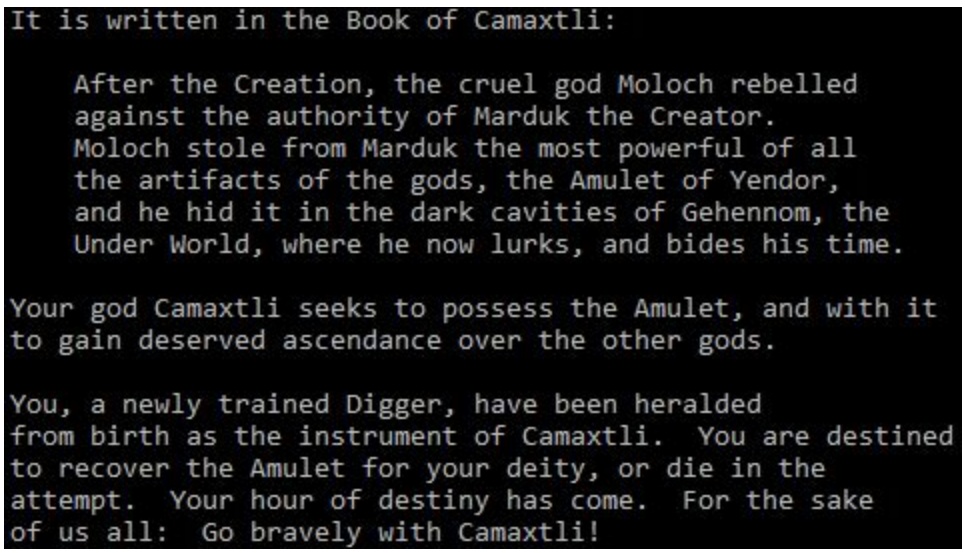
Interaction diagram for System TwitterNetHack

Assignment in the course PA1435 Objektorienterad Design

2017-04-10

Robin Westerlund, Max Nilsson, Rikard Magnom, Michael Lindroth

Author Name	Social Security Number	Thinking	Writing
Robin Westerlund	9510090195	25%	9%
Max Nilsson	9509052875	25%	23%
Rikard Magnom	9701014699	25%	61%
Michael Lindroth	9012283033	25%	40%



System Description

NetHack is a game that involves traversing a maze-like dungeon called “Mazes of Menace”, in which the goal is to obtain the Amulet of Yendor and escape the dungeon alive. Along with the amulet you are encouraged to loot as much treasure as possible, since you want the most points possible. This will place you on the list of high scores.

The goal of the system is to modify the classic game NetHack by using Twitter to generate dungeons and implementing multiplayer support. It will also be expanded in many ways to achieve more depth in the game.

Prioritised List of Use Cases

Motivation for Priorities

1. Since the actual maze and dungeon exploration is such an integral part of the game we have prioritized the “Movement” use case very highly.
2. Seeing as we are planning to create a text based implementation the “Look Around” use case has a high priority, as the player needs to know what options are available to him.
3. Battling being the other key element of the game alongside dungeon exploring this is also prioritized quite highly.
4. Not a game if you can’t lose.
5. Roughly tied into battling, however not as essential to the gameplay.
6. Prioritized lower than battling as the feature is not useful until battling is implemented.

10. Inventory alongside the use cases 7-9 is the last key feature of the game and have a medium level of priority. The prioritization on these use cases is vague as they are very tightly connected to each other.

13. “Load Game” along with use case 12 “Save Game”, while interesting features, are non-functional features and as such are prioritized very low.

Detailed Use Cases

Use Case

Movement

Main Course of Events

Actor	System
1. The player indicates they want to move.	

	2. The player is moved in the specified direction.

Alternative Flow of Events

2. The player isn't moved due to their path being blocked or being stuck.

Use Case

Look Around

Main Course of Events

Actor	System
1. The player indicates they want to look in some direction.	
	2. The player is shown what is in the specified direction.

Use Case

Battling an enemy

Main Course of Events

Actor	System
1. The player encounters an enemy	
	2. A fighting sequence is initiated.
3. The player and the enemy trade attacks in turn-based combat.	
4. The player emerges victorious.	

Alternative Flow of Events

4. The player is defeated.

Use Case

Exit game

Main Course of Events

Actor	System
1. The player's health drops to 0.	
	2. The player is informed they have lost the game.
	3. The system exits the game and closes the window.

Alternate course of events

1. The player manually closes the window.

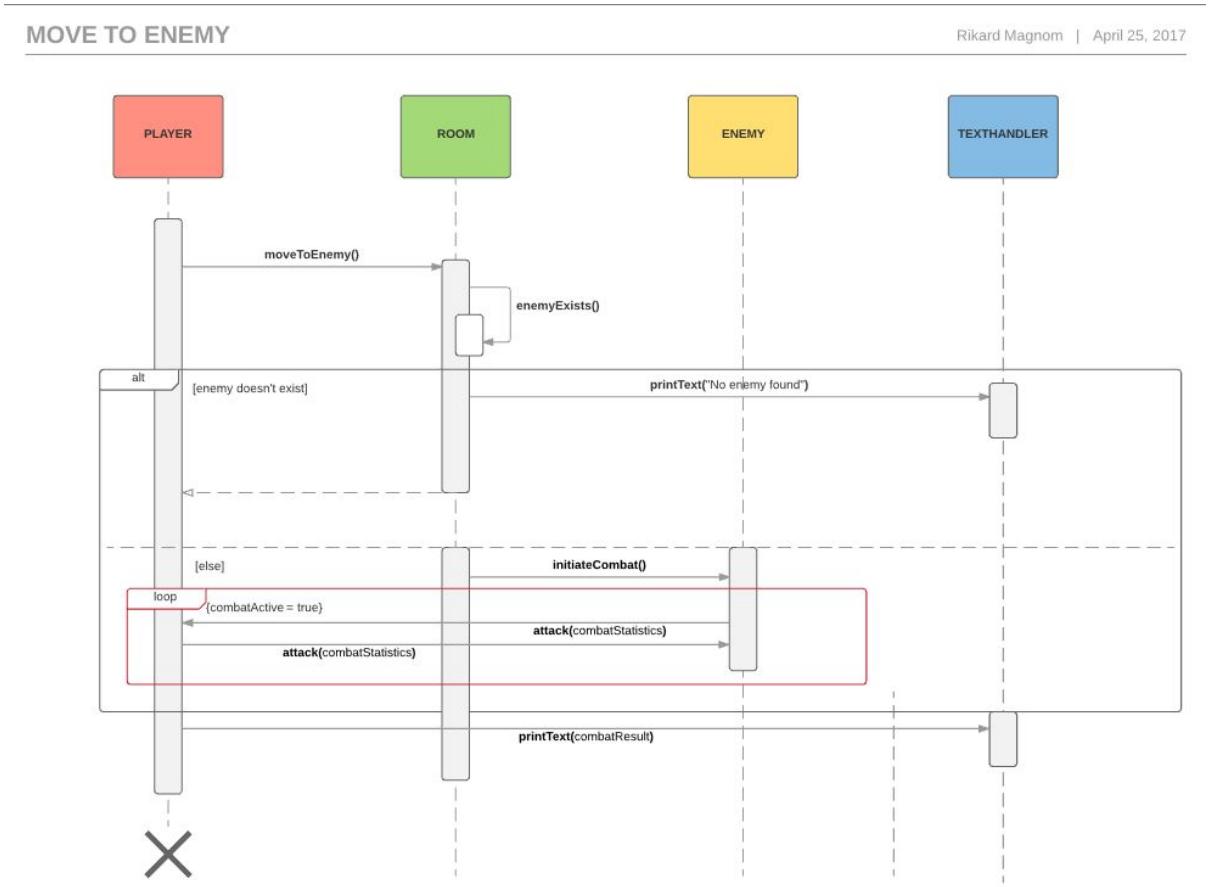
‘

Interaction diagrams

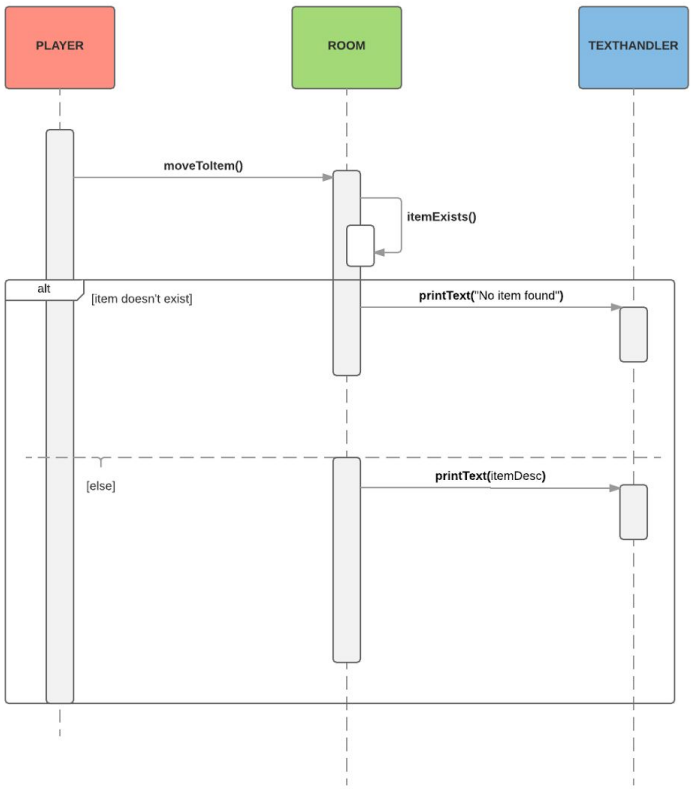
Use Case

Movement

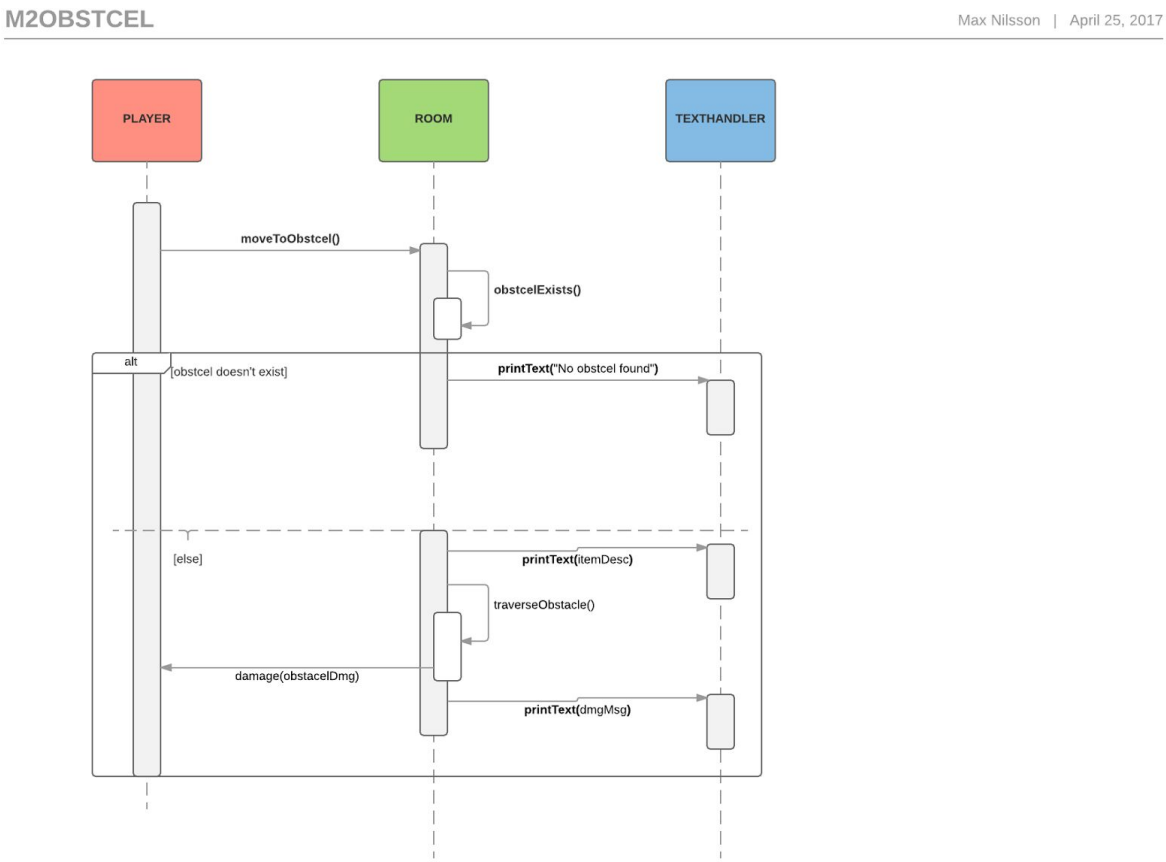
Move to enemy



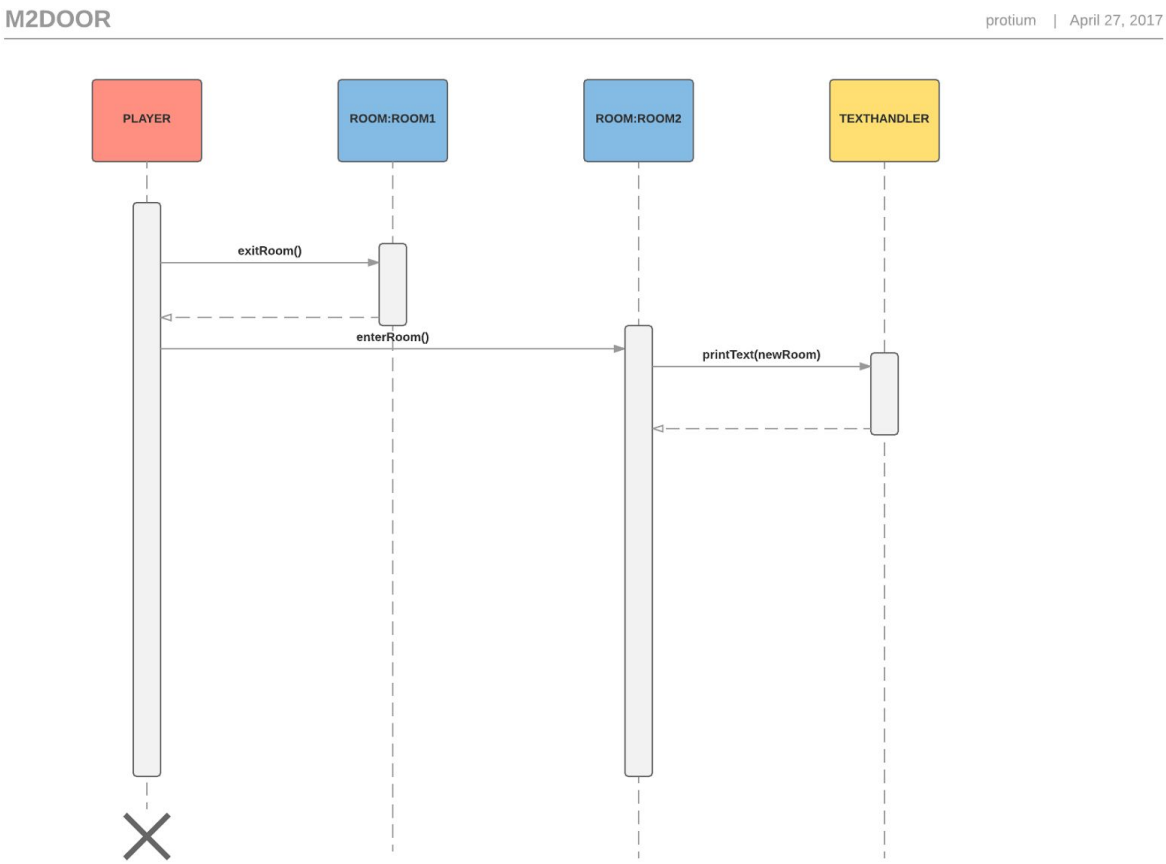
Move to item


PLAYER
ROOM
TEXTHANDLER
moveToItem()
itemExists()
alt
[item doesn't exist]
printText("No item found")
[else]
printText(itemDesc)

Move to obstacle



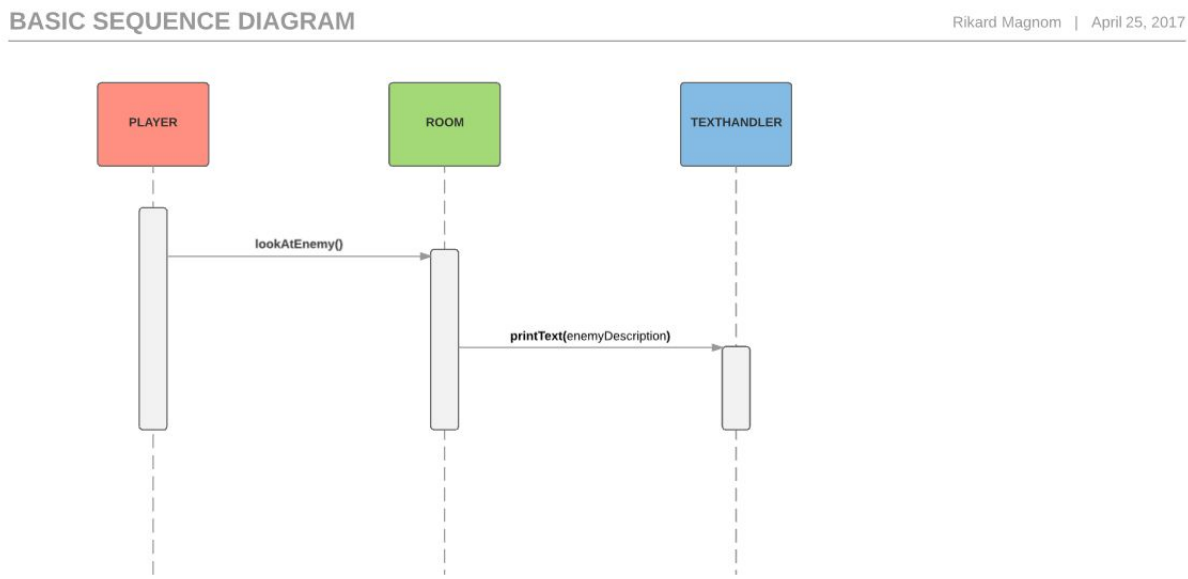
Move to door


M2DOOR
protium | April 27, 2017
PLAYER
ROOM:ROOM1
ROOM:ROOM2
TEXTHANDLER
exitRoom()
enterRoom()
printText(newRoom)

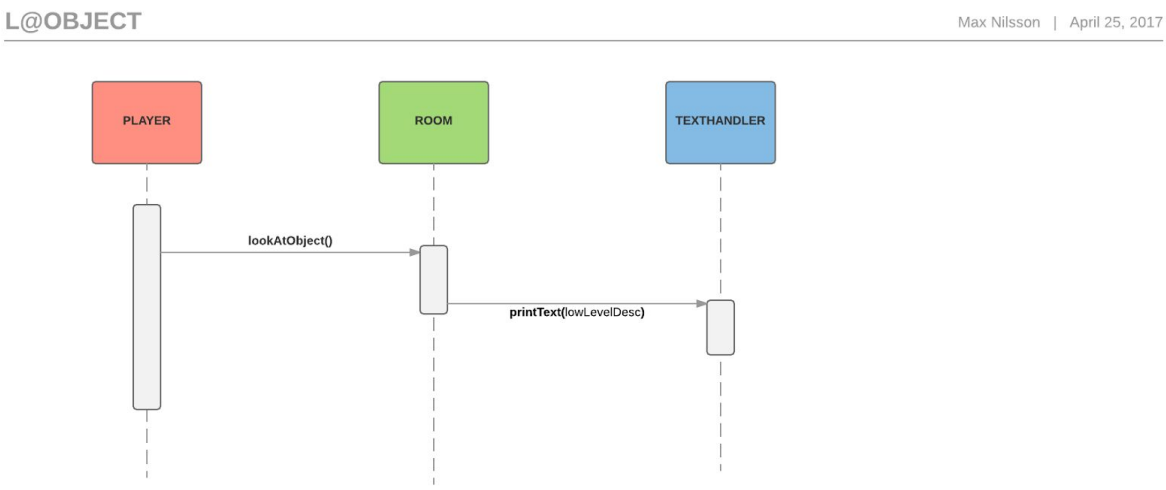
Use case

Look around

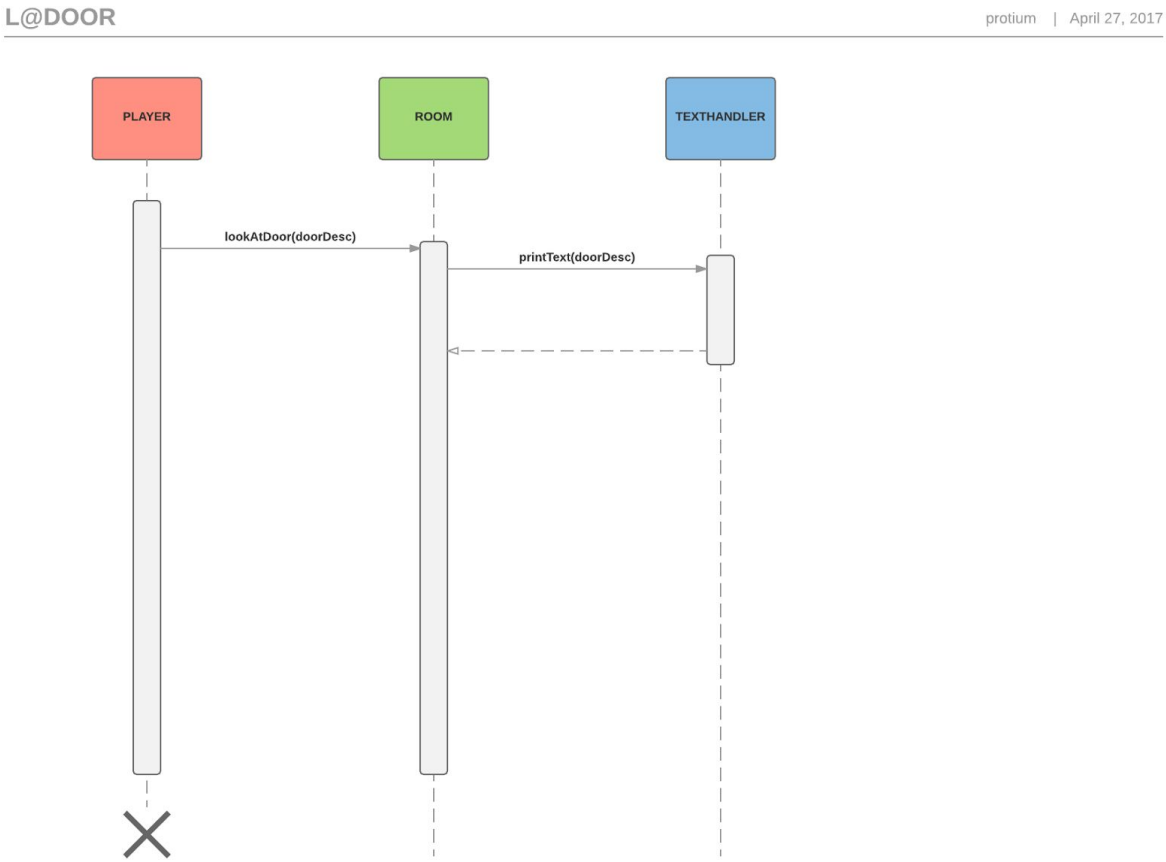
Look at enemy



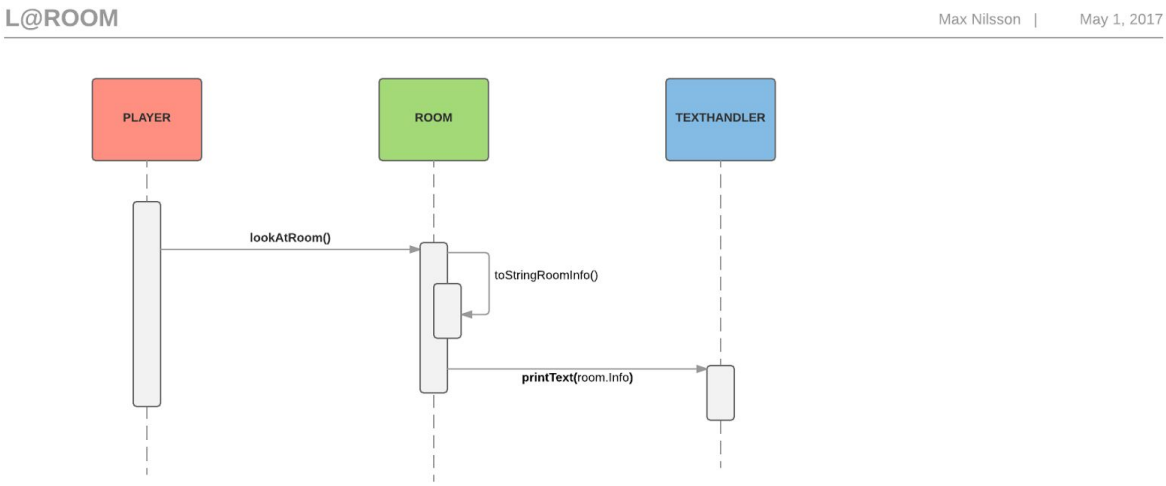
Look at object



Look at door


L@DOOR
protium | April 27, 2017
PLAYER
ROOM
TEXTHANDLER
lookAtDoor(doorDesc)
printText(doorDesc)

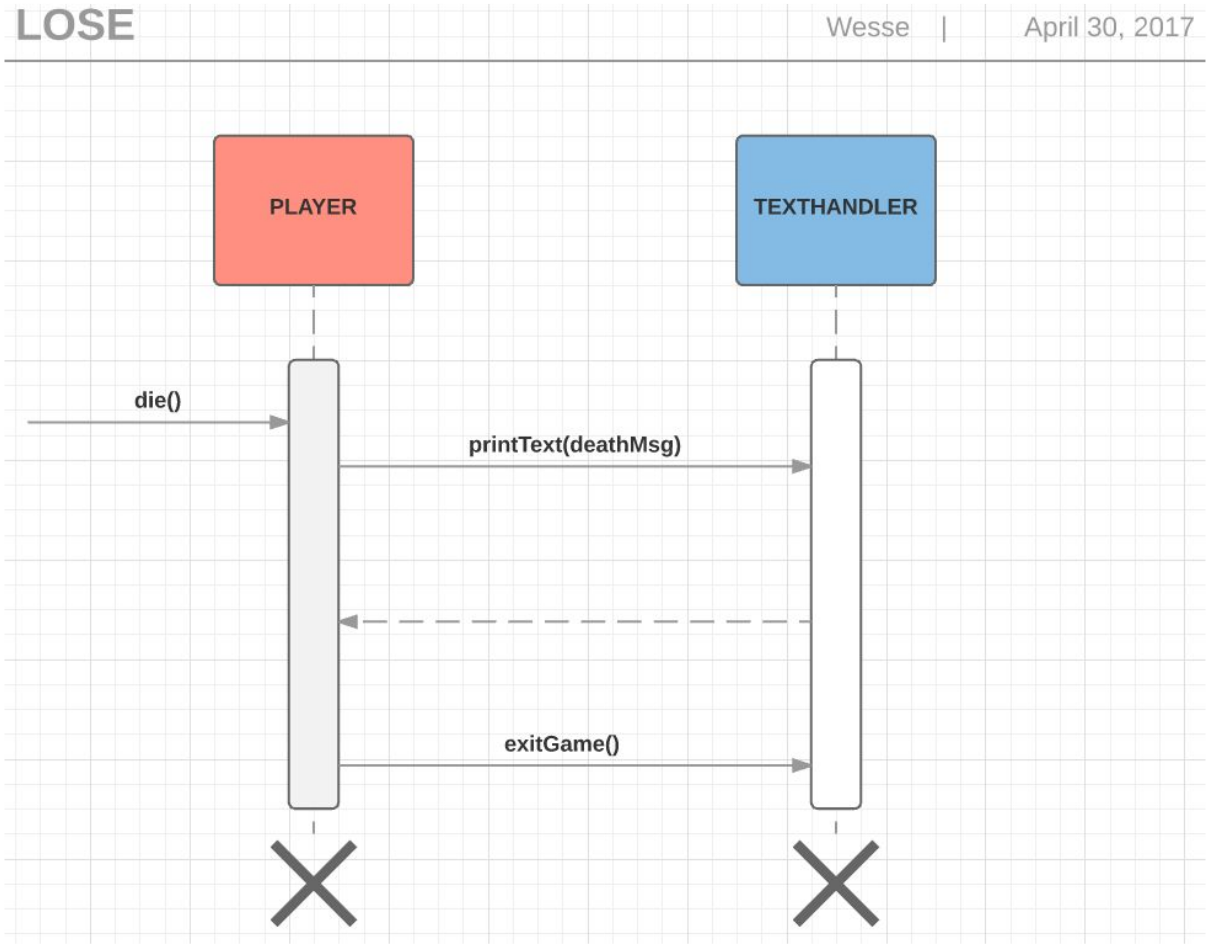
Look around room



Use case

Exit game

Player loses



Player exits