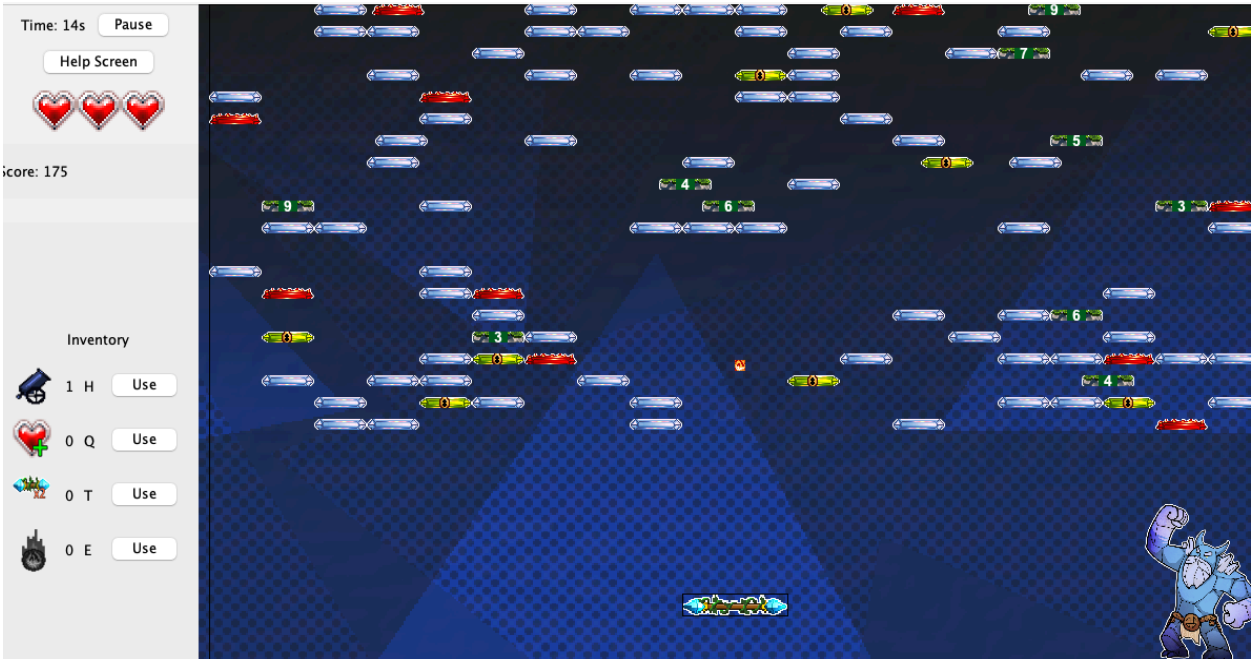
COMP 302 Project

Lance of Destiny
Final Report

31.05.2024



Group: Waterfall

Şebnem Demirtaş, 76813
Melih Okşak, 75621
Oğuz Kağan Hitit, 76757
İdil Görgülü, 79323
Ata Halıcıoğlu, 75773
Mehmet Atakan Arslan, 80878

Table of Contents

1. Introduction

a. Vision

This report is specifically structured to comprehensively capture the requirements, objectives, features, and the progression path of our team for the "Lance of Destiny" project. Emulating the envisaged format, our focus is on developing a barrier-breaking themed game. This application is designed to provide an expansive range of functionalities, incorporating diverse user interface options for enhanced player engagement.

b. Positioning

Business Opportunity

"Lance of Destiny" combines fast-paced action with strategic gameplay, challenging players to break various unique barriers using a fireball. Each barrier type requires different tactics, adding depth and variety to the game. The focus on speed and precision appeals to players who enjoy both reflex-based and strategic challenges. This game is designed to attract a wide range of players, offering an exciting mix of agility and strategy that stands out in the gaming landscape.

Problem Statement

In traditional brick-breaking games, players are often confined to simple gameplay, featuring one type of barrier and lacking additional features like spells. However, in "Lance of Destiny," we significantly elevate the gaming experience by introducing a diverse range of elements. This game features four distinct types of spells, each enhancing gameplay with abilities such as increasing chances of success or empowering the ball. Additionally, players encounter four unique kinds of barriers, including those that can drop spells, adding a strategic layer to the game. By integrating these innovative features, we offer a dynamic and enriched gaming experience, transcending the conventional limits of brick-breaking games. This approach not only diversifies the player's tactical options but also enriches the overall user experience, positioning "Lance of Destiny" as a groundbreaking game in the realm of modern interactive entertainment!

Product Position Statement

"Lance of Destiny" caters to those who enjoy the thrill of timed challenges, offering a versatile gaming experience suitable for all age groups. It targets enthusiasts of dynamic games, where changing states and evolving gameplay keep the excitement alive. Our goal is to provide an engaging escape, allowing players to momentarily forget their fatigue and immerse themselves in

a world of interactive fun. This game aspires to be a refreshing break from the everyday, delivering joy and a sense of achievement to a diverse audience.

c. Stakeholder Descriptions

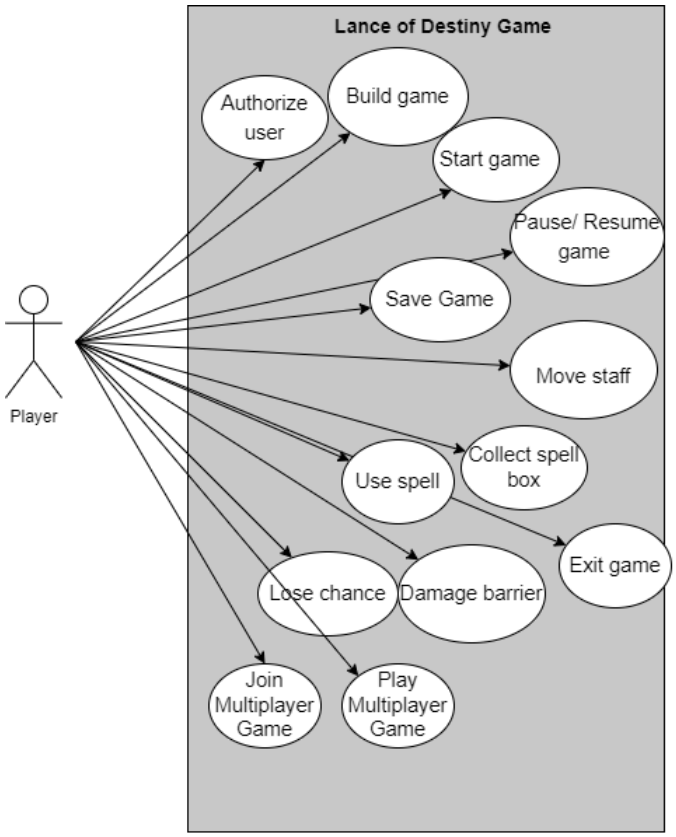
Players of this game are people from all ages who are bored from the barrier-breaking games.

2. Teamwork Organization

At the onset of our working process, we employed Github tools efficiently to ensure an adequate workload distribution. During the initial process of creating the game objects, Şebnem undertook the task of working on the Building Mode whereas Oğuz Kağan and Melih worked collaboratively on handling the overall collision logic, fireball movements and staff movements. Ata was tasked with the development of the required logic to store information about the game and the user within the database. İdil was responsible for handling barrier movements and special barrier ability implementations such as spell dropping and debris dropping. Atakan worked on the login and signup page and the necessary button additions to present a smooth GUI transition between different game panels. During the execution of the running mode and other required components for Phase 1, a cooperative approach was filled. A similar spirit can also be seen during the preparation of the necessary documentation to be submitted. During each sub-phase requiring the submission of UML diagrams, the SSDs, Use Cases, Communication diagrams and Class Diagrams were distributed evenly to each team member to work on.

The second phase of the project included the addition of various new features. During the development process of this phase, Melih was assigned the implementation of Hex, Felix Felicis, Overwhelming Fireball and Magical Staff Expansion, which were optional for the Phase 1 demo. İdil worked on the integration of Ymir and its abilities Hollow Purple, Infinite Void and Double Accel to the game. Şebnem was responsible for the construction of necessary methods to save previously-played game sessions to the database with every game component detail. Ata and Oğuz Kağan worked cooperatively on the development of multiplayer game components. Atakan worked on the integration of game panels including the Help Screen, and UI components such as back buttons and inventory displays. Getting motivation from the power of the team's spirit, the team effectively utilized collaborative effort which concluded in the successful development of Lance of Destiny.

3. Use Case Diagram



4. Selected Use Case Narratives

Use Case UC1 — Build Game

Use Case Name	Build Game
Scope	Building Mode
Level	User Goal
Primary Actor	Player
Stakeholders and Interests	Player wants to build the game
Preconditions	1. Game is installed and launched 2. Player is authenticated, and on the Main Menu. 3. Player selected the Building Mode
Success Guarantee (Postconditions)	1. Player has successfully built a game template according to the game rules, including placing all necessary barriers with the right amount of barrier count. 2. The game template is saved and ready for immediate play or future use. 3. The player is transitioned to the "Running Mode" to play in the newly created environment or has the option to save their template.
Main Success Scenario	1. Player selects the "Building Mode" from the main menu. 2. The system presents an interactive grid where the player can design their game template. 3. The player selects barrier types and positions them within the grid according to strategic preferences and game rules. 4. Once satisfied, the player finalizes the template setup. 5. The system performs a final check to ensure all game rules are met and saves the created template. 6. The player is given the option to start playing in the newly created environment immediately or save it for future use.
Extensions	*a. Player can exit and go back to the Main Menu by clicking the red cross button at the right top of the pop-up.

	3a. Player chooses Saved Game Templates 1. Player can choose from the Saved Game Templates, where it contains the built-in game templates and also the previously saved game templates of the player
Special Requirements	- Getting input from the mouse
Technology & Data Variations List	*a. Player chooses by the mouse click
Frequency of Occurrence	Every time user clicks the Building Mode

Use Case UC2 — Move Magical Staff

Use Case Name	Move Staff
Scope	Running Mode
Level	Subfunction
Primary Actor	Player
Stakeholders and Interests	Player: Has the aim of changing the angle and horizontal position of the magical staff.
Preconditions	-The player must have run and initialized the game executable. -The player needs to press the corresponding keys that move the magical staff in the right-left direction (right arrow or left arrow) or keys that change the angle of the staff (A and D).
Success Guarantee (Postconditions)	**a)** Right arrow is pressed and released: The Magical Staff should move right by an offset equal to L/2 with a speed of L/second. **i)** Right arrow is pressed and stays down: The speed becomes 2*L/second. **b)** Left arrow is pressed: The Magical Staff should move left by an offset equal to L/2 with a speed of L/second. **i)** Left arrow is pressed and stays down: The speed becomes 2*L/second. **c)** Key A is pressed: The rotation angle will change with a rate of 20 degrees/second until it reaches 45 degrees. **i)** A is pressed and stays down: The speed becomes 2*L/second. **d)** Key D is pressed: The rotation angle will change with the same rate until it reaches 135 degrees. **i)** D is pressed and stays down: The speed becomes 2*L/second.
Main Success Scenario	**1)** The player presses the corresponding key which is A, D, right arrow or left arrow, depending on the desired movement. **2)** If the player keeps pressing the key the desired movement gets implemented with a speed of 2*L/second. **3)** If the player releases the key, the magical staff moves by an offset equal to L/2 with a speed of L/second. **4)** The desired movement gets implemented on the gameplay screen.

Extensions	**1a)** Right arrow is pressed but the magical staff is already on the rightmost side of the screen: : the magical staff does not move. **1b)** Left arrow is pressed but the magical staff is already on the left most side of the screen: the magical staff does not move. **1c)** A is pressed but the rotation angle of the magical staff is already 45 degrees: the magical staff does not move. **1d)** D is pressed but the rotation angle of the magical staff is already 135 degrees: the magical staff does not move.
Special Requirements	**1)** Once one of the A or D keys is released the Magical Staff will go back to its horizontal state, and the rotation angle is going to be changed by a rate of 45 degrees/second. **2)** The Magical Staff length L is 10% of the screen width. **3)** The Magical Staff thickness T is 20px.
Technology & Data Variations List	The device the application is being run on needs to accept keyboard inputs.
Frequency of Occurrence	Every time the player presses A, D, right arrow or left arrow during the running mode of the game.

Use Case UC3 — Join Multiplayer Game

Use Case Name	Join Multiplayer Game
Scope	Running Mode
Level	User Goal
Primary Actor	Player
Stakeholders and Interests	Player: Wants to join a multiplayer game which was created by another player.
Preconditions	- The player must be already authorized.
Success Guarantee (Postconditions)	Player joins a game with another user and plays simultaneously.
Main Success Scenario	1. The player chooses the multiplayer mode. 2. The player chooses to join a game instead of hosting. 3. The player enters the IP address and port number of host player. 4. The player confirms they are ready to join the game. 5. The player successfully joins the game and starts playing.
Extensions	*a. At any time system fails: 1. Application is closed. *b. At any time network connection is disrupted. 1. A warning is shown to both sides. 2. If reconnection cannot be made in 5 seconds, both the host and the joiner leaves the running mode. 3a. No matches with the given information is found. 1. A warning is shown and the player re-enters information. 4a. The player does not confirm they are ready. 1. The player cannot join game. 2. The player waits until clicking. 5a. The player is able to join the game but cannot play. 1. A warning indicating to check network connection is displayed.
Special Requirements	- Getting input from a mouse - Clear indication of existing game
Frequency of Occurrence	Whenever the player wants to join a new game.

Use Case UC4 — Damage Barrier

Use Case Name	Damage Barrier
Scope	Running Mode
Level	User Goal
Primary Actor	Player
Stakeholders and Interests	Player: Directs the fireball to a barrier. Fireball: Hits a barrier and makes it disappear if the barrier's hit count gets to be 0.
Preconditions	- The player must be already authorized. - The game must be in running mode. - There are barriers present on the screen.
Success Guarantee (Postconditions)	A barrier is successfully broken. The player's score increases.
Main Success Scenario	1. The player directs the fireball towards barriers using the Magical Staff. 2. The fireball collides with a barrier. 3. Based on the type of the barrier, the appropriate action happens (barrier can just break, drop a spell, or explode) and the user's score increases based on the formula provided in the project info. 4. The player continues to play.
Extensions	3a. If the barrier is a simple barrier, it is broken in one hit. 3b. If the barrier is a reinforced barrier, it requires more than one hit to be broken. 3c. If the barrier is an explosive barrier, it is broken in one hit. 3d. When an explosive barrier is broken, it falls downwards, and if it hits the Magical Staff, the player loses a chance. 3e. If the barrier is a rewarding barrier, it is broken in one hit. 3f. When a rewarding barrier is broken, it drops a box downwards towards the Magical Staff. If the Magical Staff touches the box, the player wins a spell. *If all barriers are broken, the game ends.
Special Requirements	The game must handle different types of barriers and their specific behaviors. The physics of the Fire Ball's movement and its interaction with barriers and the Magical Staff must be

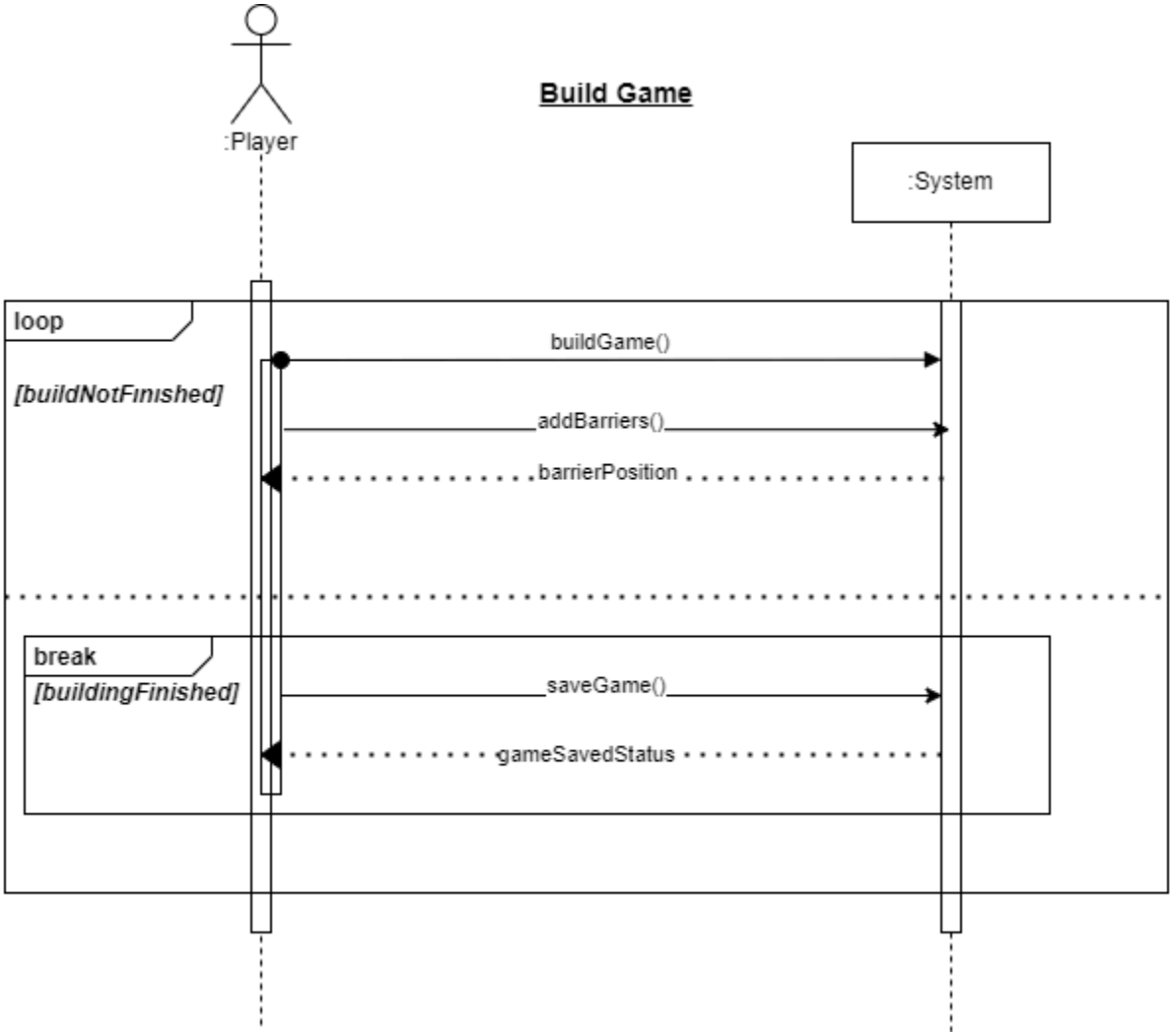
	accurately implemented.
Frequency of Occurrence	As frequent as a barrier gets hit.

Use Case UC5 - Collect Spell Box

Use Case Name	Collect Spell Box
Scope	Running Mode
Level	Subfunctional
Primary Actor	Player
Stakeholders & Interests	- Player : Wants to collect the spellboxes for later use.
Preconditions	- The game is still running. - A barrier was broken recently and the spellbox from it is still in the game domain.
Success Guarantee	Spell is added to the inventory, and removed from the game environment.
Main Success Scenario	1-Spellbox object is created upon breaking the rewarding barrier, and it starts falling. 2a- Magical staff touches the spellbox. 3-The spellbox opens and the corresponding reward spell is added to the player's inventory. 4- The player can now use the spell at any time
Extensions	2b- Spellbox falls to the ground. a) Spellbox is destroyed.
Frequency of Occurrence	Whenever a rewarding barrier breaks.
Miscellaneous	Reward inside the barrier is determined with the creation of the rewarding barrier, but the player would not know it until the spellbox is collected.

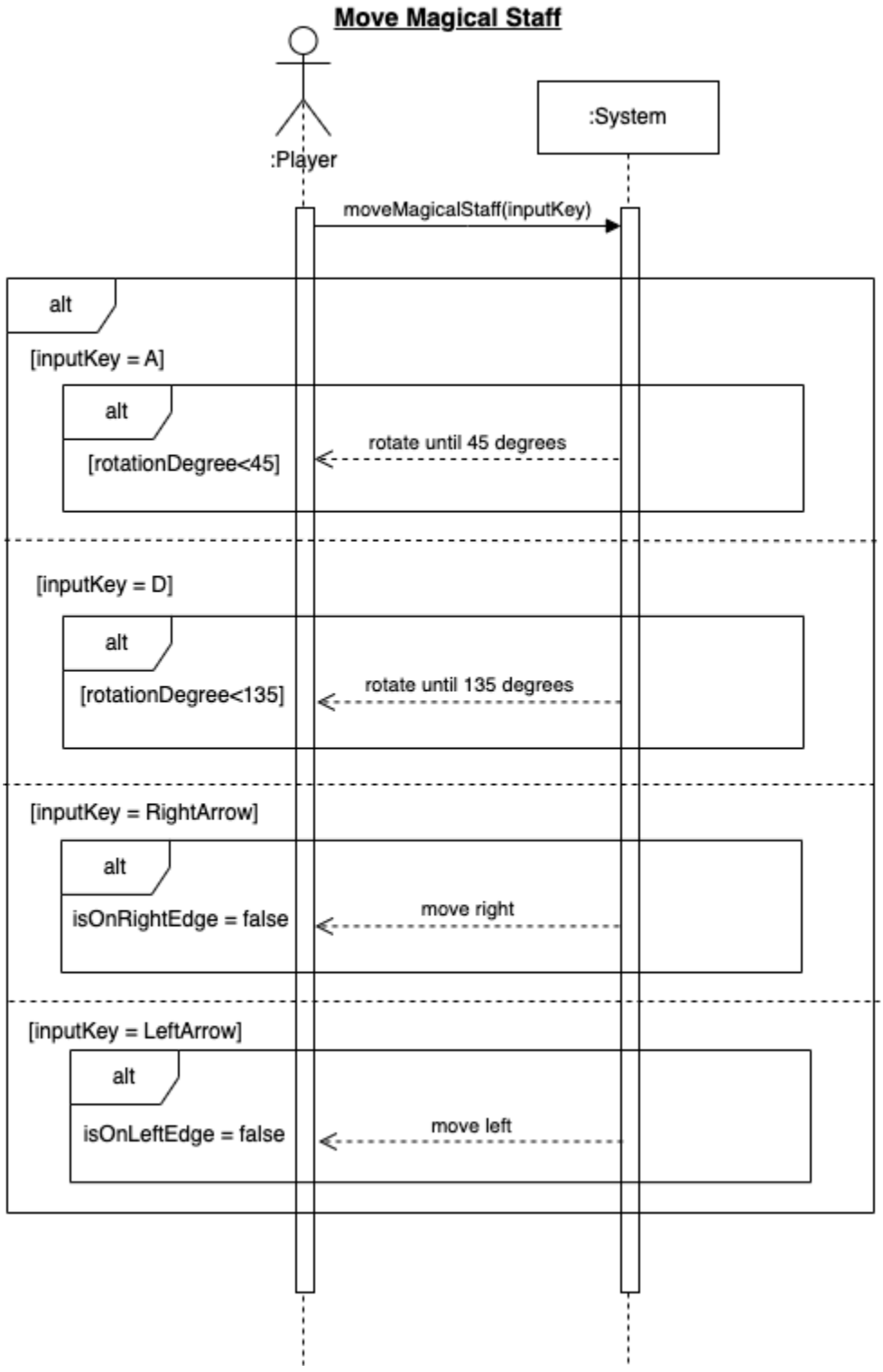
5. System Sequence Diagrams

SSD1 - Build Game

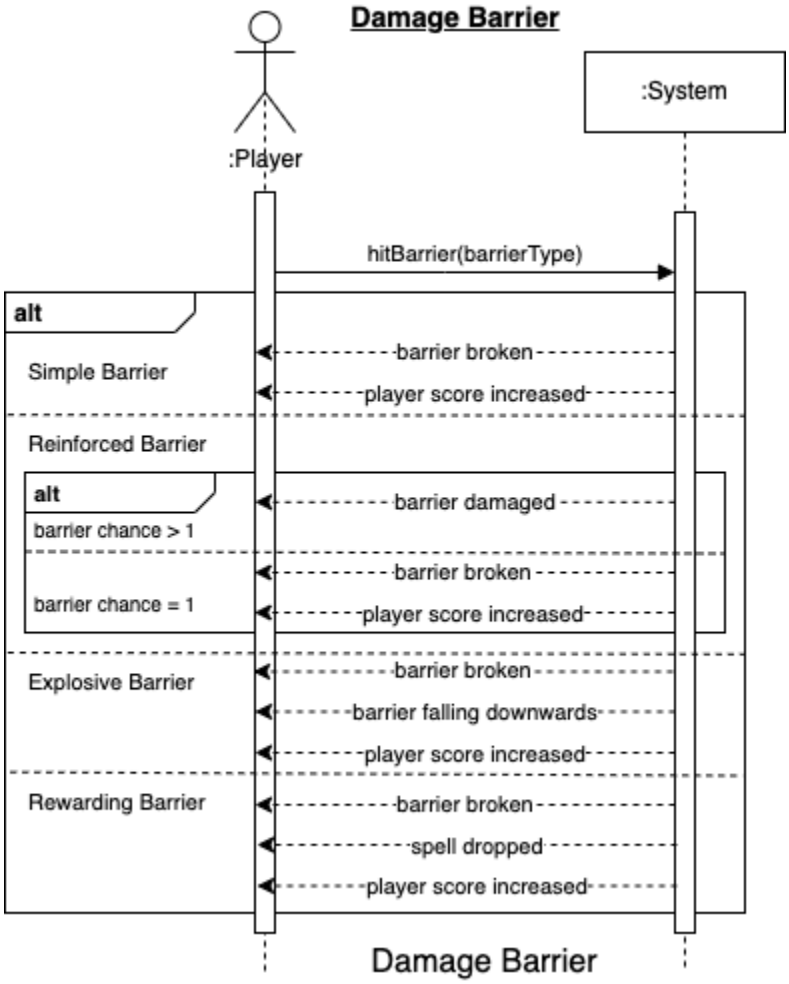


,

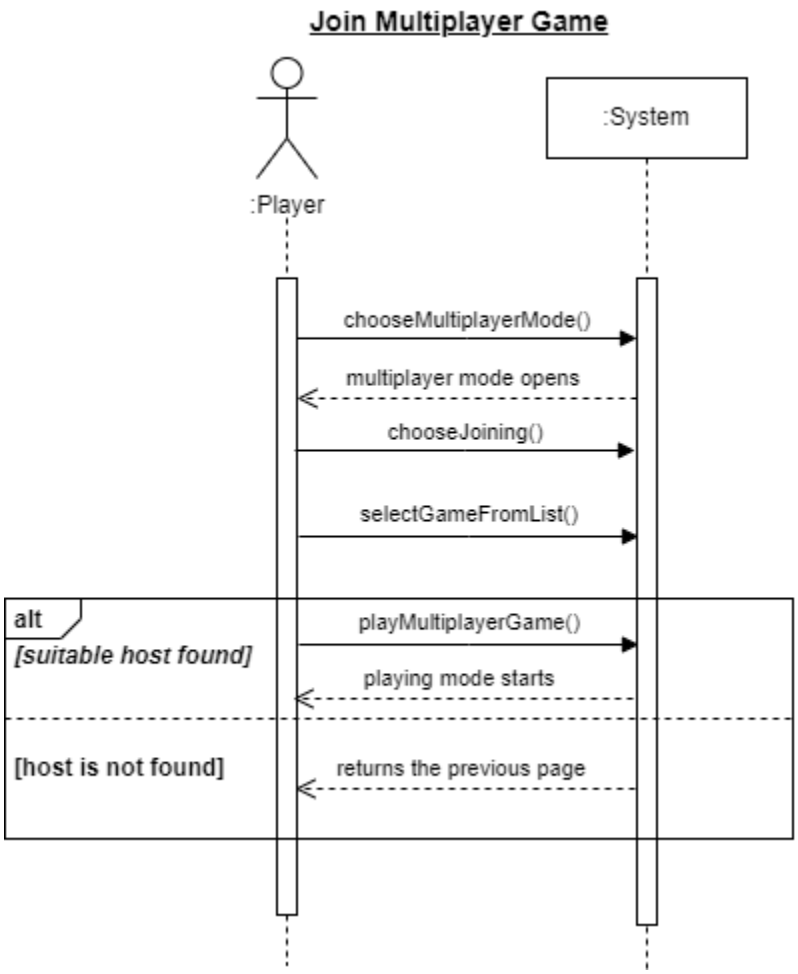
SSD2 - Move Magical Staff



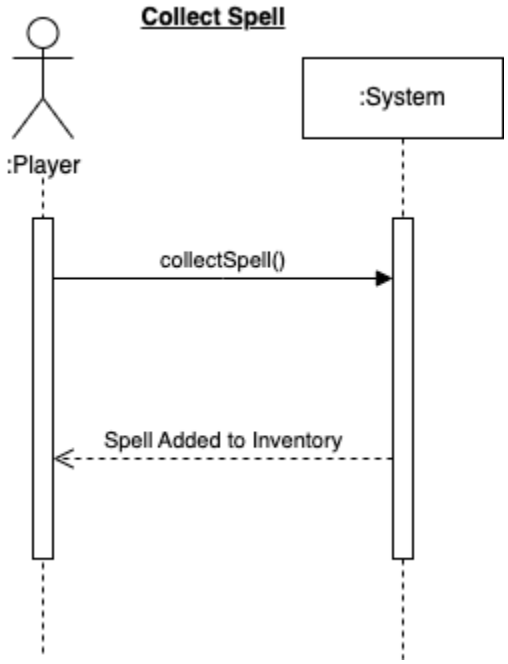
SSD3 - Damage Barrier



SSD4 - Join Multiplayer Game



SSD5 - Collect Spell



6. Operation Contracts

Contract CO1 — placeBarrier

Operation:	placeBarrier(coordinates: Coordinate, barrierType: BarrierTypeEnum)
Cross References:	UC1 - Build Game
Preconditions:	-The game is in building screen - Player selects a barrier and drags it to a place on the screen.
Postconditions:	- If the barrier is valid and positioned correctly according to the rules, the barrier was saved and added o the barrier array in game class. - Player can see the previously placed barrier in the screen, and can select and place another barrier.

Contract CO2 – slideMagicalStaff

Operation:	slideMagicalStaff(int: x)
Cross References:	UC2- Move Staff, UC5- Collect Spell Box
Preconditions:	-The player must have run and initialized the game executable. -The player needs to press the corresponding keys that move the magical staff in the right-left direction (right arrow or left arrow) or keys that change the angle of the staff (A or D).

Postconditions:

a) Staff's velocity changed according to the input parameter x.

Contract CO 3- checkBarrierFireballCollision

Operation: hitBarrier(barrierType)

Cross References: UC- Damage Barrier

Preconditions: -The player must be authorized.
-The game must be in running mode.
-There must be barriers present.

Postconditions:

- **a)** The barrier was a simple barrier: it was broken, score was updated.
- **b)** The barrier was a reinforced barrier, it requires more than one hit to be broken. If the required number of hits left was 1, it was broken., score was updated.
- **c)** The barrier was an explosive barrier: it was broken,, score was updated. Resulting debris started to fall downwards.
- **d)** The barrier was a rewarding barrier: it was broken, score was updated. The barrier dropped a box downwards.
- **e)** Infinite void was used on the barrier, barrier was not broken.
- **f)** Barrier was a hollow purple barrier, it was broken, it didnt increase the score.

Contract CO4 - JoinMultiplayerGame

Operation: joinMultiplayerGame(String gameName)

Cross References: UC- Join Multiplayer Game

Preconditions: -The player must select a game from the listed multiplayer games

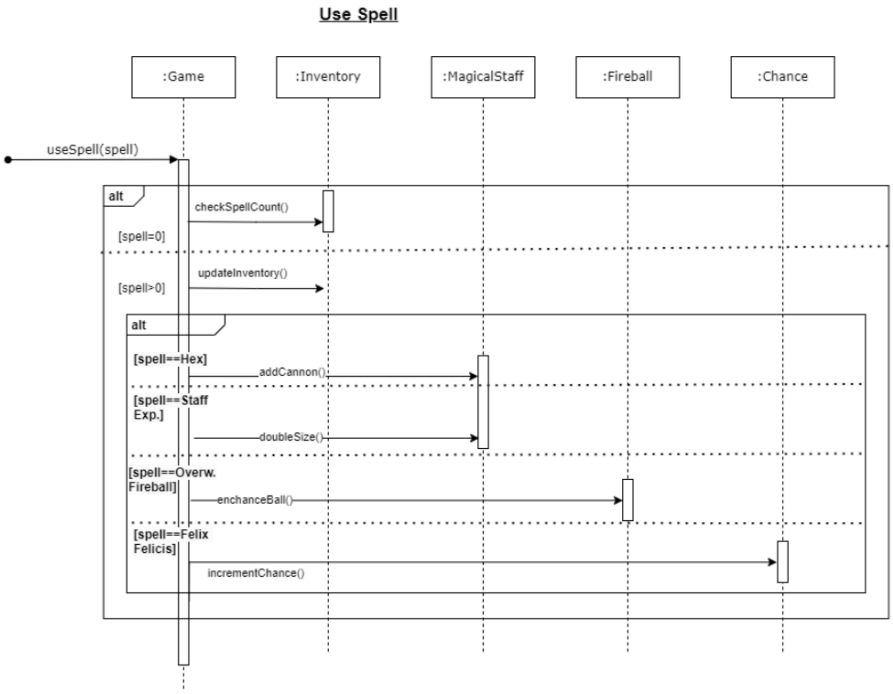
Postconditions: -The connection between the client and host was established
-The game was created according to how it is built
-A countdown had started from 3, and when it hit 0, game started

Contract CO5 – updateDroppingSpells

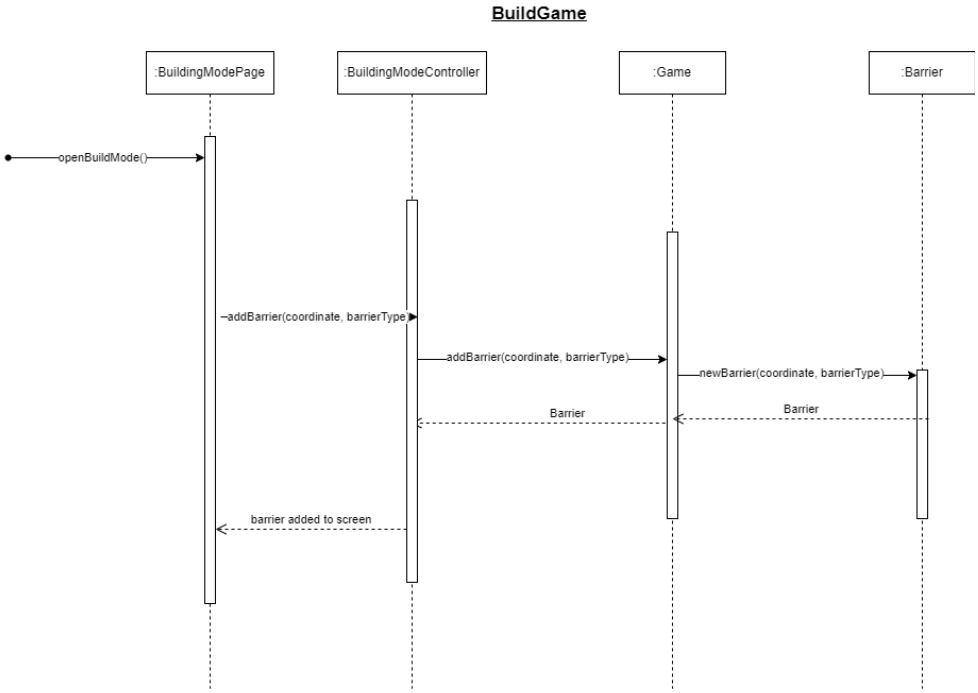
Operation:	updateDroppingSpells()
Cross References:	UC- Move Staff, UC- Collect Spell Box
Preconditions:	-A spellbox instance is in the game environment -Game is in the running mode and is not paused.
Postconditions:	-Magical staff touched the spellbox. -Spellbox opened and added the contained spell into the inventory.

7. Sequence and Communication Diagrams

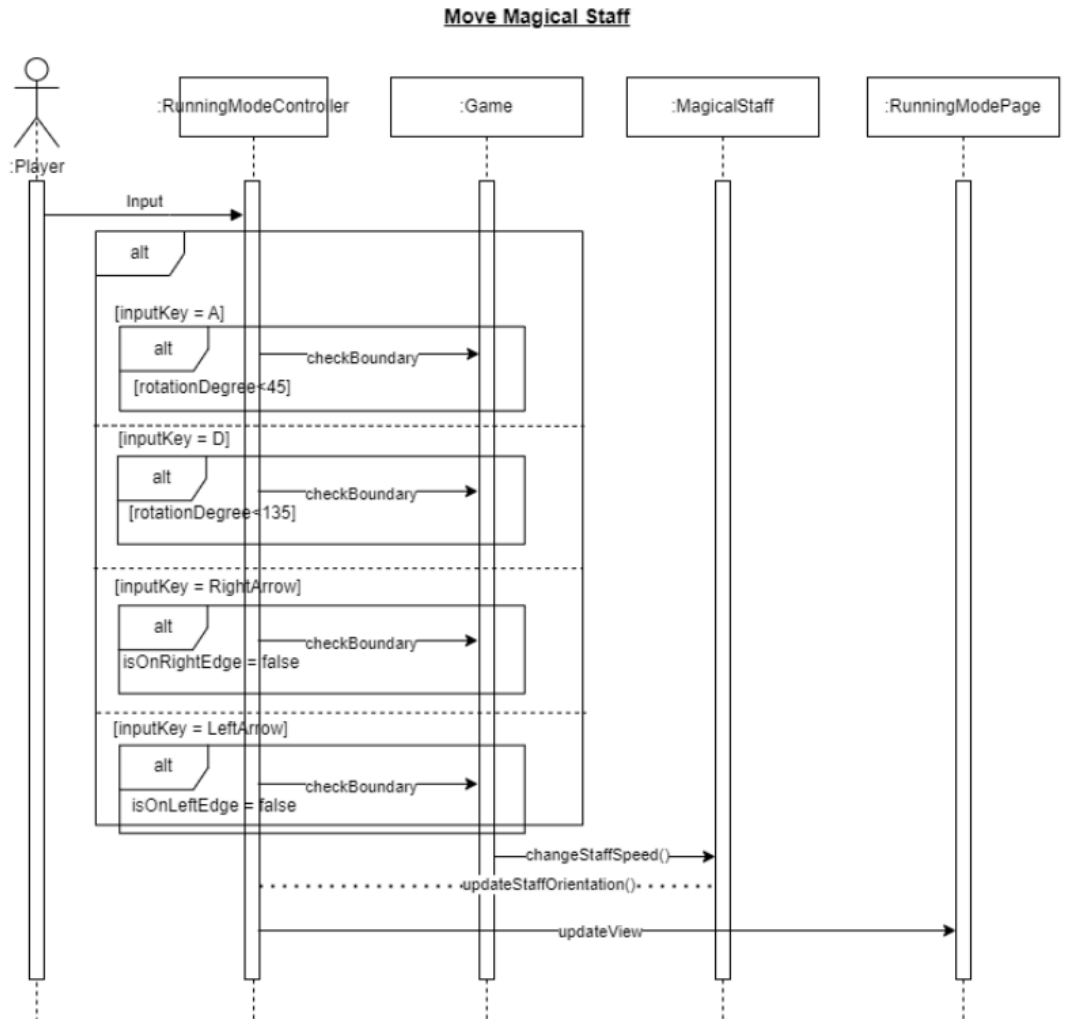
Sequence Diagram 1 - Use Spell



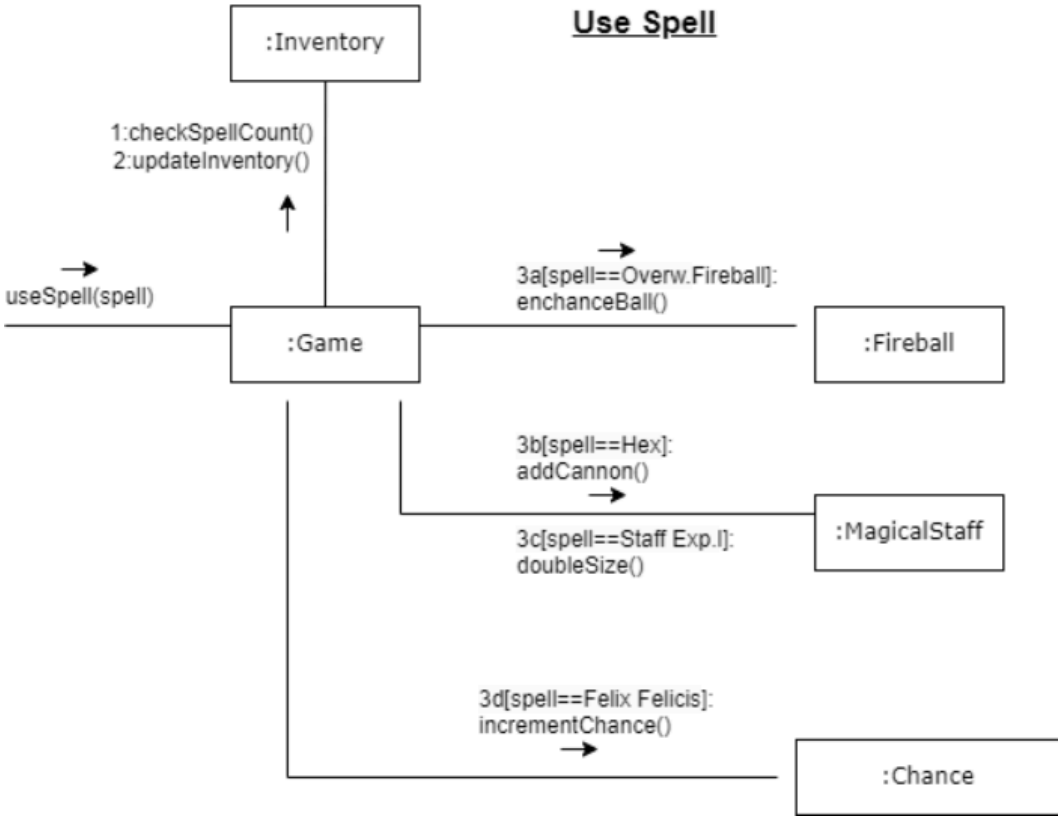
Sequence Diagram 2 - Build Game



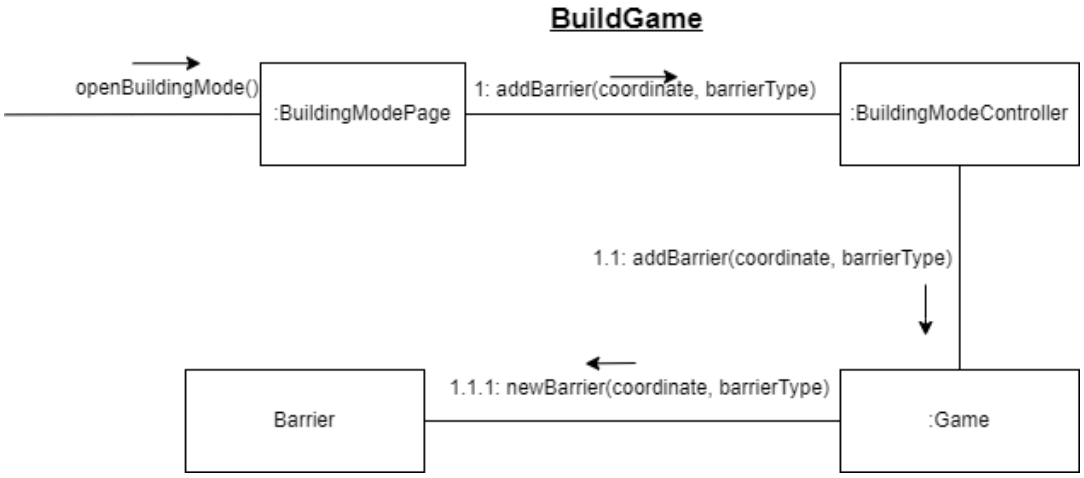
Sequence Diagram 3 - Move Magical Staff



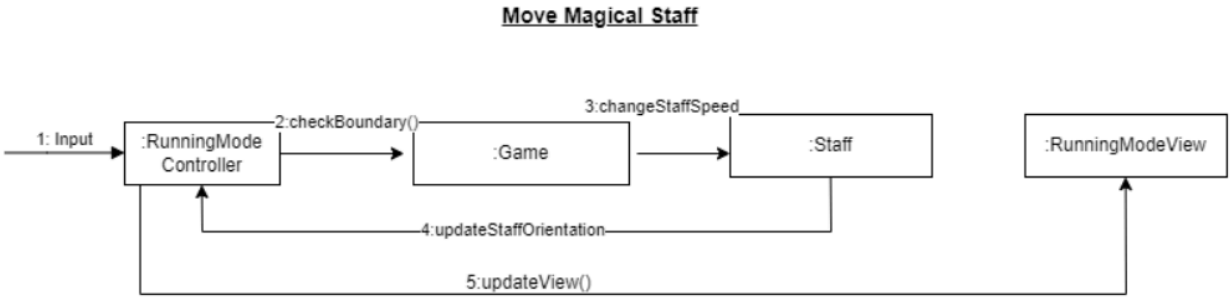
Communication Diagram 1- Use Spell



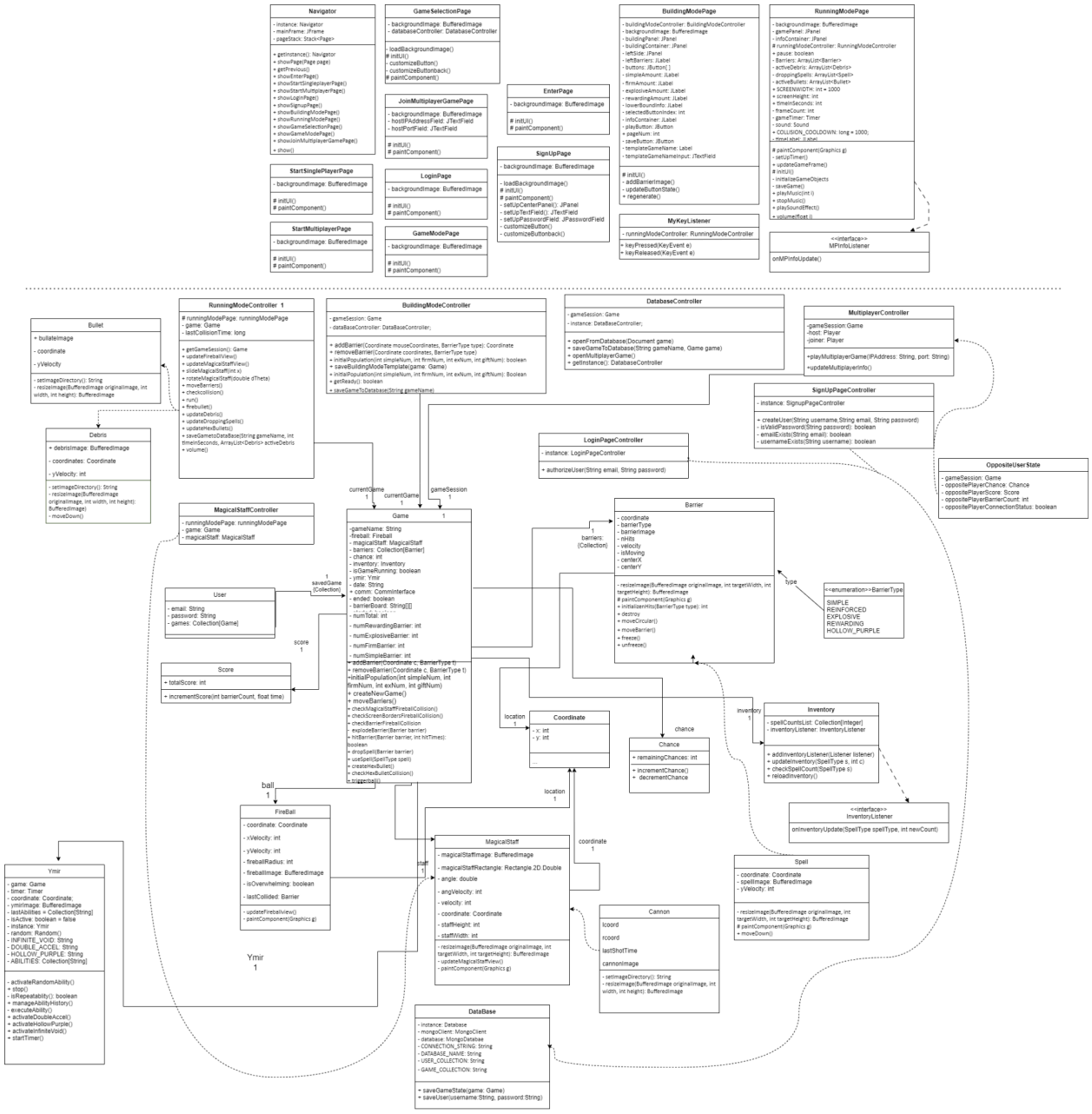
Communication Diagram 2- Build Game



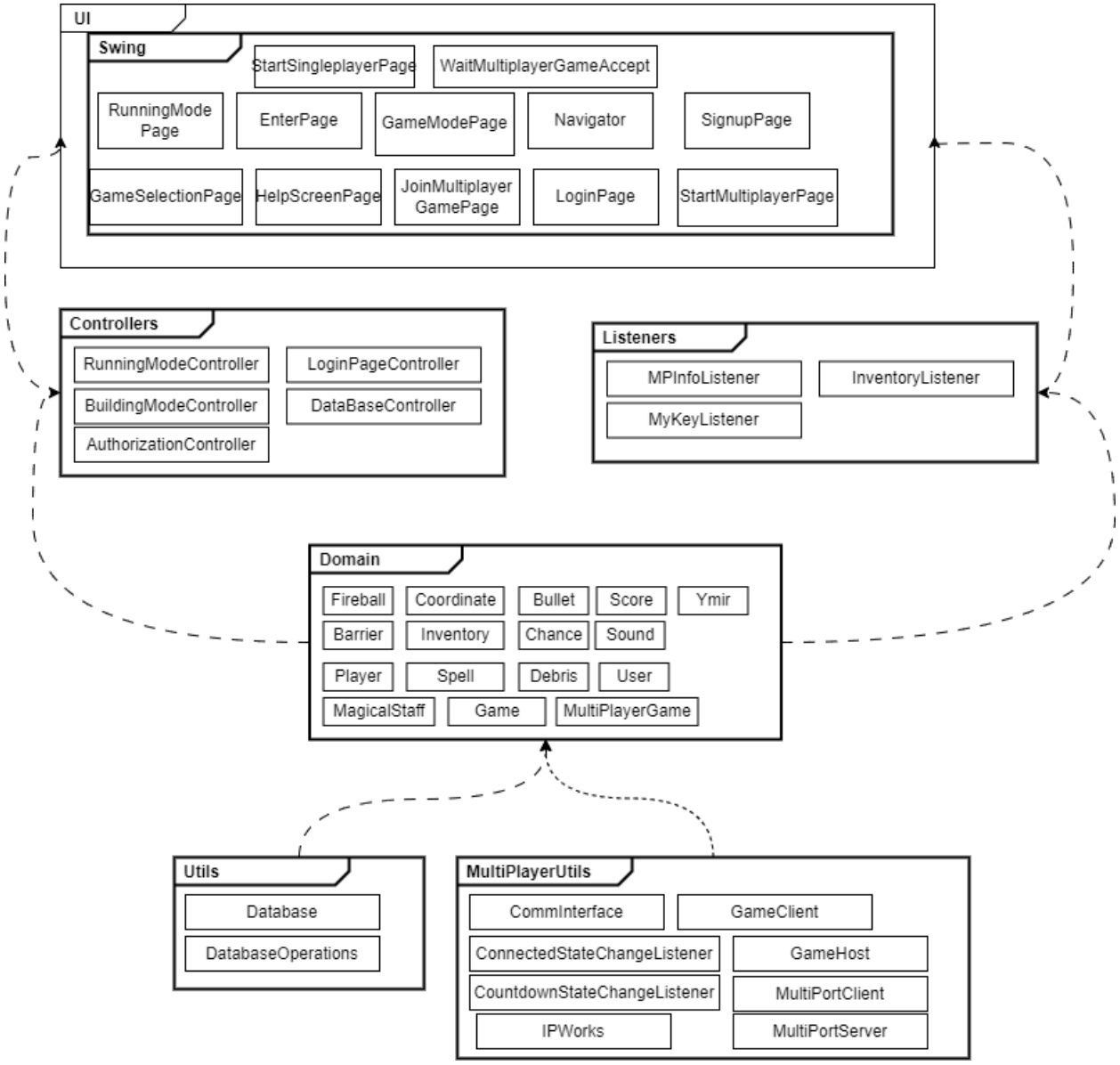
Communication Diagram 3- Move Magical Staff



8. Class Diagrams



9. Package Diagrams



10. Design Patterns and Principles

Controller Pattern

In our project design, the controller pattern was used in multiple class implementations. As it can also be seen in the class diagram, there are multiple classes labeled as "(...) Controller". These classes are included in the design with an aim of communicating with the UI components of the game and informing domain objects regarding their responsibilities. For example the Magical Staff Controller has the aim of analyzing key inputs and deciding on the movement type of the magical staff (rotating or horizontal movement). The running mode controller is responsible for controlling ball movements and reflections, as well as barrier conditions, which heavily depend on contact with the fireball. The building mode controller handles the barrier placements and checks edge cases such as whether the barriers are placed in valid positions. The authorization controller forms the connection between the user-input acquired from UI components with the player class and the game database. In a similar fashion, the pause view controller ensures the proper game response implementation for cases in which the user pauses or resumes the game. We also do database operations through our database controller.

Singleton Pattern

The singleton pattern is employed in several classes in our implementation. We decided these classes by thinking if there is the need to have multiple instances of one class at a given time.
This pattern ensures the creation of a single instance of the mentioned classes. The singleton pattern is also implemented in the Game class, since this class acts as a manager for the whole running process of the game. The Database class will also have a unique instance in order to prevent possible complications that may occur throughout data acquisition and management processes. Ymir, a class we added after phase2, is also a singleton since we do not need two ymir's at a time. Similarly, the Navigator is singleton since it is the main class controlling which UI view is shown, and only one instance is necessary.

Observer (Listener) Pattern

The Observer pattern is a subscription mechanism to allow objects to listen and react to events occurring in another object, known as the subject. In the multiplayer aspect of our game, this pattern is important in maintaining the synchronization of player states across different sessions. We make use of this pattern with as "MPInfoListener" and listen what is received by the opponent and update running mode. Similarly, we also have a "InventoryListener", which provides the ability to update running mode view when an addition or subtraction from the inventory is made.

Information Expert Pattern

The Game class contains all necessary information for the formation of instances of other classes and components. It has the information of the locations in the game any given moment and information about inventory. Therefore, it can be stated that we have utilized the information expert pattern during our domain design by assigning tasks to classes which have the most amount of required information to fulfill it.

High Cohesion and Low Coupling Principles

Being aware of the significance of high cohesion and low coupling in the game design process, we have paid significant attention to decide on which classes we are going to include in a way that no class unrelated responsibilities to fulfill. We paid attention to which class communicates to which and tried to communicate with only neighbors according to Law of Demeter. The usage of high cohesion and low coupling principles in our project will help us keep our project more organized and will also enable code reusability. Our variety of controller classes is also beneficial in decreasing the dependency between different game components, providing us a desired low coupling effect.

Model- View Separation

In the implementation of the UI layer, classes will not have any responsibility other than getting the input from the user and transferring it to the domain layer. This principle is called model-view separation, and it allows our classes to have a better interaction. Also, it provides room for developers to implement and try different UI versions without making changes in the domain layer. We made sure that no calculation regarding the game will be made in the UI classes, they will only delegate the request coming from the user to the domain.

11.Supplementary Specifications

Introduction

This document is the repository of all Lance of Destiny games requirements not captured in the use cases.

Functionality

Lance of Destiny is a game where the player tries to achieve a certain goal. Player aims to break all the barriers without losing all of the chances. Meanwhile, Player can collect and use spells. Also players can play together with other players in the same network with the Multiplayer Mode. One player will host the game and according to the game name, other player can select and join the game.
The player controls the game using keyboard and the mouse.

-Error Handling:
If the system fails for some reason or faces an error, an error message is displayed to the user and the user returns to the main menu or the game executable is closed according to the error.

-Security:
Each player needs to be authorized to play the game. This is done through a log in/sign in page. The entries of the player are checked in the database and access is given if the information matches. A player can only access their own versions of game templates, not to other player's templates.

Usability

The game is designed as a desktop application. Each page that is displayed to the user will be designed as easily visible on the desktop screen.
To provide high accessibility, buttons will be clearly labeled and avoided to be misclicked.
The colors will be chosen to be distinguishable for the comfort of the players.
The frame shown per second will be as high as needed for the easy tracking of the objects moving in the screen.

Reliability

If any errors occur, the system will notify the player and either go back to main menu or close the game. If the game is closed by the system due to some error, the executable should be run again by the player.

Performance

In general, the system will be designed to work properly and avoid the waiting of the player. Authentication process is expected to take less than 10 seconds. Starting the game is expected to take less than 30 seconds.
An important factor in such games with moving objects, frame per second is an important performance factor. This will be decided by balancing the computing requirements and player comfort on following the game.

Supportability

The game should be compatible with the major operating systems such as Windows, macOS or Linux. The game will be implemented on Java, which is a common language and can be run easily on computers.
The game will need to be downloaded again if any changes are made after the initial download. The users can access the game source code through github.

-Configurability:
The game offers a Building Mode where the players can create their own game template. Then, different games can be configured and played later.

Implementation Constraints

For the implementation of the game, Java will be used as the programming language. Its standard libraries and Java Swing will be utilized.

Purchased Components

No third party software will be needed that needs purchasing.

Free Open Source Components

Java will be utilized, which is a free open source language.

Interfaces

-Noteworthy Hardware and Interfaces:
The game does not require any special hardware, a computer with a mouse(or touchpad) and keyboard is sufficient.

-Software Interfaces:
Different libraries might be implemented using java. Different classes will interact with each other with the help of object oriented programming features.

Domain (Business) Rules

ID	Rule	Changeability	Source
RULE1	Player credentials such as user name and password are required to have access to the game.	Low. Players will need to access the game in the future also.	Game's design and security of game templates.
RULE2	The creator can modify the game.	High. New features might be added and modifications can be made as long as it does not intervene with the general concept of the game.	Team's vision and requirements of the changing game sector and user wants.

General Game Rules:

Introduction

Lance of Destiny is an interactive game.The game has two modes: running mode and building mode. In the building mode they can build their own game template to play. A game template includes different types of barriers which will be explained later. In the running mode, the player tries to break the barrier with the Fire Ball. Fire Ball bounces from the sides of the game screen, barriers, and the Magical Staff. The player moves the Staff to properly bounce the Fire Ball. The game ends when the chances of the player is over or when all the barriers are successfully broken.

Objects

a.Magical Staff
Moved by the player using the keyboard.
Can be moved horizontally or rotated up to 45 or 135 degrees.

Has the length of 10% of the screen width and thickness of 20px.

b.Fire Ball

Bounces between barriers, Magical Staff and the borders of the game.
Damages the barriers.

c.Barriers

Have 4 types:
-Simple Barrier: Broken in one hit.
-Reinforced Barrier: Requires more than one hit to be destroyed. The number is indicated on the barrier.
-Explosive Barrier: Once it is broken, it explodes and its debris falls down. If it touches the Staff, player loses a chance.
-Rewarding Barrier: It contains a box that holds some kind of a spell.

d.Good Spells

Have 4 types:
-Felix Felicis:Increases player's chances.
-Magical Staff Expansion: Doubles the length of Magical Staff.
-Hex: Adds two cannons to both ends of Magical Staff. They shoot and their hexes also damage barriers.
-Overwhelming Fire Ball: Makes Fire Ball more powerful and it destroys and passes through all kinds of barriers.

<u>Player Actions</u>

a. Move Magical Staff

Player's main action is to move the Magical Staff to reflect the Fire Ball.
-Press and release arrow left button: Staff moves left an offset equal to half of its length.
-Press and release arrow right button: Staff moves right an offset equal to half of its length.
-Keep pressing the arrow left button: Staff moves left as long as the button is down.
-Keep pressing the arrow right button: Staff moves right as long as the button is down.
-Press the key A: Magical staff turns counterclockwise.
-Press the key D: Magical staff turns clockwise.

B. Use Spells

The player can use spells either by clicking them on the inventory on the screen or clicking to the corresponding key, its first letter.

Building Mode

In this mode, the player can create their own game environment. The player enters how many of each barrier they want to the screen. The system arranges them randomly on the screen. Then, the user can move, add and remove them with mouse clicks. There are some criteria that needs to be satisfied:
- At least 75 simple barriers.
-At least 10 firm barriers.
-At least 5 explosive barriers.
-At least 10 gift barriers.

Playing Mode

In this mode, the player plays one of the game environments created by them. As explained above, the player moves the staff, uses spells and tries to get rid of all the barriers. If the player can break all of them before losing all chances, he/she wins. Meanwhile, while playing the score is calculated after each hit according to the formula:

newScore = oldScore + 300 / (currentTime - gameStartingTime)

On the screen, the score , chances, barriers, fire ball, magical staff and collected spells are shown. Also, the player can pause and save or exit the game any time.

Legal Issues

The Lance of Destiny game does not require any personal data; therefore, does not have the risk to intervene with the personal data laws.
For the implementation open-sources will be used, licensing will not be an issue.
Rights of the source code of the project is reserved for the creator team.

12. Glossary

Lance of Destiny / Game / System : The main entity a player interacts with. Game has two modes: building and running mode. In the building mode the player can design the level, the number and positioning of the barriers. In the running mode, the player controls the magical staff and tries to reflect the fireball, so that it hits the barriers and destroys them. The player can win or lose the game by their performance.

Player: The main actor throughout the game, is responsible for building and playing the game.

Building Mode : The game always starts with the building mode. In the building mode, the player can place various barriers to the game environment. The requirements of the building mode are as follows: at least 75 simple barriers, 10 reinforced barriers, 5 explosive-barriers and 10 rewarding barriers.

Running Mode: This mode has the magical staff, the fireball and the barriers. The player's goal is to destroy all barriers by reflecting the fireball and making it hit the barriers by controlling the magical staff.

Multiplayer Mode: This mode is very similar to the running mode, however, it is joinable by two users: one game host and one guest. The users play separate games, however, as they collect bad spells, which were only implementable by Ymir in the running mode, they can use them against their opponent and make it harder for them to win the game. The multiplayer mode ends when one of the player's wins or loses the game.

Magical Staff / Staff: The only entity player can directly control. It moves in the horizontal direction and can rotate up to 45 degrees.

Fireball : The ball-like object in the running mode. Initially, it starts at the top of the magical staff, and can be thrown vertically. Every time it is reflected by an object, its speed and direction changes according to the game physics. When the fireball hits barriers, it damages them, reducing their health by 1.

Barriers: There are 4 types of barriers in the game.

- Simple: It can be destroyed by one hit.
- Reinforced : It takes more than one hit to destroy it.
- Explosive: It can be destroyed by one hit, then resulting debris falls down, damaging the player on contact.
- Rewarding: It can be destroyed by one hit, creating a spellbox.
- Hollow Purple: It is the additional barrier created by the adversary. It produces no score, and is aimed to create difficulty for the player.

Score: The player's score increases each time a barrier is destroyed, and it is inversely proportional to the elapsed time in the game.

Spellbox: When summoned by a destroyed rewarding barrier it vertically falls down. Player can collect it with the staff.

Spells: The spells are acquired by collecting the spellboxes. When a spellbox is collected one of the four spells is added to the player's inventory. The player can later use them.

- Hex: Equips the staff with two cannons for 30 seconds.
    - Cannon: It periodically shoots upwards, creating bullet-like objects. They damage the barrier in the same way the fireball does.
- Felix Felicis: Increases the player's chance by 1.
- Overwhelming Fireball: Modifies the fireball so that it destroys the barriers it touches, and keeps its speed an sd direction instead of being reflected. This effect lasts 30 seconds.
- Magical Staff Expansion: Doubles the size of the staff. Lasts 30 seconds.
- Infinite Void: In the single player mode, this spell is activated by Ymir whereas in the multiplayer mode, a player can use this against his/her opponent. This spell freezes the barrier and makes them undestroyable by the fireball for 15 seconds.
- Hollow Purple: Similar to infinite void, this spell can be activated by a user only on a multiplayer game, if collected from a rewarding barrier. In the singleplayer mode, Hollow Purple is activated by Ymir and it adds 8 barriers to random locations on the game board. The types of these barriers are unique and called hollow-purple. These barriers behave very similar to simple barriers, but the player does not earn any points by destroying them.
- Double Accel: This spell decreases the fireball's speed to half and it can only be activated by Ymir in single-player games, and by the opponent user in multiplayer games. The slow-down effect lasts for 15 seconds, and after the time passes, the speed reverses back to the fireball's original speed.

Ymir: Ymir is a bad spirit which aims to make it more challenging for the user to win the game. Ymir flips a coin every 30 seconds in the running mode, and if he acquires a certain coin state, it activates one of its three abilities Infinite Void, Hollow Purple or Double Accel. These spells are activated in such a way that the upcoming ability is different from the previous-two spells activated.

Chance: Everytime the fireball misses the staff and falls down, or debris from the explosive barrier contacts the staff, the player loses one chance. If the player runs out of chances, then the player loses the game .