Ankara Yıldırım Beyazıt University
Department of Computer Engineering

CENG 201 – Object Oriented Programming
Course Project

G10: Mini Yu-Gi-Oh! Game

Analysis Report

Emre Demirbaş, 21050111069
Mehmet Ali Yılmaz, 21050111057
Ömer Faruk Özüyağlı, 21050111047
Muhammed Esat Çelebi, 21050111034

Instructor: Muhammed Abdullah Bülbül
Teaching Assistant: Elif Şanlıalp
Date: 01/12/2023

1. Table of Contents

1. Introduction

In this project, we are working on a Mini Yu-Gi-Oh! video game using object-oriented programming features. The goal is to transform the well-known Yu-Gi-Oh! board game into a digital version, includes important features like taking turns, using different types of cards such as spells, monsters, and traps, and going through specific stages like Draw, Standby, Main, Battle, and End. Players can pick characters, each having a unique set of cards, and strategize to win by achieving objectives like collecting all pieces of Exodia, reducing opponent's life points, or using up their cards. This report provides a detailed overview of our project and introduces the game's mechanics.

2. Requirements

2.1. Functional Requirements

- **Game Initialization:**
    - ○ The system must allow players to initialize a new game.
    - ○ Players must be able to select one of the characters at the beginning of the game.

- **Turn-Based Gameplay:**
    - ○ The system must implement a turn-based structure, allowing players to take turns.
    - ○ Each turn consist some phases such as Draw Phase, Standby Phase, Battle Phase, etc.
    - ○ Players must be able to draw a card at the start of their turn.

- **Card Management:**
  - Players must be able to draw cards from their decks.
  - Players must be able to play cards from their hands onto the playing field.

- **Monster Card Actions:**
  - Players must be able to summon monsters in attack position or defense position to the playing field.
  - Monsters must be able to attack other monsters.
- **Spell and Trap Card Actions:**
  - Players must be able to play spell and trap cards with various effects.
  - Spell and trap cards must resolve their effects based on game rules.

- **Life Points Management:**
  - The system must track and display the life points of each player.
  - Players lose the game if their life points reach zero.

- **Winning Conditions:**
  - The game must have predefined winning conditions, such as reducing the opponent's life points to zero.

- **Card Effects:**
  - Cards with special effects can perform their actions according to the game rules.

- **Game State Management:**
  - The system must manage and update the overall game state, including phases and current player actions.

2.2. Non-Functional Requirements

- **Usability:**
  - The user interface should be intuitive, allowing players to easily understand and execute game actions.

- **Performance:**
  - The game must respond to user actions quickly, such as drawing cards and playing cards, to provide the user a smooth gaming experience.

- **Supportability:**
  - It should be possible to add new monster cards, spell cards, trap cards and characters to the game.

- **Maintainability:**
  - The codebase should follow coding standards and best practices to facilitate easy maintenance and future development.

3. System Models

3.1. Scenarios

Yu-Gi-Oh Duelist’s Challenge: Emre vs Mehmet Ali:

- **Getting into the Game:**
  - Emre and Mehmet Ali log in to the Yu-Gi-Oh game, excited to start a strategic duel. They go through the main menu, choosing their characters. Emre picks Yugi Muto, and Mehmet Ali selects Seto Kaiba.

- **Draw Phase - Emre's Turn:**
  - The duel starts, and it’s Emre’s turn. A window appears, asking him to draw five cards. The cards are face down, creating anticipation.

- **Strategic Decision-Making:**
  - Emre looks at his drawn cards, thinking about the best moves for the upcoming turns. His hand has monsters, spells and traps, giving him different strategic choices.

- **Card Placement:**
  - Emre puts a monster card face-down in defense position and adds a spell card to boost his defense. He keeps some cards for possible counter attacks in the Battle Phase.

- **Mehmet Ali's Turn - Draw Phase:**
  - As Emre finishes placing his cards, the game shifts to Mehmet Ali's Draw Phase. A dialogue window prompts him to draw cards, and the cards are presented face down, maintaining the suspense.

- **Card Selection and Consideration:**
  - Mehmet Ali carefully assesses the drawn cards, weighing the potential advantages of each. He selects a monster card, positioning it on his side of the stage in attack position, indicating a potential offensive strategy.

- **Strategic Planning:**
  - With his remaining cards, Mehmet Ali thoughtfully places a spell card, enhancing his resources for the turns to come. Additionally, he strategically sets a trap card, preparing for possible defensive maneuvers against Emre’s actions.

- **Card Placement:**
  - As Mehmet Ali concludes his Draw Phase, the duel progresses to the Main Phase, where the strategic decisions made during the Draw Phase set the stage for the unfolding gameplay. The game maintains an immersive and thoughtful experience, capturing the essence of strategic gameplay in the world of Yu-Gi-Oh.

- **Winning Scenario 1: Life Point Victory**
  - ○ After a consecutive draw, standby and battle phases, life points of Emre run out. Therefore, Mehmet Ali wins the duel.

- **Winning Scenario 2: Exodia Victory:**
  - ○ If Emre collects all the pieces of Exodia, which are separate six monster cards, he wins. It's a tough but thrilling way to win the game.

- **Winning Scenario 3: Out of Cards Victory:**
  - ○ If Emre makes Mehmet Ali run out of cards, Emre wins. This victory involves smart card management throughout the game.

3.2. Use Cases

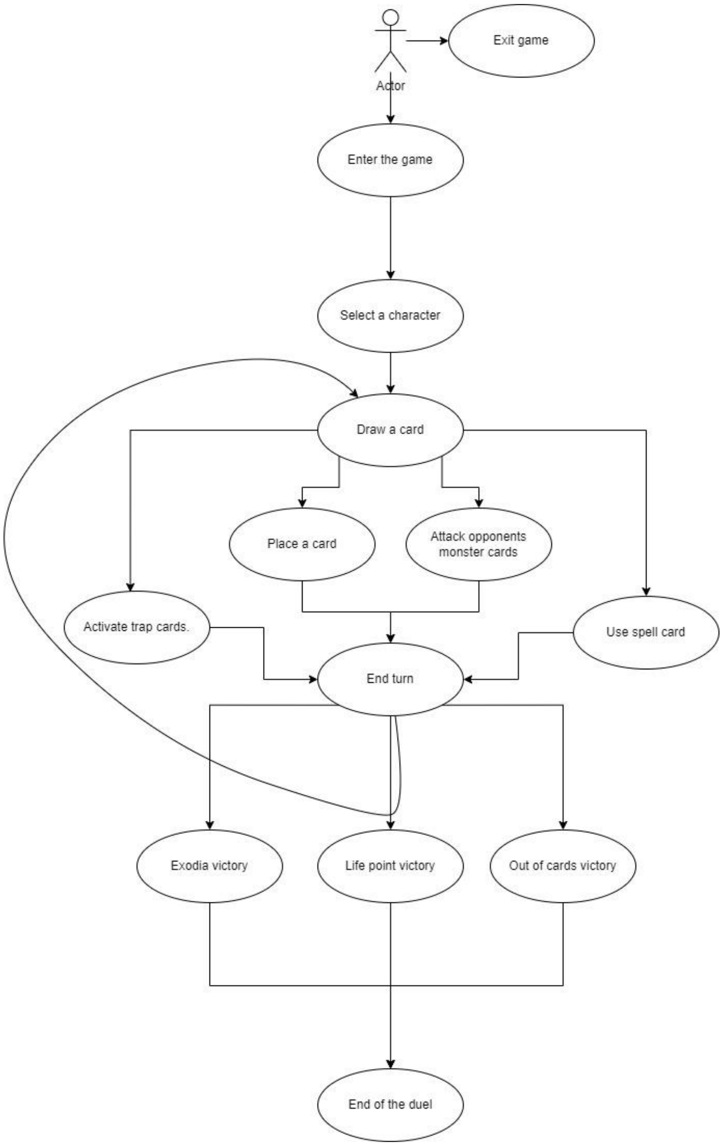

Figure 1. UML Use-Case Diagram

• Users

- Emre: The primary player initiating the duel, representing the user who chooses Yugi Muto.
- Mehmet Ali: The opponent in the duel, representing the user who selects Seto Kaiba.

• Use Case 1 Log In

- Actor: Emre, Mehmet Ali
- Description: Both Emre and Mehmet Ali enter to the Yu-Gi-Oh game, navigating through the main menu to initiate the duel.

• Use Case 2: Choose Characters

- Actor: Emre, Mehmet Ali
- Description: Both players select their characters, with Emre choosing Yugi Muto and Mehmet Ali selecting Seto Kaiba.

Figure 2. Yugi and Kaiba

• Use Case 3: Draw Cards
- Actor: Emre, Mehmet Ali
- Description: Both players, in their own turns, draws one card. The system presents the facedown cards, creating anticipation for the upcoming moves.

Exodia the Forbidden One
[Spellcaster/Effect]
If you have "Right Leg of the Forbidden One", "Left Leg of the Forbidden One", "Right Arm of the Forbidden One" and "Left Arm of the Forbidden One" in addition to this card in your hand, you win the Duel.
ATK/1000 DEF/1000
Blue-Eyes White Dragon
[Dragon/Normal]
This legendary dragon is a powerful engine of destruction. Virtually invincible, very few have faced this awesome creature and lived to tell the tale.
ATK/3000 DEF/2500
Monster Reborn
[Spell Card]
Target 1 monster in either GY; Special Summon it.
Mirror Force
[Trap Card]
When an opponent's monster declares an attack: Destroy all Attack Position monsters your opponent controls.

Figure 3. Some sample cards from the game

• Use Case 4: Card Placement
- Actor: Emre, Mehmet Ali
- Description: Both players strategically place a face-down monster card in defense position and optionally adds a spell card to boost their defense or keeping cards for potential counterattacks.

• Use Case 5: Duel Progression to Main Phase
- Actor: Emre, Mehmet Ali
- Description: The duel progresses to the Main Phase, where strategic decisions like attacking opponents monster cards or activating spell cards.

• Use Case 6: Life Point Victory
- Actor: Emre
- Description: Emre wins after consecutive draw, standby, and battle phases as the life points of Mehmet Ali run out.

• Use Case 7: Exodia Victory
- Actor: Mehmet Ali
- Description: Mehmet Ali wins if he collects all six pieces of Exodia. This represents a challenging but thrilling way to secure victory.

• Use Case 8: Out of Cards Victory
- Actor: Emre
- Description: Emre wins if he makes Mehmet Ali run out of cards, demonstrating smart card management throughout the game.

3.3. Object and Class Model

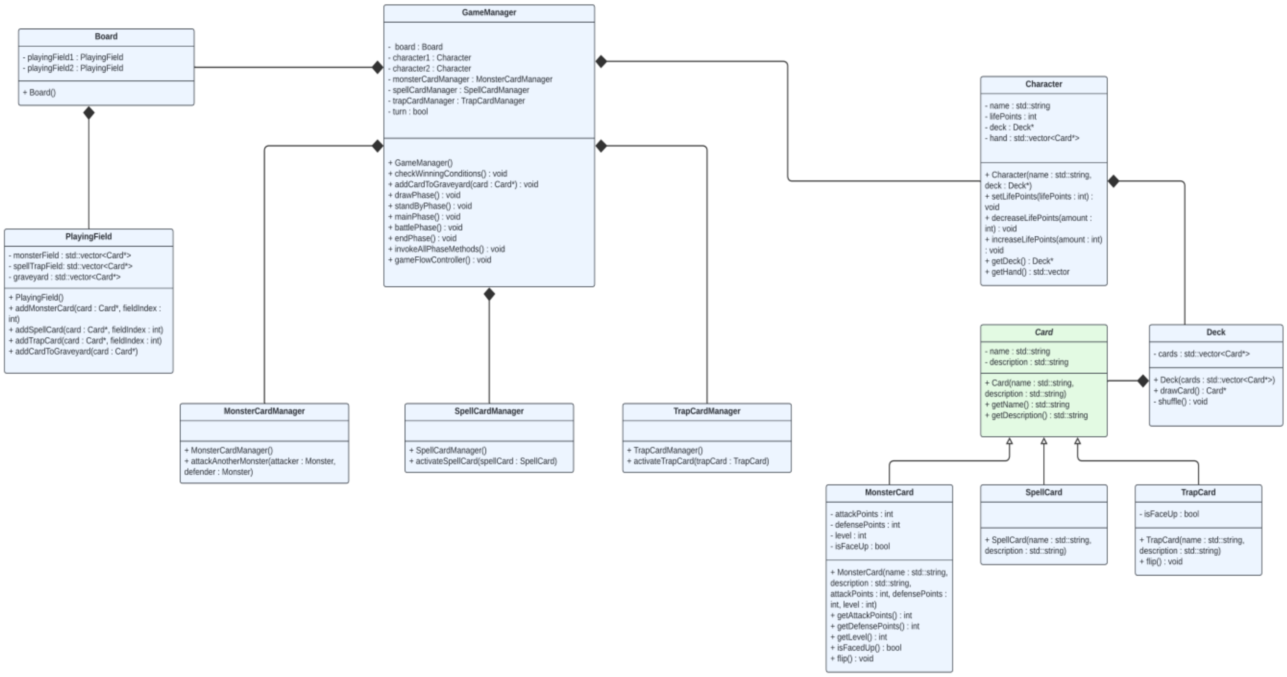
Board
- playingField1 : PlayingField
- playingField2 : PlayingField
+ Board()
GameManager
- board : Board
- character1 : Character
- character2 : Character
- monsterCardManager : MonsterCardManager
- spellCardManager : SpellCardManager
- trapCardManager : TrapCardManager
- turn : bool
+ GameManager()
+ checkWinningConditions() : void
+ addCardToGraveyard(card : Card*) : void
+ drawPhase() : void
+ standByPhase() : void
+ mainPhase() : void
+ battlePhase() : void
+ endPhase() : void
+ invokeAllPhaseMethods() : void
+ gameFlowController() : void
Character
- name : std::string
- lifePoints : int
- deck : Deck*
- hand : std::vector<Card*>
+ Character(name : std::string, deck : Deck*)
+ setLifePoints(lifePoints : int) : void
+ decreaseLifePoints(amount : int) : void
+ increaseLifePoints(amount : int) : void
+ getDeck() : Deck*
+ getHand() : std::vector
PlayingField
- monsterField : std::vector<Card*>
- spellTrapField: std::vector<Card*>
- graveyard : std::vector<Card*>
+ PlayingField()
+ addMonsterCard(card : Card*, fieldIndex : int)
+ addSpellCard(card : Card*, fieldIndex : int)
+ addTrapCard(card : Card*, fieldIndex : int)
+ addCardToGraveyard(card : Card*)
Card
- name : std::string
- description : std::string
+ Card(name : std::string, description : std::string)
+ getName() : std::string
+ getDescription() : std::string
Deck
- cards : std::vector<Card*>
+ Deck(cards : std::vector<Card*>)
+ drawCard() : Card*
- shuffle() : void
MonsterCardManager
+ MonsterCardManager()
+ attackAnotherMonster(attacker : Monster, defender : Monster)
SpellCardManager
+ SpellCardManager()
+ activateSpellCard(spellCard : SpellCard)
TrapCardManager
+ TrapCardManager()
+ activateTrapCard(trapCard : TrapCard)
MonsterCard
- attackPoints : int
- defensePoints : int
- level : int
- isFaceUp : bool
+ MonsterCard(name : std::string, description : std::string, attackPoints : int, defensePoints : int, level : int)
+ getAttackPoints() : int
+ getDefensePoints() : int
+ getLevel() : int
+ isFacedUp() : bool
+ flip() : void
SpellCard
+ SpellCard(name : std::string, description : std::string)
TrapCard
- isFaceUp : bool
+ TrapCard(name : std::string, description : std::string)
+ flip() : void

Figure 4. UML Class Diagram

3.4. User Interfaces

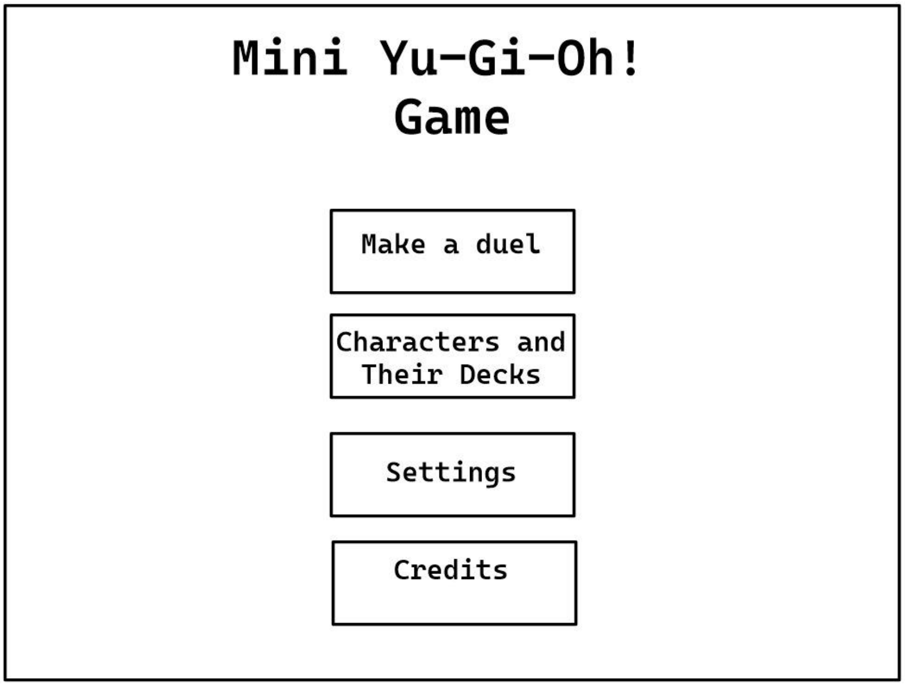

Figure 5. Main Menu

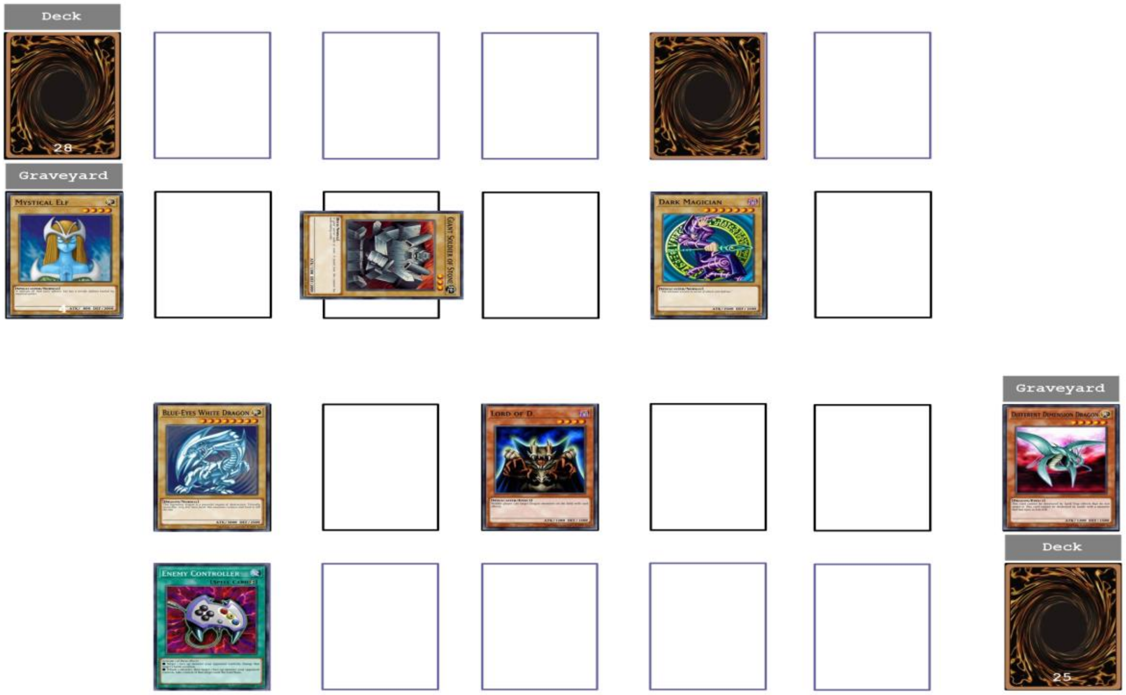

Figure 6. Game Board

4. Conclusion

In this report, we explored the details of creating a Yu-Gi-Oh! video game using object-oriented programming properties. We described different situations in the game, for instance turns and card types such as spell, monster, and trap cards. The report also explained the phases of the game, Draw, Standby, Main, Battle, and End Phase. Each player, represented by characters like Yugi Muto and Seto Kaiba, has their own deck, and winning involves assembling the Exodia, reducing life points, or running out of cards. We outlined the practical needs, like starting the game and playing turns, and other aspects like usability, performance, and supportability. The report also discussed how the games' interface will look and feel.

All team members collaborated as partners while working on the report. In every section, members shared their opinions. But if we should state the topics individually,

Introduction: Mehmet Ali, Ömer Faruk
Functional & Non-Functional Requirements: All of the members
Scenarios: Emre, Ömer Faruk
Use Cases: Ömer Faruk, Emre, Mehmet Ali
Object & Class Model: All of the members
User Interfaces: Emre, Esat
Conclusion: Mehmet Ali, Ömer Faruk

References

https://ygoprodeck.com/
https://yugioh.fandom.com/wiki/Yu-Gi-Oh!_Wiki
https://yugipedia.com/wiki/Yugipedia