Execute Summary for the System and Object Design

The Lost Empire - Group 10 - Daniel, Yue Ben, Nianzu Ma

The Lost Empire is a virtual-reality scuba-diving game, featuring 3D thinking as well as 3D visioning.

The Use Case diagram below (Figure 1) shows the core functions contained in our game, and it's refined from the original requirement specification, focusing on the Adventure Mode only.

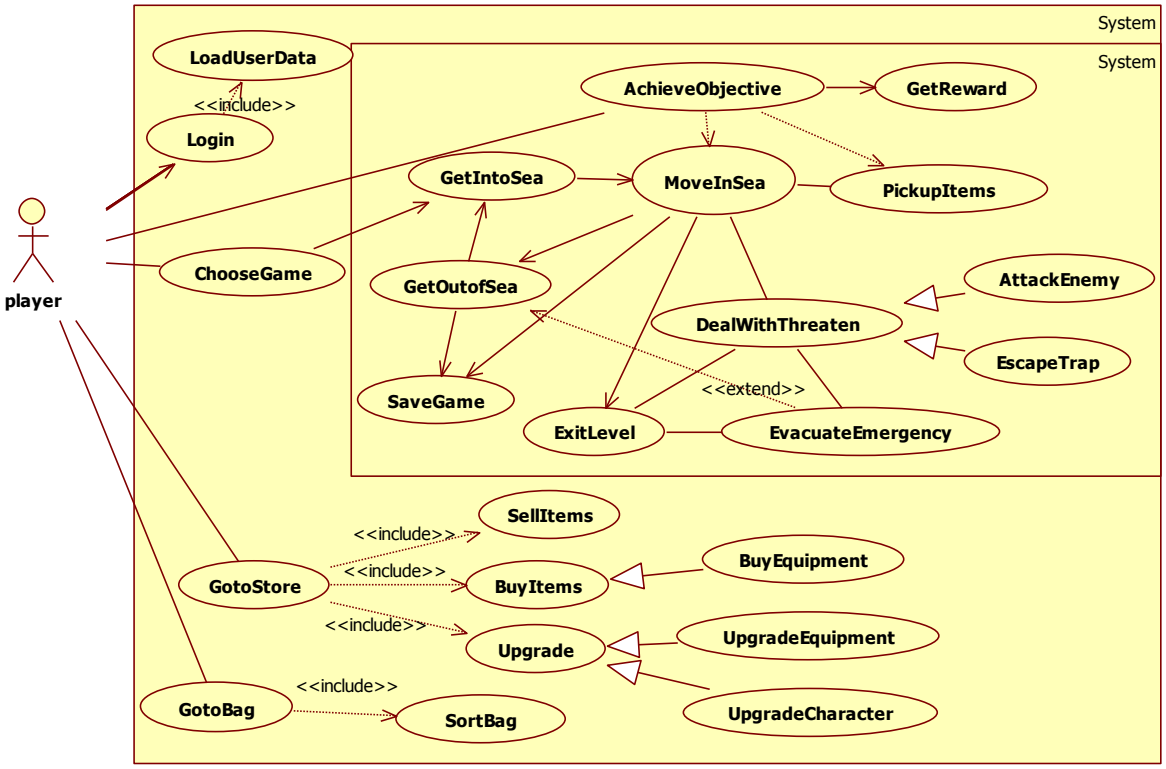

Figure 1 System Functions (outline, UML Use Case diagram)

Basically the game is imitating scuba diving in the real world. Its key functions are: 1) User data storage, so as to maintain information for each user; 2) Scuba diving supporting, including dealing with dangers, treasure collection; 3) Trading[1], for selling items collected while diving, and buying or upgrading equipment; 4) Achievement and Reward system, now embedded in the Game subsystem, for rewarding the player if he achieves objectives within games.

System and Object Design

The system is briefly decomposed into four subsystems - User Interface subsystem, Game subsystem, User Data subsystem, and Item subsystem, using an open layered architecture (Figure 2). The decomposition is based on different domains covered; within each subsystem, there might be structure similar to the three-tier architecture, such as the Game subsystem and the Item subsystem, where operations and data files are separated.

The User Interface Subsystem

The User Interface subsystem contains all user interfaces interacting with the user, including the initial start-up page, and returns execution results obtained from other subsystems. It contains all menu related interaction, such as the Store view and the Bag view. Although the Game view (for use within game level) is also included in this subsystem, it's relatively independent, and only used after game loaded.

[1] Trading is especially important for future extension for commercial use, e.g. selling virtual items for real money.

A World Map view is provided for the user to view and select game level, under the constraints of user ability judged from User Data (in User Data subsystem). Based on the user's selection, a filename will be sent to the World Map Subsystem (in Game subsystem) for loading the selected game, also transferring control to the World Map Subsystem.

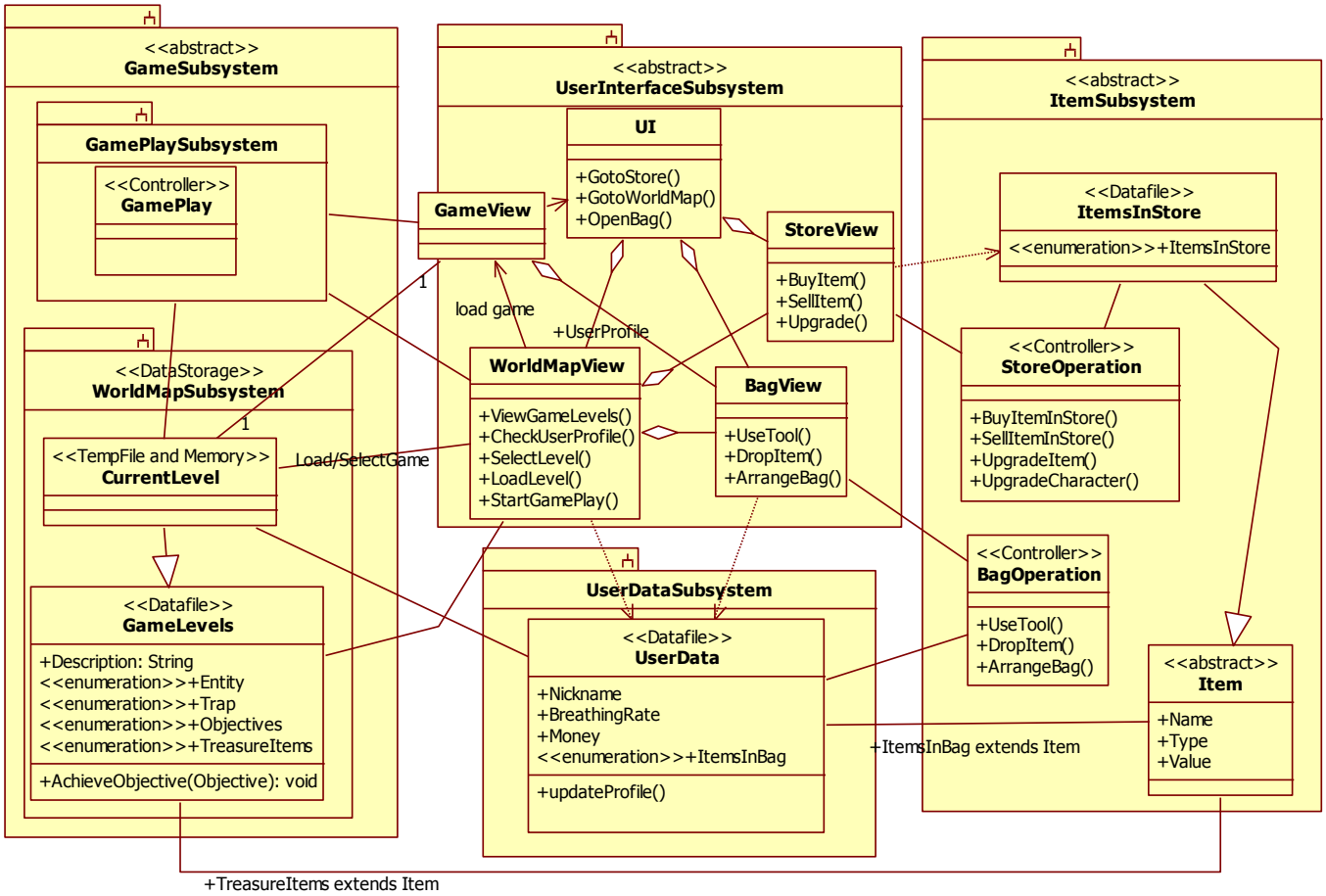

Figure 2 System Architecture (outline, UML class diagram)

The Game subsystem:

The Game subsystem contains two sub-subsystems - the World Map Subsystem and the Game Play Subsystem, and itself has no real responsibilities.

The World Map Subsystem is mainly in the data storage level, maintaining all game level maps and saved games. Besides, it also contains temporary runtime data between the data level and the function level.

The Game Play Subsystem is responsible for core game play, covering all attributes and operations needed within a game level. Briefly it covers the sub system marked in the Use Case Diagram (Figure 1).

The User Data subsystem

The User Data subsystem is responsible for maintaining user profile and game-related user data, such as the player's breathing rate and so on, to help judging user's ability and adjust game progress.

The Item subsystem

The Item subsystem is responsible for Item operations in Store and in Bag – realizing function called in the Store view and the Bag view, also defining the basic attributes in the abstract class Item (not realized), as well as maintaining data files for Items in Store. Note not all items in the game is managed here, player-owned Items (in his bag) are in the player's user data (thus in User Data subsystem), while Treasure Items are in Game/Level data files.