Need for Spear

Final Report

Group dev/null/
Kaan Türkmen
Can Usluel
Melis Oktayoğlu
Halil Doruk Yıldırım
Gökçe Sevimli

Table of Contents

Introduction

Vision

Revision History

Version	Date	Description	Author
Draft	Oct 25, 2021	First draft, to be revisioned	Melis Oktayoğlu, Gökçe Sevimli, Halil Doruk Yıldırım
Revision	Oct 26, 2021	Revision is done.	Kaan Türkmen, Can Usluel
Update	Jan 10, 2022	Update is completed.	Kaan Türkmen

1. Introduction

We aimed to develop a fun game named “Need for Spear”. Hence, we created a computer rival called Ymir. This rival tries to make the player lose the game by using spells on the map. We allowed users to create their own accounts where they can store their earlier games, as well as the game maps they had designed for later use. Player has a noble phantasm and an enchanted sphere which they will use to prevent the sphere from falling to the ground. The falling of the sphere results in losing one of the three lives of the players. The sphere is an object that they will use to destroy the obstacles, thereby earning scores. At the same time players can earn magical abilities to enhance their powers and the game ends when all obstacles are destroyed.

1.1 Purpose

The purpose of this vision document is to provide all the necessary information and curate the high-level needs, requirements and features of our game NeedForSpear.

1.2 Scope

This vision is for the Need for Spear game developed by the /dev/null team. The game is visioned to be played on desktops and this will provide people of all ages to spend fun time and bond with each other over some friendly competition.

1.3 References

Templates inspired from: https://www.ibm.com/docs/en/elm/7.0.0?topic=requirements-vision-document
https://personal.utdallas.edu/~chung/RE/Presentations10F/Team-hope/1%20-%20VisionDoc.pdf

[1] H. F. Büyük, “Turkey’s Digital Gaming Industry Soars on Cheap Lira, Pandemic,” *https://balkaninsight.com/2021/09/27/turkeys-digital-gaming-industry-soars-on-cheap-lira-pandemic/*, 27-Sep-2021.

2. Positioning

2.1 Business Opportunity

The current game industry is highly capable of developing high quality, high resolution games. However, they often cost a lot of money which limits distributing the game to a wide range of demographics. For this reason we aim to develop an affordable desktop game that is still enjoyable.

2.2 Problem Statement

Due to developments and innovations in the game industry, many game developers are publishing various games to offer different alternatives to potential players. At this stage, /dev/null team is developing a game called “Need for Spear” to provide gamers a fascinating game experience.

2.3 Product Position Statement

For	People of all ages.
who	Want to have fun and spend time.
NeedForSpear	Is a desktop game application.
that	Provides a platform for spending fun and competitive time.
unlike	The current brick-breaker-like games which don't offer various types of obstacles and magical spells.
our product	Offers a fascinating game experience by adding different features to the classic brick-breaker game.

3. Stakeholder and User Descriptions

3.1 Market Demographics

Last year, according to the annual Turkey Game Market Report, digital gaming industry's revenues in Turkey reached $880 million, as the number of gamers grew from 32 million in 2019 to 36 million in 2020. Experts believe the sector will hit the $1 billion mark by the end of this year [1].

The target segment includes people in all age groups, who want to spend time by playing and have fun with others.

3.2 Stakeholder Summary

Name	Represents	Role
Computer Engineers	This stakeholder is a primary lead in the development of Need for Spear.	Responsible for the architecture, implementation, design and overall development of Need for Spear.
Team Lead	This stakeholder leads development of Need for Spear.	Plans meetings between team members, sets priorities with them and keeps the project team focused.

3.3 User Summary

Name	Description	Responsibilities	Stakeholder
Players	Primary End user of the game.	Uses the application to play with others or play by themself.	Self.

3.4 User Environment

1. Need for Spear will be played by anyone, regardless of age, that wants to play the game for any reason they desire.
2. The game can be played single player.
3. The players will create user credentials, which will be saved in the database, to log in to their account.
4. After login to the game, the players will interact with an understandable interface where they can:
   a) Enter the help screen.
   b) Start a new game.
   c) Load a new game.

3.5 Stakeholder Profiles

Computer Engineers

Description	This stakeholder is responsible for the development of Need for Spear.
Type	Software guru experience in programming and teamwork.
Responsibilities	Uses agile methodologies to design and develop Need for Spear.
Success Criteria	Success is defined as going through the project iterations to create Need for Spear in an agile matter.
Involvement	This stakeholder is a lead figure in the creation of Need for Spear. Designs and programs the game.
Deliverables	Iterations of design, code and other necessary material to COMP302 staff.
Comments/Issues	None.

3.6 User Profiles

The Players

Description	The users that play Need for Spear.
Type	Casual user who may have never played any video game before.
Responsibilities	Plays Need for Spear to pass time, have fun or for any reason they desire.
Success Criteria	Success is defined as the player enjoying the game and continuing to play it.
Involvement	None.
Deliverables	None.
Comments/Issues	None.

3.7 Key Stakeholders or User Needs

Need	Priority	Concerns	Current Solution	Proposed Solutions
Flexible (configurable)	High	Ability to customize obstacles based on different user desires.	See proposed.	Allow users to determine the number and the type of obstacles.
Easy to use	Low	Ability for users to easily open the main menu, help screen and pause menu and save and load the game.	See proposed.	Provide to save and load game, understandable pop-ups for help screen and pause screen, and clear-view of the main menu.

4. Product Overview

4.1 Product Perspective

This app is self contained and independent of another system.

4.2 Summary of Capabilities

Customer Benefit	Supporting Features
Having a fun time.	A nice-looking and fun game design.
Ability to arrange their time.	Pause and save/load game features.
Have a fun experience versus the computer.	Changing difficulty to obtain a harder game experience.

4.3 Assumptions and Dependencies

1. It is assumed that players have access to a computer.
2. It is assumed that players have a functional keyboard, screen(s) and mouse.
3. It is assumed that players have an internet connection.
4. It is assumed that players have valid email adressses.
5. The game language is English.
6. It is assumed that players are literate in English.

4.4 Licencing and Installation

Players can pull the game from our Github repository. After installation, the installation guide will be provided, and the users will be licensed after entering a userna

5. Product Features

5.1 System Features

1. Start Game.
2. Exit Game.
3. Accept Keyboard Input.
4. Show Help Screen.
5. Pause Game.
6. Save Game.
7. Load Game.

5.2 Game Features

8. Moving the noble phantasm to the left or to the right.
9. Rotating noble phantasm.
10. Designing a game map.
11. Using spells.
12. Hitting an obstacle.
13. Switch to run mode.

6. Precedence and Priority

Priority	**Feature (By Number Above)**
High	1, 2, 3, 8, 9, 10, 12, 13
Medium	5, 6, 7, 11
Low	4

7. Other Product Requirements

7.1 Applicable Standards

None specified.

7.2 System Requirements

1. The game must be run on a computer.
2. Player should have Java installed in their computers.

7.3 Performance Requirements

1. Latency of the game should not be more than 1 second.
2. Frame per second should be 30.
3. Memory usage should not exceed 10% of total memory space.

7.4 Environmental Requirements

Having a stable internet connection.

8.Documentation Requirements

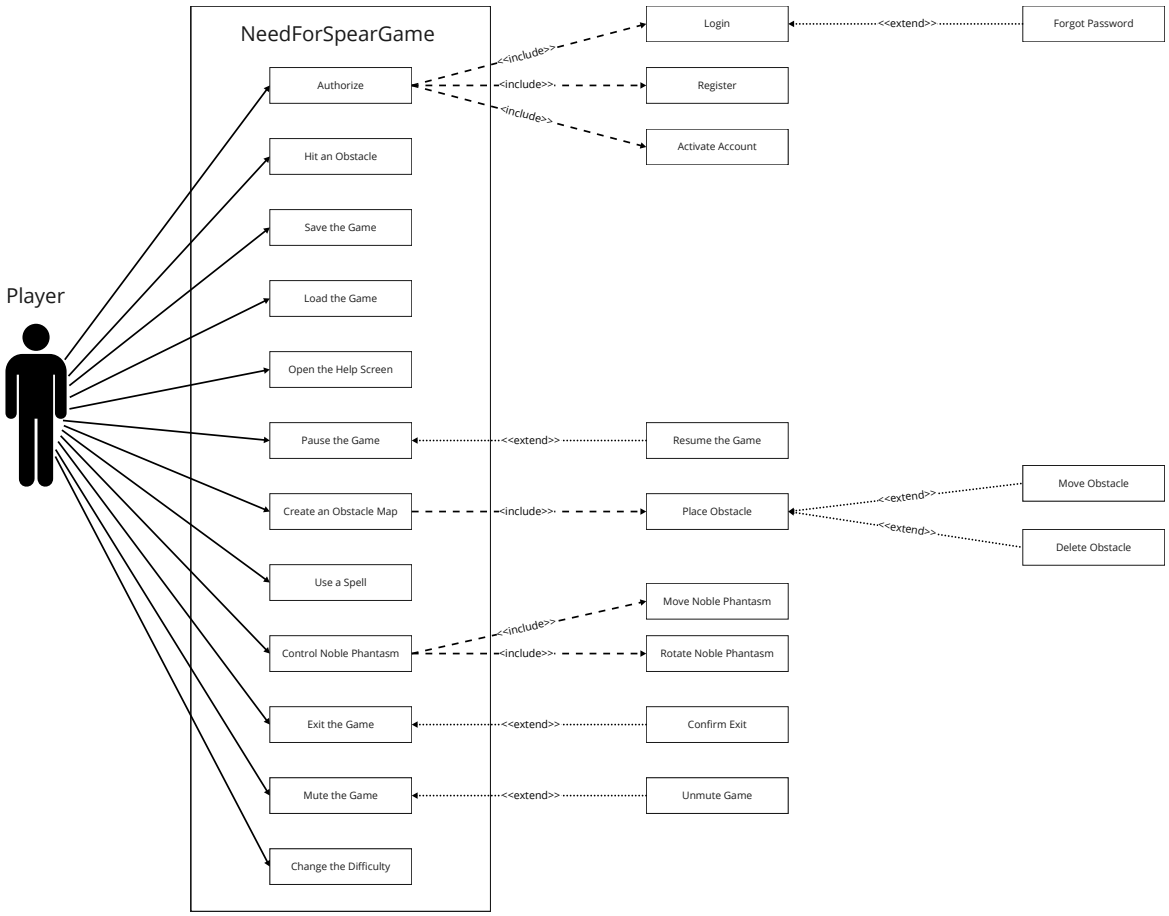
8.1 User Manual

Users can consult the help screen on the game by clicking on the respective icon.

Teamwork Organization

Person	Main Contribution
Kaan Turkmen	Threading Concepts, Database Implementation, Authentication System and Views.
Can Usluel	Obstacle Placement Algorithms, GameView GUI and Hollow Purple Spell.
Melis Oktayoglu	Physics Engine, Sphere movement & Interaction, Infinite Void & Double Accel Spell.
Doruk Yildirim	Game Assets, Magical Hex Spell, Chance Giving Spell & Score Calculation.
Gokce Sevimli	Physics Engine, Noble Phantasm (Paddle), Expansion Spell.

Use Case Diagram



Use Case Narratives

Use Case UC1: Authorize.
Scope: Login Screen.
Level: Subfunction.
Primary Actor: Player.
Stakeholders and Interests:

1. Player: Wants to create a mechanism to retrieve all personalized data about them. (Progress in the game, highscore etc.)

Preconditions:

1. Player must have an internet connection.
2. Player must be in the login screen.

Success Guarantee:

1. The player successfully logs in or, alternatively, signs up the game.

Main Success Scenario:

1. Player enters the username to the required area.
2. Player enters the password to the required area.
3. Player clicks to the "Login" button.
4. System checks if the username and password match.
5. System checks if the email is verified.
6. System directs the player to the Main Menu.

Extensions:

*****At any time**,** the player wants to sign up for the first time.

1. Player enters their email address.
2. Player enters a username.
3. Player enters a password.
4. System saves this information to the database and sends a verification email to the player.
5. Player verifies their email to complete its sign up process.

2a. If the player forgot their password, or wants to change it.

1. Player clicks the "Forgot my password" button.
2. System sends an email to the player.
3. Player selects a new password with the help of instructions sent by email.
4. System updates the password at the database.

4a. If the provided username is not found.

1. System informs the player that the provided username has not been authorized.

4b. Player enters a wrong password.

1. System notifies the player that the password is wrong and asks to try again.

5a. Player's mail is not verified.

1. System notifies the player that their mail is not verified.

Special Requirements:

1. Rectangular text fields should be large enough to take inputs.
2. Rectangular text fields should be visible through various backgrounds.

Technology and Data Variations List:

1. Username, email should be text field while password area should be secured field.
2. The input should be entered via keyboard.

Frequency of Occurrence: Whenever the player wants to login to the game.
Open Issues: NO OPEN ISSUES.

Use Case UC2: Hit an Obstacle.
Scope: Running Mode.
Level: User Goal.
Primary Actor: Player.
Stakeholders and Interests:

1. Player: Wants to direct the moving sphere with the noble phantasm to hit the obstacles.

Preconditions:

1. Player has built the level or loaded a built one.
2. Player is in the running mode.
3. Player has one or more lives.
4. Player is logged in using their credentials.
5. Player must have an internet connection.

Success Guarantee:

1. The player successfully hits the obstacle with the sphere.

Main Success Scenario:

1. Player moves the noble phantasm according to the sphere's movement.
2. Player redirects the sphere by using the noble phantasm.
3. Sphere hits an obstacle.

Extensions:

1a. Player fails to hit the sphere with noble phantasm.
    1. Player's chances are reduced by 1.
        1a. Player loses the game if the number of chances reduces to 0.
3a. Sphere misses obstacles.
    1. Sphere bounces back from the edges and comes back to the level of the noble phantasm.
3b. Player selects to use "Magical Hex" ability.
    1. Player hits the obstacles using the ability instead of the sphere.
3c. Sphere hits "Firm Obstacle".
    1. The obstacle's health number decreases by one.
        1a. The obstacle is destroyed if its health is reduced to 0.
3d. Sphere hits "Explosive Obstacle".
    1. The obstacle's remains drop downwards.
    2. If the remains hit the noble phantasm, the player's lives decrease by one.
3e. Sphere hits "Gift Obstacle".
    1. The obstacle drops a gift box downwards.
    2. If the gift box touches the noble phantasm, the player gets a special ability.
        2a. If the special ability is the "Chance Giving Ability", the player's life increases by one.
        2b. If the special ability is the "Noble Phantasm Expansion", the noble phantasm's size increases.

2c. If the special ability is the “Magical Hex”, the player can shoot using the cannons on both sides of the phantasm.

2d. If the special ability is the “Unstoppable Enchanted Sphere”, the sphere can penetrate the obstacles.

3. If the player already has unused abilities, the newly acquired ability is stored alongside the unused ones.

Special Requirements:

1. All the obstacles’ colors should be chosen such that they are visible between various background colors.
2. Obstacle’s health should be displayed on the obstacle either using a number or texture.
3. Gifted, normal and explosive obstacles should be distinguishable from each other.
4. Each special ability should appear at least once after every gift obstacle is destroyed.

Technology and Data Variations List:

1. Player should interact with the game via keyboard.

Frequency of Occurrence: Every time the player hits an obstacle.
Open Issues: NO OPEN ISSUES.

Use Case UC7: Create an obstacle map.
Scope: Building Mode.
Level: User Goal.
Primary Actor: Player.
Stakeholders and Interests:

1. Player: Wants to design a map which they want to save to play later or to play immediately.

Preconditions:

1. Player is logged in using their credentials.
2. Player must have an internet connection.
3. Player is in building mode.

Success Guarantee:

1. Creates a new map in which the number, type and health of the bricks are set.

Main Success Scenario:

1. Player clicks the “New Game” button in the main menu.
2. An empty map, which contains different types of bricks for the player to choose, will be displayed.
3. Player designs the map by placing the bricks.
4. Player saves or plays the map.

Extensions:

3a. Player wants to remove a selected brick.

1. Player selects the brick and presses the right click.
2. System removes the brick from the screen.

3b. Player wants to put obstacles.

1. Player chooses the type of obstacle from the JComboBox.
2. Player clicks left twice.
3. System puts a new obstacle to the screen.

3a. If a new obstacle’s location intersects with another, the new obstacle will not be placed.

Special Requirements:

1. All the bricks in the level should be visible.
2. Bricks' health should be set randomly.
3. Gifted, normal and explosive bricks should be distinguishable from each other.
4. Editing area for the player should be distinguishable.

Technology and Data Variations List: NO TECH or DATA VARIATION.
Frequency of Occurrence: Whenever the player wants to design a map.
Open Issues: NO OPEN ISSUES.

Use Case UC8: Use a Spell.
Scope: Running Mode.
Level: User Goal.
Primary Actor: Player.
Stakeholders and Interests:

Player: Uses a spell of their choosing.

Preconditions:

1. Player is logged in using their credentials.
2. Player must have an internet connection.
3. Player is in running mode.
4. Player has a spell.

Success Guarantee:

1. Player activates the spell.

Main Success Scenario:

1. Player clicks the icon or types the first letter of the spell.
2. Spell is activated.

Extensions:

1a. "Chance Giving Ability" is used immediately upon pickup.

Special Requirements:

1. Spells' icons should be visible and understandable.
2. Player should be able to understand whether they have any spells or not.
3. Player should be able to understand if there are any spells active.
4. Visualization of spells' effects should be distinguishable.

Technology and Data Variations List:

1. As a data variation to the default keys of spells (E, M ,R) we added two additional keys (Q and R) to activate the spells (hex and unstoppable) so that player can use close positioned keys on the keyboard.

Frequency of Occurrence: Whenever the player wants to use a spell.
Open Issues: NO OPEN ISSUES.

Use Case UC9: Control the Noble Phantasm.
Scope: Running Mode.
Level: User Goal.
Primary Actor: Player.
Stakeholders and Interests:

1. Player: Wants to change the Noble Phantasm's location and degree.

Preconditions:

1. Player is logged in using their credentials.
2. Player must have an internet connection.
3. Player is in running mode.

Success Guarantee:

1. Player moves, rotates and speeds up the noble phantasm.

Main Success Scenario:

1. Player presses right or left arrows to move the phantasm horizontally.
2. Player presses the A and D keys to change the noble phantasm's rotation degree.
3. Player presses the down arrow to double the phantasm's speed.

Extensions:

1a. Player tries to control Noble Phantasm in pause mode.

1. Phantasm's state is not changed.

1b. Player tries to move the phantasm out of the map.

1. Phantasm stays at the edge of the map.

Special Requirements:

1. Noble Phantasm should be visible through various backgrounds.

Technology and Data Variations List: NO TECH or DATA VARIATION.
Frequency of Occurrence: Whenever the player wants to move the phantasm.
Open Issues: NO OPEN ISSUES.

Use Case UC10: Exit The Game.
Scope: Main Menu
Level: Subfunction.
Primary Actor: Player.
Stakeholders and Interests:

1. Player: Wants to close the game without any exceptions.

Preconditions:

1. Player is in the main menu.

Success Guarantee:

1. Game shuts down.

Main Success Scenario:

1. Player clicks the "Exit" button.
2. A screen pops up asking "Are you sure?".
3. Player clicks "Yes".
4. Game closes.

Extensions:

2a. Player clicks "No"

1. Player returns to the previous screen.

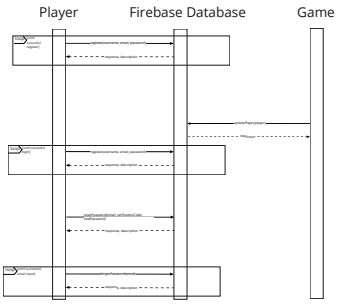
Special Requirements:

1. Rectangular buttons should be visible through various backgrounds.

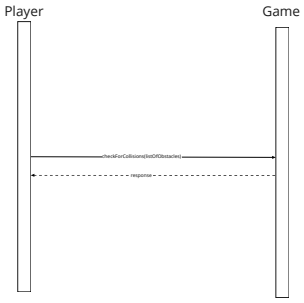
Technology and Data Variations List: NO TECH or DATA VARIATION.
Frequency of Occurrence: Whenever the player wants to exit from the game.
Open Issues: NO OPEN ISSUES.

System Sequence Diagrams

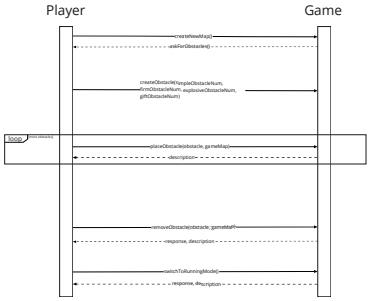
Authorize



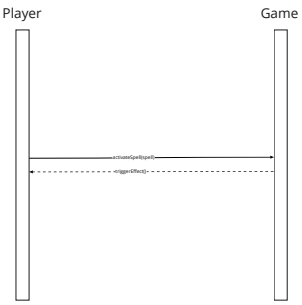
Hit an Obstacle



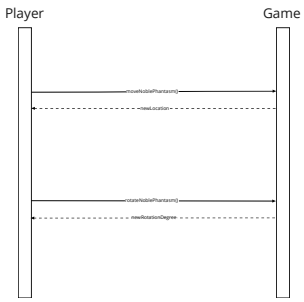
Create an Obstacle Map



Use a Spell



Control the Noble Phantasm



Operation Contracts

Operation: login(*player* : Player)
Cross References: Authorize.
Preconditions:
1. Username and password pair exists on the game database.

Postconditions:
1. *player* was associated with NeedForSpearGame. *(association formed)*
2. *player* was associated with GameMap. *(association formed)*
3. *player* was associated with Spell. *(association formed)*
4. *player* was associated with NoblePhantasm. *(association formed)*

Operation: forgotPassword(*email* : String)
Cross References: Authorize.
Preconditions:
1. *email* has an associated username and password *player* object on the game database.

Postconditions:
1. *player*.password is updated with a new password according to user input. *(attribute modification)*

Operation: loadGame(*player* : Player)
Cross References: Load the Game.
Preconditions:
1. Game is in the Pause Menu or Main Menu.
2. There is a previously saved game file in the database.

Postconditions:
1. An instance of GameMap *map* was created. *(instance creation)*
2. *map.listOfObstacles is set to the data from the database. (attribute modification)*
3. *player.*score is set to score from the database. *(attribute modification)*
4. *player.*lives is set to lives from the database. *(attribute modification)*

Operation: activateSpell(*spell* : Spell)
Cross References: Use a Spell.
Preconditions:
1. Game is in the running mode.
2. Player should have at least one instance of spell.

Postconditions:
1. *spell* is associated with a Hit object. *(association formed)*
2. *hit*.damage was set according to *spell*.type. *(attribute modification)*
3. *player.*listOfSpells is adjusted accordingly. *(attribute modification)*

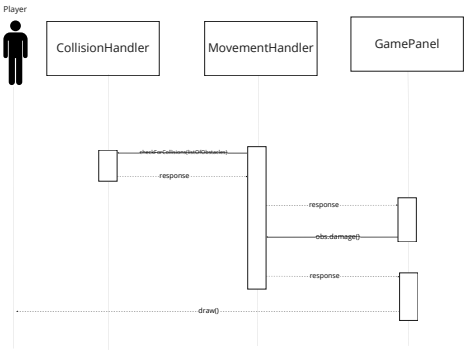
Operation: dispatchKeyEvent(keyEvent : KeyEvent)
Cross References: Control the Noble Phantasm.
Preconditions:
1. An instance of a NoblePhantasm *noblePhantasm* has been created.
2. System is in the running mode.

Postconditions:
1. *noblePhantasm*.speed is set according to the input. *(attribute modification)*
2. *noblePhantasm*.location is set according to the input. *(attribute modification)*

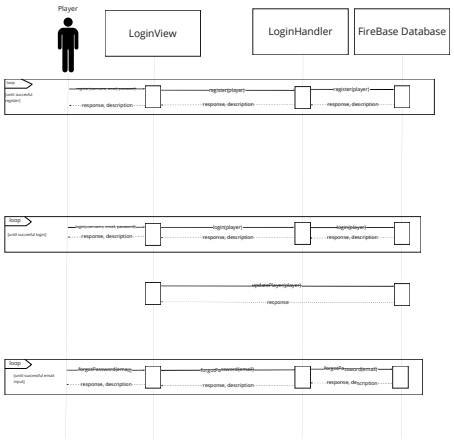
Interaction Diagrams

Sequence Diagrams

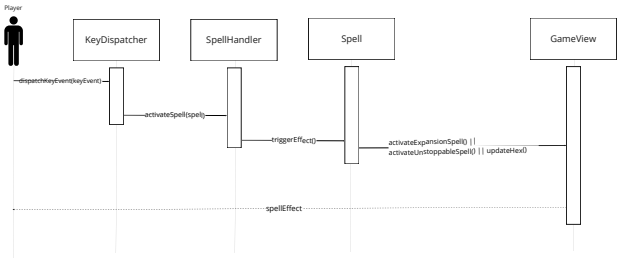
Hit an Obstacle



Authorize

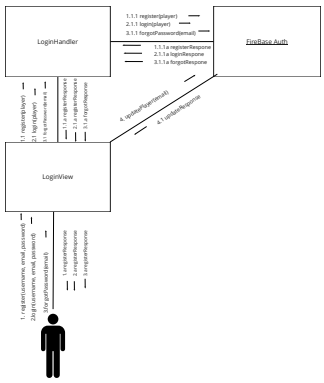


Use a Spell

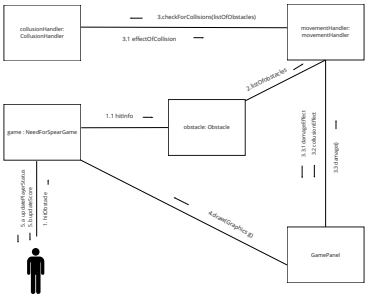


Communication Diagram

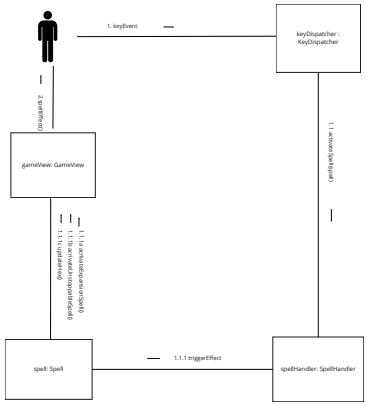
Authorize



Hit an Obstacle

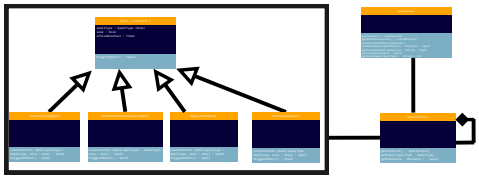


Use a Spell



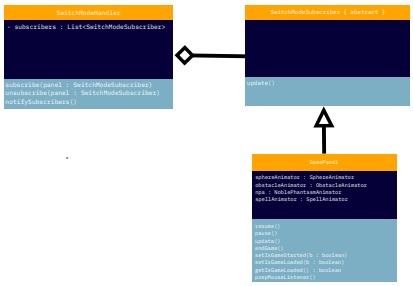
Class Diagrams

Singleton Spell Factory



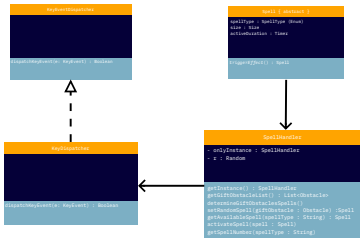
Hit an Obstacle

Switch Mode Observer

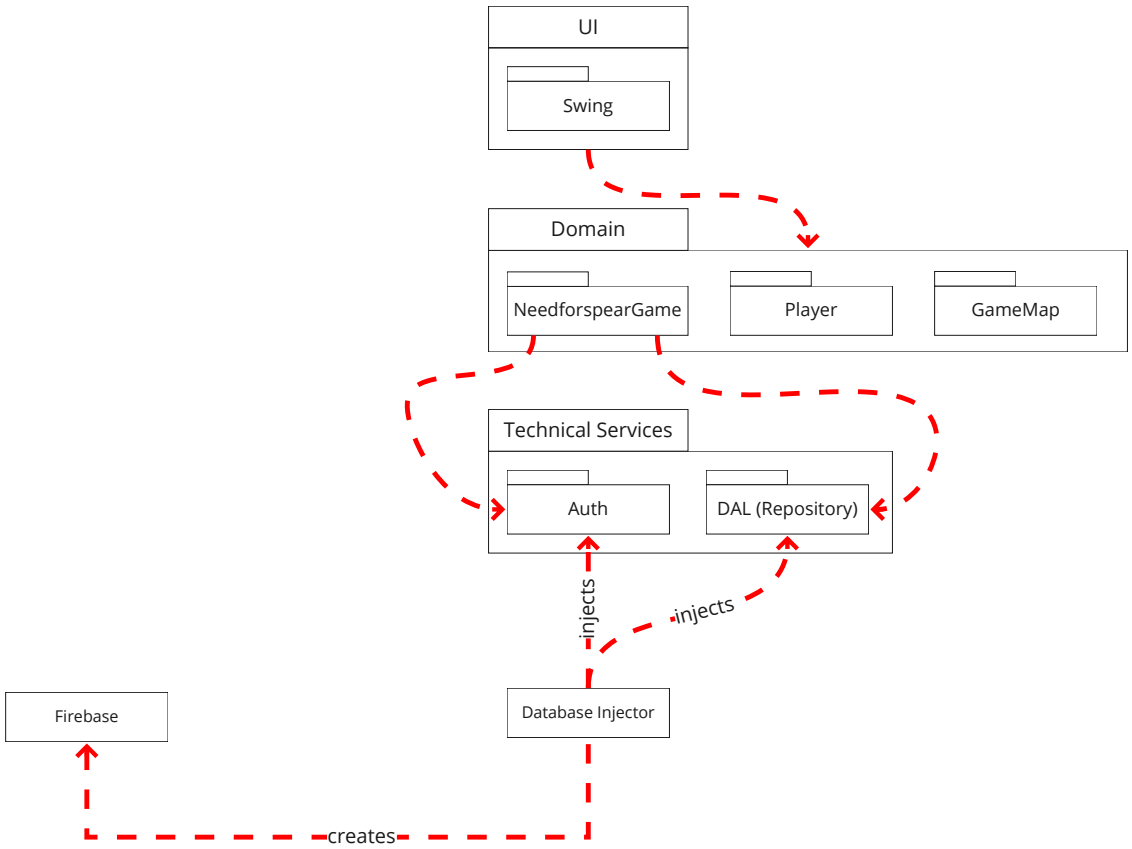


Authorize

Use a Spell



Package Diagrams



Discussion of Design Alternatives, Design Patterns, and the Principles

Observer Design Pattern: We have used Observer Design Pattern at multiple places, however, we want to cover why it is applied to the database concepts. Database calls are asynchronous, meaning that after we call its method, we can have no return or we can have a return but after a long time. Thus, if we create a variable and try to access the answer from it. The time when we access this object, we might have accessed way more early than we should. Thus, the observer design pattern is a perfect fit for the situation. Some of the objects are subscribing to our database, and the database notifies all of its subscribers after a response is produced. Thus, we are no longer needing a wait and objects which are subscribed to the database getting their response in time.

State Design Pattern: State design pattern is used for switching between different views such as from register view to main menu view. By calling the methods that define the state we are switching, we achieved low coupling when switching between states since the classes required less dependency to each other.

Factory Design Pattern: We used Factory Design Pattern in ObstacleFactory and SpellFactory classes. In that way, we created the obstacles of spells that we need by using only one class for the job. This application reduced the coupling in the code since it reduced the level of dependency of classes to each other and increased the cohesion between creator and other classes.

Singleton Design Pattern: This design pattern is used mainly on the NeedforSpearGame.java, because all objects and their instances should access the same game, and it's only instance. Thus, the singleton design pattern is a perfect fit for this kind of concept.

MVVM Design: We will not mention the Controller Design Pattern, however, we will mention MVVM Design. MVVM Design states that there should be three layers of the software: Models, ViewModels and the Views. ViewModels are also known as Handlers in the Controller Design Pattern. This mechanism separates logic from the View. Thus, we are achieving what we call flow of dependencies between layers.

Supplementary Specification

Revision History

Version	Date	Description	Author
Draft	Oct 25, 2021	First draft, to be revisioned.	Can Usluel, Kaan Türkmen, Halil Doruk Yıldırım
Revision	Oct 25, 2021	Revision is done.	Melis Oktayoğlu, Gökçe Sevimli
Update	Jan 10, 2022	Update is completed.	Kaan Türkmen, Can Usluel

1. Objectives

In this document, FURPS requirements will be analyzed. Furthermore, the parts which are not covered in the use cases will be discussed. This document along with the use cases will be the complete analysis of the game.

2. Scope

This Supplementary Specification applies to the Need for Spear game and it is created by the /dev/null team.

The Need for Spear game will be a game which can be played by a single player. The game will allow players to hit obstacles by controlling the Noble Phantasm and directing the sphere.

3. References

Templates inspired from:

https://csis.pace.edu/~marchese/SE616_New/Samples/Example%20%20Supplementary%20Specification.htm

https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/webtmpl/templates/req/rup_sspec.htm

4. Functionality

4.1 Store Constants in Separate File

System should keep the constants in some other file to ease the number of the updated files of the game after devnull publishes an update on the game.

5. Usability

5.1 Operating System Compliance

No specific operating system requirements due to Java Virtual Machine being a platform-independent environment.

5.2 Design for Ease-of-Use

Need for Spear user interface will be easily readable and designed clearly for the ease-of-use of the player.

6. Reliability

6.1 Sign-up Availability

The user should be able to sign up 24 hours a day, 7 days a week.

6.2 Frequency of failure

The game should not fail at any time.

7. Performance

7.1 Database Access Response time

The game's latency should not be more than 1 second.

7.2 Throughput

Game should display at least 30 frames per second.

7.3 Resource Usage

Game should not use more than %10 of the total memory space available.

8. Supportability

This section will list the features which enhance user experience.

8.1 Scalable Database

Devnull will use an auto scaling database to enhance user experience. The times of when the high traffic between users occur, users will not have a lagging experience due to the auto scaling databases.

9. Design Constraints

This section will list constraints that need to be obtained to create the game.

9.1 Java Compatibility

Player should have Java installed in their computers.

9.2 Hardware Requirements

Processor: Dual Core 1.4 GHz or better
Memory: 2 GB RAM
Storage: 500 MB available space

Glossary

Revision History

Version	Date	Description	Author
Draft	Oct 25, 2021	First draft, to be revisioned	Can Usluel, Doruk Yıldırım, Kaan Türkmen
Revision	Oct 28, 2021	Revision is done.	Melis Oktayoğlu, Gökçe Sevimli
Update	Jan 10, 2022	Update is completed.	Kaan Türkmen, Can Usluel

Definitions

Term	Definition and Information	Aliases
Map	The area where the player will play the game or build a new map.	
Obstacle	Bricks that players need to destroy to get a score.	Brick
Sphere	The ball that the player uses to destroy bricks.	Ball, Enhanced Sphere
Noble Phantasm	The paddle user moves to hit the moving sphere.	Spear
Spell	A resource the player can gather during the game to use for a helpful effect.	Magical Ability
Database	A place to store or request data either on on-premises software or cloud.	Db, dbase

Firebase	A google service helps developers to authenticate or store data about the user.	Fb
User Authorization	Validation by the third party mechanism to save or load user details to the database.	
Gradle	A tool that helps developers to build their project.	
IntelliJ / Eclipse	An environment to help developers to code on.	IDE
User ID	Unique user identification number.	uid
/dev/null	The developer team of the game.	devnull
Slack	A messaging app where the developers communicate with each other.	
Git	A version control system where developers can work collaboratively.	
Gitlab	A DevOps tool which provides an interface to the Git system.	
Gitlab Issues	A part which is being used as a todo list and part of an agile development.	
Agile Development	A development type states that doing discovery and solution through collaborative manner in each time iteration.	
Miro	A collaborative whiteboard application used to design diagrams and workflows.	
GitHub	A DevOps tool which provides an interface to the Git system.	
Realtime Database	A database system we use to keep track of the user and map information. When it is mentioned, it is aimed to imply the Google Firebase's Realtime Database.	
Ymir	A computer opponent to the player that will randomly cast spells which makes the game harder for the player.	

Resources	The images and sounds that are used to change the texture of objects or play background music in the game.	
Dependency Injection	It is a term that when the application calls the methods of interface and it does not have an idea whatever is satisfying the operation. Dependency being injected in a runtime, thus, even though the body of the method changes, if it is satisfying the interface requests, the game will run with no problem.	
Design patterns	The patterns that are followed while constructing the code. The use of various patterns achieves low coupling and high cohesion which facilitates the maintenance of the code.	