UML diagrams : Code quest

An Educative Coding Game

Prepared by: Mohamed Amine El Bacha and Younes menfalouti

October 22, 2025

Contents

Chapter 1

Use Case Diagram

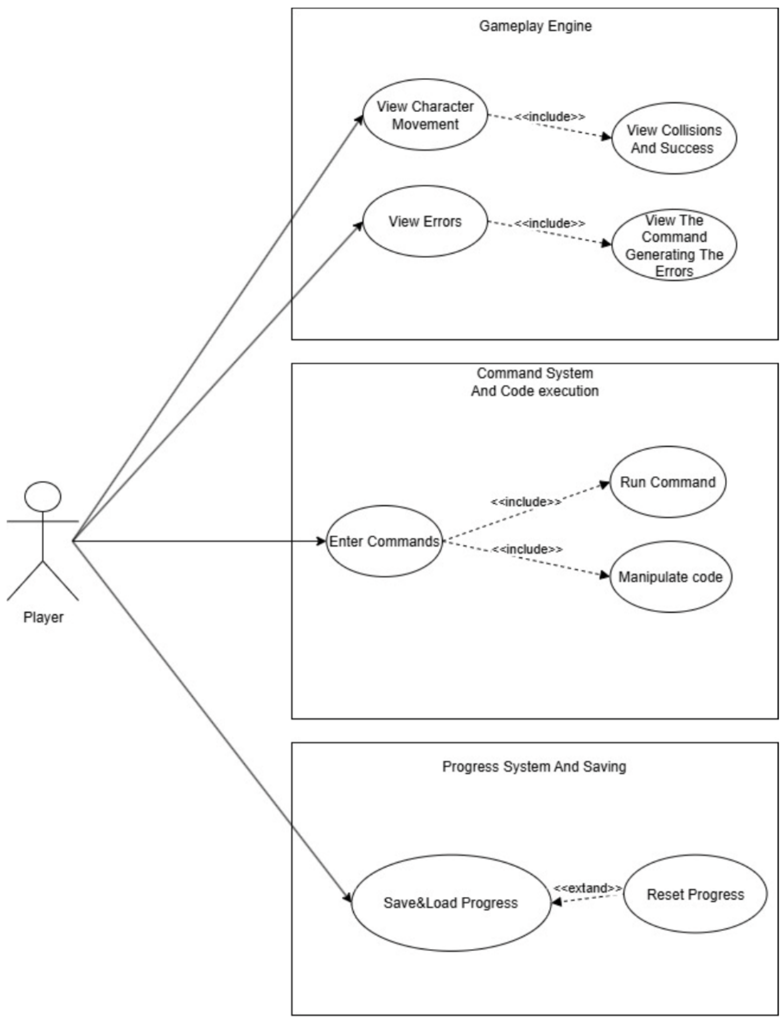

Figure 1.1: Use Case Diagram for the Code Quest Project

Description: The Use Case Diagram represents the primary interactions between the player and the Code Quest system. It identifies the main actors:

- **Player** – Interacts with the game by entering commands and viewing results.
- **System** – Executes code, manages level transitions, and provides feedback.

The player can:

- Start a new game or load saved progress.
- Input code commands that control the in-game character.
- View feedback and results for executed actions.

The system, in turn, parses commands, executes logic, updates player position, and checks win conditions.

Chapter 2

Class Diagram

The UML Class Diagram provides a structural view of the system's architecture. It defines how the main classes are related and how data flows between them.

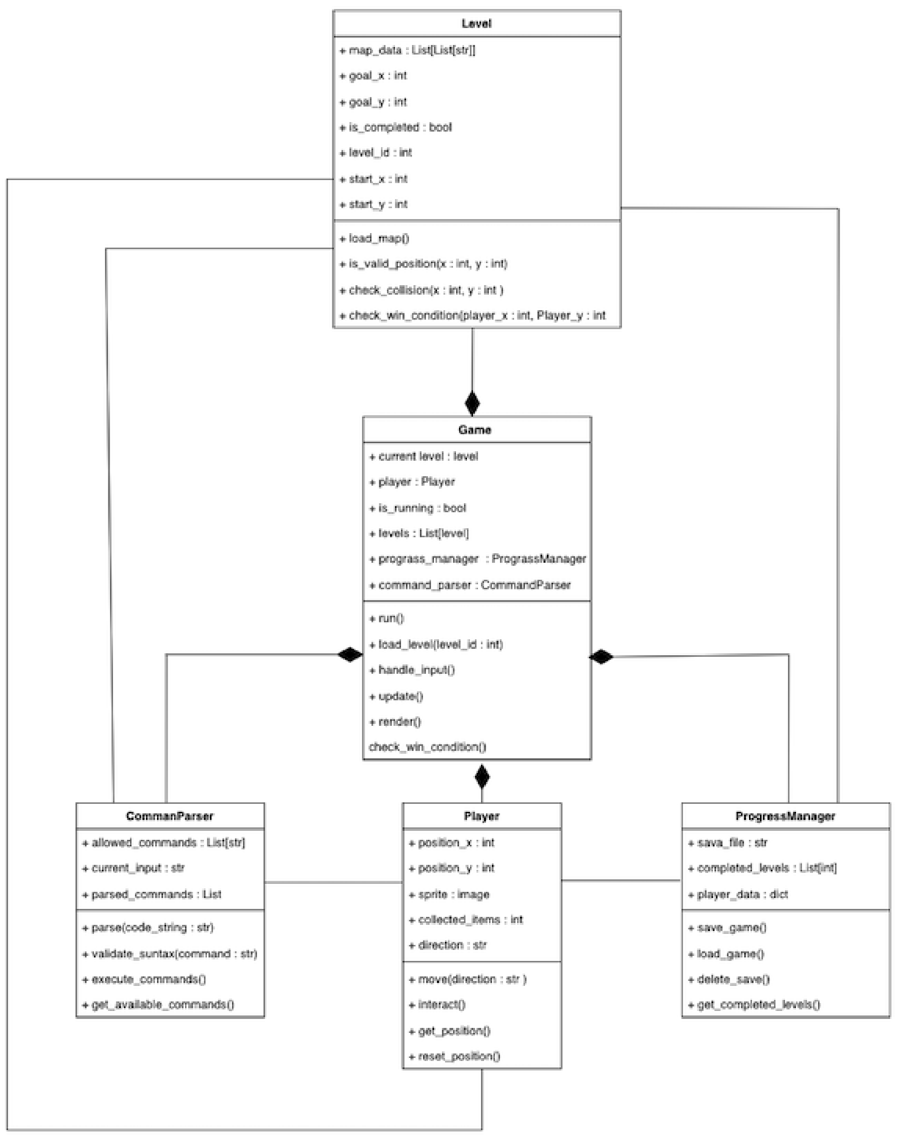

Figure 2.1: UML Class Diagram for the Code Quest Project

Chapter 3

Explanation of the Class Diagram

The class diagram defines the main objects and their responsibilities within the game:

- **Game**: Controls the overall game loop, manages the player, level transitions, and coordinates between modules.
- **Level**: Represents each playable environment; handles loading, validation, and collision detection.
- **Player**: Stores player state and behavior, including movement and interaction logic.
- **CommandParser**: Interprets and validates player code input, executing valid commands.
- **ProgressManager**: Handles saving and loading of player progress, including completed levels and session data.

Interactions between classes:

- The `Game` class coordinates all interactions between components.
- The `Player` interacts with the `Level` to detect collisions and reach goals.
- The `CommandParser` interprets code input and directs the player's actions.
- The `ProgressManager` ensures persistent storage of progress and achievements.

Together, these classes ensure a clean separation of concerns and support scalability for new features and levels.

Chapter 4

Conclusion

This report presents a simplified architectural overview of the Code Quest project. By analyzing the Use Case and Class Diagrams, we gain insight into how player interaction, command parsing, and progress tracking work together. The modularity of the design supports easy maintenance, extension, and educational use in programming instruction.