T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

A WAR GAME WITH NEGOTIATION USING LLM-DRIVEN NPC

BORAN KURUT

SUPERVISOR
DR. ÖĞR. ÜYESI YAKUP GENÇ

GEBZE
2025

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

A WAR GAME WITH NEGOTIATION USING LLM-DRIVEN NPC

BORAN KURUT

SUPERVISOR
DR. ÖĞR. ÜYESI YAKUP GENÇ

2025
GEBZE

GRADUATION PROJECT
JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 09/10/2024 by the following jury.

JURY

Member
(Supervisor) : Dr. Öğr. Üyesi Yakup Genç

Member : Prof. Dr. Yusuf Sinan Akgül

ABSTRACT

This project presents the development of a grid-based war game in Unity, integrating an interactive AI-driven enemy commander powered by OpenAI's large language model (LLM). The core gameplay revolves around unit deployment and combat mechanics on a grid-based battlefield. The AI commander enhances the player experience through dynamic, conversational interactions, including negotiating terms of surrender based on its internal decision-making process.

A key feature of the enemy AI is its "aggressiveness value," a variable that adjusts during gameplay to influence the rate and intensity of unit deployments, creating an adaptive challenge for the player. While the AI does not generate specific battlefield strategies or tactics, it uses its LLM capabilities to evaluate the game state and respond contextually in dialogue, enriching the narrative depth and player engagement. This project explores the integration of conversational AI in gaming, emphasizing its potential to create immersive and reactive gameplay experiences.

Keywords: LLM, OpenAI, Unity, Adaptive gameplay, Player-AI negotiation, Interactive dialogue, Game state evaluation.

ÖZET

Bu proje, OpenAI'nin büyük dil modeli (LLM) tarafından desteklenen, interaktif bir yapay zeka düşman komutanının entegre edildiği, Unity üzerinde geliştirilmiş bir kare tabanlı savaş oyununu sunmaktadır. Ana oyun mekaniği, kare tabanlı bir savaş alanında birim yerleştirme ve savaş mekanikleri etrafında şekillenmektedir. Yapay zeka komutanı, karar verme sürecine dayanarak teslim olma koşullarını müzakere etme gibi dinamik ve etkileşimli diyaloglarla oyuncu deneyimini zenginleştirmektedir.

Düşman yapay zekasının temel özelliklerinden biri, "saldırganlık değeri"dir. Bu değişken, oyun sırasında birim yerleştirme hızını ve yoğunluğunu etkileyerek oyuncuya uyum sağlayan bir zorluk sunar. Yapay zeka, belirli savaş stratejileri veya taktikleri üretmese de, LLM yeteneklerini kullanarak oyun durumunu değerlendirir ve bağlamsal diyaloglarla yanıt verir, böylece anlatı derinliği ve oyuncu katılımını artırır. Bu proje, oyunlarda etkileşimli yapay zekanın entegrasyonunu araştırmakta ve sürükleyici ve tepkisel oyun deneyimleri yaratma potansiyelini vurgulamaktadır.

Anahtar Kelimeler: Büyük Dil Modeli, OpenAI, Unity, Uyarlanabilir oyun deneyimi, Oyuncu-Yapay Zeka müzakeresi, Etkileşimli diyalog, Oyun durumu değerlendirmesi

ACKNOWLEDGEMENT

I sincerely thank Yakup Genç for his guidance and support throughout this project. I would also like to thank Erva Aksu, Berkay Çufadar, and Buğra Arslan for providing valuable feedback.

Boran Kurut

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
LLM	:	Large Language Model
NPC	:	Non Playable Character

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. STRUCTURE OF THE GAME

The structure of the grid-based strategy war game is designed to provide a seamless and engaging experience while integrating dynamic interactions with an LLM-driven NPC enemy commander.

1.1. Grid Based Map

The battlefield is represented as a grid-based map, where both the player and the enemy commander can deploy their units on valid areas. 1.1.

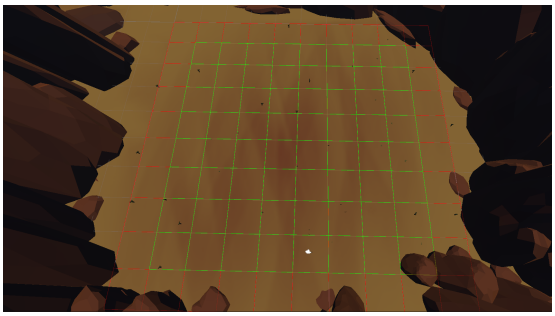

Figure 1.1: Grid Based Map

1.2. Player Mechanics

The player assumes the role of a commander with the ability to interact with the game world and the enemy commander.

- The player can place soldiers, tanks, and throw airstrikes onto the map.
- The player can engage in dialogue with the enemy commander to negotiate surrender or to provoke them.

1.3. Enemy Commander

The enemy commander is powered by a large language model (LLM) and plays a pivotal role in the game.

- The enemy commander interacts with the player through contextual dialogue, responding to game state changes and player actions.
- The NPC adjusts its ”aggressiveness” and ”surrender likelihood” values based on the evolving game state. (If player provokes NPC, aggressiveness increase and if the battle is going against NPC, surrender likelihood increase.)
- The NPC exhibits more aggressive dialogue when its aggressiveness value is high.

1.4. Unit Types

The game features three primary unit types, each with distinct roles and abilities:

- **Soldiers**: Infantry units equipped with rifles that deal direct damage to targeted enemies.
- **Tanks**: Tank units capable of delivering area damage around a targeted enemy.
- **Airstrikes**: Aerial projectiles that inflict area damage at a specified location, offering strategic advantages in eliminating clusters of enemies.

1.5. Victory Conditions

The game ends when one of the following conditions is met:

- **Player Victory**: Achieved by either forcing the enemy commander to surrender or eliminating all enemy units.
- **Enemy Victory**: The player loses if their units are obliterated.

1.6. Level Design Window

The level design window enables players to customize game settings, providing flexibility and variety for gameplay scenarios.

1.6.1. Interface Overview

The interface includes:

- **Unit Configuration**: Input fields for setting player and enemy soldiers, tanks, and airstrikes.

- **Aggressiveness Slider**: Adjusts the enemy's behavior dynamically.
- **Language Button**: Toggles between English and Turkish for NPC dialogue.
- **Start Button**: Launches the game with the selected settings.

1.6.2. Impact on Gameplay

The level design window allows players to:

- Customize difficulty by modifying unit counts and aggressiveness.
- Tailor gameplay scenarios to test strategies and configurations.

1.6.3. Example Configuration

A sample setup could include 30 soldiers, 10 tanks, and 5 airstrikes for the player; 40 soldiers, 8 tanks, and 7 airstrikes for the enemy; and an aggressiveness value of 3. 1.2

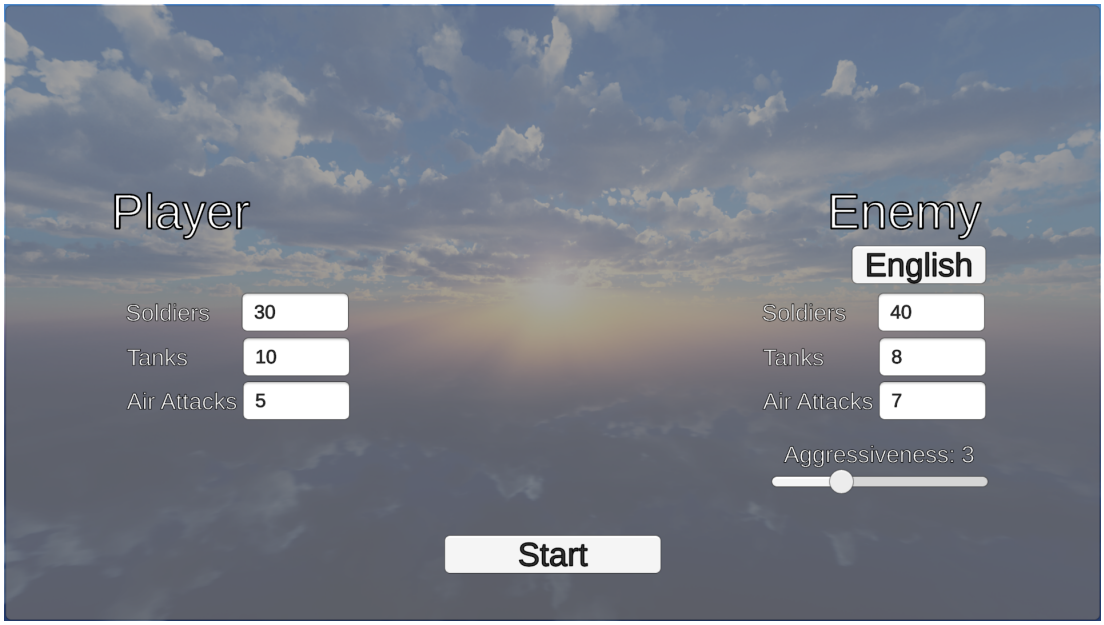

Figure 1.2: Level Design Window

2. ARMY IMPLEMENTATION

The army system in the game is implemented to handle various unit types, manage their attributes, and provide dynamic behaviors during gameplay. The implementation is structured around the concepts of army information, unit attributes, and unit controllers. Each unit type is uniquely designed to fulfill specific battlefield roles, categorized as Soldiers, Tanks, and Airstrikes.

2.1. Army Information

The `Army Information` mechanism manages the overall state of the army. It tracks the following attributes:

- **Initial Army**: The starting composition of soldiers, tanks, and airstrikes.
- **At Hand**: Units that are ready for deployment but not yet on the battlefield.
- **At Battlefield**: Units currently deployed and engaged in combat.
- **Current Total**: The combined count of units across all states, calculated dynamically.

This structure enables efficient resource tracking and provides insights into the player's and enemy's available forces during the game.

2.2. Soldiers

Soldiers are the most basic units, designed for direct combat. Their implementation involves:

- **Attributes**: Managed using the `Army Member` system, soldiers have health and damage values. These attributes define their combat effectiveness.
- **Controller**: The `Soldier Controller` mechanism governs the behavior of soldier units, including:
  - Movement towards the closest enemy.
  - Shooting animations and effects using particle systems.
  - Health reduction upon taking damage.
  - Death handling, which decreases the soldier count in the battlefield and cleans up the game object.

2.3. Tanks

Tanks are armored units with higher durability and the ability to deal area-of-effect damage. Their implementation includes:

- **Attributes**: Tanks possess higher health and damage values compared to soldiers, making them suitable for breaking enemy lines.
- **Projectile System**: Manages the behavior of all projectiles.
  - Area damage calculations based on a specified radius.
  - Particle effects for explosions and impacts.
  - Collision detection to determine affected units.
- **Controller**: The `Tank Controller` mechanism manages tank-specific behaviors:
  - Movement towards enemy targets.
  - Turret rotation for precise aiming.
  - Firing projectiles at enemy units.
  - Death handling, Reduces the tank count in the battlefield and triggers visual effects, such as smoke particles, upon destruction.

2.4. Airstrikes

Airstrikes are powerful one-time-use projectiles that target specific locations. Their implementation focuses on high-impact, strategic gameplay:

- **Attributes**: Airstrikes are defined by their damage and area-of-effect radius, as they do not have health or persistent presence on the battlefield.
- **Behavior**: Airstrikes are deployed at a specific location and detonate on impact, dealing damage to all units within their effective radius.
- **Controller**: The `projectile system` is used to handle airstrike mechanics, ensuring precise targeting and explosion effects.
- **Particle Effects**: Explosion and impact visuals are enhanced using particle systems, providing clear feedback to the player.

2.5. Army Member Controllers

Army controllers play a vital role in managing unit behavior and interactions. Each unit type has its own controller that handles state transitions, such as:

- **Idle**: Units remain stationary when no enemy is in range.
- **Moving**: Units navigate towards the closest enemy target.
- **Shooting**: Units attack the enemy when within range, applying damage.
- **Dead**: Units transition to a death state, triggering cleanup operations and reducing the battlefield count.

This hierarchical and modular implementation ensures that the game can manage diverse units effectively while maintaining scalable and dynamic gameplay.

3. AI IMPLEMENTATION

The AI system in the game is responsible for controlling the enemy commander, dynamically interacting with the player, and managing the deployment of enemy units. The implementation leverages several key components, including GPT-based interaction, attack patterns, and automated unit deployment.

3.1. GPT-Driven Chat Facility

The game's interactive dialogue system is powered by the OpenAI GPT model [1]. The `GPT` manages conversations with the player through the following features:

- **Dynamic Dialogue**: The AI commander generates contextual responses based on the state of the game, including battlefield updates and player interactions.
- **Characteristics Management**: The AI adjusts its behavior, such as `anger` and `surrender likelihood`, based on the game state. These characteristics influence the tone of the AI's responses.
- **Secret Prompt System**: The AI is periodically updated using secret prompts (e.g., `tatata`) to integrate battlefield information and adapt its strategies accordingly.
- **Pre Configuration**: The AI is configured with a set of rules about the game and a sample conversation.

This system ensures the AI provides engaging and reactive dialogue, enhancing the player's experience.

3.2. GPT Informer

The `GPT Informer` is a critical component for managing and relaying information to the AI. Its primary functions include:

- **Army Information Updates**: The `GPT Informer` consolidates and communicates the current state of both the enemy and player armies, including units at hand, deployed units, and overall status.
- **Characteristics Updates**: The AI's characteristics, such as aggressiveness and surrender likelihood, are updated dynamically using this module.

- **Battlefield Awareness**: By providing periodic updates, the `GPT Informer` ensures the AI remains aware of the game's current state and can make informed decisions.

3.3. Attack Pattern Management

The `Attack Pattern` structure defines the AI's unit deployment behavior based on its aggressiveness. Key features include:

- **Attack Timing**: Separate attack periods are defined for soldiers, tanks, and airstrikes. These are dynamically adjusted based on the AI's aggressiveness value.
- **Aggressiveness Scaling**: The AI's `aggressiveness` is scaled using a divider to ensure balanced gameplay, with values clamped between 1 and 10.
- **Adaptive Deployment**: As the AI's aggressiveness increases, its deployment periods shorten, resulting in faster and more frequent attacks.

This system enables the AI to adapt its behavior dynamically, creating a challenging and responsive opponent.

3.4. Enemy Unit Sending

The `Enemy Unit Sending` implementation includes:

- **Automated Deployment**: Separate routines are defined for deploying soldiers, tanks, and airstrikes. These routines operate on timers based on the `Attack Pattern` of the enemy.
- **Strategic Placement**: Units are placed at valid grid positions within the enemy territory. For airstrikes, the AI targets random positions within the player's army.
- **Game State Integration**: The deployment routines adapt to the current state of the game. For example, airstrikes are only deployed when the player's army exceeds a certain threshold.
- **Dynamic Aggressiveness**: The AI's deployment speed is influenced by its `aggressiveness`.

3.5. Integration and Coordination

The AI system integrates all components to provide a cohesive experience:

- **Information Flow**: The `GPT Informer` ensures the AI has up-to-date battlefield and characteristics information.
- **Reactive Behavior**: The AI's dialogue and deployment strategies adapt dynamically based on inputs from the `Chat Facility` and `Enemy Unit Sending`.
- **Scalability**: The modular design of components like `Attack Pattern` and `Enemy Unit Sending` allows for future enhancements or adjustments to AI behavior.

This AI implementation ensures that the enemy commander is narratively engaging character with interaction and decision-making capabilities.

4. PLAYER MECHANICS

The player mechanics are designed to offer an interactive and engaging experience, allowing the player to deploy units strategically and navigate the battlefield efficiently. This chapter explains the implementation of player mechanics, focusing on unit placement, selection, and camera control.

4.1. Unit Placement

The player can deploy soldiers, tanks, and airstrikes by interacting with the battlefield. Key features include:

- **Mouse Interaction**: The player clicks on valid positions on the battlefield to deploy units. A ray-cast detects if the click intersects a valid grid location.
- **Unit Readiness**: Before deployment, the system checks whether the selected unit type is ready and if sufficient units are available in the player's army inventory.
- **Deployment Logic**: The instantiation and positioning of units are handled based on the player's selection.

This system ensures that the player's unit placement adheres to game rules and strategic limitations.

4.2. Unit Selection

The player can switch between unit types for deployment using keyboard inputs.

- **Key Bindings**:
  - `1`: Selects `Soldier`.
  - `2`: Selects `Tank`.
  - `3`: Selects `Airstrike`.
- **Validation**: Selection impacts the deployment process, ensuring the player can only deploy the selected unit type if conditions are met.

4.3. Camera Control

The player has the ability to navigate and observe the battlefield. Key functionalities include:

- **Movement**: The player can move the camera using the keyboard:
  - `W`, `A`, `S`, `D`: Move forward, left, backward, and right, respectively.
  - `E`: Move up.
  - `Q`: Move down.
- **Rotation**: By holding the right mouse button, the player can rotate the camera using mouse movements.
- **Bounds Enforcement**: The camera position is clamped within predefined bounds to prevent the player from leaving the battlefield area.
- **Dynamic Grid Visibility**: The system toggles the visibility of player and enemy side indicators based on camera movement.

4.4. Integration with Battlefield State

The player's mechanics are tightly integrated with the game state:

- The `attack pattern` of the enemy NPC is synchronized with the player's deployment logic, maintaining balanced gameplay.
- The system dynamically updates the readiness status and stacks for soldiers, tanks, and airstrikes, ensuring real-time feedback on unit availability.

This comprehensive implementation allows the player to strategize effectively, engage with the battlefield dynamically, and make informed decisions during gameplay.

5. EVALUATION

This project implemented a grid-based strategy war game with dynamic and interactive gameplay elements powered by an LLM. The integration of the LLM provided an enemy NPC to engage in contextual dialogue and make informed surrender decisions.

5.1. Summary of Achievements

The development of the game included several notable accomplishments:

- A robust grid-based battlefield where both the player and the NPC can deploy units strategically.
- An LLM-powered enemy commander capable of interacting dynamically with the player and adjusting its surrender likelihood and aggressiveness based on the game state.
- The implementation of diverse unit types, including soldiers, tanks, and airstrikes, with distinct behaviors and battlefield roles.
- Seamless integration of player mechanics, allowing unit deployment, selection, and battlefield navigation.
- Real-time AI adjustments to game dynamics through the `Attack Pattern`, `GPT Informer`, and chat system, ensuring a responsive and adaptive opponent.
- The game runs at an average of 115 fps on a system equipped with an RTX 3060 GPU and a Ryzen 5 processor.
- The average response time for NPC interactions is approximately 2.5 seconds.

5.2. Player Feedback

To evaluate the effectiveness of the game's AI and overall player experience, 10 participants were asked to play the game and provide feedback. The average responses to key evaluation questions were as follows:

- **Accuracy of NPC surrender decisions**: The NPC's ability to make accurate surrender decisions was rated at **3.8** out of 5, indicating a reasonable alignment with player expectations but with room for improvement.

- **Relevance of NPC responses**: The NPC's dialogue relevance was rated at **4.1** out of 5, reflecting a strong connection between its responses and the game context.

These results highlight the strengths of the AI in creating a believable and engaging opponent while also identifying areas for future enhancement.

5.3. Challenges and Limitations

The project faced several challenges and limitations:

- The accuracy of surrender decisions, while reasonable, could be further refined by improving the underlying decision-making logic and incorporating additional battlefield parameters.
- The use of OpenAI's API introduces a dependency on an external, priced platform. This not only increases the operational cost of the project but also adds latency to response due to communication with a remote server.

5.4. Future Work

Building on the current implementation, several areas for future improvement have been identified:

- Enhancing the decision-making process for NPC surrender likelihood by incorporating more sophisticated game state analysis.
- Expanding the diversity of unit types and their behaviors to add strategic depth.
- Refining the LLM's conversational logic to further improve dialogue relevance and player engagement.
- Conducting larger-scale play-testing to gather more comprehensive feedback and fine-tune the AI's performance.

6. CONCLUSION

This project demonstrates the potential of integrating advanced AI into strategy games, offering both strategic challenges and narrative depth. The implementation of a grid-based war game with an LLM-driven NPC commander provides a unique gaming experience, balancing gameplay mechanics with dynamic and engaging AI behavior. With further refinements and expansions, this system can serve as a foundation for more complex and immersive strategy games in the future.

BIBLIOGRAPHY

[1] OkGoDoIt, *Openai-api-dotnet: .net bindings for openai's api, version 1.6*, https://github.com/OkGoDoIt/OpenAI-API-dotnet/releases/tag/v1.6, Accessed: 2025-01-12, 2025.