**Analysis Document**

1. Introduction

The proposed game is called Galaxy Explorer: Mission to Pandora. A space crew is on a mission from Earth to find a planet that can support life. The player is the sole crew member that awoke from his/her cryosleep and now has to power on the ship in order to find out where they are and to wake the rest of the ship crew.

2. Proposed System

2.1 Overview

“Galaxy Explorer: Mission to Pandora” is an educational text-based adventure game that is intended to teach core components of computer science such binary conversion. The system will have a command line user interface and will accept user input from the keyboard. The user will input commands to determine the path in which they can take and the objects that they may interact with..

2.2 Functional Requirements

2.2.1Start Game

The player must be able to start a new or saved game from the main menu

2.2.2 Load Game

The player must be able to load the information from a previously saved game

2.2.3 Save Game

The player must be able to save a game they have started

2.2.4 Help

At any time the player should be able to hit the help button and get a list of actions that can be performed given their place in the game

2.2.5 Exit

The player should be able to exit the game at any time, with or without saving first

2.2.6 Add New Features

The admin should be able to add new features to the game

2.2.7 Edit Descriptions

The admin should be able to edit the descriptions of any object in the game

2.2.8 Quick Jump

The system should allow the player to view a description of an item and then have the option to take it or leave it. Player has to be in the same room as the item

2.2.9 Explore Room

Players will have a description of the room appear every time they enter a room.

2.2.10 Move

The system will take player input of the cardinal directions and move one room in that direction.

2.2.11 Random Item/Monster Generation

Every time the player enters the room, there will be a chance for a puzzle, monster, or both to appear. There will also be a chance where neither appear and the player can just continue as normal.

2.3 Non-Functional Requirements

2.3.1 Reliability  
 - The system should not crash or end unexpectedly in the middle of the game.

-System should successfully save and load data files

2.3.2 Availability

The game should run on any Windows-based desktop computer.

2.3.3 Recoverability

The game should be able to start back up where it left off.

2.3.4 Performance

The game should be able to run almost perfectly without bugs

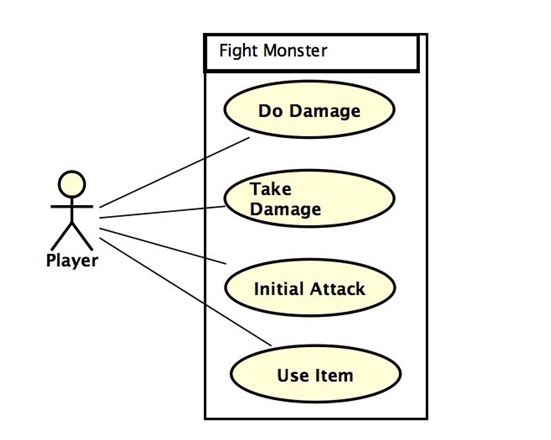
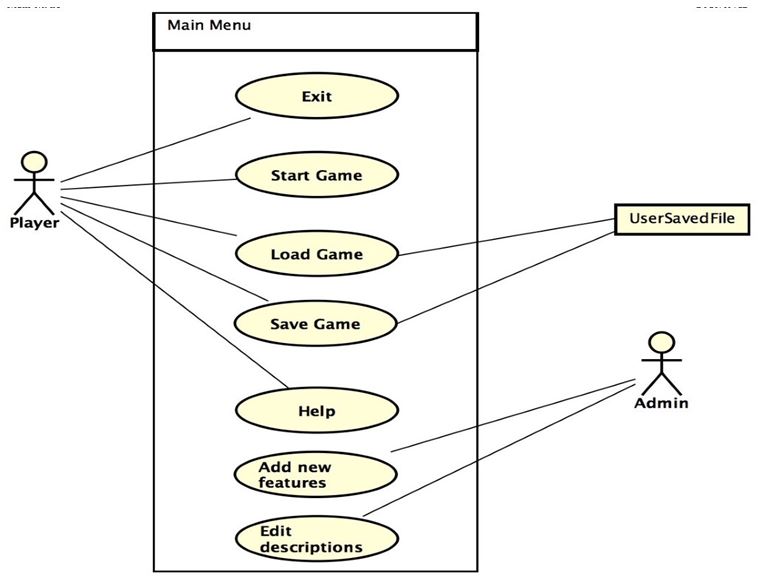
2.4 System Models

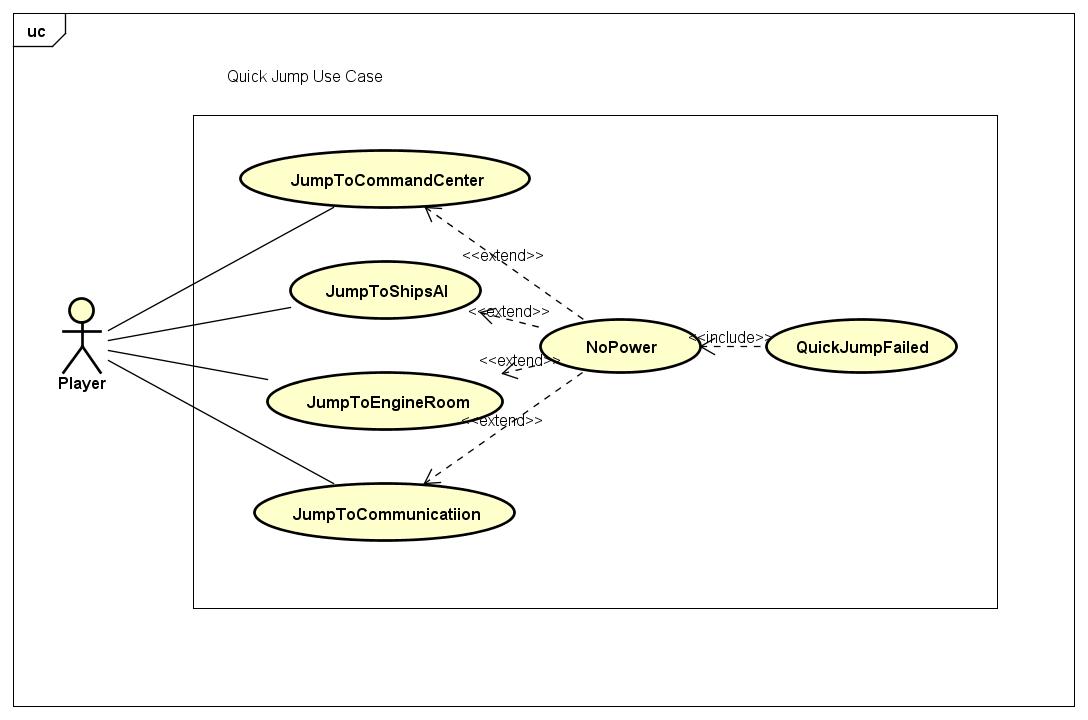
2.4.1. Use case model

2.4.1.1. Use cases description

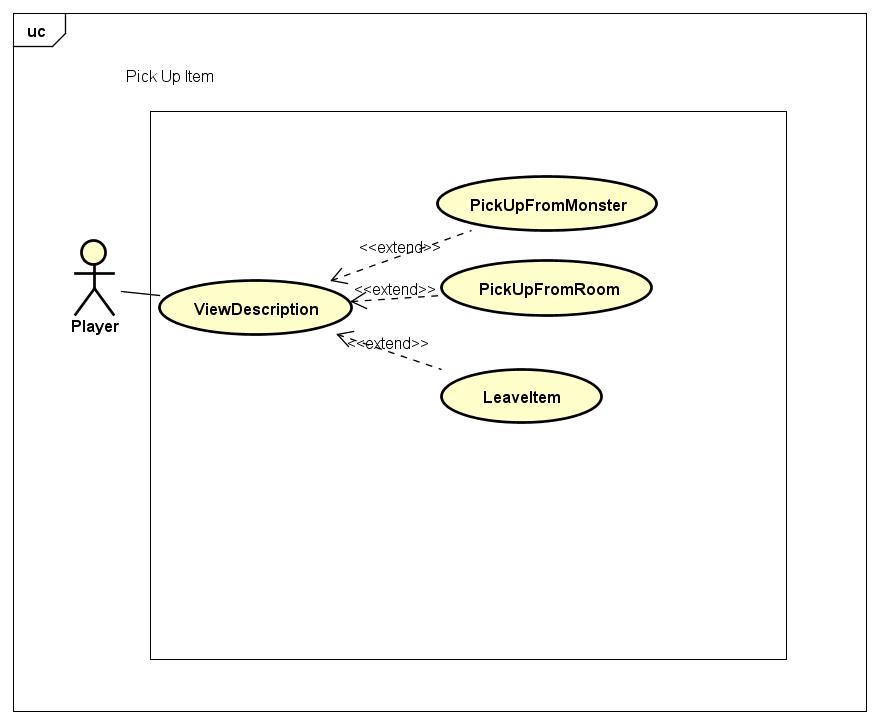
1. Fight
   1. Do Damage - This use case will deduct HP from the monster that is in the player’s current room.
   2. Take Damage - This use case will cause the HP of the player to decrease based on the monster that is currently in the same room as the player
   3. Initial Attack
   4. Use Items - This use case will display a list of items that are currently within the player’s inventory. The user may select an item from the list, at which point the item’s associated function will be performed and then be removed from the player’s inventory.
2. Main Menu
   1. Exit - When this use case is invoked, the application will be closed.
   2. Start Game - This use case will begin a new game from its beginning
   3. Load Game - This use case displays a list of save states organized by name. The user will be prompted to select one, after which the program will load the game’s state based off of the information in the text file.
   4. Save - This use case prompts the user to enter a textual input to identify the game’s save state. The program will then enter the information of the game’s current state into a text file, or overwrite the information of a current text file if the name entered already exists.
   5. Help - this use case will display a list of possible inputs that the user may invoke.
   6. Add new Features- This will allow the Admin to implement new features if need be to improve/fix gameplay.
   7. Edit Descriptions- The Admin can edit descriptions if need be to fit the game
3. Room Menu(Josh)
   1. Quick Jump - When this use case is invoked, the user will be given a list of potential rooms that they may move to. After selecting one, the player will be moved to the chosen room
   2. Pick Up Item - This use case will allow the user to enter a command to move an item from a room into the player’s inventory
   3. Explore Room - This use case will display a textual description of the room that the player is currently in. Any puzzles or monsters in the room will also be written out.
   4. Move - This use case allows the user to change its position to an adjacent, unlocked room
   5. Random Item Generation - This use case will determine the possibility for an item, monster, or puzzle to be created when the player moves to a new room.
4. Puzzle
5. Display Puzzle - When this use case is selected, it will initiate a puzzle that is in the room. After the puzzle is solved, the user will receive an item.
6. Quit Puzzle - This use case will exit out from the puzzle that the user is on.
7. Display Solution - If the player cannot solve the puzzle, and desire to know the answer he selects ‘Show Answer.’ This gives the answer to the puzzle.
8. Hint - If the player cannot solve a puzzle, this use case would show a clue on how to solve the puzzle.

2.4.1.2. Use case diagrams

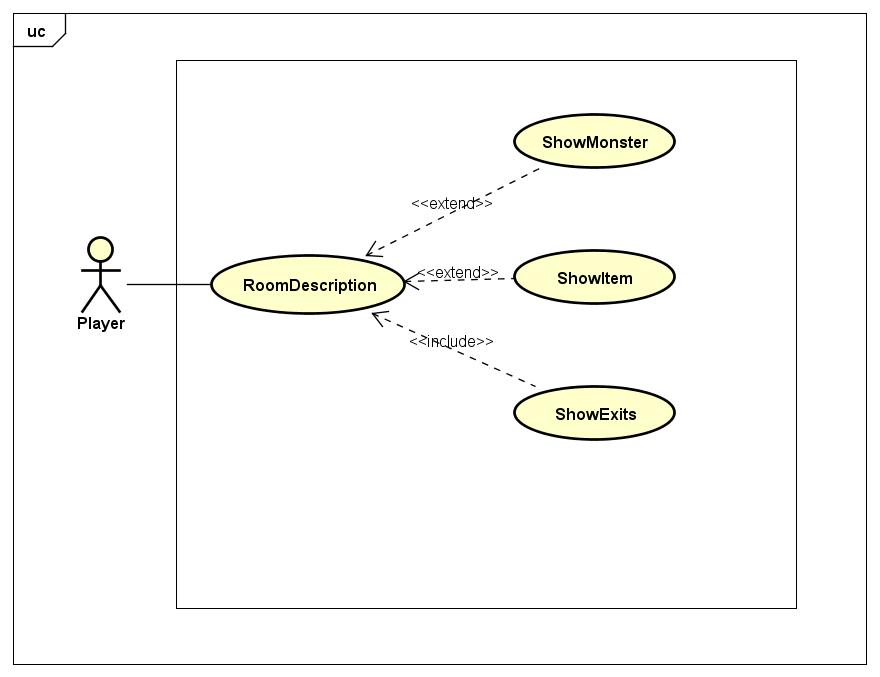
1. Battle  
   
2. Main Menu  
   
3. Room Menu(Josh)
   1. Quick Jump



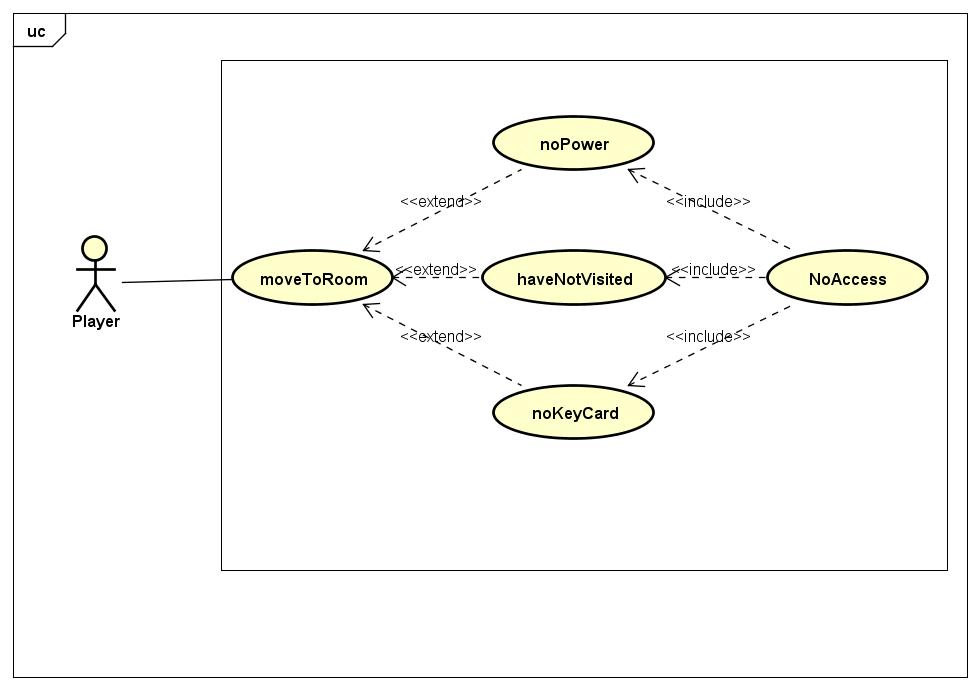
* 1. Pick up Item

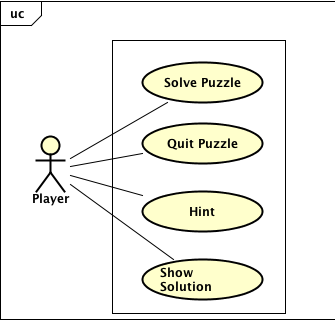


* 1. Explore Room



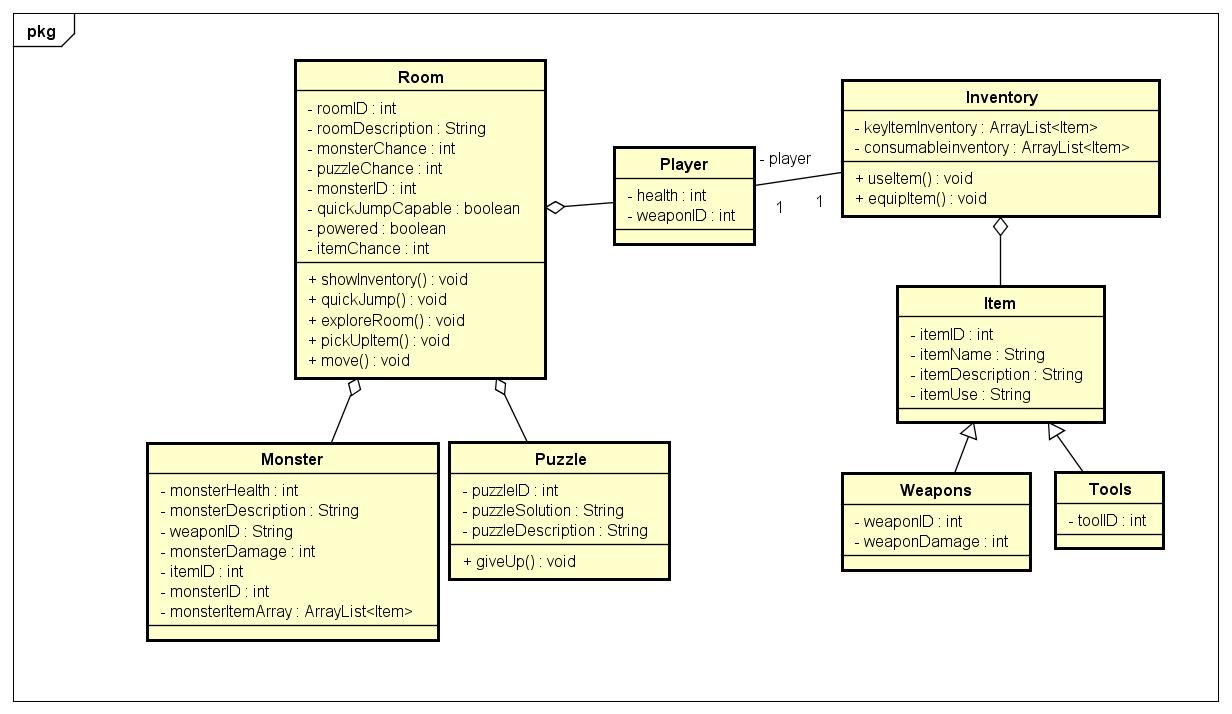
* 1. Move



1. Puzzle

2.4.2. Object Model

2.4.2.1. Class Diagram

J

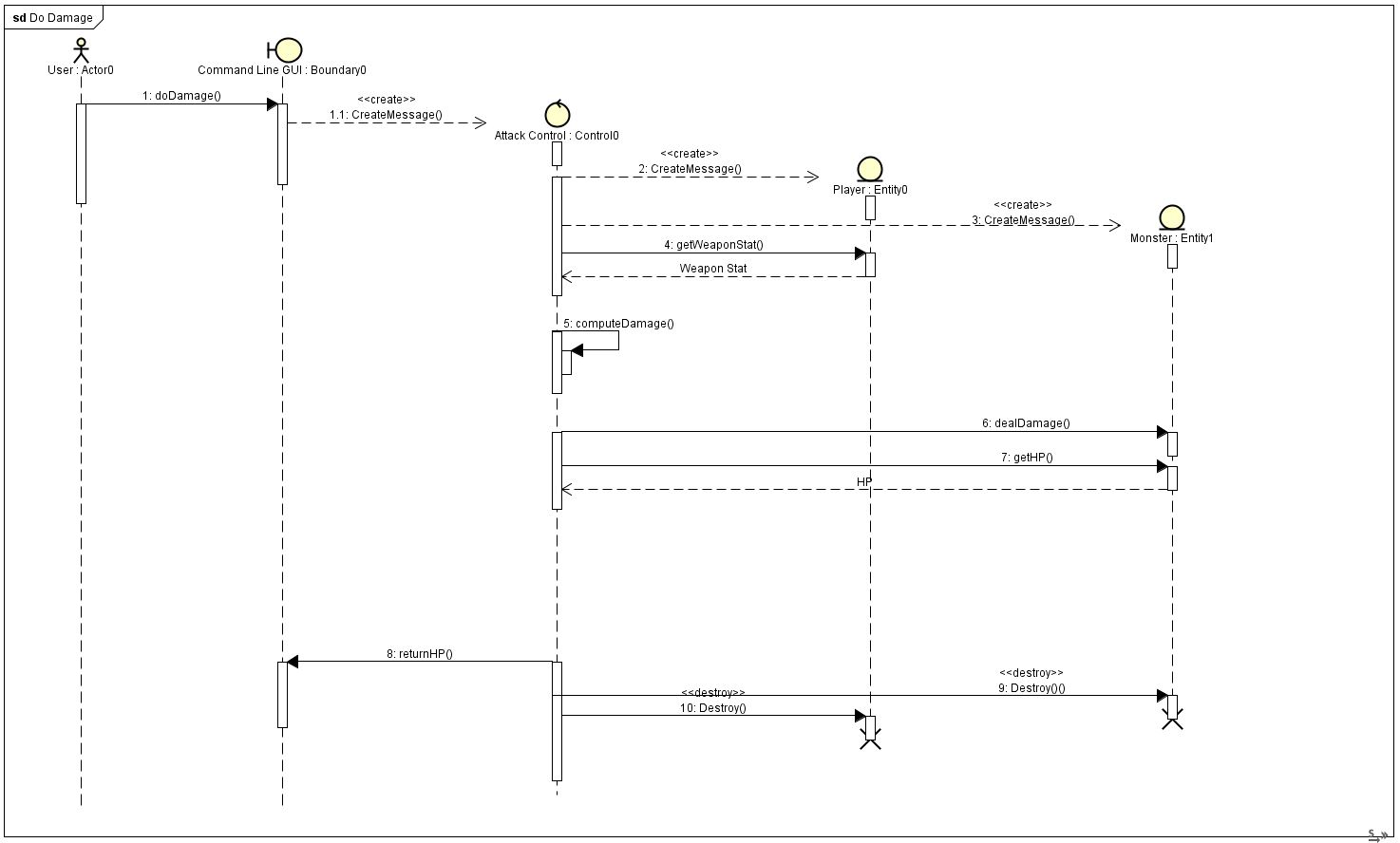
2.4.2.2. Class Diagram description

1. Player - The ‘player’ class has two attributes which is ‘health’ and ‘weaponID.’ The ‘health’ attribute will display the users lifespan and then ‘weaponID’ will show what item the user is carrying.
2. Monster - The ‘monster’ class has several different attributes. The different attributes display monsters lifespan, and its damage. It will also display the monsters description and show the play what item the monster may be carrying.
3. Room - The ‘room’ class is where the user interacts with and fights ‘monsters’ and ‘items.’
4. Inventory - an ‘Inventory’ will show you all of the collected items the user has.
5. Puzzle - a mental challenge that a player must overcome
6. Item - an Item is an entity that can be found within rooms or carried by both players and monsters. Items may be picked up and used by the player.

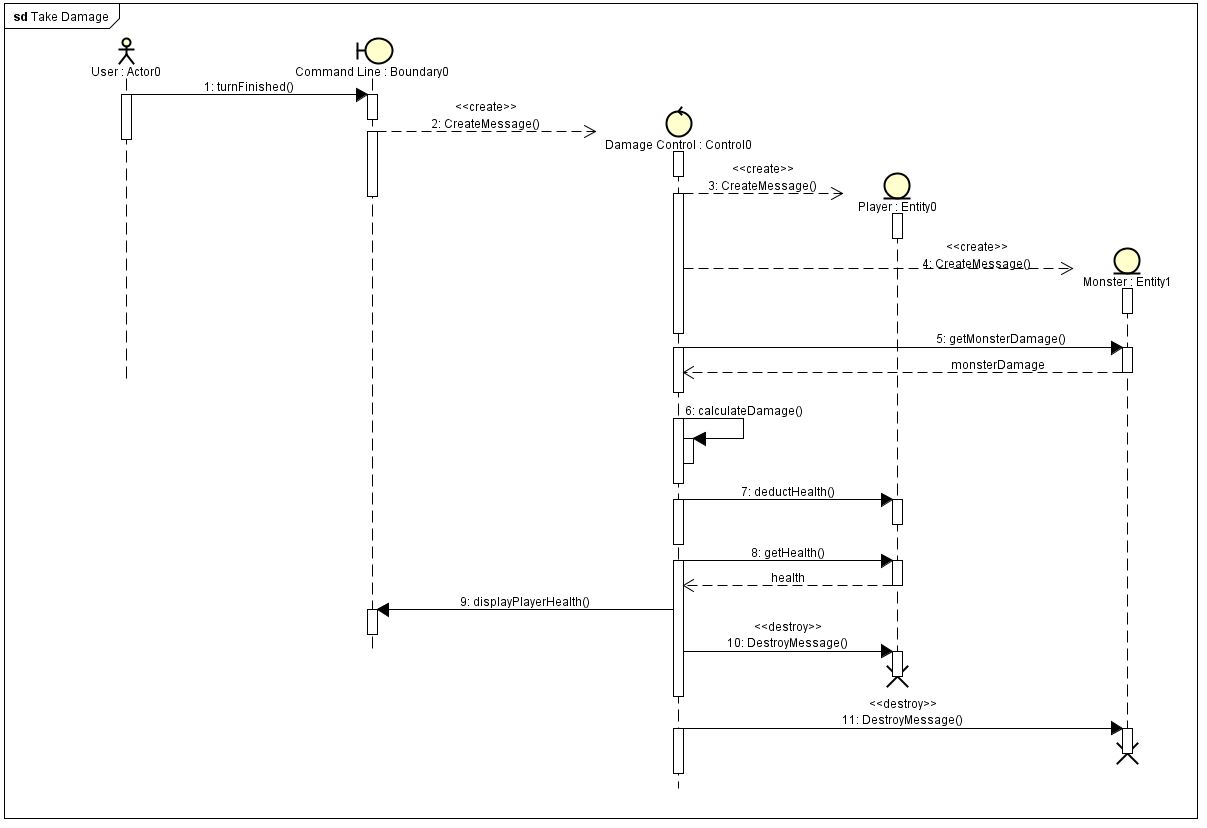
2.4.3. Dynamic Model

2.4.3.1. Sequence Diagrams

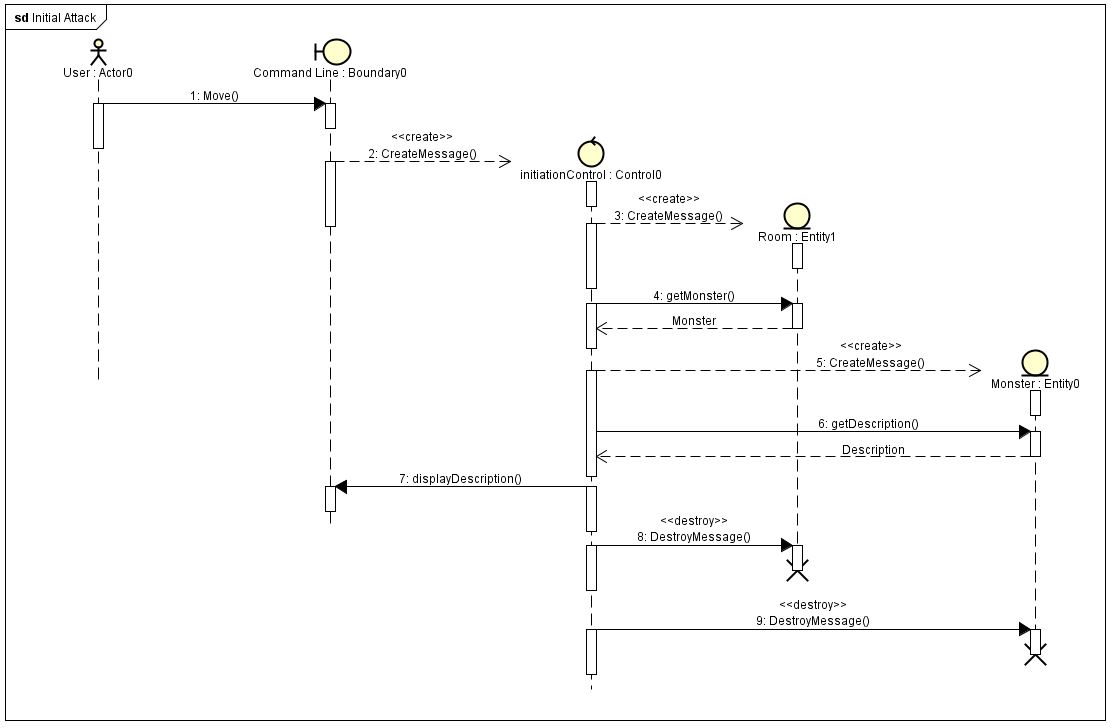
1. Fight
   1. Do Damage



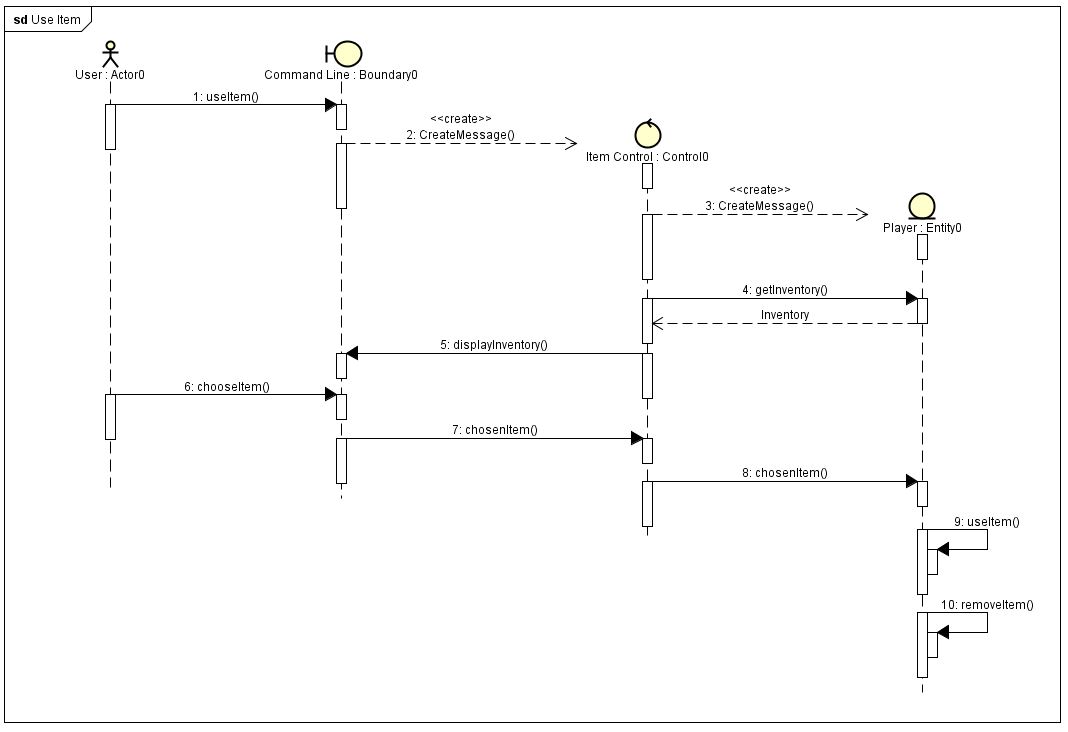
* 1. Take Damage

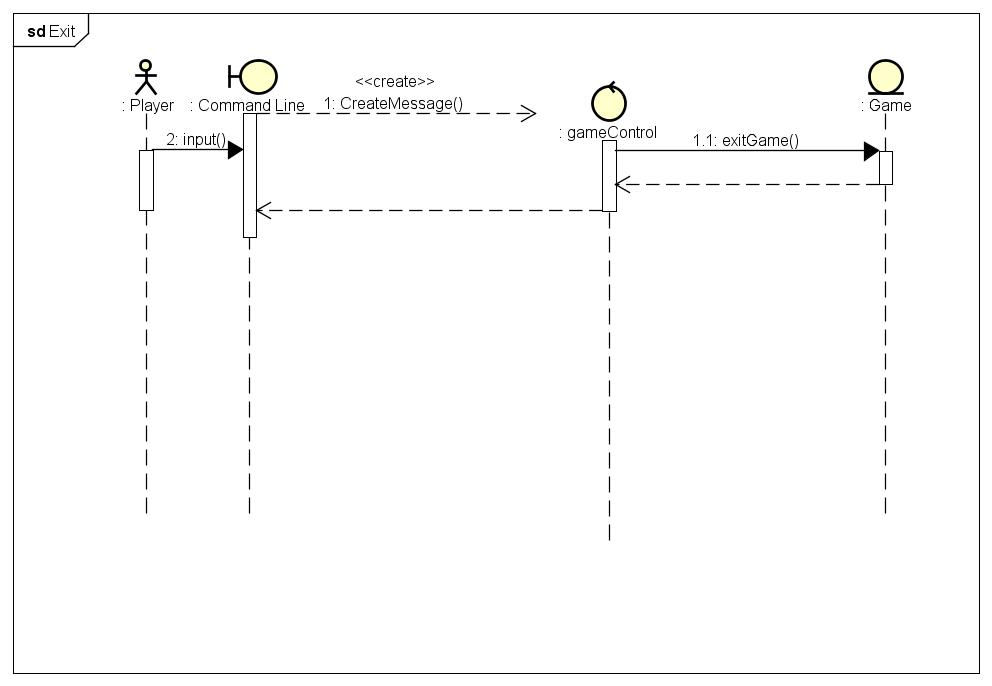
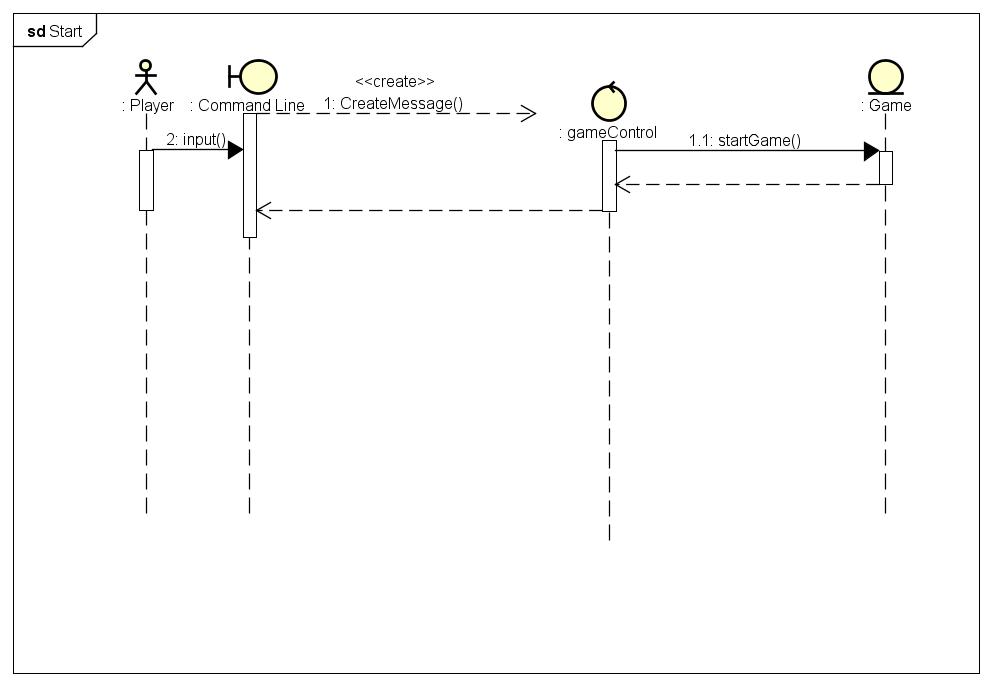
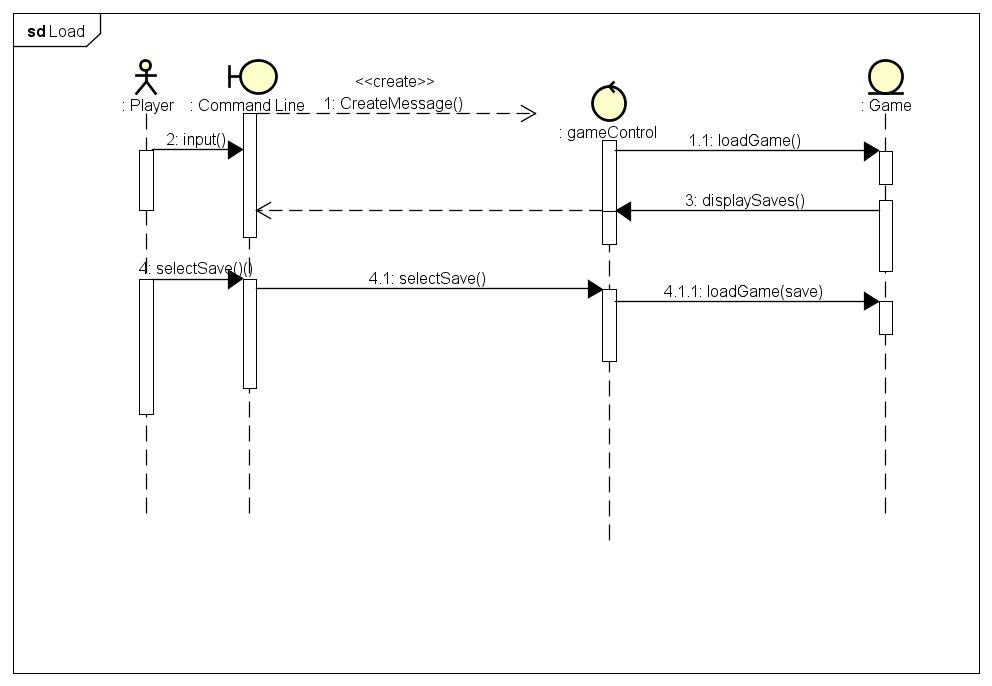
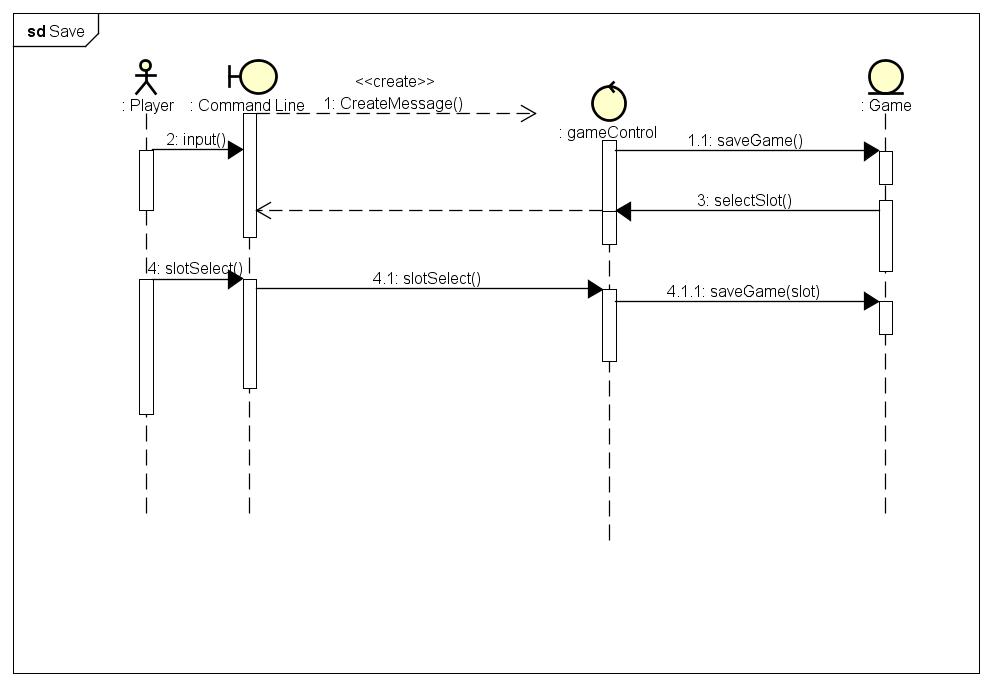
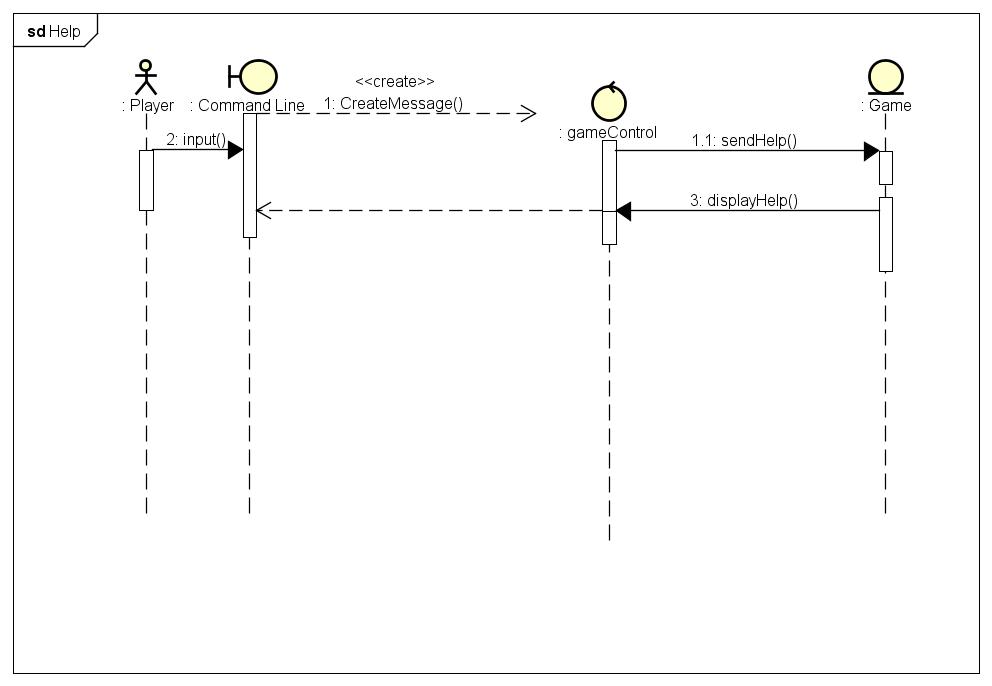
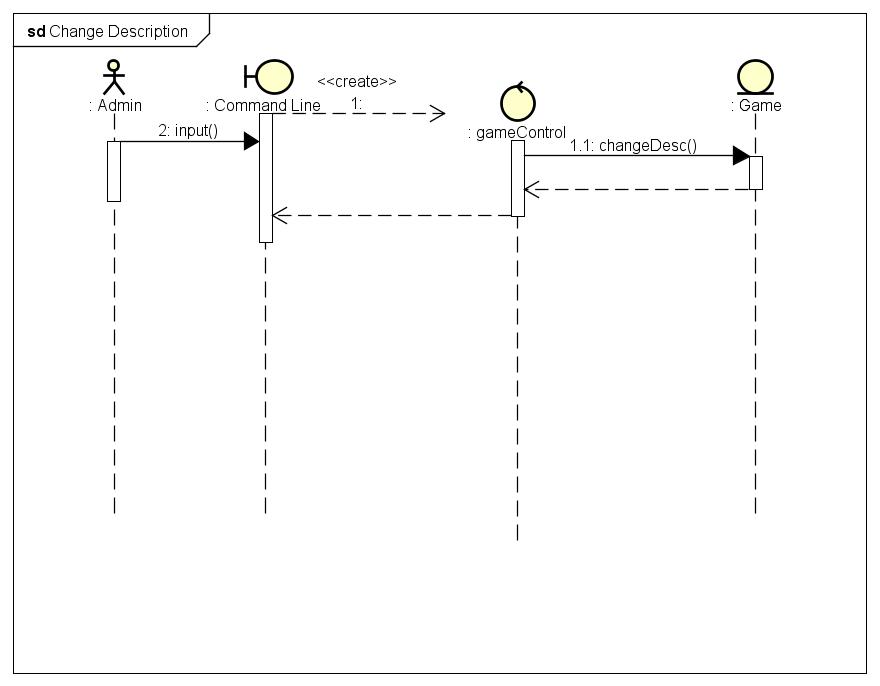


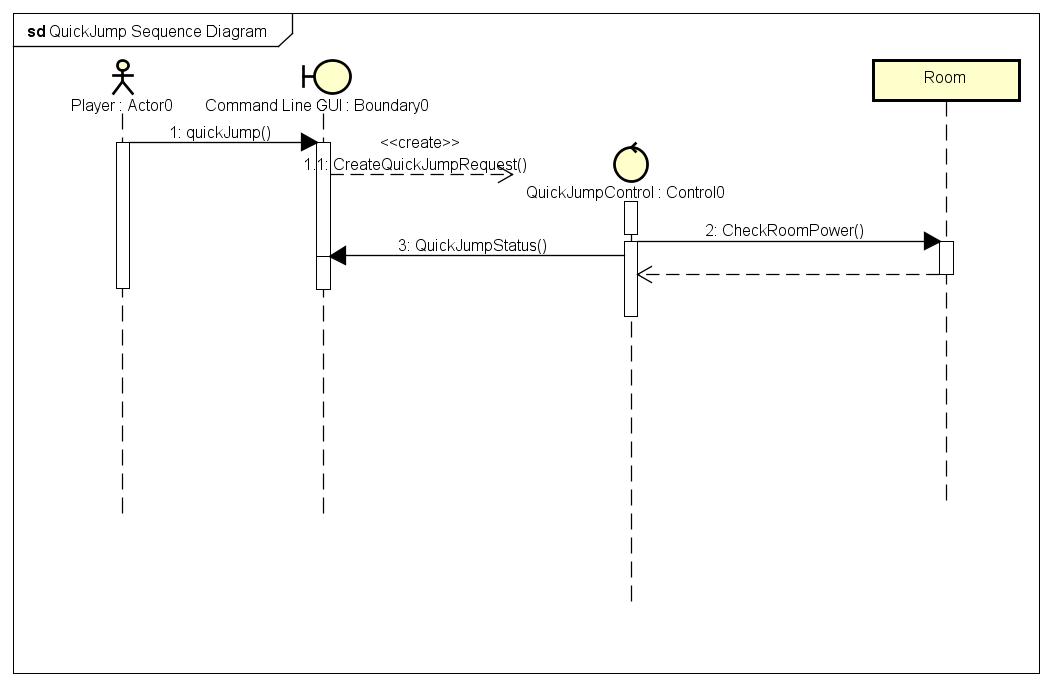
* 1. Initial Attack



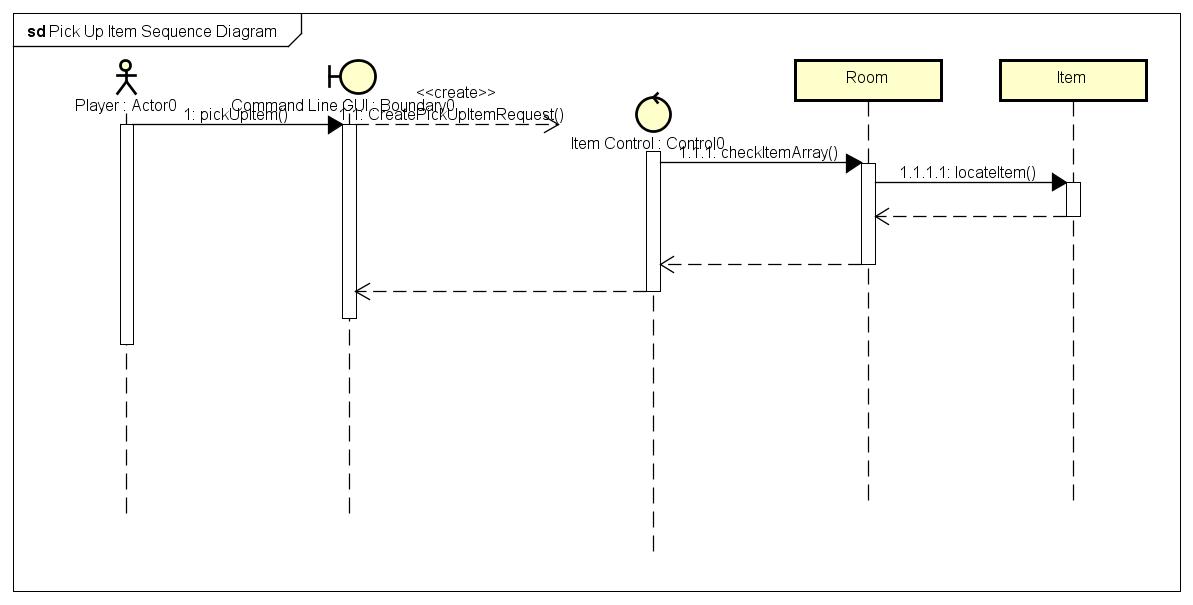
* 1. Use Item



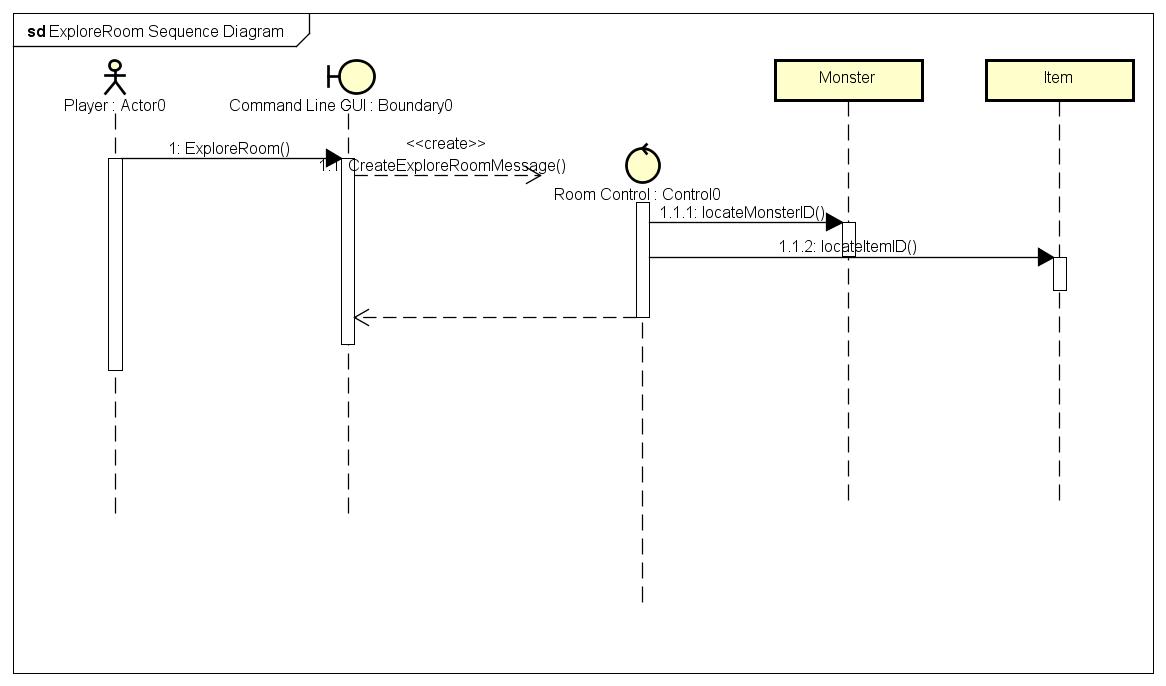
1. Main Menu
   1. Exit
   2. Start
   3. Load
   4. Save
   5. Help
   6. ChangeDescription
2. Room
   1. Quick Jump



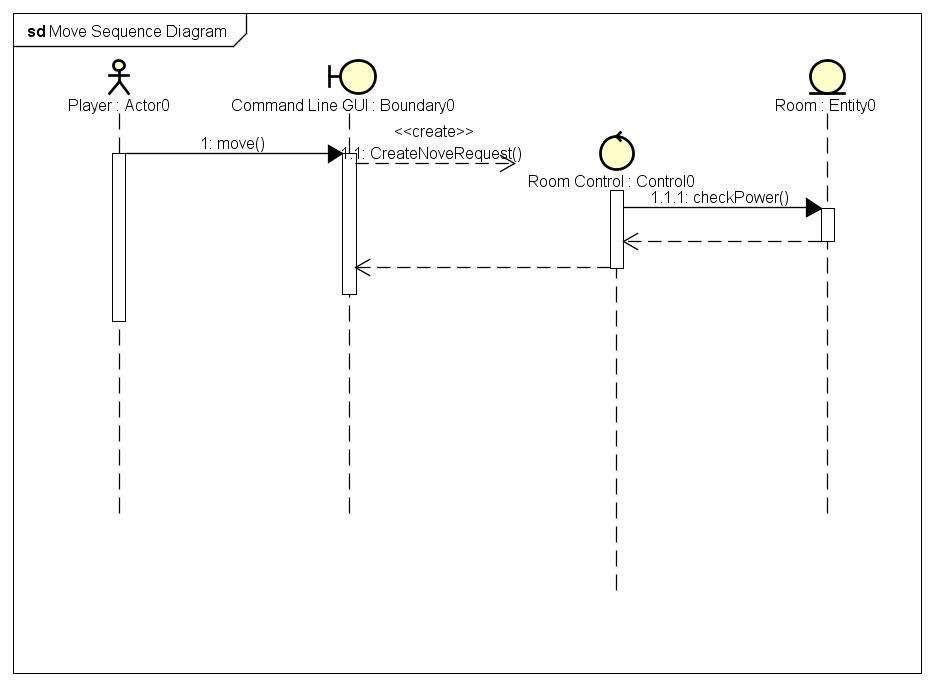
* 1. Pick Up Item



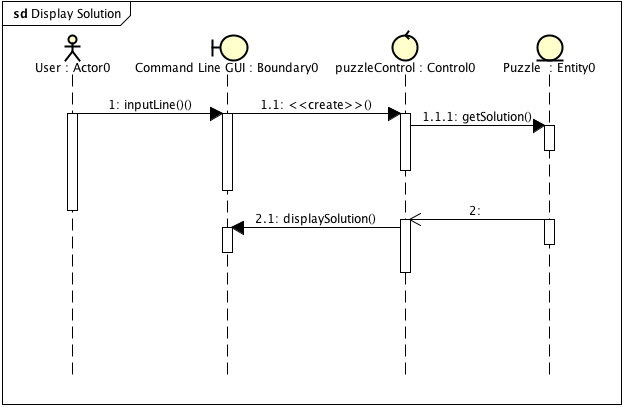
* 1. Explore Room



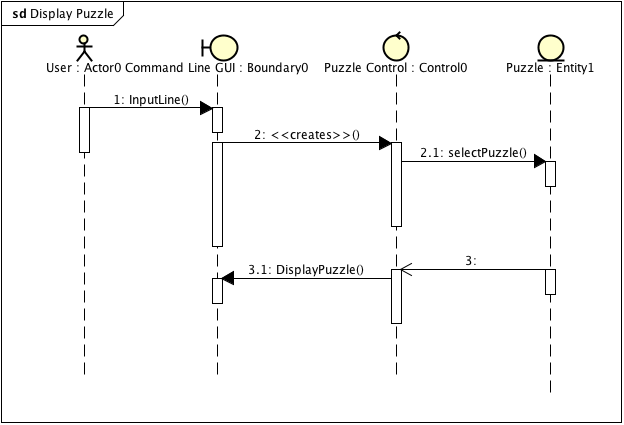
* 1. Move



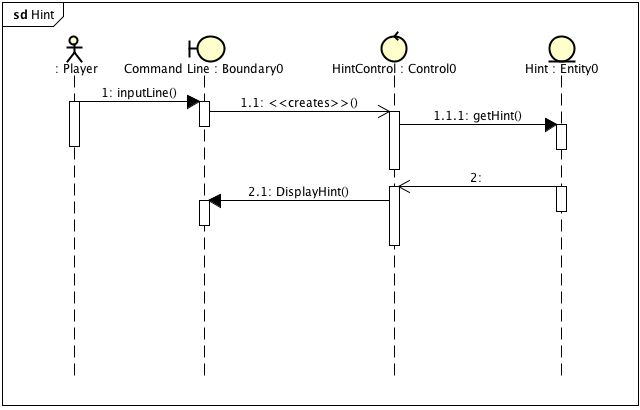
5. Puzzle

1. Display Puzzle

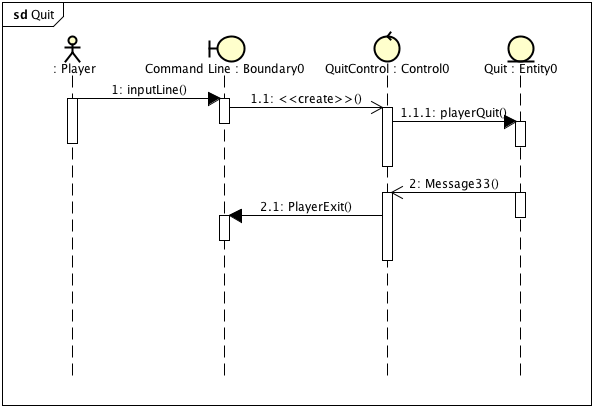
B.



C.



D.



2.4.4. User interface

Commands for user interface

1. Inventory - INV
2. Quick Jump - QJ
3. Move
4. Explore Room
5. Pick Up Item

2.4.4.1. Screen mockups (Extra Credit)

2.4.4.2. Navigation

3. Glossary