Darth Escape

Team Darth

Kevin Baffo, Jeremy Wu, Matthew Wilbern

February 15, 2018

TABLE OF CONTENTS

REVISION HISTORY

Revision Date	Changes Made	Name
February 15, 2019	- Pilot - Document created	Kevin Baffo, Jeremy Wu, Matthew Wilbern
Mar 10, 2019	- Updated missing components of document as per comments	Jeremy Wu
Mar 18, 2019	- Updated the class diagrams	Matthew Wilbern

Introduction

Welcome aboard to the Dark Shuttle, where your darkest dreams come true. Unfortunately, the ship was attacked Trump Industries while you were asleep and you landed on Mucho Gordo, a planet that relies on oxygen. You are Obi Juan Kenobi, a space warrior that just woke up to a beeping alert, notifying you that there is a gas leak inside the ship, which could ultimately kill you. Due to the gas and the collision. you are currently dazed and confused without understanding what really happened, but your oxygen levels are extremely low. You find out your current location, but must find out how to exit this ship to survive. In your journey of survival, there will be items that you will require to exit the ship...but you may even find items that will extend your oxygen bar.

The goal of this project is to create a text-based adventure game where the player will attempt to escape and survive. We will implement features such as load/save and have a basic implementation of A.I. to allow the player to interact with different classes of characters. This document will cover the roles and responsibilities of team members, listing a number of unfortunate scenarios that may hinder the development process, followed by a plan to deal with the situation if it occurs. We will explain our design choices, followed by diagrams listed in the appendices of the document.

Project Management

The project management section will cover the team organization and risk management. Team organization will cover each member's individual responsibilities, and risk management will cover precautionary measures for worst case scenarios.

Team Organization

Team lead: Jeremy Wu

- The team lead acts as a project manager to ensure deadlines are met. He will also schedule meetings, and ensure group members are all equally contributing to the project.

Design Lead: Matthew Wilbern

- The design lead ensures the design is implemented correctly in the source code and is being followed. If the design gets changed then the design lead modifies it to appropriately accommodate the changes made, and also ensures good object oriented design principles are adhered to.

QA Lead: Kevin Baffo

- The QA lead ensures that unit tests are made and the source code is being tested as the project progresses. They will also ensure that source code is readable and is of good quality to ensure that quality control is maintained and ensures error checking to be used in the source code.

Documentation Lead: Jeremy Wu

- The documentation lead ensures that the manual and the project report are created. He also ensures code is commented correctly and can be readable. He will also ensure that version control software will be used to show iterative version control history. Will also update source code's help section is maintained.

All team members will fill the roles of Software Developer and Software Tester. We will distribute implementation and testing work as evenly as possible between all team members. Our team will have weekly meetings in which work distribution, work schedule, and progress will be discussed. Internal deadlines will also be created to ensure a finishing product can be presented upon actual deadline. This also ensures that deadlines will not be missed, and will also provide cushioning for time if unforeseen circumstances ever arise.

Risk Management

Foreseeable risks include but not limited to:

- The team planned a project that cannot be completed before due date
    - Simplify game functionalities

- Team members underestimated how long parts of the project would take
    - The team will work together to complete the assignment by due date

- Major changes to design are needed during implementation
    - This is expected as this is our first attempt at group work, and will do the necessary steps to meet deadline while meeting specifications

- Loss of a team member
    - In the event that a team member has lost their role, workload is distributed evenly between remaining team members

- Unproductive Team Members
    - Will be addressed in team meetings first. If unresolved, a consultation with Dr.Anvik will be required

- Inexperienced Team Members

    - Offer help for trivial blockages and point teammate to the right direction

- Illness
    - Team will carry the ill member

- Unanticipated Life Events
    - Contact and carry on without team member until their return
    - Upon return, team member will be updated and reassigned their original position

- Inexperience with new tools
    - If other team members do not understand how to use required tools, we will research how to use these tools

- Learning curve for tools steeper than expected
    - Members will take the necessary time to learn the tools as fully understanding these tools make the entire project significantly more efficient

- Tools don't work together in an integrated way
    - Research and find an equivalent tool that would work with the integration

Development Process

In this section, we address how software will be developed and what coding conventions will be used, the procedures for configuration management, code review process, tools on how we communicate, and how to deal with change management.

Coding Conventions

We will used lower camelcase for variables and methods/functions and will use upper camelcase for classes. We will use the capitalization with underscore notation for constants and enumeration types. Google formatting will be applied. Comments will be precise with a good explanation in a format that each group member will understand.

Procedures for configuration management

Forking of the main repository, which is the Design Lead's repository, where each member will be responsible for pulling and pushing code to. This allows us to freely work on individually assigned parts. We can push to the forked repository and pull from the main one to apply changes. This will allow minimum conflicts as we are all working on separate documents. The only conflict that will arise is if multiple uses work on the same file.

Code Review Process

The code will be mainly reviewed by Matthew. Jeremy and Kevin will provide input if there are issues in the code. The code is then pulled only when it is ready. Once the pull request is complete, then everyone can pull from the main repository and add that code to their CodeBlocks program.

Communications Tools

Discord will be the primary tool of communication. This allows for group voice chats in case meetings at school are not possible. Discord also allows group chat which will allow ease of communication across all team members. Emails will be used if major updates are applied to the code to let team members know

Change Management

Bug reports will be assigned to the author of the piece of code. For example, if Jeremy receives a bug report, only Jeremy will correct the issue as it is the responsibility of the author. Other team members can choose to help if required.

Software Design

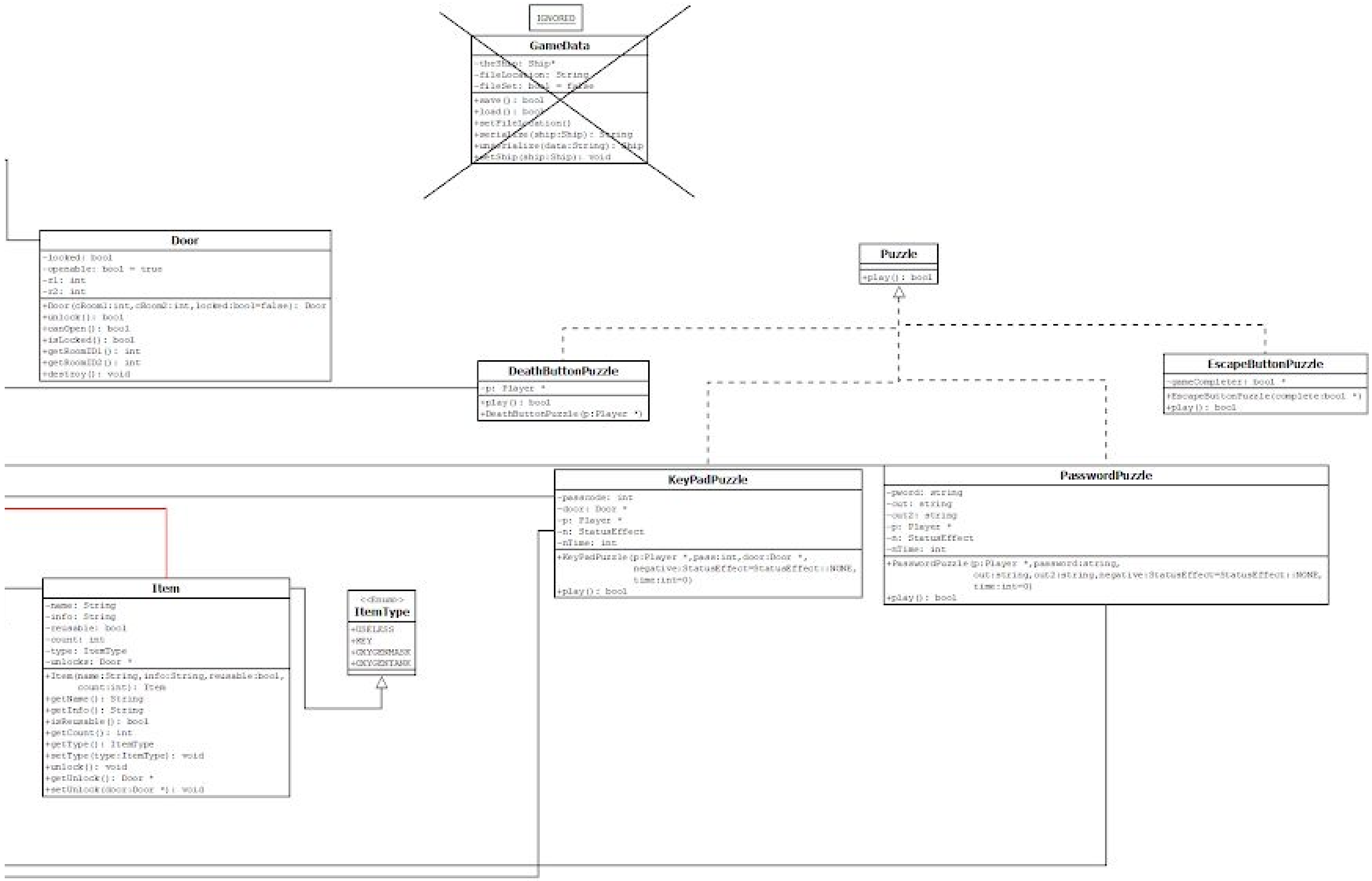
In this section, the overall design of the project and classes used will be discussed by diagrams.

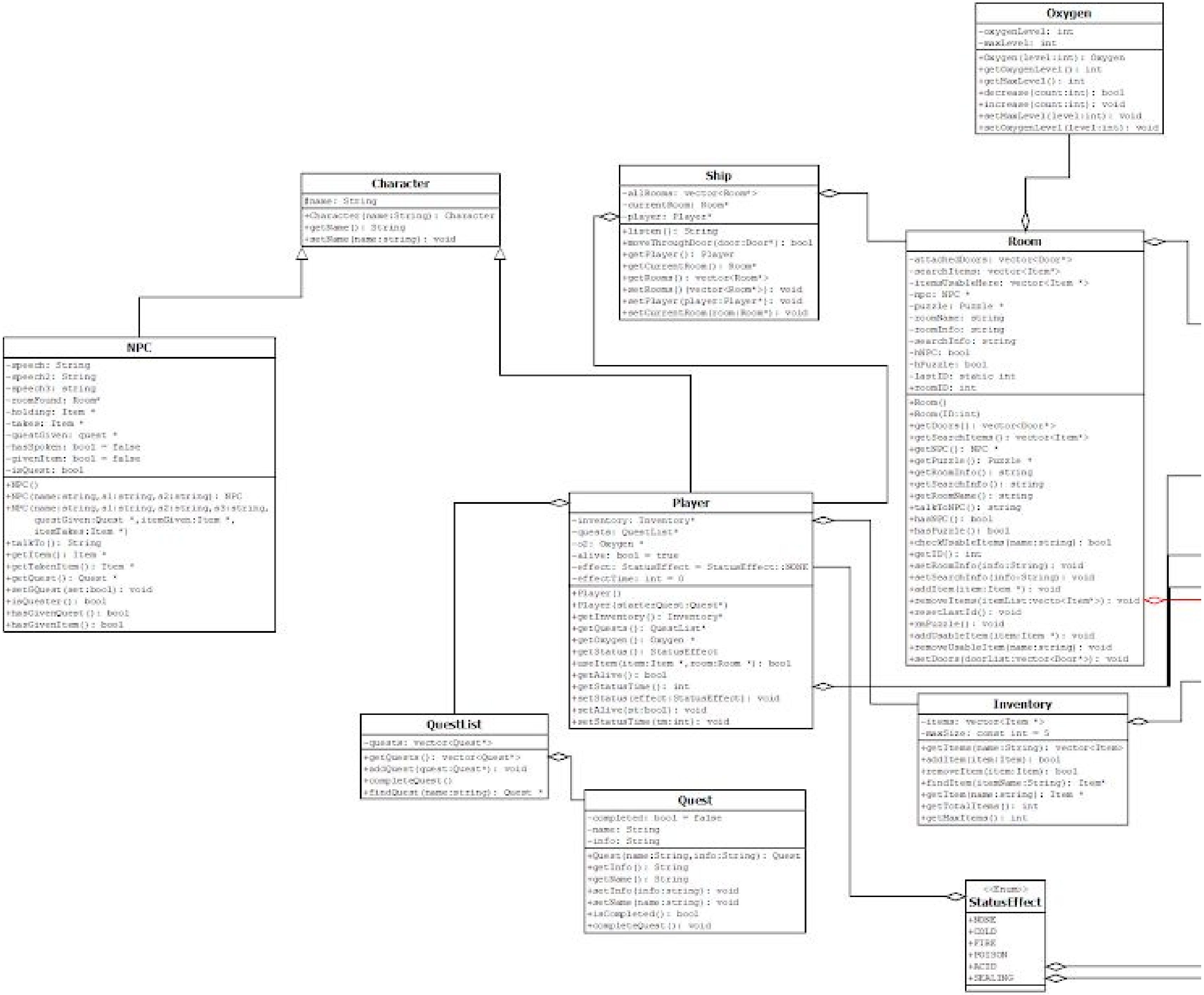
Appendices

This subsection of software design will include the UML diagrams as well as the sequence diagrams for implementation.

APPENDIX A: FIGURES AND TABLES

CLASS DIAGRAM




Oxygen
-oxygenLevel: int
-maxLevel: int
+Oxygen(level:int): Oxygen
+getOxygenLevel(): int
+getMaxLevel(): int
+decrease(count:int): bool
+increase(count:int): void
+setMaxLevel(level:int): void
+setOxygenLevel(level:int): void
Character
#name: String
+Character(name:String): Character
+getName(): String
+setName(name:string): void
Ship
-allRooms: vector<Room*>
-currentRoom: Room*
-player: Player*
+listen(): String
+moveThroughDoor(door:Door*): bool
+getPlayer(): Player
+getCurrentRoom(): Room*
+getRooms(): vector<Room*>
+setRooms()(vector<Room*>): void
+setPlayer(player:Player*): void
+setCurrentRoom(room:Room*): void
Room
-attachedDoors: vector<Door*>
-searchItems: vector<Item*>
-itemsUsableHere: vector<Item *>
-npc: NPC *
-puzzle: Puzzle *
-roomName: string
-roomInfo: string
-searchInfo: string
-hNPC: bool
-hPuzzle: bool
-lastID: static int
+roomID: int
+Room()
+Room(ID:int)
+getDoors(): vector<Door*>
+getSearchItems(): vector<Item*>
+getNPC(): NPC *
+getPuzzle(): Puzzle *
+getRoomInfo(): string
+getSearchInfo(): string
+getRoomName(): string
+talkToNPC(): string
+hasNPC(): bool
+hasPuzzle(): bool
+checkUsableItems(name:string): bool
+getID(): int
+setRoomInfo(info:String): void
+setSearchInfo(info:String): void
+addItem(item:Item *): void
+removeItems(itemList:vecto<Item*>): void
+resetLastId(): void
+rmPuzzle(): void
+addUsableItem(item:Item *): void
+removeUsableItem(name:string): void
+setDoors(doorlist:vector<Door*>): void
NPC
-speech: String
-speech2: String
-speech3: string
-roomFound: Room*
-holding: Item *
-takes: Item *
-questGiven: quest *
-hasSpoken: bool = false
-givenItem: bool = false
-isQuest: bool
+NPC()
+NPC(name:string,s1:string,s2:string): NPC
+NPC(name:string,s1:string,s2:string,s3:string,
questGiven:Quest *,itemGiven:Item *,
itemTakes:Item *)
+talkTo(): String
+getItem(): Item *
+getTakenItem(): Item *
+getQuest(): Quest *
+setGQuest(set:bool): void
+isQuester(): bool
+hasGivenQuest(): bool
+hasGivenItem(): bool
Player
-inventory: Inventory*
-quests: QuestList*
-o2: Oxygen *
-alive: bool = true
-effect: StatusEffect = StatusEffect::NONE
-effectTime: int = 0
+Player()
+Player(starterQuest:Quest*)
+getInventory(): Inventory*
+getQuests(): QuestList*
+getOxygen(): Oxygen *
+getStatus(): StatusEffect
+useItem(item:Item *,room:Room *): bool
+getAlive(): bool
+getStatusTime(): int
+setStatus(effect:StatusEffect): void
+setAlive(st:bool): void
+setStatusTime(tm:int): void
Inventory
-items: vector<Item *>
-maxSize: const int = 5
+getItems(name:String): vector<Item>
+addItem(item:Item): bool
+removeItem(item:Item): bool
+findItem(itemName:String): Item*
+getItem(name:string): Item *
+getTotalItems(): int
+getMaxItems(): int
QuestList
-quests: vector<Quest*>
+getQuests(): vector<Quest*>
+addQuest(quest:Quest*): void
+completeQuest()
+findQuest(name:string): Quest *
Quest
-completed: bool = false
-name: String
-info: String
+Quest(name:String,info:String): Quest
+getInfo(): String
+getName(): String
+setInfo(info:string): void
+setName(name:string): void
+isCompleted(): bool
+completeQuest(): void
<<Enum>>
StatusEffect
+NONE
+COLD
+FIRE
+POISON
+ACID
+SEALING

SEQUENCE DIAGRAMS

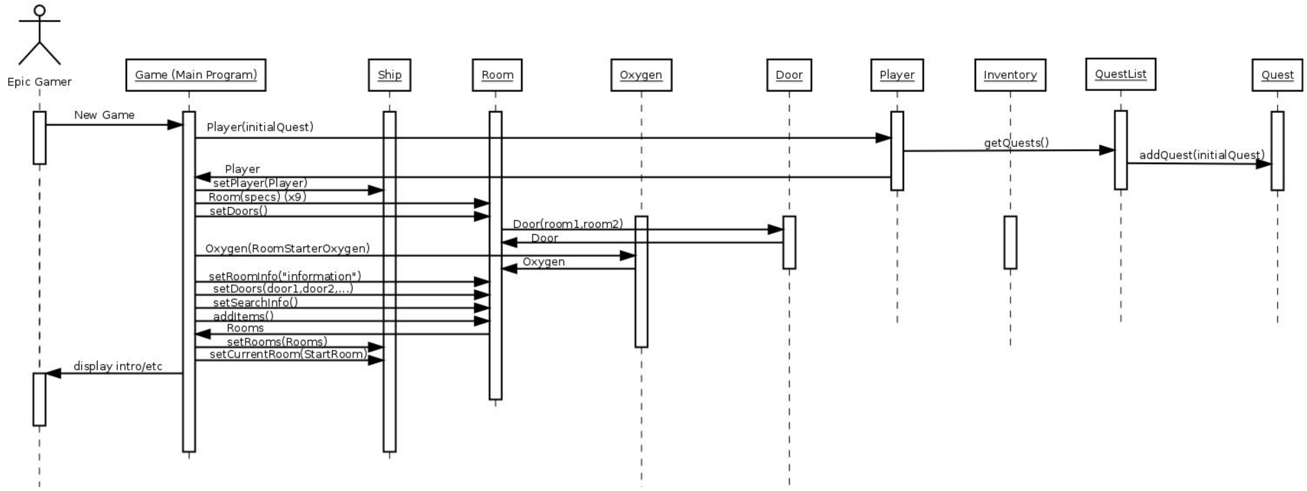


Diagram A: Sequence diagram for game implementation

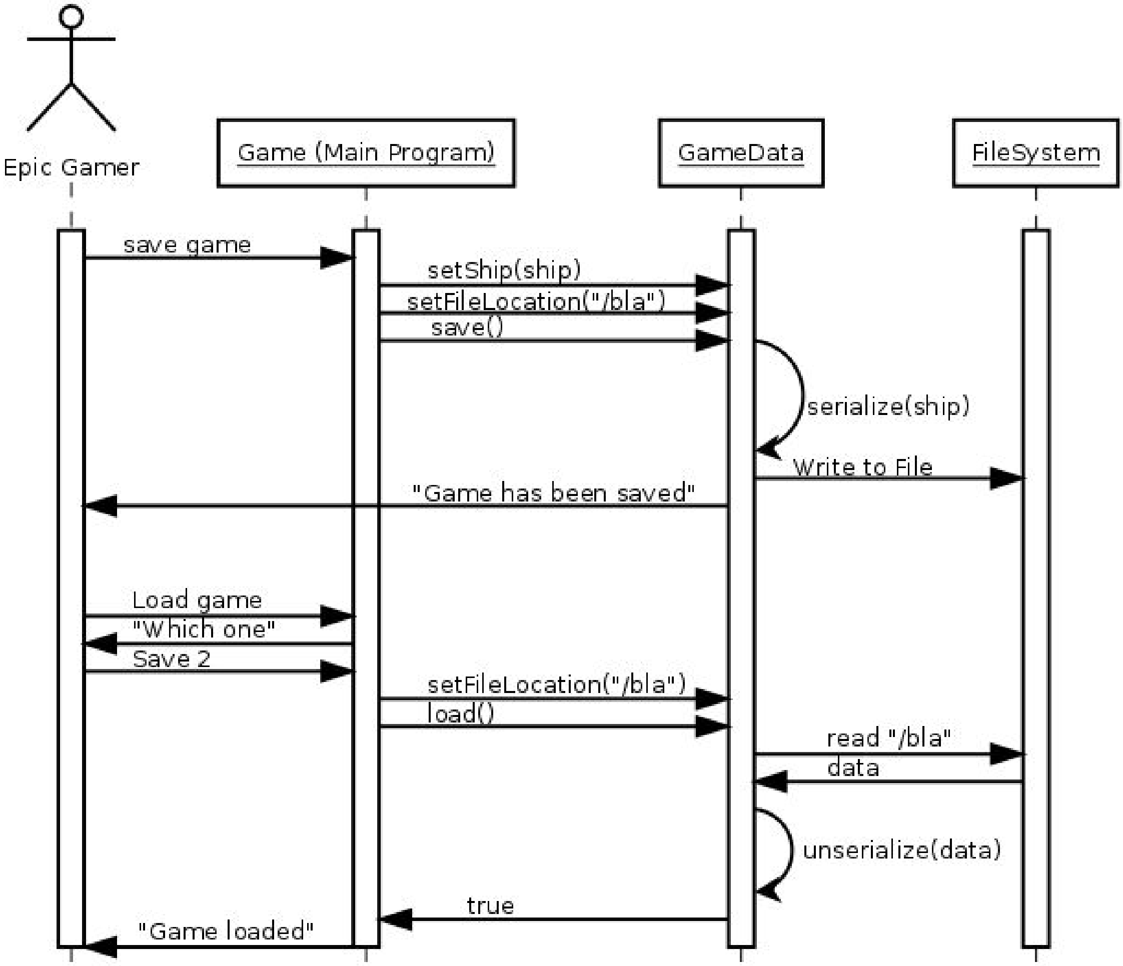


Diagram B: Sequence diagram for load/save data