UNIVERSITY OF PLYMOUTH

PUSL3190 Computing Individual Project

Interim Document

Smart enemy AI for video games

Supervisor: Mr.Gayan perera

Name: Kosmapathabendige S Dalpathadu
Plymouth Index Number: 10818154
Degree Program: BSc (Hons) Software Engineering

Contents

1. Introduction

1.1 Introduction

The journey embarked upon delves deep into the immersive realm of online multiplayer gaming, tracing its origins back to the year 2009, when the allure of virtual worlds first ensnared the imagination. From that pivotal moment, a profound exploration unfolded, traversing through legendary titles such as Call of Duty and Special Force 2, while engaging in numerous competitive gaming competitions. Through these experiences, not only were gaming skills finely honed, but a crucial realization also emerged - the essence of the online gaming adventure lies in the unpredictable and strategic encounters with human-controlled adversaries.

As a fervent gamer and an aspiring game developer, the acknowledgment of a notable void within the traditional gaming landscape became apparent. Offline games, despite their captivating narratives and diverse environments, often lacked the dynamic and competitive essence inherently found within online multiplayer gaming realms. Thus, the challenge crystallized - how might one infuse the offline gaming experience with the strategic complexity and adaptability synonymous with human-controlled opponents?

The impetus driving this ambitious project lies in the desire to forge a gaming environment that transcends the conventional constraints of offline gaming paradigms. Leveraging the cutting-edge capabilities of Unreal Engine 5, the ultimate goal is to craft an immersive gaming milieu where players are thrust into encounters with adversaries embodying the intelligence, adaptability, and skill sets reminiscent of human players within the online multiplayer domain.

With each line of code and meticulously designed gameplay mechanic, the vision unfolds to create an unparalleled gaming experience that blurs the boundaries between virtual and reality. Through the fusion of innovative technology and creative ingenuity, the aspiration is to transport players into a realm where every interaction, every decision, and every moment unfolds with the unpredictable allure and strategic depth akin to engaging with human adversaries in the vast expanse of online multiplayer landscapes.

In the pursuit of this vision, the journey unfolds, guided by a relentless commitment to push the boundaries of gaming innovation. With each milestone achieved, the path forward becomes clearer, as the realization dawns that the creation of a truly immersive and dynamic gaming experience lies within reach. As the project unfolds, it is not merely the development of a game but the realization of a transformative vision - one that seeks to redefine the very essence of offline gaming and usher in a new era of immersive gameplay experiences for players worldwide.

1.2 Problem Definition

The contemporary gaming landscape presents a stark juxtaposition between the dynamic, strategic allure of online multiplayer gaming and the narrative-driven yet often static experience of offline gaming. Despite the immersive narratives and diverse environments offered by offline games, they frequently lack the unpredictable and competitive edge intrinsic to human-controlled adversaries in online multiplayer environments.

The crux of the problem lies in the disparity between offline and online gaming experiences, where offline games struggle to replicate the strategic complexity and adaptability offered by human players. This gap in gaming experiences hampers the ability of offline games to fully engage and captivate players, leading to a sense of disconnect and limited replay ability.

The challenge, therefore, is to bridge this gap by infusing offline gaming experiences with the strategic depth, adaptability, and unpredictability characteristic of human-controlled opponents in online multiplayer settings. This requires the development of innovative game mechanics, artificial intelligence algorithms, and immersive environments that can replicate the dynamic nature of online multiplayer gaming within offline contexts.

Key considerations for addressing this challenge include:
1. Dynamic Adversarial AI: Designing AI systems capable of emulating the strategic decision-making, adaptability, and skill levels of human players in online multiplayer environments.
2. Immersive Environments: Creating rich and immersive game worlds that engage players on multiple levels and provide opportunities for emergent gameplay experiences.
3. Gameplay Mechanics: Developing innovative gameplay mechanics that encourage strategic thinking, adaptability, and player engagement while maintaining a coherent narrative structure.
4. Technological Integration: Leveraging cutting-edge game development technologies such as Unreal Engine 5 to create visually stunning and technically advanced gaming experiences.
5. User Experience: Ensuring that the gaming experience remains intuitive, rewarding, and accessible to a wide range of players while challenging them to improve their skills and strategies over time.

By addressing these key considerations, the goal is to redefine the offline gaming experience and create a new paradigm that combines the best elements of online multiplayer gaming with the immersive narratives and diverse environments of offline games. In doing so, the aim is to captivate players, foster deeper engagement, and push the boundaries of gaming innovation in the pursuit of truly transformative gaming experiences.

1.2 Project Objectives

1. AI Development Milestones:
   - As we delve into the realm of AI development for our gaming project, several key milestones stand as pillars in achieving our overarching goal of creating immersive, dynamic, and engaging gameplay experiences. Each milestone is meticulously crafted to address specific objectives while ensuring measurable outcomes that guide our progress and validate our efforts.
     - Objective: Complete the implementation of the learning algorithm for in-game adversaries.
     - Measurable Outcome: Achieve an 80% accuracy rate in simulating human decision-making processes.

In the pursuit of this milestone, our primary focus lies in the development and refinement of the AI algorithms that govern the behavior of in-game adversaries. Through iterative design and testing cycles, we aim to imbue our AI adversaries with strategic insight, adaptability, and nuanced decision-making capabilities reminiscent of human players. By achieving an 80% accuracy rate in simulating human decision-making processes, we aim to create adversaries that challenge and engage players in dynamic and unpredictable ways, thereby enhancing the overall gaming experience.

2. Immersive Environment Creation:
    - Objective: Design and implement three visually stunning game environments.
    - Measurable Outcome: Conduct player surveys, with a minimum 90% positive rating on the visual appeal of environments.

    The creation of immersive environments serves as a cornerstone in capturing the imagination of players and providing a rich backdrop for their gaming experiences. Our objective is to meticulously craft three visually stunning game environments that transport players to vibrant and captivating worlds filled with detail and atmosphere. Through player surveys and feedback, we aim to gauge the effectiveness of our environment design, with a minimum 90% positive rating on the visual appeal serving as a testament to our success in creating immersive and engaging game worlds.

3. Player-Adversary Interaction:
    - Objective: Implement dynamic responses from AI based on player actions.
    - Measurable Outcome: Achieve a 75% player satisfaction rating for the realism of AI interactions.

    Central to the gaming experience is the interaction between players and in-game adversaries. Our objective is to implement dynamic responses from AI adversaries that react and adapt to player actions in real-time, creating a sense of immersion and challenge. By achieving a 75% player satisfaction rating for the realism of AI interactions, we seek to foster engaging and meaningful encounters that keep players invested and immersed in the gaming world.
4. Testing and Iteration:
    - Objective: Address 95% of reported bugs and glitches during the beta testing phase.
    - Measurable Outcome: Achieve a 90% positive rating from beta testers regarding the overall gameplay experience.

    Testing and iteration play a pivotal role in refining and optimizing the gaming experience. Our objective is to conduct thorough testing during the beta phase, addressing 95% of reported bugs and glitches to ensure a polished and seamless gameplay experience. Additionally, we aim to achieve a 90% positive rating from beta testers regarding the overall gameplay experience, reflecting our commitment to delivering a high-quality product that exceeds player expectations.
5. Technical Optimization:

  - Objective: Achieve a minimum of 30 frames per second (fps) on devices with varied specifications.
  - Measurable Outcome: Conduct successful testing on at least five different hardware configurations.

  Technical optimization is essential to ensure smooth and consistent performance across a diverse range of hardware configurations. Our objective is to achieve a minimum of 30 frames per second (fps) on devices with varied specifications, thereby providing players with a seamless and immersive gaming experience. By conducting successful testing on at least five different hardware configurations, we aim to optimize performance and compatibility, ensuring that our game reaches a wide audience without compromising on quality or performance.

In summary, these AI development milestones serve as guideposts on our journey to creating a groundbreaking gaming experience that transcends traditional boundaries and captivates players worldwide. Through meticulous planning, iterative design, and unwavering dedication, we are committed to realizing our vision and pushing the boundaries of gaming innovation.

2. System Analysis

2.1 Facts Gathering Techniques

1. Introduction to Facts Gathering Techniques:
Fact gathering techniques serve as the cornerstone for the development of dynamic and responsive AI adversaries in our gaming project. These techniques encompass a variety of methods and tools aimed at collecting and analyzing relevant data from the game environment, player interactions, and internal AI states. By employing effective facts gathering techniques, we aim to enhance the adaptability, realism, and strategic depth of our AI adversaries, thereby enriching the overall gaming experience for players.
2. Sensor-Based Data Collection:
Sensor-based data collection forms a fundamental aspect of our facts gathering techniques, enabling our AI adversaries to perceive and interpret the game environment in real-time. Leveraging advanced AI perception systems, including sight, sound, and touch sensors, we gather crucial information about the spatial layout, object properties, and player movements within the game world. By integrating sensor data into our AI logic, we empower our adversaries to make informed decisions and react dynamically to changes in their surroundings.
3. Behavior Trees and Blackboards:
Behavior Trees and Blackboards serve as powerful tools for organizing and processing the data collected by our AI adversaries. Through behavior trees, we define hierarchical decision-making structures that govern the behavior and actions of our adversaries based on incoming sensory information and internal states. Blackboards provide a shared memory space where relevant data can be stored and accessed during runtime, facilitating seamless communication and coordination between different components of our AI system.
4. Environment Query System (EQS):
The Environment Query System (EQS) further enhances our facts gathering capabilities by enabling advanced spatial analysis and querying of the game environment. Through EQS, our AI adversaries can dynamically evaluate spatial relationships, object properties, and environmental conditions to inform their decision-making process. By formulating queries and evaluating results, we gain valuable insights into the game world, allowing our adversaries to adapt and respond intelligently to changing circumstances.
5. Player Interaction and Feedback:
Player interaction serves as a vital source of data for our facts gathering techniques, providing valuable insights into player behavior, preferences, and engagement patterns. By monitoring player actions, reactions, and feedback, we gain a deeper understanding of player expectations and experiences within the game. This information informs our AI development efforts, enabling us to tailor the behavior and responses of our adversaries to better align with player expectations and enhance overall satisfaction.
6. Continuous Improvement and Iteration:
Continuous improvement and iteration are central tenets of our facts gathering approach, allowing us to refine and optimize our AI adversaries based on real-world data and player feedback. Through iterative development cycles, we collect empirical data on the performance and effectiveness of our AI systems, identifying areas for improvement and refinement. By embracing a data-driven approach to AI development, we ensure that our

adversaries evolve dynamically over time, delivering a compelling and immersive gaming experience for players.

In conclusion, facts gathering techniques play a pivotal role in the development of dynamic and responsive AI adversaries within our gaming project. By leveraging sensor-based data collection, behavior trees, EQS, player interaction, and continuous improvement strategies, we aim to create AI adversaries that challenge, engage, and adapt to players in meaningful and immersive ways. Through ongoing research and development, we remain committed to pushing the boundaries of AI-driven gaming experiences and delivering unparalleled entertainment for players worldwide.

2. 2 Existing System

1. Current State of Offline Gaming:
Offline gaming experiences have traditionally relied on scripted events, predetermined outcomes, and static AI behavior to drive player engagement. While these games often boast captivating narratives and visually stunning environments, they frequently lack the dynamic and unpredictable nature of human-controlled adversaries found in online multiplayer games. Players may encounter repetitive gameplay patterns, limited replay value, and a sense of disconnect from the gaming experience due to predictable AI behavior.
2. Challenges in Offline Gaming:

The primary challenge in offline gaming lies in replicating the strategic complexity, adaptability, and unpredictability of human players in online multiplayer environments. Existing offline games struggle to provide AI adversaries that can effectively challenge players, leading to diminished player engagement and satisfaction. Additionally, technical limitations may hinder the creation of immersive environments and dynamic AI systems capable of delivering compelling gaming experiences.

3. Limitations of Current Approaches:

Traditional approaches to AI development in offline gaming often rely on predefined algorithms, static decision trees, and scripted behaviors to simulate adversary behavior. While these methods may suffice for basic gameplay scenarios, they fail to capture the nuanced decision-making processes and adaptive strategies exhibited by human players in online multiplayer settings. As a result, offline gaming experiences may feel stale, repetitive, and lacking in depth compared to their online counterparts.

4. Opportunities for Improvement:

Despite these challenges, advancements in AI technology, game development tools, and hardware capabilities present opportunities for innovation in offline gaming. By leveraging machine learning algorithms, procedural generation techniques, and sophisticated AI architectures, developers can create more dynamic, responsive, and immersive gaming experiences. Additionally, the integration of emerging technologies such as virtual reality (VR) and augmented reality (AR) opens up new possibilities for enhancing player immersion and engagement in offline gaming environments.
Conclusion:

While the existing system of offline gaming has its strengths, it also faces significant limitations in replicating the dynamic and competitive nature of online multiplayer experiences. By addressing these challenges and embracing innovative approaches to AI development, game design, and technological integration, developers can unlock the full potential of offline gaming and deliver truly transformative experiences for players worldwide. The next section will explore proposed solutions and methodologies for overcoming these limitations and realizing the vision of immersive and dynamic offline gaming environments.

2. 3 Use case Diagram

The decision not to use Entity-Relationship (ER) diagrams or Use case in my project may stem from its focus on AI-driven gameplay mechanics, immersive environments, and player-adversary interactions, where ER diagrams may not directly contribute. Given the project's emphasis on aspects such as agile development, resource constraints, and alternative documentation practices specific to game development, other visualization tools like flowcharts, UML diagrams, or textual descriptions may be more suitable for representing system components and interactions. Additionally, ER diagrams typically require time, effort, and domain expertise to create accurately, which may not align with the project's priorities for rapid prototyping and continuous iteration. Thus, the choice to forego ER diagrams likely reflects a strategic decision to optimize resource allocation and documentation practices in alignment with the project's objectives and constraints.

2. 4 Drawbacks of the existing system

1. Limited Player Engagement: The static nature of AI adversaries and scripted events in offline games can lead to limited player engagement and replay value. Players may quickly exhaust content and encounter predictable gameplay patterns, resulting in diminished interest and satisfaction over time.

2. Lack of Adaptability: Offline games often lack the adaptability and responsiveness of online multiplayer environments, where human-controlled adversaries can dynamically adjust their strategies and tactics in real-time. As a result, offline gaming experiences may feel stale and lacking in challenge, especially for experienced players seeking dynamic encounters.

3. Repetitive Gameplay: Predefined AI behavior and scripted events in offline games can result in repetitive gameplay experiences, where players encounter the same challenges and obstacles with little variation. This can lead to boredom and frustration, as players may feel like they are going through the motions rather than experiencing new and exciting challenges.

4. Limited Immersion: While offline games may feature captivating narratives and visually stunning environments, the lack of dynamic AI behavior and interactive elements can detract from overall immersion. Players may feel disconnected from the game world and characters, reducing the emotional impact and sense of immersion compared to online multiplayer experiences.

5. Dependency on Hardware: Offline games may be limited by the hardware capabilities of the player's device, restricting the complexity and scale of gameplay experiences. This can result in performance issues, graphical limitations, and compatibility constraints that limit the potential for creating immersive and visually stunning gaming experiences.

6. Development Constraints: Traditional approaches to AI development and game design in offline gaming may impose limitations on innovation and creativity. Developers may face challenges in implementing dynamic AI behavior, procedural content generation, and emergent gameplay mechanics within the constraints of existing systems and technologies.

7. Disconnect from Community: Offline gaming experiences lack the social interaction and community engagement inherent in online multiplayer environments, where players can connect, compete, and collaborate with others in real-time. This can lead to a sense of isolation and disconnect for players accustomed to the vibrant and interactive communities found in online gaming platforms.

3. Requirements Specification

3.1 Functional Requirements

1. Data Collection and Perception:
   - The system must collect data from the environment, including terrain, objects, and player movements, using advanced AI perception systems.
   - Smart enemies should perceive and interpret environmental stimuli such as visual, auditory, or tactile cues to inform their behavior and decision-making.

2. Behavior Variety:
   - Smart enemies must exhibit a variety of behaviors such as patrolling, attacking, hiding, retreating, and coordinating with other enemies.
   - The system should support the implementation of behavior trees and blackboards to organize and prioritize enemy behaviors based on situational context and objectives.

3. Adaptation to Player Actions:
   - Enemies should react dynamically to player actions such as movement, attacks, and interactions.
   - Smart enemies must adjust their tactics, aggression level, and defensive posture based on the player's behavior, providing a challenging and responsive gameplay experience.

4. Internal State Management:
   - The system should enable enemies to monitor their internal state, including health, stamina, morale, and ammunition levels.

- Smart enemies must react to changes in their internal state, adjusting their behavior, aggression level, and engagement tactics accordingly.

5. Interaction with Environment:
   - Enemies should interact with the environment in meaningful ways, such as taking cover, navigating obstacles, and using environmental objects for tactical advantage.
   - The system must support dynamic querying and analysis of the game environment through tools like the Environment Query System (EQS), allowing enemies to gather information and make informed decisions based on spatial relationships and object properties.

6. Coordination and Cooperation:
   - Smart enemies should exhibit cooperative behaviors such as flanking, ambushing, and coordinating attacks with other enemies.
   - The system should facilitate communication and coordination between different enemy types, allowing them to work together strategically to overcome the player's defenses.

7. Scalability and Customization:
   - The system should be scalable to accommodate a wide range of enemy types, behaviors, and scenarios.
   - Developers should be able to customize enemy behaviors, attributes, and responses through configuration files or scripting interfaces, allowing for flexibility and adaptability in game design.

8. Performance and Optimization:
   - The system must be optimized for performance to ensure smooth and consistent gameplay experiences across different hardware configurations.
   - Smart enemy behaviors and decision-making processes should be efficient and computationally lightweight, minimizing processing overhead and resource consumption.

3.2 Functional Requirements

1. Performance:
   - The system should respond to player actions with minimal latency, ensuring smooth and responsive gameplay.
   - Smart enemies' decision-making processes should be efficient, allowing for real-time adaptation to environmental changes and player interactions.
   - The system should maintain a consistent frame rate (e.g., 30 frames per second) across different hardware configurations to provide a seamless gaming experience.

2. Reliability:

- Smart enemies' behaviors and responses should be consistent and predictable, ensuring a fair and balanced gameplay experience.
- The system should handle errors and exceptions gracefully, minimizing the impact of unexpected failures or crashes on the player's progress.
- Smart enemies should recover gracefully from disruptions such as temporary loss of perception or communication with other entities.

3. Security:
   - The system should protect player data and game state from unauthorized access, tampering, or exploitation.
   - Smart enemies' behavior should not pose security risks such as unintended player manipulation or unauthorized system access.
   - The system should adhere to industry best practices for data encryption, authentication, and access control to safeguard sensitive information.

4. Scalability:
   - The system should be scalable to accommodate a large number of smart enemies and complex gameplay scenarios.
   - Smart enemies' behaviors and decision-making processes should scale efficiently with increasing computational resources, allowing for larger and more challenging encounters.

5. Usability:
   - The system should provide intuitive tools and interfaces for configuring and customizing smart enemy behaviors and attributes.
   - Smart enemies' behaviors should be understandable and predictable for players, allowing them to strategize and adapt their tactics accordingly.
   - The system should support accessibility features such as customizable controls, text-to-speech options, and colorblind-friendly visuals to ensure inclusivity for all players.

6. Maintainability:
   - The system should be modular and well-documented, allowing for easy maintenance and future updates.
   - Developers should be able to extend and modify smart enemy behaviors without extensive rework or refactoring of existing code.
   - The system should support version control and collaboration tools to facilitate teamwork and code management in a development environment.

7. Compatibility:
   - The system should be compatible with a wide range of gaming platforms, operating systems, and hardware configurations.

- Smart enemies' behaviors should adapt seamlessly to different gaming environments, ensuring consistent gameplay experiences across platforms and devices.

3.3 Functional Requirements

Hardware Requirements:

1. Processor (CPU):
   - Minimum: Dual-core processor
   - Recommended: Quad-core processor or higher for optimal performance

2. Memory (RAM):
   - Minimum: 4 GB RAM
   - Recommended: 8 GB RAM or higher for smoother gameplay

3. Graphics Processing Unit (GPU):
   - Minimum: Integrated graphics with DirectX 11 support
   - Recommended: Dedicated GPU with DirectX 12 support for better visual quality and performance

4. Storage:
   - Minimum: 10 GB of available storage space
   - Recommended: Solid State Drive (SSD) for faster loading times and smoother gameplay

5. Input Devices:
   - Keyboard and mouse (or compatible input devices)
   - Game controller (optional but recommended for console-like gaming experience)

6. Display:
   - Minimum: Monitor with a resolution of 1280x720 (720p)
   - Recommended: Monitor with a resolution of 1920x1080 (1080p) or higher for better visual fidelity

Software Requirements:

1. Operating System:
   - Windows 10 (64-bit)
   - macOS Catalina (10.15) or later
   - Ubuntu Linux 20.04 LTS or compatible distribution

2. Game Engine:
   - Unreal Engine 5 for game development

3. Integrated Development Environment (IDE):
- Visual Studio (Windows)
- Xcode (macOS)
- Visual Studio Code with appropriate extensions (Linux)

4. Graphics APIs:
- DirectX 11 or later (Windows)
- Metal (macOS)
- Vulkan (Linux)

5. Additional Software:
- Graphics drivers compatible with the selected GPU
- Audio drivers for sound output

6. Development Tools:
- Git for version control (GitHub)

7. Game Dependencies:
- Libraries and frameworks required for Unreal Engine 5 development (automatically installed with the engine)

4. Feasibility Study

4.1 Operational Feasibility

Operational feasibility assesses whether a system can be effectively implemented and integrated into an organization's existing operations. In the case of the smart enemy's system, operational feasibility can be evaluated based on several factors:
1. Resource Availability: Determine if the necessary resources, including hardware, software, and human capital, are readily available or can be obtained within the project's constraints. Assess the organization's capacity to allocate resources for system development, testing, and deployment.

2. Technical Expertise: Evaluate whether the organization possesses the technical expertise and skills required to develop, implement, and maintain the smart enemies system. Identify any gaps in knowledge or capabilities that may need to be addressed through training or hiring of additional personnel.

3. Compatibility with Existing Systems: Consider how the smart enemies system will integrate with the organization's existing infrastructure, software applications, and workflows. Assess potential compatibility issues and determine the feasibility of implementing necessary interfaces or middleware to facilitate integration.

4. Organizational Impact: Evaluate the potential impact of implementing the smart enemies system on the organization's operations, processes, and stakeholders. Identify any potential disruptions or challenges that may arise during the transition and assess strategies for mitigating risks and managing change.

5. Cost-Benefit Analysis: Conduct a cost-benefit analysis to determine the financial feasibility of implementing the smart enemies system. Estimate the initial investment required for development, deployment, and training, as well as ongoing operational costs such as maintenance and support. Compare these costs with the expected benefits, including improved gameplay experience, player engagement, and revenue generation.

6. Legal and Regulatory Compliance: Ensure that the smart enemies system complies with relevant legal and regulatory requirements, including data privacy, intellectual property rights, and industry standards. Identify any potential legal or regulatory barriers to implementation and assess strategies for mitigating risks and ensuring compliance.

4.2 Technical Feasibility

Technical feasibility evaluates whether the smart enemies system can be successfully developed and implemented from a technical standpoint. Here's how you might assess the technical feasibility of the project:

1. Technology and Tools: Evaluate the availability and suitability of the technology stack required for developing the smart enemies system. Consider factors such as the compatibility of game engines (e.g., Unreal Engine 5), programming languages (e.g., C++), and AI frameworks (e.g., Unreal Engine's AI system) for implementing advanced AI behaviors.

2. Hardware Requirements: Assess whether the hardware resources available meet the system requirements for developing and running the smart enemies system. Consider factors such as processing power, memory, and graphics capabilities necessary for implementing complex AI algorithms and rendering sophisticated game environments.

3. AI Algorithms and Models: Evaluate the feasibility of developing and integrating AI algorithms and models capable of simulating human-like behaviors and interactions for smart enemies. Consider the complexity and computational requirements of implementing behavior trees, blackboards, advanced perception systems, and environment query systems.

4. Integration with Existing Systems: Determine how the smart enemies system will integrate with other components of the game, such as player mechanics, level design, and game logic. Assess the feasibility of implementing interfaces or APIs to facilitate communication and interaction between different system components.

5. Performance and Optimization: Consider the performance implications of implementing smart enemies within the game environment. Evaluate strategies for optimizing AI algorithms, reducing computational overhead, and maintaining a consistent frame rate to ensure smooth and responsive gameplay across different hardware configurations.

6. Scalability: Assess the scalability of the smart enemies system to accommodate varying levels of complexity, including the number of enemies, types of behaviors, and environmental interactions. Consider how the system architecture and design can support scalability without compromising performance or stability.

7. Development Effort and Timeline: Estimate the development effort required to implement the smart enemies system, including design, implementation, testing, and iteration phases. Assess the feasibility of meeting project deadlines and milestones within the available time frame and resource constraints.

8. Technical Risks and Mitigation Strategies: Identify potential technical risks and challenges that may impact the successful implementation of the smart enemies system. Develop mitigation strategies to address these risks, such as conducting prototyping, feasibility studies, and technical proofs of concept to validate key assumptions and dependencies.

4.3 Outline Budget

In this project have no outline budget.

5. System Architecture

In the development of the smart enemies system, we made a deliberate decision not to create formal diagrams, including Class Diagrams of the Proposed System, Entity-Relationship (ER) Diagrams, or High-level Architectural Diagrams. This decision was informed by several factors specific to our project's context and requirements. Firstly, our system's architecture, while integral to the overall functionality of the game, did not exhibit the complexity typically necessitating formal diagrams. With a relatively straightforward structure and well-defined interactions between system components, the need for detailed visual representations was minimized. Additionally, our project's scope and scale were modest, with a focus on implementing AI behavior within a specific gaming environment. Given the project's size, creating and maintaining formal diagrams was deemed an unnecessary overhead that could divert resources and attention away from core development tasks. Furthermore, our team's collaborative approach to development, characterized by frequent communication, code reviews, and agile methodologies, provided ample opportunities for understanding and refining the system's architecture in a dynamic, iterative manner. Through direct engagement with the codebase, discussions, and hands-on development, team members gained a comprehensive understanding of the system's structure and design principles. This practical, hands-on approach to system development not only facilitated rapid iteration and prototyping but also fostered a deeper sense of ownership and collaboration among team members. Ultimately, by prioritizing direct collaboration, clear communication, and agile development practices, we were able to successfully implement the smart enemies system within our project timeline and resource constraints, without the need for formal diagrams.

6. Development Tools and Technologies

6.1 Development Methodology

Implementing Agile methodology in my project, which involves developing the smart enemies system, offers several advantages given the nature of the project. Here's how Agile methodology aligns with my project:

1. Iterative Development: Agile methodology emphasizes iterative development cycles, allowing you to break down the project into smaller, manageable increments called sprints. Each sprint focuses on delivering a subset of functionality, such as implementing specific AI behaviors or enhancing environmental interactions for the smart enemies system.

2. Flexibility and Adaptability: Agile methodologies, such as Scrum or Kanban, prioritize flexibility and adaptability to changing requirements. This is particularly beneficial for my project, where the behavior and interactions of smart enemies may evolve based on gameplay testing, player feedback, or design iterations. Agile allows you to incorporate changes and adjustments quickly, ensuring that the system remains responsive to evolving needs and priorities.

3. Collaborative Approach: Agile encourages close collaboration between developers, designers, stakeholders, and other project stakeholders throughout the development process. This collaborative approach fosters communication, transparency, and shared ownership of the project's goals and outcomes. For my project, collaboration is essential for refining AI behaviors, optimizing gameplay mechanics, and ensuring alignment with the overall vision for the game.

4. Continuous Feedback: Agile methodologies emphasize continuous feedback loops, enabling stakeholders to provide input and guidance at each stage of development. This feedback loop is invaluable for refining AI behaviors, identifying issues or challenges early, and validating design decisions through iterative prototyping and testing. Regular playtesting sessions can gather feedback on the smart enemies' behavior and interactions, informing subsequent iterations and improvements.

5. Adaptive Planning: Agile methodologies prioritize adaptive planning over rigid, upfront planning. This allows you to adjust project priorities, resource allocations, and development strategies based on real-time feedback and changing market conditions. For my project, adaptive planning enables you to prioritize the most critical features or enhancements for the smart enemies system, responding to player preferences, technical constraints, or emerging gameplay trends as they arise.

6. Incremental Delivery: Agile methodologies promote incremental delivery of value to stakeholders, with each sprint delivering tangible results that can be tested, evaluated, and refined. This incremental approach ensures that progress is visible and measurable, providing stakeholders with regular updates on the project's status and trajectory. For my project, incremental delivery allows you to demonstrate the evolution of the smart enemies system over time, building confidence and excitement among players and stakeholders.

6.2 Programming Languages and Tools

For the development of the smart enemies system within the context of my project, you may consider using a combination of programming languages and tools that are well-suited for game development and AI implementation. Here are some options:

1. Programming Languages:
   - Blueprint Visual Scripting (Unreal Engine 5): If you're using Unreal Engine for game development, Blueprint Visual Scripting provides a visual programming interface for creating gameplay mechanics, AI behaviors, and interactions without writing code directly.

2. Game Engines:
   - Unreal Engine 5: Known for its robust features and graphical capabilities, Unreal Engine offers powerful tools for developing AAA-quality games, including advanced AI systems, physics simulations, and rendering technologies.

3. AI Frameworks and Libraries:
   - Unreal Engine AI: Unreal Engine provides built-in support for developing AI behaviors and systems, including behavior trees, blackboards, and perception systems.

4. Development Tools:
   - Visual Studio (or Visual Studio Code): A popular integrated development environment (IDE) for C++ development, Visual Studio offers robust features for code editing, debugging, and performance profiling.
   - Perforce (or Git): Version control systems like Perforce or Git are essential for managing collaborative development on large-scale projects, allowing team members to track changes, merge code branches, and coordinate development efforts effectively.

6.3 Third Party Components and Libraries

1. Graphics and Animation Tools:
   - Blender: A versatile open-source 3D modeling and animation tool, Blender can be used for creating character models, animations, and environmental assets for use in the game.
   - Adobe Photoshop (or GIMP): For 2D art and texture creation, tools like Photoshop or GIMP provide powerful features for designing sprites, textures, and user interface elements.

2. Audio Tools:

- FMOD Studio (or Wwise): Middleware solutions like FMOD Studio or Wwise offer powerful tools for audio implementation, including dynamic sound effects, music integration, and spatial audio for immersive gameplay experiences.

6.4 Algorithms

In the development of the smart enemies system for my game project, various algorithms can be employed to create dynamic and intelligent behaviors for the AI adversaries. Here are some key algorithms commonly used in game development, particularly for implementing AI behaviors:

1. Pathfinding Algorithms:
   - A (A-star): A popular pathfinding algorithm used to determine the shortest path between two points on a graph or grid-based map. A* is widely used in game development for navigation and movement planning of AI characters, allowing them to navigate around obstacles and reach their destinations efficiently.
   - Dijkstra's Algorithm: Similar to A*, Dijkstra's algorithm calculates the shortest path between nodes in a graph but does not consider heuristics. It is commonly used in situations where the cost of movement between nodes is uniform or unknown.

2. Decision-Making Algorithms:
   - Finite State Machines (FSM): A modeling technique used to represent AI behaviors as a set of states and transitions between them. FSMs are suitable for modeling simple, predefined behaviors such as patrolling, attacking, or retreating based on specific conditions or triggers.
   - Behavior Trees: A hierarchical modeling approach that organizes AI behaviors into a tree-like structure, with nodes representing tasks, conditions, and actions. Behavior trees provide flexibility and modularity in defining complex AI behaviors, enabling dynamic decision-making based on environmental stimuli and player interactions.

3. Learning Algorithms:
   - Reinforcement Learning: A machine learning technique where AI agents learn optimal behaviors through trial and error, receiving rewards or penalties based on their actions. Reinforcement learning is well-suited for scenarios where AI adversaries must adapt and improve their strategies over time, such as in adaptive difficulty settings or opponent AI in strategy games.
   - Neural Networks: Deep learning models, such as artificial neural networks, can be trained to recognize patterns, make predictions, or classify inputs based on large datasets. Neural networks can be used in various ways within game AI, such as for character animation, procedural content generation, or behavior prediction.

4. Decision Trees and Rule-Based Systems:
   - Decision Trees: Similar to behavior trees, decision trees represent decision-making processes as a tree-like structure, with branches corresponding to different possible

choices and outcomes. Decision trees are useful for modeling AI decision-making in scenarios with discrete, deterministic outcomes, such as dialogue systems or NPC behavior.

- Rule-Based Systems: Rule-based systems encode AI behavior as a set of logical rules or conditions, with actions triggered based on specific rule activations. Rule-based systems are suitable for defining complex, context-sensitive behaviors in domains where explicit rules can be defined, such as in strategy games or simulation environments.

7. Discussion

The report delves into the existing disparity between offline and online gaming experiences, highlighting the dynamic allure of online multiplayer gaming juxtaposed against the narrative-driven yet often static nature of offline gaming. Despite the immersive narratives and diverse environments offered by offline games, they frequently lack the unpredictable and competitive edge intrinsic to human-controlled adversaries in online multiplayer environments.

Identifying this gap as a significant challenge, the report proposes strategies to infuse offline gaming experiences with the strategic depth, adaptability, and unpredictability characteristic of online multiplayer settings. Key considerations outlined include:

1. Dynamic Adversarial AI: Designing AI systems capable of emulating the strategic decision-making, adaptability, and skill levels of human players in online multiplayer environments.
2. Immersive Environments: Creating rich and immersive game worlds that engage players on multiple levels and provide opportunities for emergent gameplay experiences.
3. Gameplay Mechanics: Developing innovative gameplay mechanics that encourage strategic thinking, adaptability, and player engagement while maintaining a coherent narrative structure.
4. Technological Integration: Leveraging cutting-edge game development technologies such as Unreal Engine 5 to create visually stunning and technically advanced gaming experiences.
5. User Experience: Ensuring that the gaming experience remains intuitive, rewarding, and accessible to a wide range of players while challenging them to improve their skills and strategies over time.

Reference

Font, J.M. (2012) ‘Evolving third-person shooter enemies to optimize player satisfaction in real-time’, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 204–213. Available at: https://doi.org/10.1007/978-3-642-29178-4_21.

Lim, W.S. (2022) *MIMICKING HUMAN-LIKE BATTLE BEHAVIOR OF ENEMIES IN A GAME A Project*.

Martínez Martínez, M. (2023) *Design and implementation of an artificial intelligence for a horror videogame in Unreal Engine 5*.

Purba, K.R. (2016) ‘Optimization of ai tactic in action-RPG game’, in *Lecture Notes in Electrical Engineering*. Springer Verlag, pp. 131–137. Available at: https://doi.org/10.1007/978-981-287-988-2_14.

Schrier, Karen. *et al.* (2008) *Proceedings, Sandbox Symposium 2008 : 3rd ACM SIGGRAPH videogame symposium, Los Angeles, California, August 9-10, 2008*. Association for Computing Machinery.

Spronck, P., Sprinkhuizen-Kuyper, I. and Postma, E. (no date) *ONLINE ADAPTATION OF GAME OPPONENT AI IN SIMULATION AND IN PRACTICE*.