Insight Into Game AI

Submitted By

Mahedi Hassan Rafin

193-35-518

Department of Software Engineering

Supervised By

Ms Aditi Dhali

AD

Department of Software Engineering

A Project report has been submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Software Engineering

Fall 2023

APPROVAL

This Project titled on **"Insight Into Game AI"**, submitted by **Mahedi Hassan Rafin (193-35-518)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Declaration

I hereby declare that I have completed this project under the guidance of **Ms Aditi Dhali**, Lecturer in the Department of Software Engineering at Daffodil International University. Furthermore, I affirm that neither this project nor any part of it has been presented for any degree awards elsewhere.

Supervised By

Ms. Aditi Dhali

Aditi Dhali

Lecturer
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Submitted By

Mahedi Hassan Rafin

Rafin

ID: 193-35-518
Batch:30
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Advisor & Supervisor Clearance

Supervised By

Ms. Aditi Dhali

Aditi Dhali

Lecturer
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Submitted By

Mahedi Hassan Rafin

[illegible]

ID: 193-35-518
Batch:30
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Advised By

Ms. Nusrat Tasnim

নুসরাত তাসনিম

Lecturer
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Table of Contents

Chapter 1 Introduction

1.1 Project Overview

This project aims to create a platform where students and researchers can explore various Artificial Intelligence used in games within simple environments. The idea for this project came about due to the lack of resources available to experience simple game AIs at the beginner level. Although there is information about building simple AI scattered all over the internet, there is no single place to check out all these AIs in their simplest form. This project fills the gap and provides a new place in the game development industry to experience and learn about AIs in their simplest forms. It will showcase a variety of AI that can be experienced and understood in terms of their functionality.

Some of the most commonly used AI techniques in gaming include the A* Algorithm, Autonomous Agents, and Crowd Simulation. For future scalability, the project will not allow users to change the parameters of the AI.

The goal is to showcase at least three AI techniques in a single place, making it easy for users to access and experience them. By doing so, it will help users gain a better understanding of how these AIs work and their potential applications in the gaming industry.

1.2 Project Purpose

This project aims to create a central hub where beginners and novice game developers can explore and learn about basic AI techniques used in games. Currently, no platform provides a comprehensive resource for learning about different game AI techniques. Therefore, this project will fill that gap by providing a single location for game developers to learn about game AI.

The primary target audience for this project is novice game developers and individuals interested in observing how a fundamental game AI looks in its simplest form. This project will impact the game development community, particularly for those who are willing to learn about how a game AI functions from a single location.

Moreover, this project will be open source, allowing game developers to contribute their own AI technologies. This will make this project a valuable resource for the entire game development community.

1.2.1 AI in Games

One of the common questions regarding AI is the difference between artificial intelligence (AI) and machine learning. While these terms are often used interchangeably, they are not the same.

Artificial Intelligence is broadly defined as the intelligence demonstrated by machines. In contrast, natural intelligence is the kind exhibited by living creatures. AI as an academic discipline emerged in 1956, focusing on replicating human abilities such as reasoning, knowledge representation, planning, learning, natural language processing, perception, and the ability to manipulate objects. Essentially, AI aims to enable machines to adapt to new situations. However, achieving this holistically is challenging, given that even human psychological processes are not fully understood. As a result, AI is divided into smaller, more focused fields, each tackling a specific aspect of intelligence.

Machine Learning, a subset of AI, is just one of these specialized fields. AI's applications span robotics, planning, vision, speech, expert systems, natural language processing, and machine learning. In the context of video games, AI has been used more for simulating intelligence to create believable characters rather than achieving true intelligence as pursued in academic research. Game AI and academic AI are thus quite distinct.

In gaming, AI has historically drawn most from the fields of planning and expert systems. However, machine learning has recently gained prominence in game development. It holds the potential to create characters that can adapt intelligently to changes in their environment. Understanding machine learning requires a strong foundation in mathematics and statistics, especially to appreciate its applications in games. Despite its potential, machine learning would be overkill for the functionality required in about 95% of today's games.

The project focuses on implementing the fundamental techniques of AI in games, such as non-player characters' movement, interaction with players, and exhibiting signs of digital life. These time-honoured methods, applied in Unity with C#, form the core of game AI and can significantly enhance game projects.

1.2.2 Proposed System

This project will be developed using the Unity game engine, which provides ample support for implementing basic artificial intelligence in-game environments.

Unity Game Engine serves as a versatile and powerful foundation for developing interactive and immersive gaming experiences. Unity has capabilities that particularly shine in the realm of artificial intelligence (AI) integration within game environments. Unity's comprehensive toolset simplifies the integration of AI techniques into games. It supports the development of various AI components, like pathfinding algorithms for character navigation, and decision-making processes for non-player characters (NPCs). Unity allows developers to create multiple "scenes" - essentially different levels or environments within the game. This modular approach not only streamlines the development process but also enhances the user experience. Additionally, Unity's community and resources are invaluable for both developers and learners. There are numerous tutorials, forums, and documentation available, which can be integrated into the system to provide users with learning materials and support as they explore AI in gaming.

The system will operate using a scene system, where users can switch between scenes to access different artificial intelligence techniques. This approach keeps all artificial intelligence techniques organized, making it easier for users to learn about them, and eliminates distractions by having them in a single location.

The system will be designed to be as user-friendly as possible. All user interfaces will be intuitive and easy to understand. Additionally, the system will provide various resources to help users learn about artificial intelligence.

The goal of this project is to make artificial intelligence more accessible to a wider range of users. By providing a simple and easy-to-use system, this project can help people learn about artificial intelligence in games.

Chapter 2 System Analysis

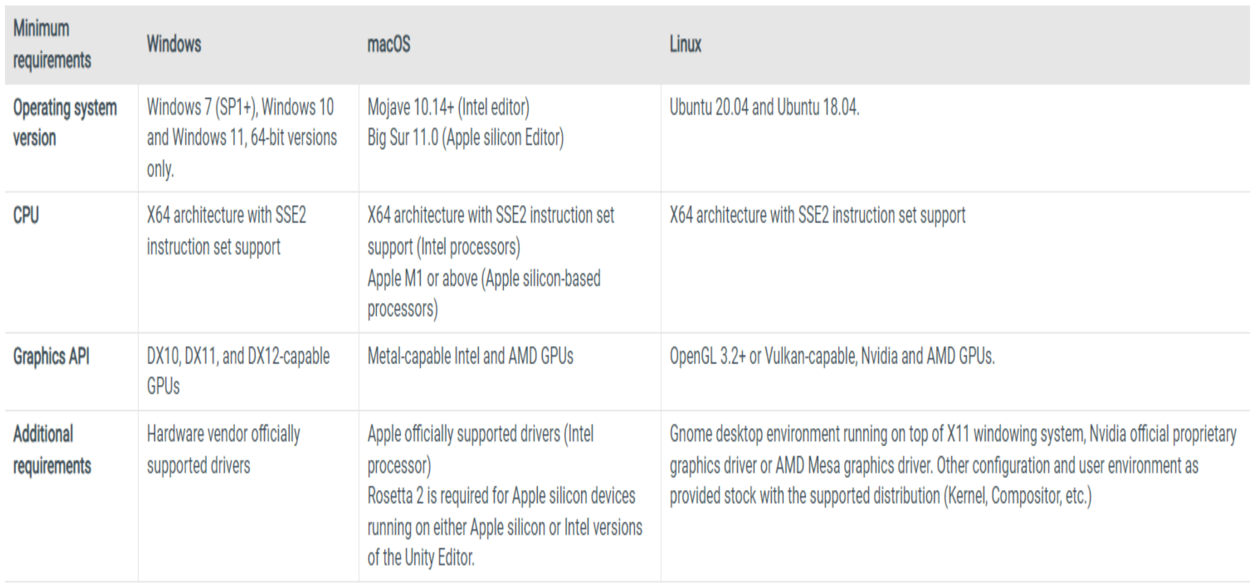
2.1 Feasibility Analysis

Today's standard computers have sufficient power to run the Unity game engine, which will be the primary tool used to develop this project.
No new technology needs to be developed for this project, as all the necessary tools and resources are already available. The Unity game engine, Visual Studio, and other hardware are all free to use. Additionally, in-game assets are available in the Unity asset store, and this project will only use free-to-use assets.
Completing the project requires a beginner level of game development knowledge, and it should take approximately 4-5 months to finish. The system will be developed in a way that enables users to navigate seamlessly between AI scenes. Once the system is released, any maintenance required will consist solely of bug fixes.

2.2 Functional Requirements

- The game should be able to demonstrate the A* algorithm by allowing users to interact with it.
- The game should be able to implement autonomous agents.
- The game should be able to simulate crowd systems.
- The User should be able to change the crowd system's parameters.
- The game should be able to allow users to switch between different AI techniques.
- Users should be able to interact with AI techniques that require it.
- Users should be able to control player movement if needed.

2.3 System Requirements



Minimum requirements	Windows	macOS	Linux
Operating system version	Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only.	Mojave 10.14+ (Intel editor) Big Sur 11.0 (Apple silicon Editor)	Ubuntu 20.04 and Ubuntu 18.04.
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support (Intel processors) Apple M1 or above (Apple silicon-based processors)	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers (Intel processor) Rosetta 2 is required for Apple silicon devices running on either Apple silicon or Intel versions of the Unity Editor.	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)

2.4 Non-Functional Requirements

- The game should be as smooth as possible, with no noticeable lag or stuttering.
- The game should not collect any personal data from the user.
- The game UI should be intuitive and easy to use.
- The game system should be designed to be scalable so that it can be easily expanded in the future.

2.5 Performance

The game should be able to run at a constant 60 frames per second (FPS) at 1080p resolution on the minimum system requirements. This can be tested using a third-party tool such as MSI Afterburner.

It is important to note that the performance of the game may vary depending on the specific hardware configuration and software environment. For example, the game may run slower on a system with a lot of background processes running.

Chapter 3 System Design

3.1 Development Model

In the development of this project, I deviated from traditional, structured software development methodologies, adopting instead an Agile and learning-focused approach. My process was guided by a blend of educational resources, tutorials, and direct experimentation.

Given more than enough AI techniques employed in game development, it was impractical to integrate all of them into the initial version of the system. Consequently, I prioritized testing and experimenting with various AI strategies to discern the most effective ones for my project. This approach facilitated a continuous refinement of the project, shaped by an iterative acquisition of knowledge and skills.

The integration of different AI types within a single system introduced unique challenges. I addressed these by actively seeking out specific tutorials and resources.
This allowed me to quickly iterate on the project and make changes as needed.

3.2 AI Techniques

After testing and experimenting with various AI techniques, I decided to build three AI techniques that are probably used in many games. These techniques are A* Algorithm, Autonomous Agents, and Crowd Simulation.

These three AI techniques are all powerful and versatile. They can be used to create various game experiences, from simple puzzle games to complex action games. By combining these techniques, game developers can create games that are both challenging and engaging.

3.2.1 A* Algorithm

The A* (A-Star) algorithm, invented by computer scientists Nils Nilsson, Peter Hart, and Bertram Raphael in 1968, was developed for a navigation system for the Shakey robot research project at Stanford. Initially, they worked on two algorithms, A1 and A2, eventually finding A2 to be more optimal and renaming it A*. This algorithm is widely used in AI for games, particularly for planning actions and determining the best path through an environment.

The essence of the A* algorithm is its intuitive nature, resembling human thought processes in pathfinding. It involves considering multiple paths and selecting the one that appears most efficient based on certain criteria. The algorithm uses three key values: G (the movement cost from the starting point to a given square), H (the estimated cost to move from that square to the final destination), and F (the sum of G and H). The algorithm aims to find the path with the lowest F value at each step, leading to the goal.

The process involves maintaining two lists: an open list of squares to be examined and a closed list of squares already examined. The A* algorithm avoids looking too far ahead, focusing only on the distances travelled and the estimated distance remaining. This efficiency makes it a popular choice in game development for pathfinding and navigation tasks.

3.2.2 Autonomous Agents

Autonomously moving agents in games are AI-driven characters or entities that can act and make decisions without direct input from the player or a game designer. These agents are programmed to have a degree of independence, allowing them to interact with the game environment and other entities in a way that mimics real-world behaviour or logic.

Autonomously moving agents make decisions based on the game state, and their objectives, and sometimes learn from past experiences. They might choose to attack, flee, follow, explore, or perform any number of actions based on their programming and the situation.

Autonomously moving agents interact with both the game environment and the players.

3.2.3 Crowd Simulation

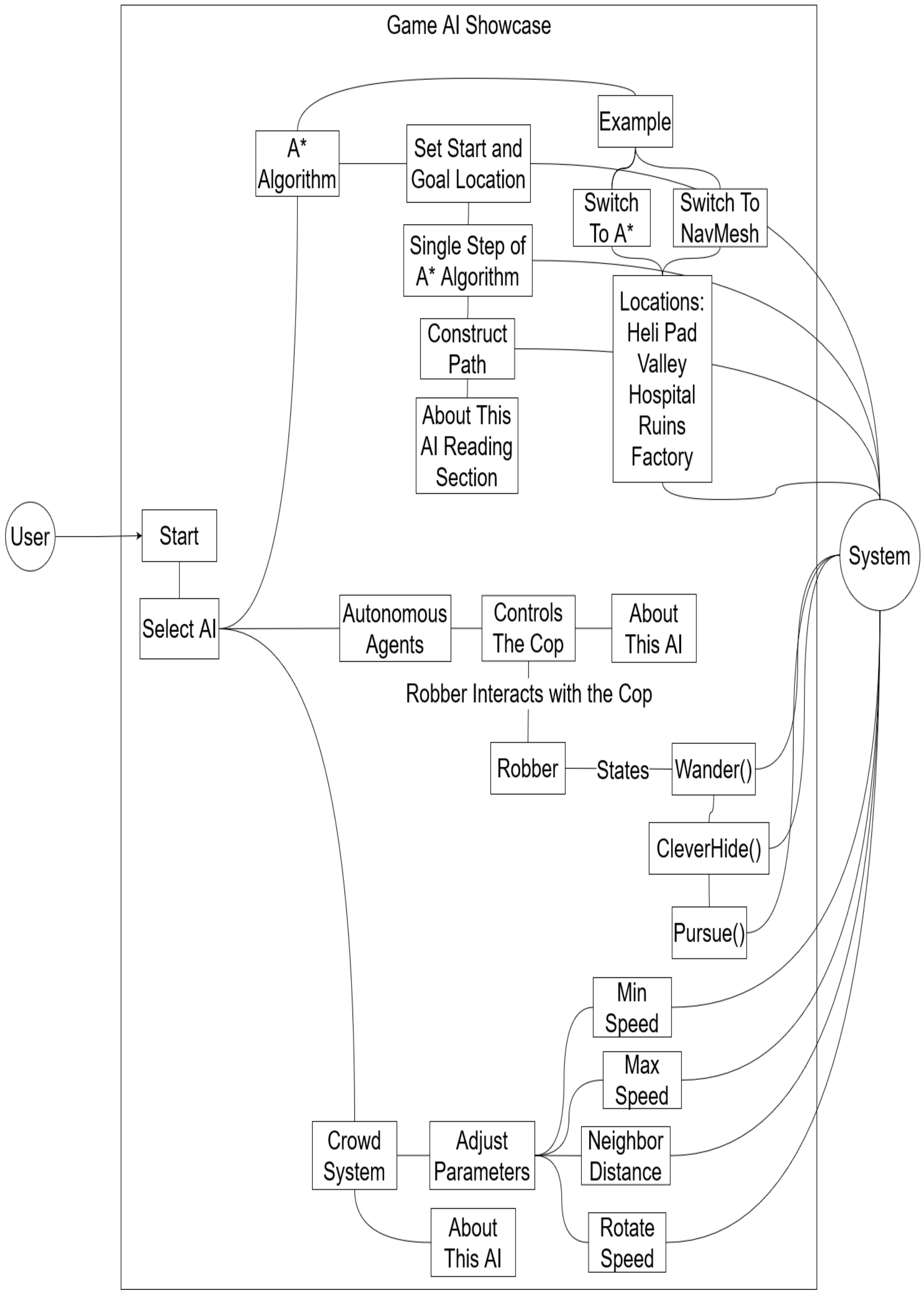
In this section, the focus is on simulating crowd behaviour, which is distinct from individual behaviour. In crowds, properties like motion, collision, direction, and speed propagate from one person to another, giving the group its own identity. Simulating every individual in a crowd can be labour-intensive, so simpler methods using vector mathematics are often employed.

Different methods for simulating crowds include treating people as particles in a fluid and using real-world data. Crowds move in a quasi-organized fashion rather than chaotically, forming self-organizing systems with flow lines in the direction of movement. However, when crowd density exceeds six people per square meter, turbulence can occur.

One popular and simple method for simulating crowds is Reynolds' flocking algorithm, used in motion pictures to simulate animal herds and armies. This algorithm is based on three rules:

- Turn towards the average heading of the group.
- Move towards the average centre of the group.
- Avoid close proximity to others.

3.3 Use Case Diagram



3.3.1 Use Case Descriptions

Use Case Name	Start
Actor	User
Precondition	n/a
Flow and Count	Initiating the game.

Use Case Name	Select AI
Actor	User
Precondition	● The user has started the game. ● The game is in AI selection Menu
Flow and Counts	1. Click on “Select AI”. 2. Select the desired AI. a. A* Algorithm: Choosing the A* pathfinding algorithm for the AI. b. Autonomous Agents: Selecting AI that operates autonomously within the game. c. Crowd System: Managing a system that controls the behaviour of crowds in the game.

Use Case Name	Set Start and Goal Location
Actor	User
Precondition	Be in the A* Algorithm AI
Flow and Counts	Click on ‘Q’ to Define the initial and target positions for the pathfinding algorithm.

Use Case Name	Single Step of A* Algorithm
Actor	User
Precondition	Be in the A* Algorithm AI
Flow and Counts	Click on ‘W’ to Execute a single iteration of the A* algorithm.

Use Case Name	Construct Path
Actor	User
Precondition	Be in the A* Algorithm AI
Flow and Counts	Click on ‘E’ to Build a complete path from start to goal using the A* algorithm

Use Case Name	Example of A*
Actor	User
Precondition	Be in the A* Algorithm AI
Flow and Counts	1. Click on ‘Esc’ to bring in the Pause menu. 2. Click on ’ Example’ to switch to the Example scene.

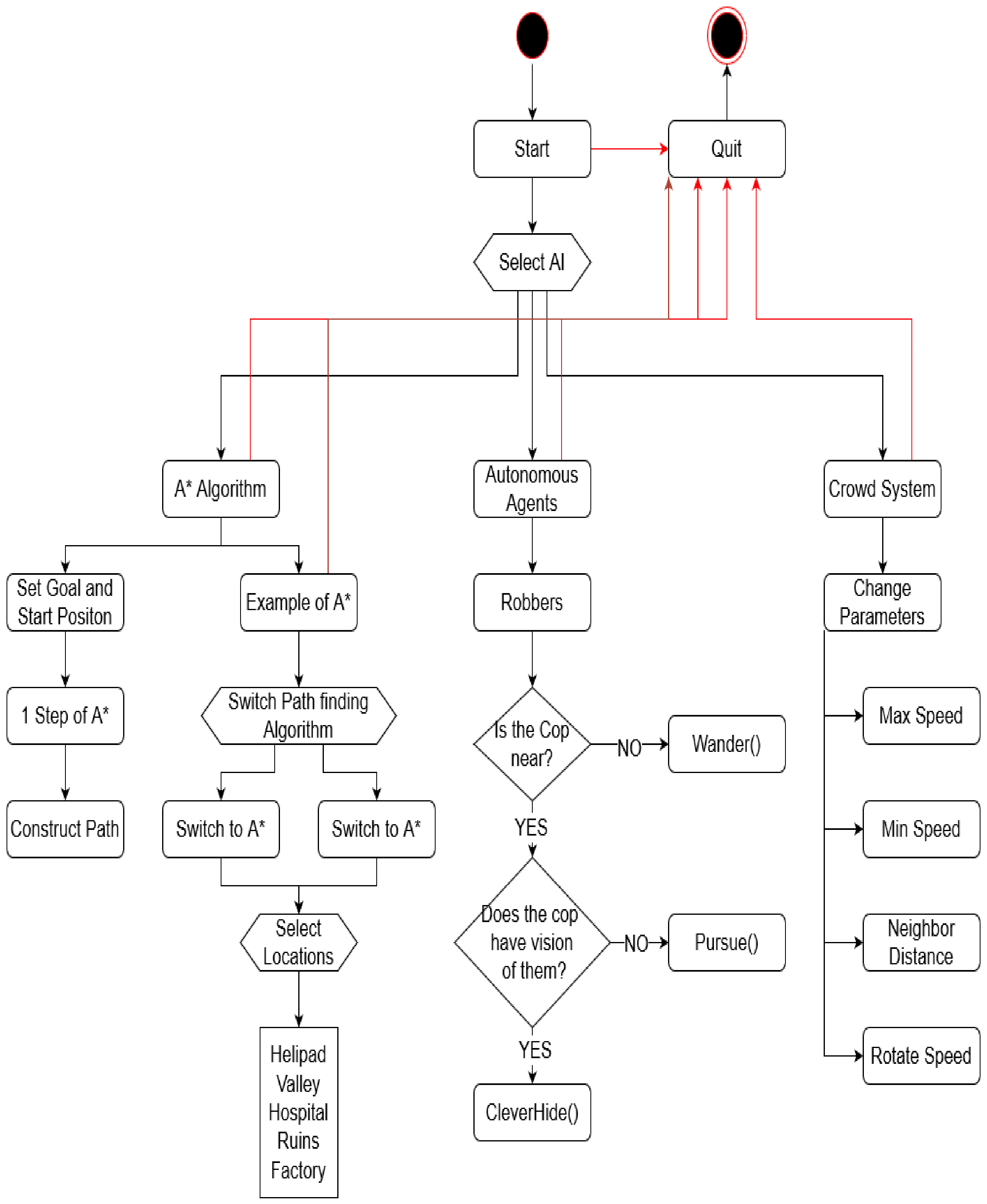
Use Case Name	Switching Pathfinding Algorithms
Actor	User
Precondition	Be in the A* Algorithm AI’s Example Scene
Flow and Counts	1. Click on ‘Switch To A*’ to Change the current pathfinding algorithm to A* 2. Click on ‘Switch To NavMesh’ to Change the current pathfinding system to a navigation mesh. 3. Choose location, ○ Heli Pad ○ Valley ○ Hospital ○ Ruins ○ Factory

Use Case Name	Controls The Cop
Actor	User
Precondition	Be in the Autonomous Agents AI.
Flow and Counts	1. Go forward and Backward with ‘W’ and ‘S’ 2. Rotate the cop using ‘A’ and ‘D’ 3. Control the camera with the mouse.

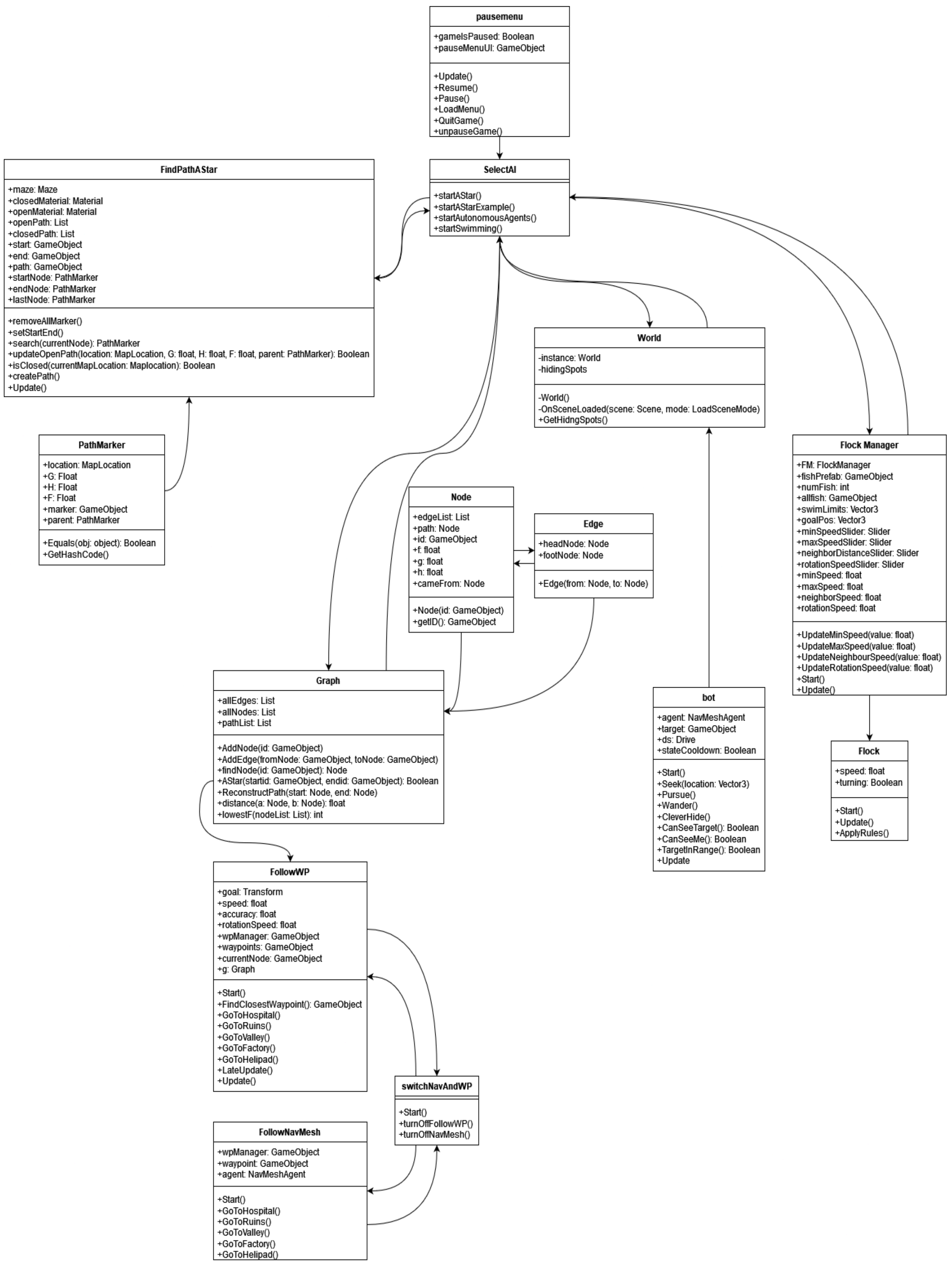
Use Case Name	Robber Interacts with the Cop
Actor	System
Precondition	1. Be in the Autonomous Agents AI. 2. Must be controlling the cop.
Flow and Counts	The robber has Three states of interaction with the cop. ● Wander(): AI characters moving randomly or exploring. ● CleverHide(): AI characters hiding intelligently based on the cop and the distance between them. ● Pursue(): AI characters chasing the cop.

Use Case Name	Adjust Parameters
Actor	User
Precondition	Be in the Crowd System AI
Flow and Counts	Adjust parameters of the crowd system using sliders. ● Min Speed: Setting the minimum speed for the AI-controlled characters. ● Max Speed: Setting the maximum speed for the AI-controlled characters. ● Neighbor Distance: Determining the proximity required for AI characters to consider others as neighbours. ● Rotate Speed: Adjusting the rate at which AI characters can change direction.

3.4 Activity Diagram



3.5 Class Diagram



Chapter 4 Development Tool & Technology

4.1 Integrated Development Environment (IDE)

Visual Studio has been chosen as the Integrated Development Environment (IDE). The selection of Visual Studio is strategic, as it offers robust support for Unity, the engine powering this project. Visual Studio's comprehensive debugging tools, intuitive interface, and extensive library support greatly facilitate the development process.

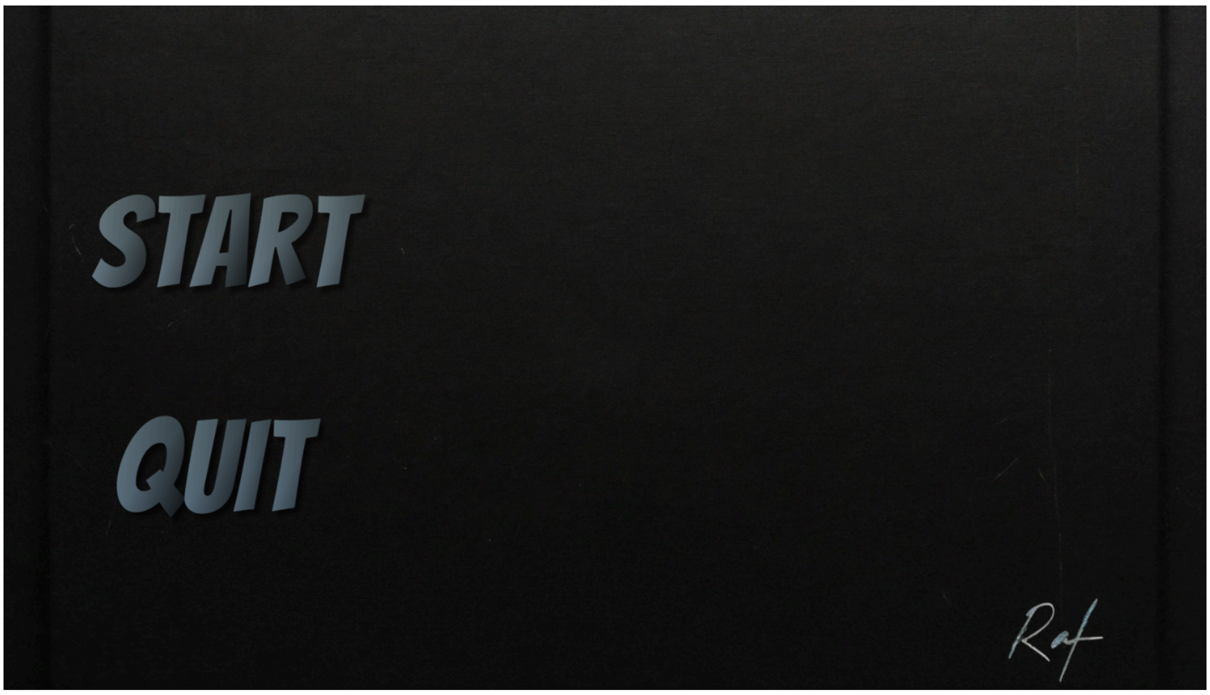
4.2 Programming Language

In this project, C# has been chosen as the primary programming language, accounting for approximately 66% of the development. This decision aligns with the technical requirements of the Unity game engine, which exclusively supports C#.

C# is known for its simplicity and readability. C# is an object-oriented programming language, which is well-suited for game development. It allows for organizing complex game elements into manageable objects, making it easier to create, manage, and modify complex game worlds and interactions.

Unity aims to enable game development for multiple platforms, and C# is a cross-platform language. This means games developed in Unity can be easily ported to various platforms like Windows, macOS, Linux, iOS, Android, and more. C# comes with an extensive set of libraries and frameworks.

4.3 User Interface Design

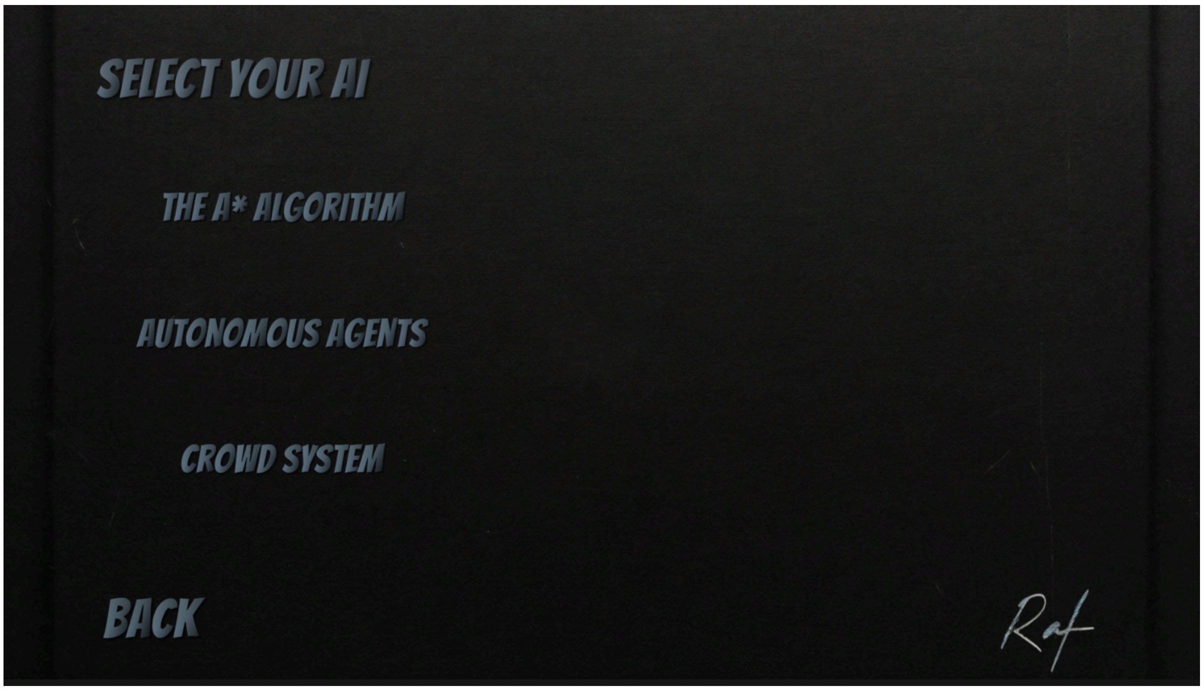


The user interface here is a very minimalistic design. The interface presents two options: "START" and "QUIT".

The background is predominantly dark. This choice suggests a simplistic and sophisticated design approach. The darkness of the background helps the options stand out more prominently. The options are in capital letters.

The START option is used to begin the game. It is placed above "QUIT".
QUIT is the alternative option, allowing the user to exit the game. Its placement below "START" suggests it is the secondary choice.

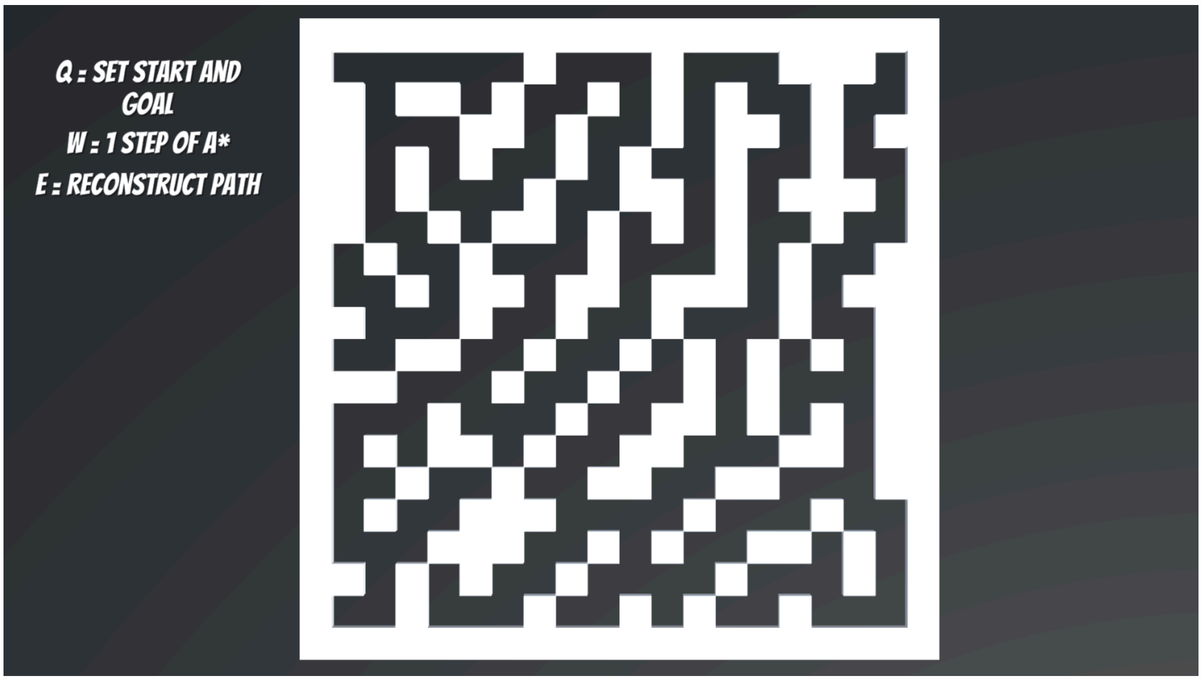
The text has a three-dimensional, metallic look with light reflecting off the upper edges, which gives it a tactile and modern feel. This style makes the options appear to be more interactive, almost like physical buttons that can be pressed.
In the bottom right corner, there is a signature "Raf".



The user interface here, the user can select different types of artificial intelligence (AI) options. The interface is minimalist and uses a dark background to highlight the text options.

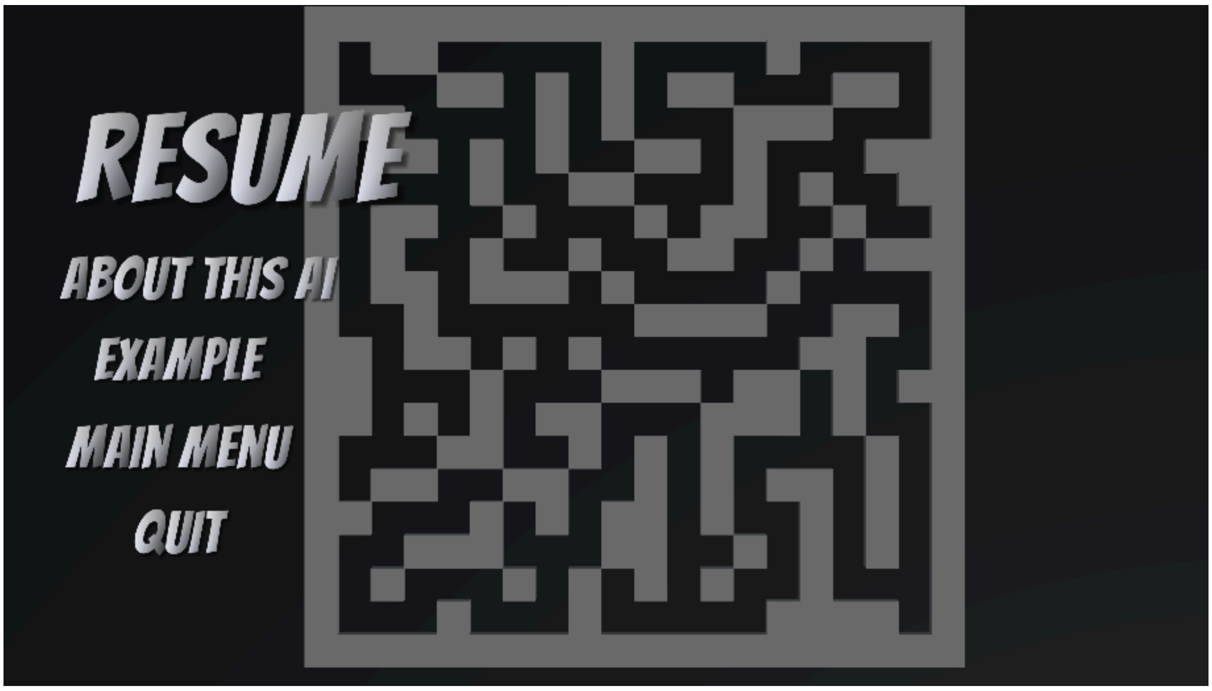
"SELECT YOUR AI" is the title at the top of the menu. THE A* ALGORITHM This refers to a specific AI pathfinding algorithm. AUTONOMOUS AGENTS, This refers to AI entities that operate independently. CROWD SYSTEM, This refers to an AI system designed to simulate large groups of entities.

At the bottom, there is an option labelled "BACK", which suggests that selecting this will return the user to a previous menu.



the image demonstrates the A* (A-star) pathfinding algorithm. The interface includes a visual representation of a maze and a set of controls that the user can input.
The maze is presented in a high-contrast color scheme (black and white), which makes the paths and walls distinguishable.

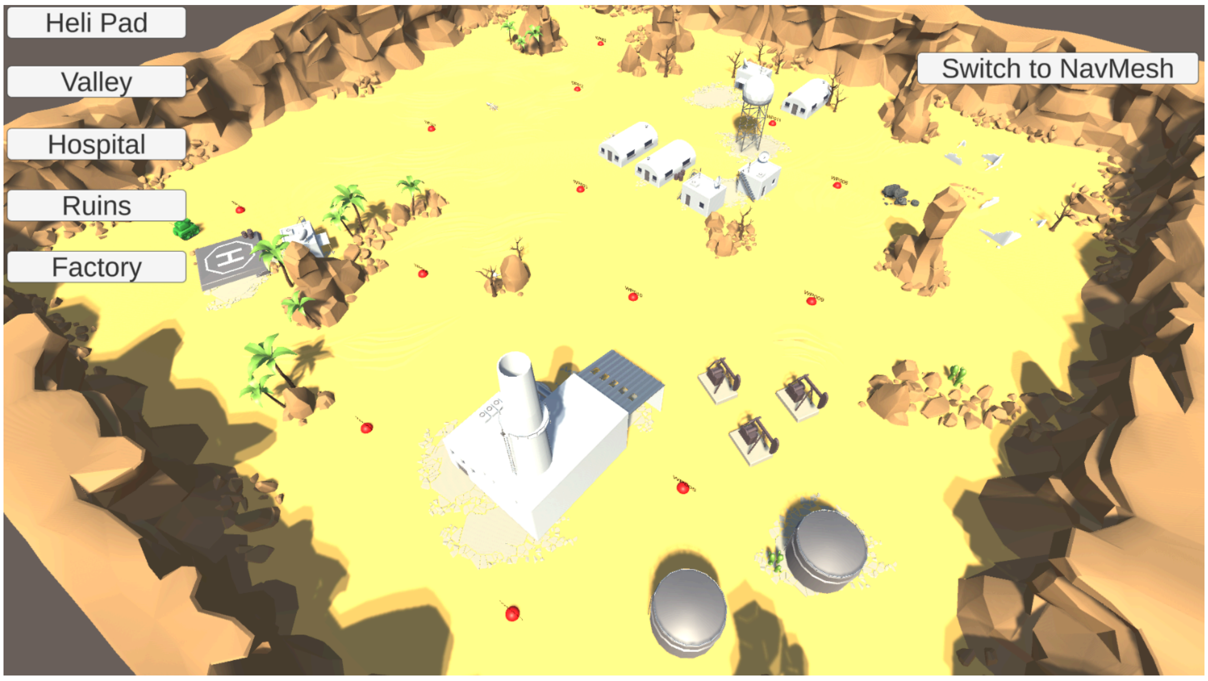
Q: "SET START AND GOAL" suggests that pressing the 'Q' key allows the user to define the starting point and the goal or endpoint within the maze.
W: "1 STEP OF A*" indicates that pressing the 'W' key will execute one step of the A* algorithm.
E: "RECONSTRUCT PATH" means that pressing the 'E' key will trigger the system to show the path that has been calculated by the A* algorithm from the start to the goal.

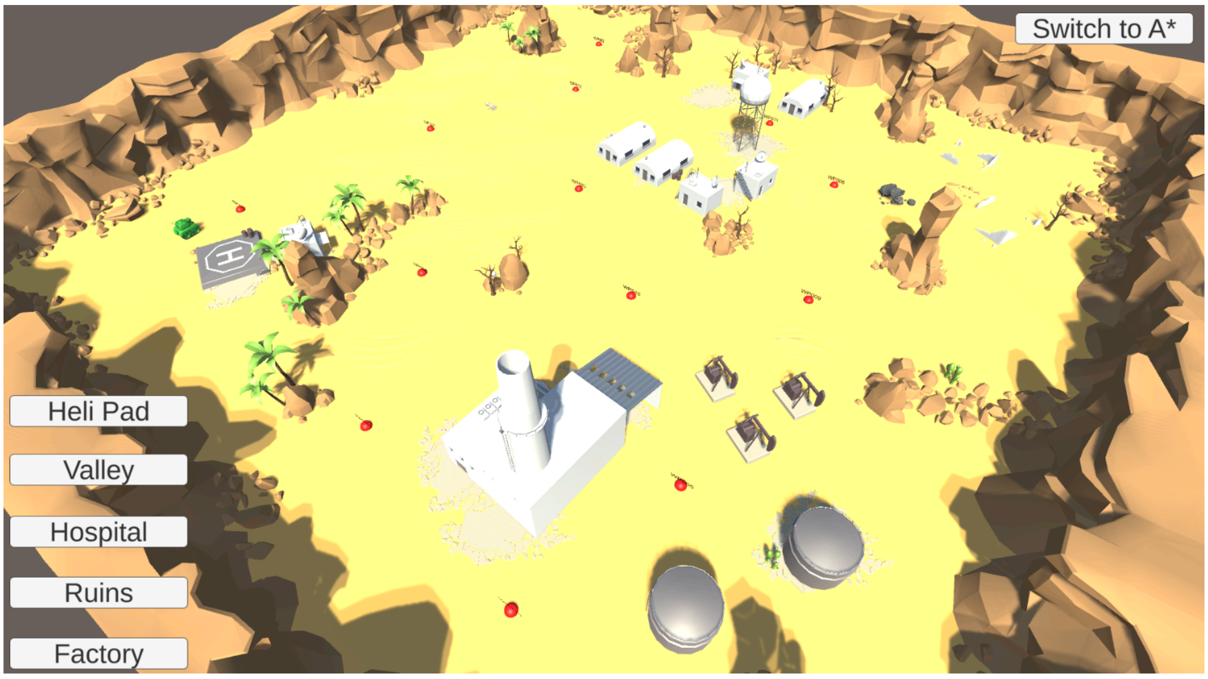


It's the pause menu from the game. The menu provides several options for the user to select from.

RESUME: This option is used to continue with the game after pausing.
ABOUT THIS AI: This provides information on the artificial intelligence component of the game, potentially explaining how the AI works or its role within the game.
EXAMPLE: This shows an example scenario, illustrating how to navigate the maze.
MAIN MENU: This returns the user to the main screen of the game.
QUIT: This option is for exiting the game or application.

The options are listed in a vertical format on the left side of the screen, making them easy to navigate. The text is in all caps, which makes each option stand out and indicates their functionality as buttons or selectable items.





The environment navigation screen features a top-down view of a simulated environment with various areas like 'Heli Pad', 'Valley', and 'Hospital' labelled in a column on the left side. button to 'Switch to NavMesh' and ‘Switch to A*’ indicates an interactive element to switch between AI navigation in the scene.

4.4 Release Environment

The project is an open-source project. The final build will be available on GitHub with the necessary documents. Any further improvements or changes will be handled there.

Chapter 5 System Testing

5.1 Testing Features

Testing was crucial to this project. Every AI technique in this project was tested separately in different environments. Each AI technique has its parts, such as agents, scripts, and scenes. These needed testing each time the core part was completed developing.

The following is a list of features that needed testing:

- Start Menu: The start menu was tested to ensure that it was functional and easy to use and visual clarity was on point.
- AI Selection: The AI selection system was tested to ensure that it was easy to use and that users could select the AI technique they wanted.
- Scene Changing: The scene-changing system was tested to ensure smoothness and that the AI functionality was not lost when changing scenes.
- HUD Elements: The HUD elements were tested to ensure that they were functional and displayed the correct information.
    - Objective: To ensure all HUD elements are functional and display correct, real-time information.
    - Visibility and Design: Ensured clarity and readability of HUD elements under various game conditions.
- A* Algorithm: The A* algorithm was tested to ensure that it was efficient and found the shortest path to the goal.
    - Objective: To ensure the algorithm efficiently calculates the shortest possible path in different scenarios.
    - Performance Metrics: Measure the computation time and compare the path length with theoretical optimal paths.
- Autonomous Agents: The autonomous agents were tested to ensure that they were able to automatically change states and be aware of the environment.
    - Objective: To ensure agents can autonomously change states and interact appropriately with the environment.
    - Behavior Testing: Evaluated the agents’ decision-making processes and state transitions in various scenarios.
- Crowd System: The crowd system was tested to ensure that the agents could interact with each other and avoid collisions.

5.2 Testing Strategies

The testing strategy for the Game AI showcase project was designed to be comprehensive and iterative, ensuring thorough validation of each component and the system as a whole.

5.2.1 Test Approach

Unit Testing: Each component, such as individual AI algorithms and interface elements, was tested in isolation to ensure it performed as expected.
Integration Testing: After unit testing, components were tested in combination to ensure they work together seamlessly.
System Testing: The entire system was tested as a whole to ensure all components function correctly together. This included testing the overall user experience, the transition between different AI showcases, and the integration of HUD elements.

5.2.2 Pass/Fail Criteria

Functionality: Each test case had clear pass/fail criteria based on whether the system component met its functional requirements.
Performance: For AI algorithms, performance was measured against specific benchmarks, such as pathfinding efficiency and response time.
Usability: User acceptance tests included criteria for ease of use, understanding of AI concepts presented, and overall user engagement.

5.2.3 Testing Schedule

Phased Approach: Testing was conducted in phases, corresponding with the development stages – from unit testing during early development to full system testing closer to project completion.
Iterative Cycles: Each testing phase was followed by a review and iteration cycle, where findings were analyzed, and necessary adjustments were made.
Final Review: Before the final deployment, a comprehensive testing cycle was conducted to ensure all previously identified issues were resolved and that the system met all its objectives.
Conclusion

Chapter 6 Conclusion

6.1 Project Link

Access the project, view its components, and understand its functionality from the below link, https://github.com/Raf-byte/Game-AI-Showcase

In the link there will be the source code of the project and a release version.

6.2 Limitations

Technological limitation: The performance of AI simulations, particularly in the context of crowd systems, was limited by hardware capabilities. This occasionally led to reduced simulation complexity or a lower number of autonomous agents than initially envisioned.
Resource Limitation: The project timeline was a significant constraint, affecting the AI techniques that could be explored and implemented.
Functional Limitations: The project does not currently allow users to modify AI parameters or interact deeply with the AI systems, which restricts the level of engagement and experimentation users can have.

6.3 Future Scope

Advanced AI Implementations: Future versions of the project could include more complex and advanced AI algorithms. This expansion would cater not only to beginners but also to intermediate and advanced learners.
Interactive AI Experiments: Introducing features that allow users to interact with and modify AI parameters could significantly enhance the educational value of the project. This would enable users to experiment with AI behaviours and understand their impacts firsthand.
Optimization for Higher Performance: Future work could focus on optimizing the code and algorithms for better performance, allowing for more complex simulations and a larger number of agents in the crowd system.
Cross-Platform Compatibility: Making the project accessible on various platforms, including mobile devices, would increase its reach and usability.
Incorporating Educational Content: Adding in-depth educational materials, such as tutorials, use case examples, and theoretical backgrounds of AI algorithms, would make the project a more comprehensive learning tool.
Real-Game Scenarios: Integrating real-world game scenarios or partnering with game developers could provide practical examples of AI applications, bridging the gap between theory and practice.
Research Collaboration: Collaborating with academic institutions or research groups could lead to the project being used as a research tool, contributing to the field of AI in gaming.

References

- https://en.wikipedia.org/wiki/Artificial_intelligence_in_video_games
- https://unity.com/
- https://en.wikipedia.org/wiki/Agile_software_development
- https://visualstudio.microsoft.com/
- https://assetstore.unity.com/
- https://brilliant.org/wiki/a-star-search/
- https://www.scirp.org/journal/paperinformation?paperid=70460
- https://nap.nationalacademies.org/read/11827/chapter/4
- https://en.wikipedia.org/wiki/Crowd_simulation