D array of grid size*4	Output of 2D array of grid size*4
GetRewardMatrix	Check for working reward matrix	Pass	Output of 2D array of all possible rewards	Output of 2D array of all possible rewards
MakeMove	SARSA update function working	Pass	1.524	1.524
SessionReplace	Hill-climber mutation working	Pass	List of instructions with a changed instruction	List of instructions with a changed instruction
PopulateArea	Chunk selection working correctly	Pass	Correct chunk chosen	Correct chunk chosen
InBadPosition	Correct distance calculation	Pass	5	5

15.8. Initial wireframes

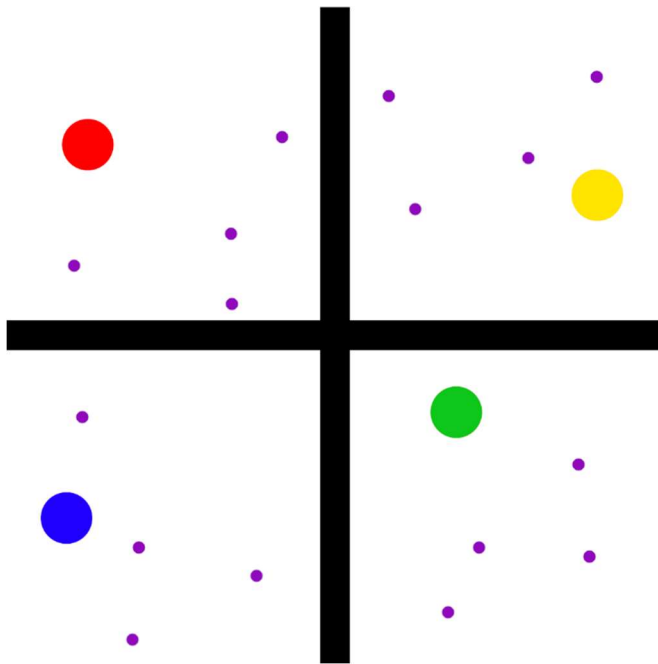


Figure 6 Algorithms in simple, separate environments

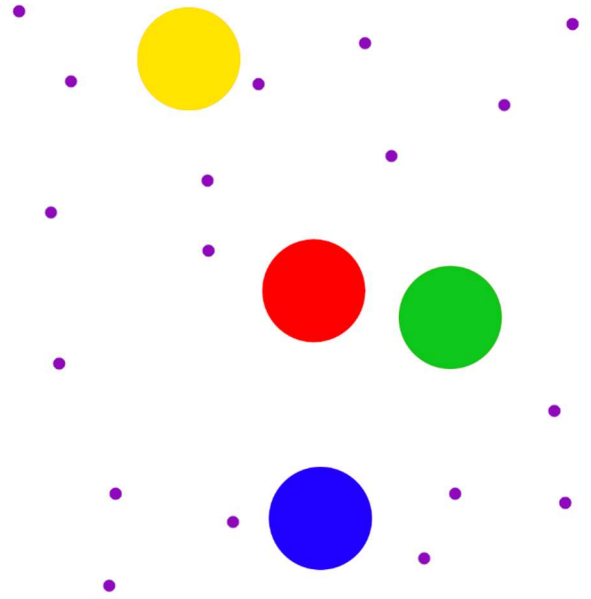


Figure 5 Algorithms in same, simple environment

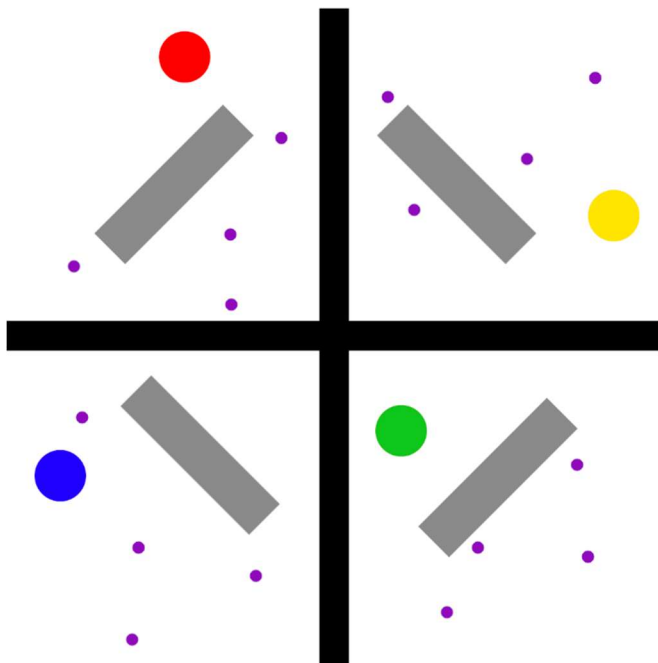


Figure 8 Algorithms in complex, separate environments

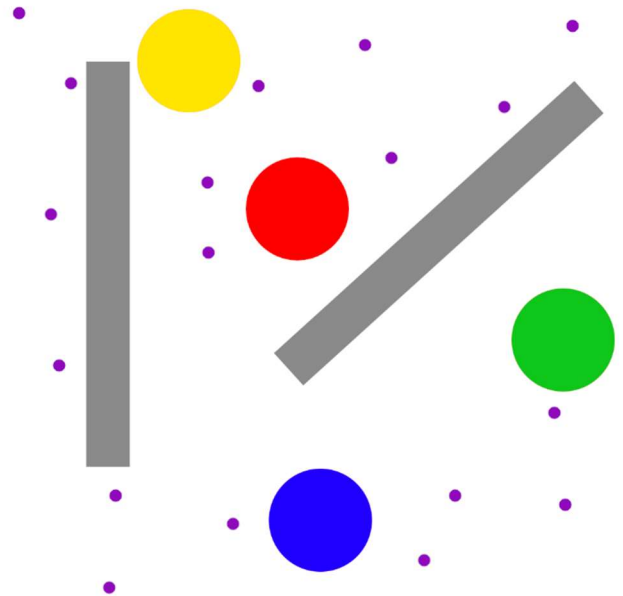


Figure 7 Algorithms in same, complex environment