Object-Oriented Programming Project

Roguelike Game 'Bong's Dungeon'

20202865 윤 수 용

2023. 12

Contents

Contents

- ❖Project Introduction
  - Game Introduction
  - Development Basis
  - Game Structure
- ❖Object-Orient Implementation
  - Object-Orient Feature Implementation
  - Use Case Diagram
  - Class Diagram
  - Sequence Diagram
  - Activity Diagram
- ❖Project Result
  - Demo Video

Project Introduction

Game Introduction: Bong's Dungeon

- Control a player to explore the map and battle enemies
- Collect various items on the map to solve battles
- Earn experience by defeating enemies: collect a certain amount to power up your player.
- Descend to the next level when a map is fully explored.
- Ends the game when the player's health is depleted and falls below 0.

Project Introduction

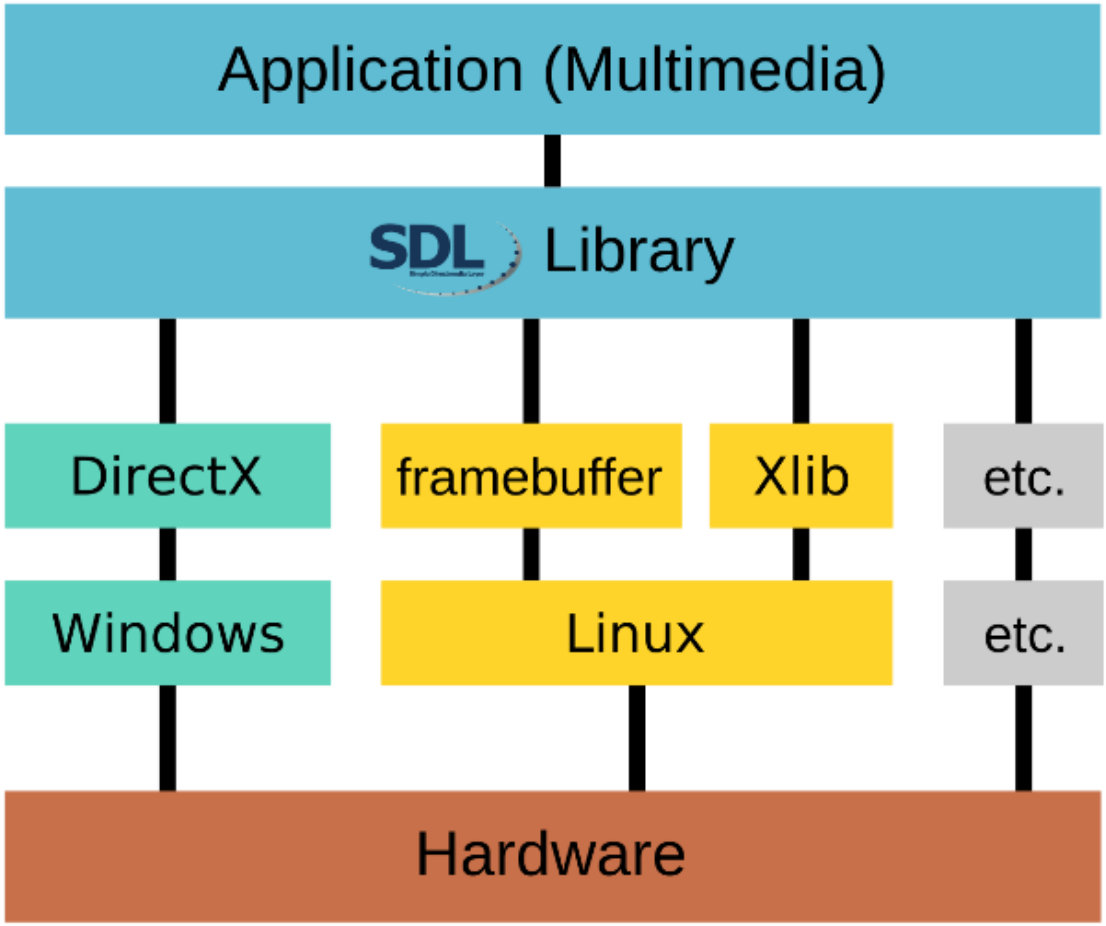
Development Basis

- ❖ libtcod
  - Library for TCOD(The Chronicles of Doryen)
  - Library based on SDL
  - Specialized in roguelike game development
  - Supports various tools and algorithms
  - Uses SDL to create graphic screen

  https://github.com/libtcod/libtcod

- ❖ SDL
  - Simple DirectMedia Layer
  - Cross-platform multimedia library
  - Supports Windows, MAC, Linux
  - Abstract layers of video, audio, user input, etc, to run on multiple operating systems

  https://www.libsdl.org/



Project Introduction

Game Structure

- **Movement**
  - Move around the map by accepting up, down, left, and right keystrokes
- **Combat**
  - Performed by making vertical and horizontal contact with enemy symbols.
  - Defeating enemies gives you experience
- **Collect/view/use items**
  - Hover over an item symbol to pick it up
  - Items can be used in the inventory.
  - Recovery items: restore the player's health
  - Tracking item: deals damage to the nearest enemy
- **Movement between floors**
  - Go to the next floor by clicking on the staircase symbol above the field.
  - All floors are randomly generated
- **Save/Run**
  - Automatically saves current progress
  - Allows you to execute saved progress via Continue

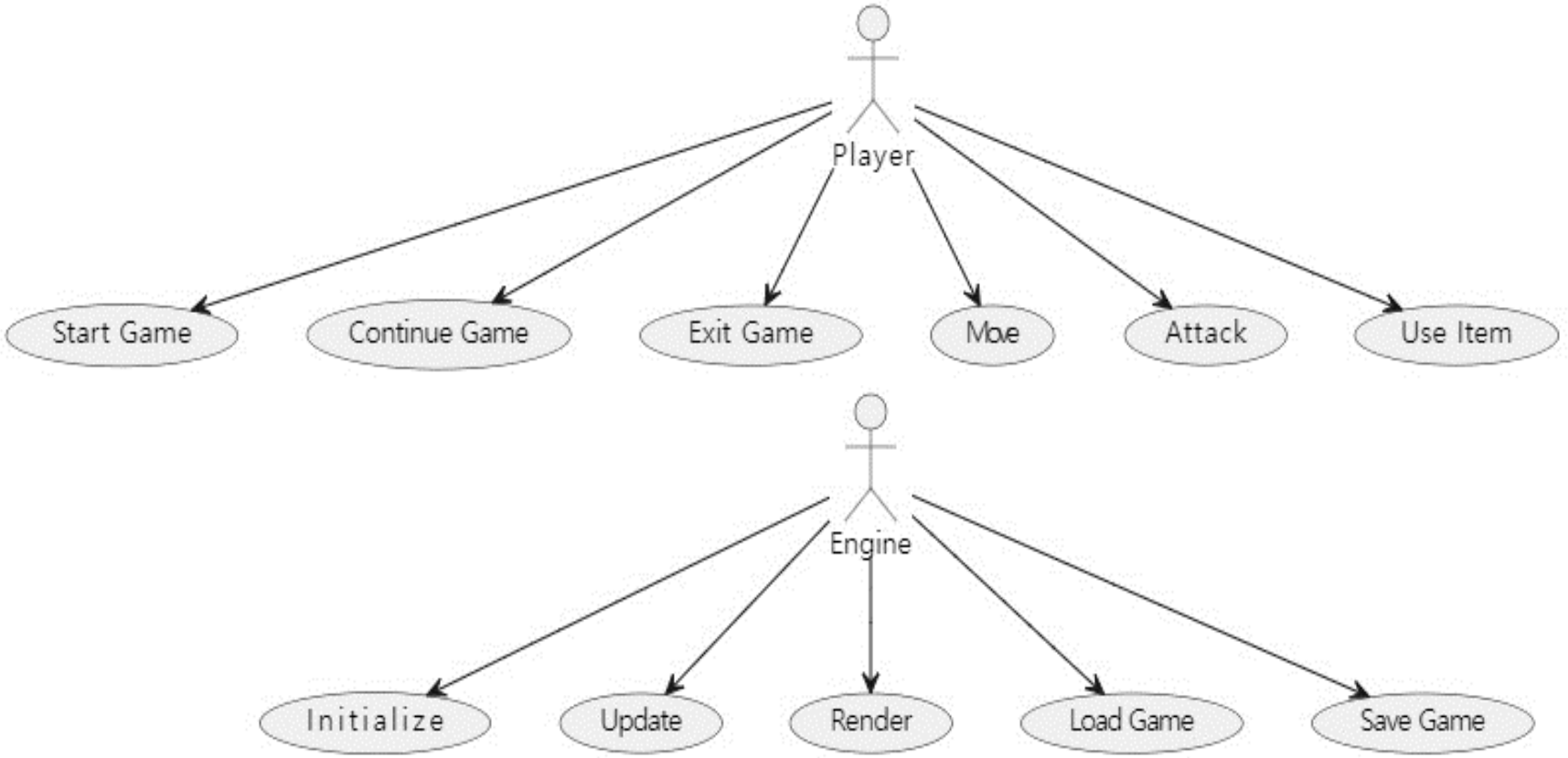
Object-Orient Implementation

Object-Orient Feature Implementation

- **Encapsulation**
  - Implementing data and behavior by hiding it inside an object
  - Each class encapsulates its own methods.
- **Inheritance**
  - Implementation by creating the most basic class and having other objects inherit from it.
  - Most of the classes that creates the game inherit from Object.
- **Polymorphism**
  - Implemented by allowing objects of different classes to respond to the same method call in different ways.
  - The Ai of a character is changed by update() depending on the type of character.

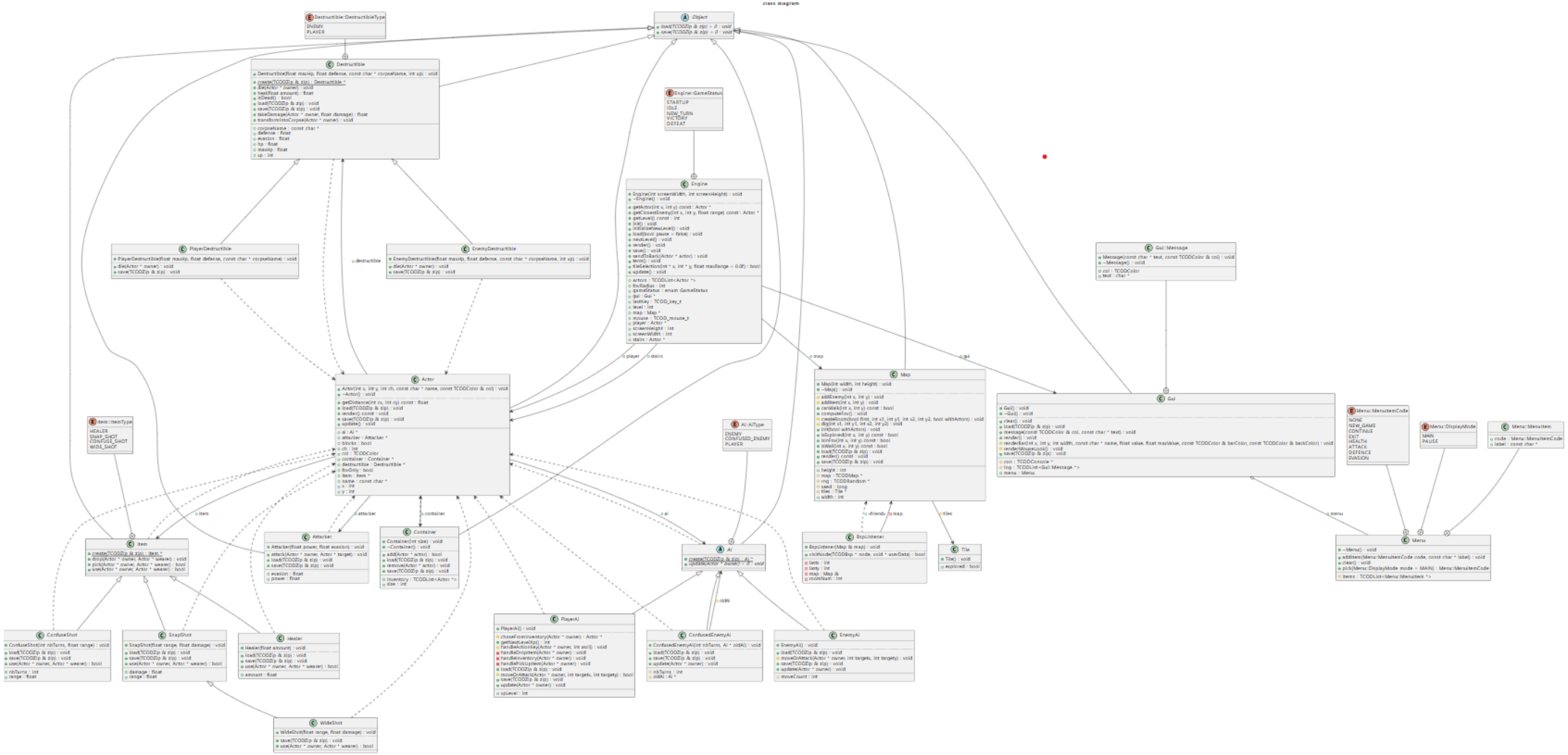
Object-Orient Implementation

Use Case Diagram



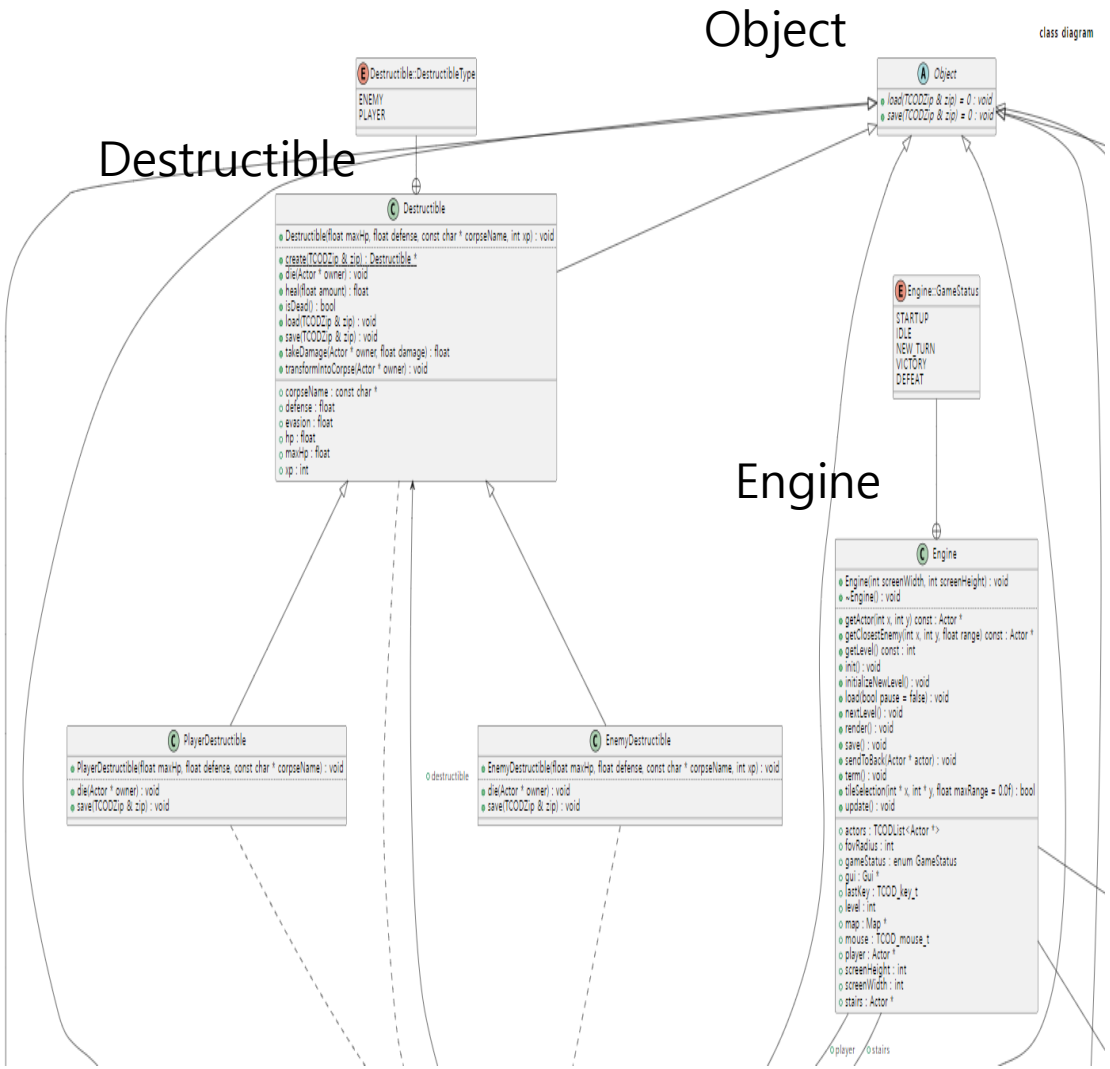
Object-Orient Implementation

Class Diagram(Full View)



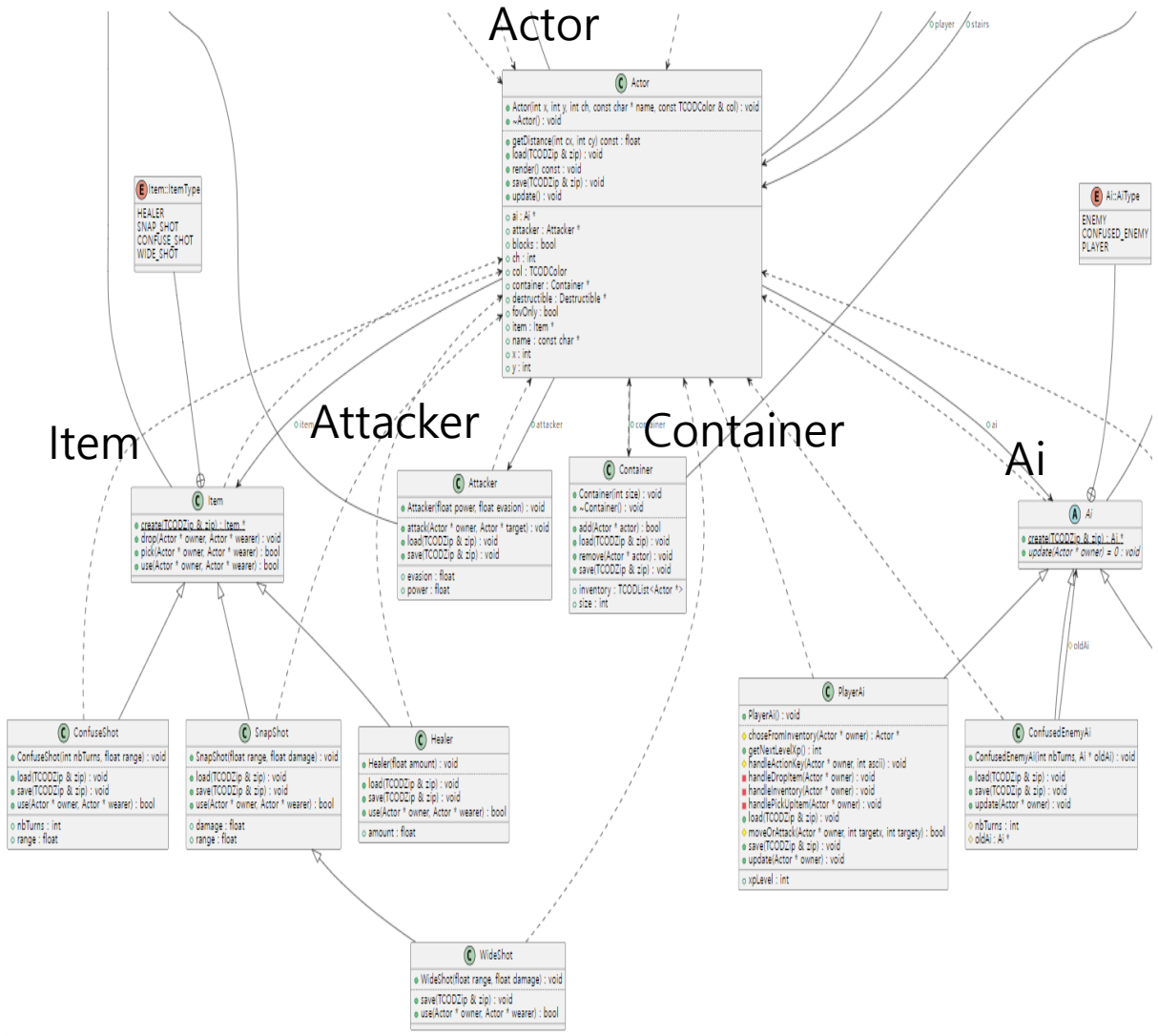
Object-Orient Implementation

- **Object**
  - Anything that can be stored and executed
  - The basis of all classes
- **Engine**
  - The underlying system that powers the game
- **Destructible**
  - Destructible objects
  - Player (PlayerDestructible)
  - Enemies (EnemyDestructible)



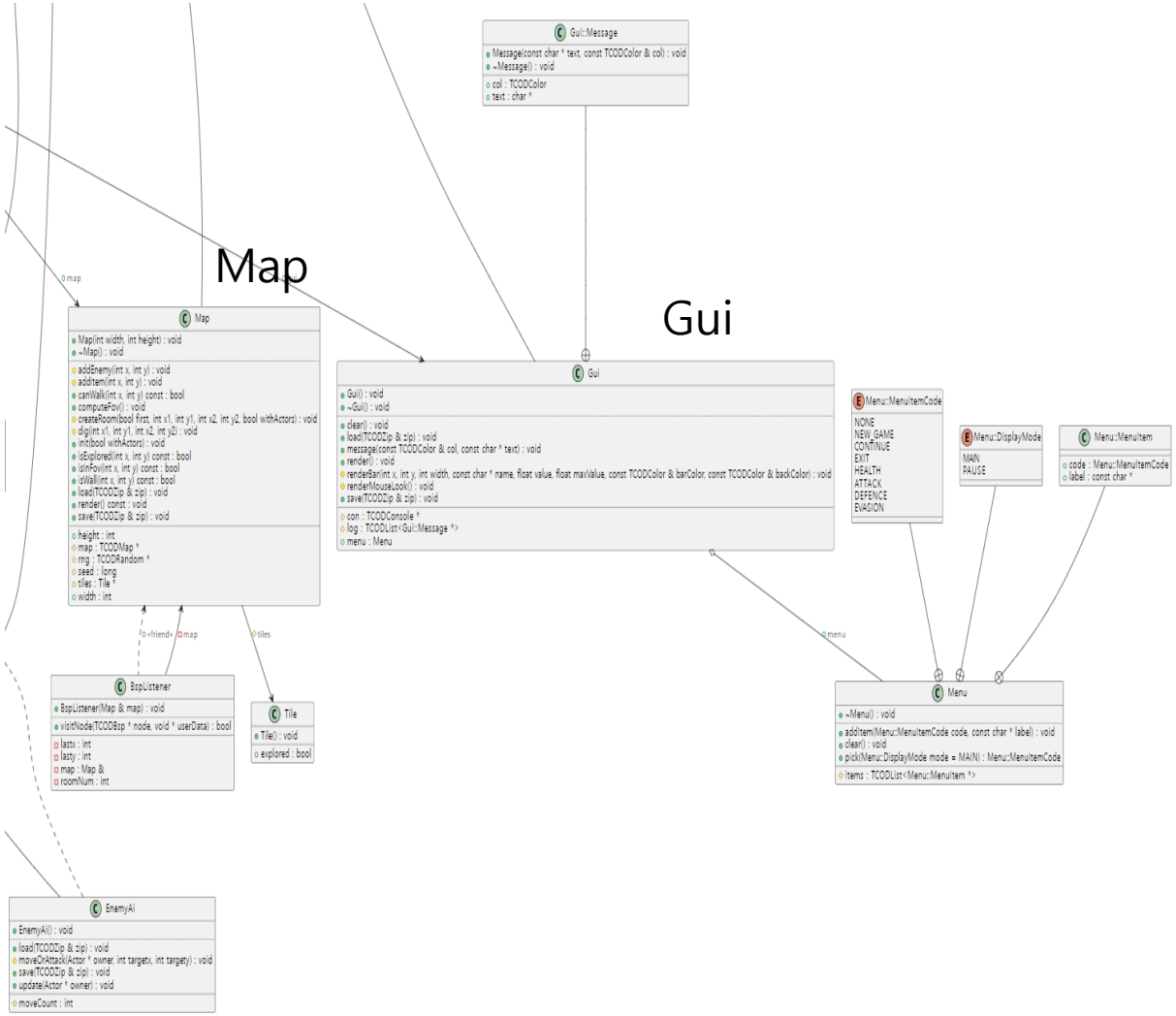
Object-Orient Implementation

- **Actor**
  - Sets the state of each object
- **Item**
  - How in-game items operates
  - Healing items (Healer)
  - Attack items (SnapShot)
  - Effect items (ConfuseShot)
  - AoE items (WideShot)
- **Attacker**
  - System that allows to perform combat
- **Container**
  - System that allows you to store acquired items
- **Ai**
  - How each Destructible operates
  - Player (PlayerAi)
  - Enemy (EnemyAi)
  - Confused Enemy (ConfusedEnemyAI)



Object-Orient Implementation

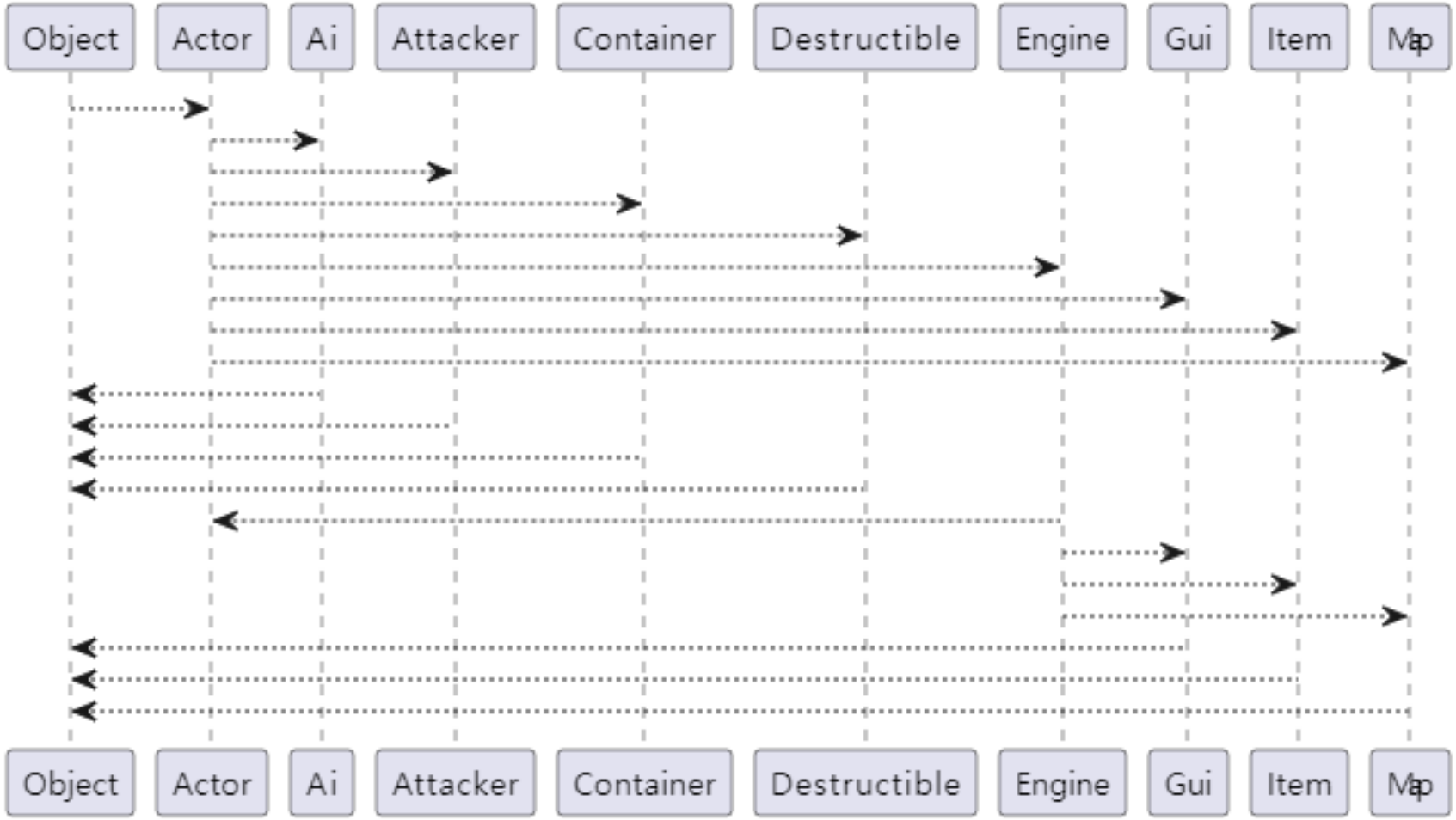
- **Map**
  - Creates In-Game Map
  - Game Map Structure (Tiles)
- **Gui**
  - Components of a screen
  - Menu Screen (Menu)



Object-Orient Implementation

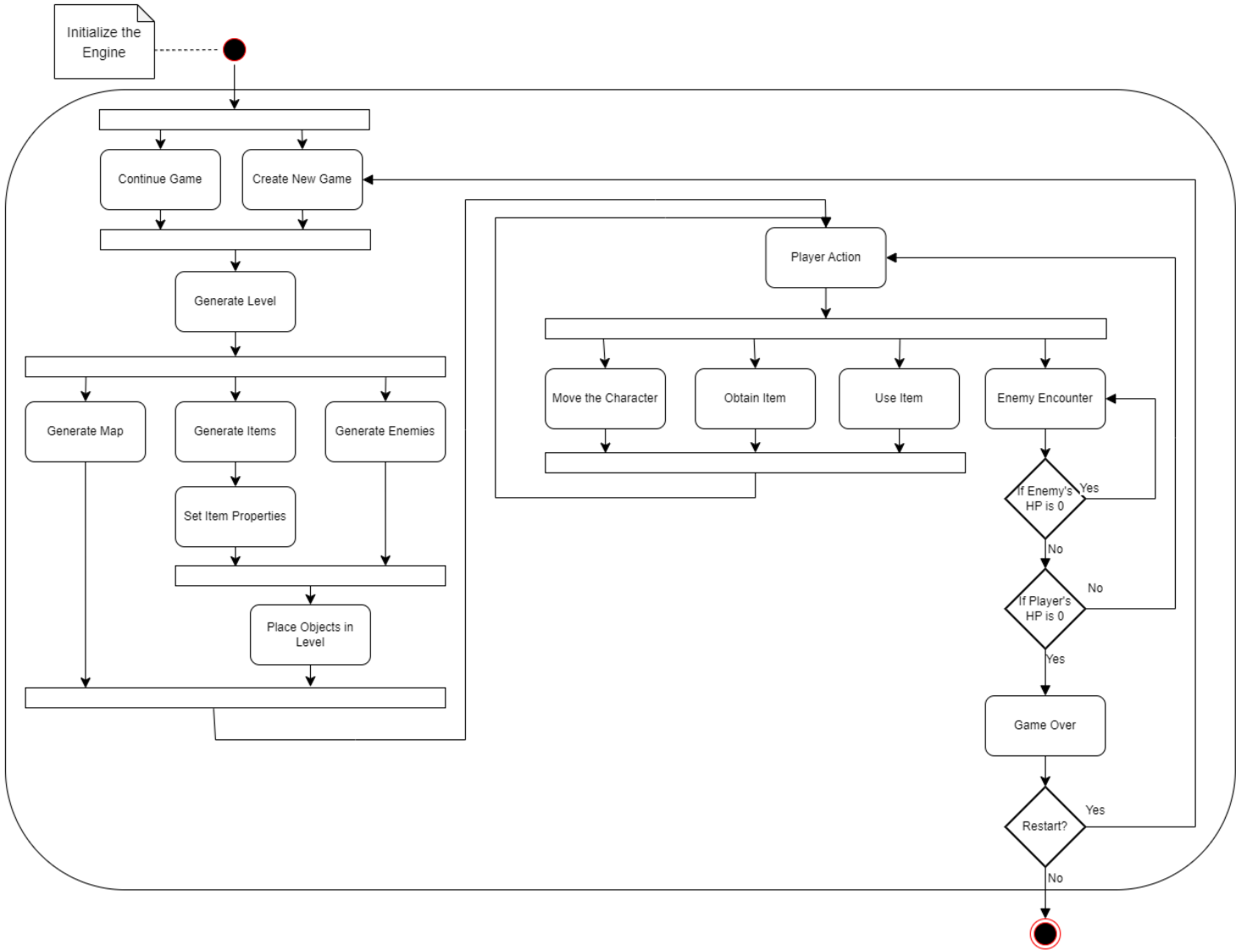
Sequence Diagram

- [sequence_diagram.svg]



Object-Orient Implementation

Activity Diagram



Project Result

Demo Video