NEED FOR SPEAR
COMP 302 PROJECT

CELAL KAPLAN
MEHMET ULAŞ UYSAL
EFE ERTEM
MERT ALTINSOY
MERT AKKAN

Table of Contents

INTRODUCTION

This is the report for the game NEED FOR SPEAR that we made for our class COMP 302. In this introduction we will introduce the game and highlight the important aspects of our project.

The main components of the game NEED FOR SPEAR are a paddle which is located at the bottom, an enchanted sphere which is a moving ball and obstacles. The aim of the player is to move the paddle in a way that prevents the ball from reaching the bottom of the screen and destroy all the obstacles that are located in the map. The player has 3 lives. If the player can remove all the obstacles before his chances are finished he wins the game.

The game starts with a building mode screen. In this phase the player sees 4 different types of obstacles each with a different role. The number of specific obstacles that are randomly generated appears on the screen and the player can add as much as the space allows in this part. After generating obstacles the player can change the location of the obstacles, save the already generated map and start the game. In order to start the game the player presses w and the ball starts moving. When the ball hits an obstacle it bounces back with a shifted direction and obstacles show their special powers if they have when they are hit. They can hit the paddle which decreases the paddle's lives, give paddle magical abilities, or lose health. Also, we implemented a new magician component to the game which is called YMIR. This magician flips a coin every 30 seconds. If the outcome of the coin flip is positive it randomly makes one of the three tasks it can give to the game. It can add random obstacles to the map, decrease the ball's speed to half and it can randomly freeze some obstacles. The aim of the player is to remove all the obstacles while facing some randomly and intentionally generated difficulties in the game.

For this game we worked together as a group and used object oriented programming techniques. We tried to follow the instructions that we gained from the course lectures and worked on some design patterns, diagrams and iterative development throughout the project.

Use Case Narratives

1. Shoot the ball

a. **Scope:** Need of Spear
b. **Primary Actor:** Player
c. **Stakeholders and Interests:** Player: Wants to shoot the ball to start the game by left mouse click or “W” button
d. **Preconditions:**

   i. User runs the program
   ii. Graphic User Interface (GUI) is rendered and displayed on the screen.
   iii. Player builds the map or loads previously build map
   iv. Player starts the game

e. **Postconditions:**
   i. The ball is shot
   ii. Ball moves towards a target
   iii. If the ball hits an obstacle the obstacle is destroyed.
   iv. If the player can not keep the ball in the boundaries he loses the game.

f. **Main Success Scenario:**
   i. Player opens the application.
   ii. 2 – The gui is rendered.
   iii. 3 – Player chooses game mode.
   iv. 4 – If “load game” is chosen, previous game settings are loaded
   v. If not, Player sets game parameters.
   vi. 6 – Game objects are created.
   vii. 7 – Game objects appear, all objects are shown on the window.
   viii. 8- To start the game user presses W or left mouse click and the ball is shot
g. **Extensions: ****Game can crash any time

   ➔ Player should restart the application

h. **Used Technology:**

   i. -A functional monitor.

   ii. -A functional keyboard.

i. **Frequency of Occurrence:** Only once at the beginning of the running mode

2. Save Game

a. **Scope:** Need of Spear
b. **Primary Actor:** Player
c. **Stakeholders and Interests:** Player: Wants to save the current game
d. **Preconditions:**

   i. User runs the program
   ii. Graphic User Interface (GUI) is rendered and displayed on the screen.
   iii. Player builds the map or loads previously built map
   iv. Player starts the game
   v. Player pauses
   vi. Player pushes the Save button

e. **Postconditions:**
   i. Program connects to the database
   ii. Program successfully saves the following things in the database:
      1. username
      2. - types, numbers and the coordinates of the obstacles
      3. - score
      4. - number of remaining lives
      5. - the direction of the spear

f. **Main Success Scenario:**
   i. Player pauses
   ii. Players clicks onto the “Save Game” button
   iii. The application connects to the database
   iv. The necessary information that is needed to continue the game is saved in the database

g. **Extensions:** ** Application can crash at any time

   *User restarts the application

   3. Error can occur when connecting to the database.
      3.1. A pop-up appears that says “Cannot save the game at the moment. Please try again later !”

h. **Used Technology:**

   i. Database
   ii. Mouse
   iii. Monitor
   iv. Keyboard

i. **Frequency of Occurrence:** Any time a player wants to save the game to carry on later which can happen in almost every session.

3. Load and Continue Game

a. **Scope:** Need of Spear
b. **Primary Actor:** Player
c. **Stakeholders and Interests:** Player: Wants to load the current game
d. **Preconditions:**

   i. User runs the program
   ii. The GUI is rendered
   iii. Player clicks on the Load and Continue Game button
   iv. Player enters the login information.

e. **Postconditions:**
   i. Player starts playing the loaded game.
f. **Main Success Scenario:**
   i. Player clicks on the Load and Continue button
   ii. Player enters login information
   iii. Game connects with database
   iv. Informations matches with the database
   v. The saved game appears on the screen paused
   vi. Player clicks on to Continue button
   vii. Player starts playing from where he/she left of
g. **Extensions:** * Game crashes

   **Player restarts the application

   2. Player enters invalid values

      2.1 Pop up appears which says “Invalid info ! Please try again !”

   3. Game cannot connect to database

      3.1 Pop up appears which says “Cannot Load the Game, Please try again later!”

   4. Entered information does not match with the database

      4.1 Pop up appears which says “Game cannot be found ! Check your information or Create New Game”

h. **Used Technology:**

   i. -A functional monitor.

   ii. -A functional keyboard.

   iii. -Database

i. **Frequency of Occurrence:** When a player wants to continue an existing game, probably often.

4. Add/Remove Obstacles

a. **Scope:** Need of Spear
b. **Primary Actor:** Player
c. **Stakeholders and Interests:** Player : Adds or removes the obstacle, chosen with mouse click, in the building mode
d. **Preconditions:**

   i. User runs the application
   ii. The GUI is rendered
   iii. Player chooses “create game”

e. **Postconditions:**
   i. Obstacle is removed or added depending on the input
f. **Main Success Scenario:**
   i. Player chooses “create game”
   ii. The player enters values on the table that is on the up-right side of the screen
   iii. Obstacles appear on the screen, randomly placed, with correct numbers and without any overlapping
   iv. Player chooses an obstacle
   v. Player clicks on to add or removes buttons
   vi. The obstacle is removed successfully, or, added in a suitable position

g. **Extensions:** *Game crashes

   ** player should restart the game
   5.1 Player want to remove beyond the minimum limit
   5.1.1 Program does not allow keep the number same and keeps the obstacle
   5.2 Player want to add beyond the maximum limit
   5.2.1 Program does not allow keep the number same and keeps the obstacle
h. **Used Technology:**

   i. Keyboard
   ii. Monitor
   iii. Mouse Click

i. **Frequency of Occurrence:** Almost constantly while in the building mode

5. Using Magical Hex Ability

a. **Scope:** Need of Spear
b. **Primary Actor:** Player
c. **Stakeholders and Interests:** Player: Wants to activate “Magical Hex” magical ability by selecting its icon on the display with the left mouse click or pressing the “H” keyword.
d. **Preconditions:**

   i. Game is in the running mode
   ii. Magical hex Ability is acquired randomly by destroying a “gift box” obstacle.

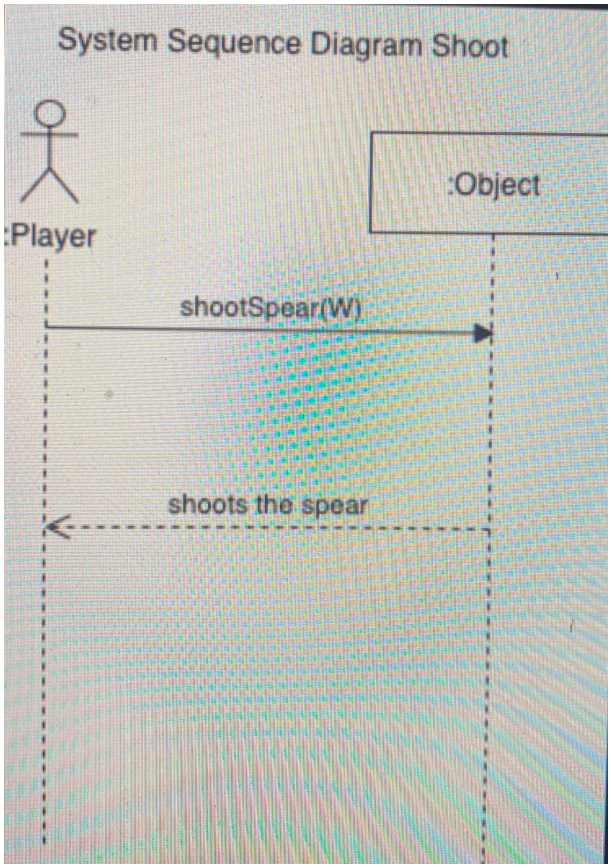
e. **Postconditions:**
   i. Two magical cannons that are pointed upwards are equipped on both sides of the noble phantasm.
   ii. Magical cannons rotate as the noble phantasm rotates.
   iii. Cannons fire magical hexes that have the same hit effect of enchanted sphere’s to the obstacles.
f. **Main Success Scenario:**
   i. Magical hex magical ability is acquired in the running mode.
   ii. Magical cannons are successfully equipped to the noble phantasm when the corresponding icon is selected by the player.
   iii. Magical hexes are fired successfully and behave the same as the enchanted sphere to an obstacle.
   iv. Magical cannons disappear after 30 seconds.

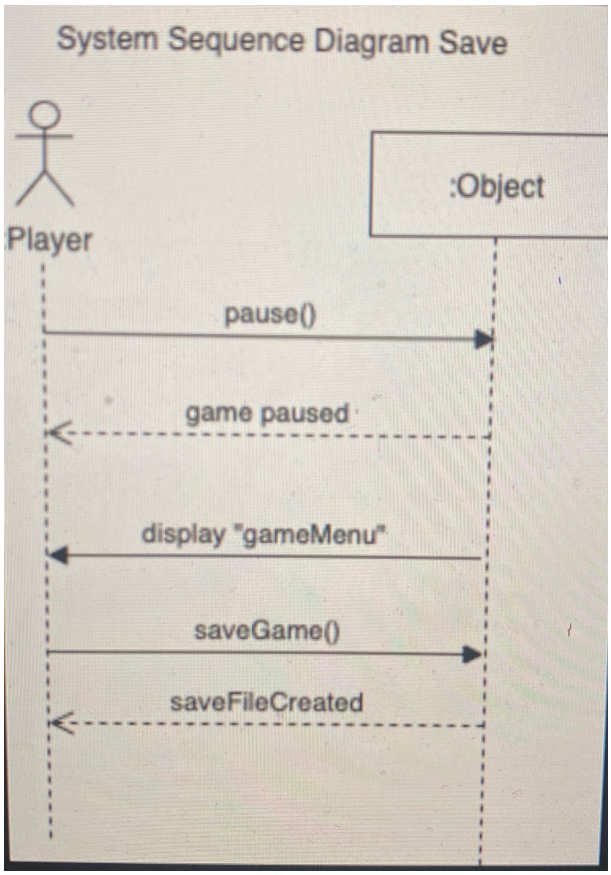
g. **Extensions:** *Game crashes

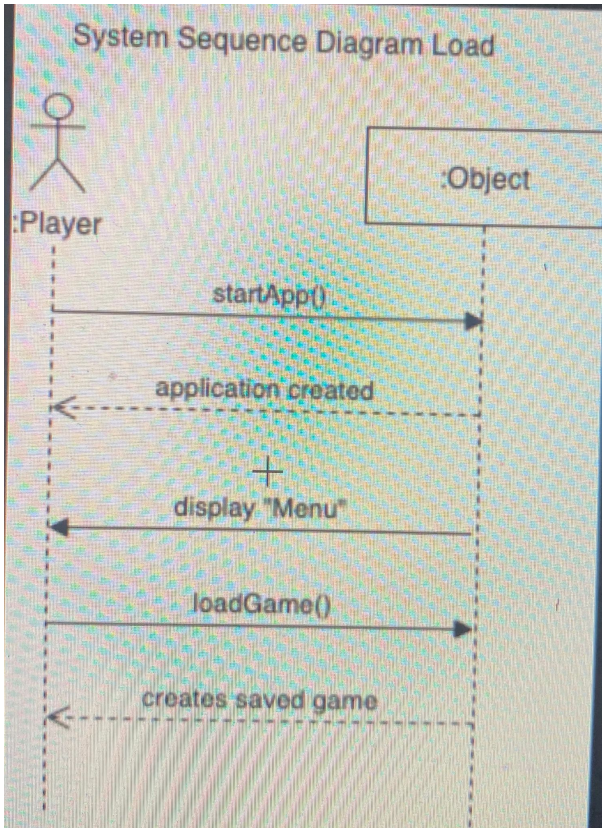
   **Player restarts the game
h. **Used Technology:**

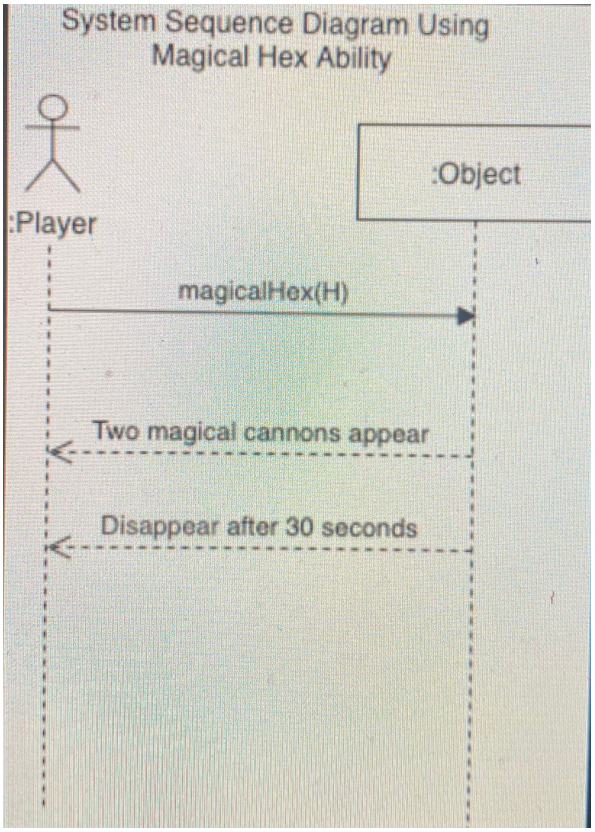
   i. Mouse
   ii. Keyboard
   iii. Monitor
   iv. Random Number Generator
i. **Frequency of Occurrence:** Low probability

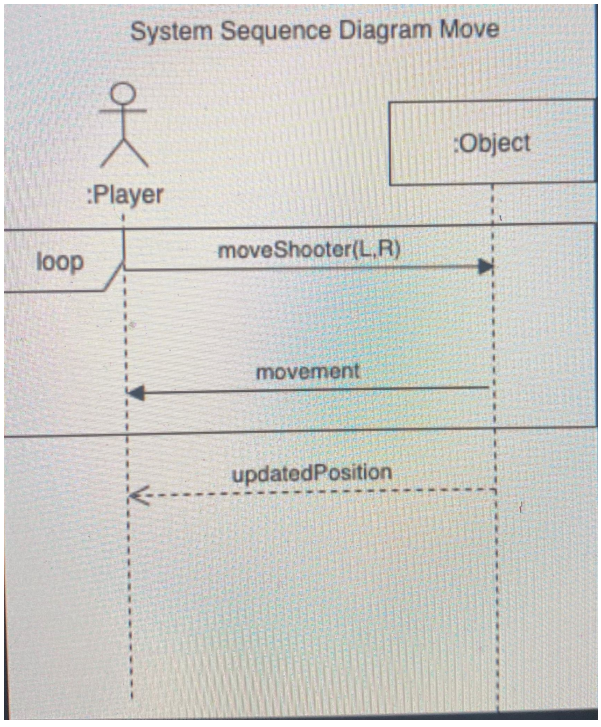
System Sequence Diagrams



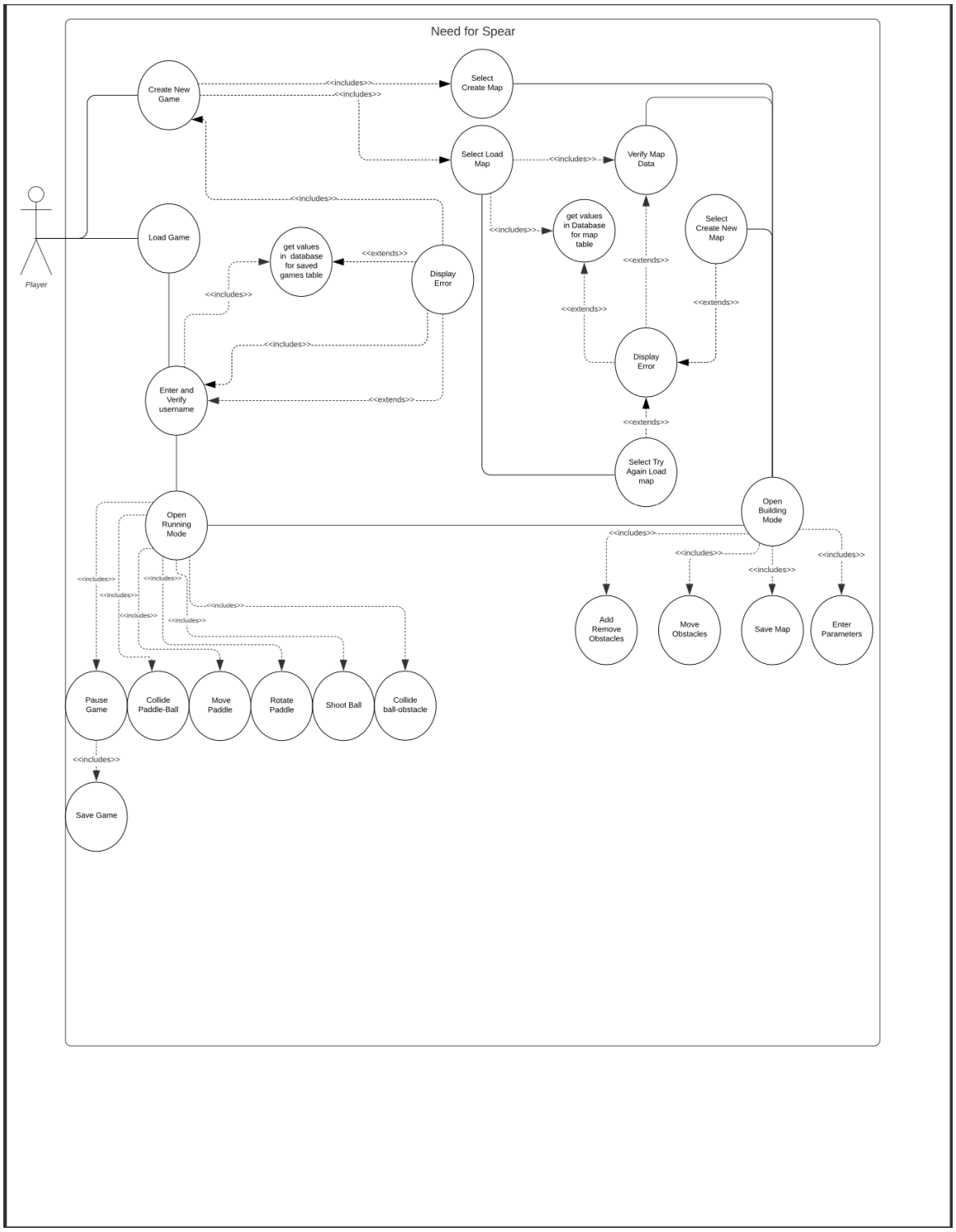



System Sequence Diagram Load
:Player
:Object
startApp()
application created
display "Menu"
loadGame()
creates saved game


System Sequence Diagram Using Magical Hex Ability
:Player
:Object
magicalHex(H)
Two magical cannons appear
Disappear after 30 seconds


System Sequence Diagram Move
:Player
:Object
loop
moveShooter(L,R)
movement
updatedPosition

Use Case Diagrams



Operation Contracts

Operation	moveObstacles()
Cross Reference	Use Case: Move Obstacles
Precondition	Application is executed and GUI is rendered. “Create Game” is selected by the user and the user specified amount of randomly located obstacles are initiated successfully.
Postcondition	obstacle.Xcoordinate and obstacle.Ycoordinate changes.

Operation	shootBall()
Cross Reference	Use Case: Shoot The Ball
Precondition	Application is executed, GUI is rendered and displayed on the screen. A new map is initiated or a previously built map is loaded and the game is started by the player.
Postcondition	noblePhantasm.shootBall() function takes place. obstacle.remove() function removes obstacles that are shot. Player.lose() if ball goes out.

Operation	rotatePaddle()
Cross Reference	use Case: Rotate Paddle
Precondition	Application is executed and GUI is rendered. The player builds the map and starts to play. He needs to rotate the paddle to shoot the ball with a desired angle.
Postcondition	noblePhantasm.rotateNegative() or noblePhantasm.rotatePositive() functions make the rotation. The noblePhantasm.xcoordinates and noblePhantasm.ycoordinates of the points in the paddle changes except the center.

Operation	pauseGame()
Cross Reference	Use case : Pause Game
Precondition	Application is executed and GUI is rendered. Player opens the game screen and the game is ready to be played. The Player can pause the game whenever he wants while playing the game.
Postcondition	pause() freezes the game. Menu.display() shows the menu.

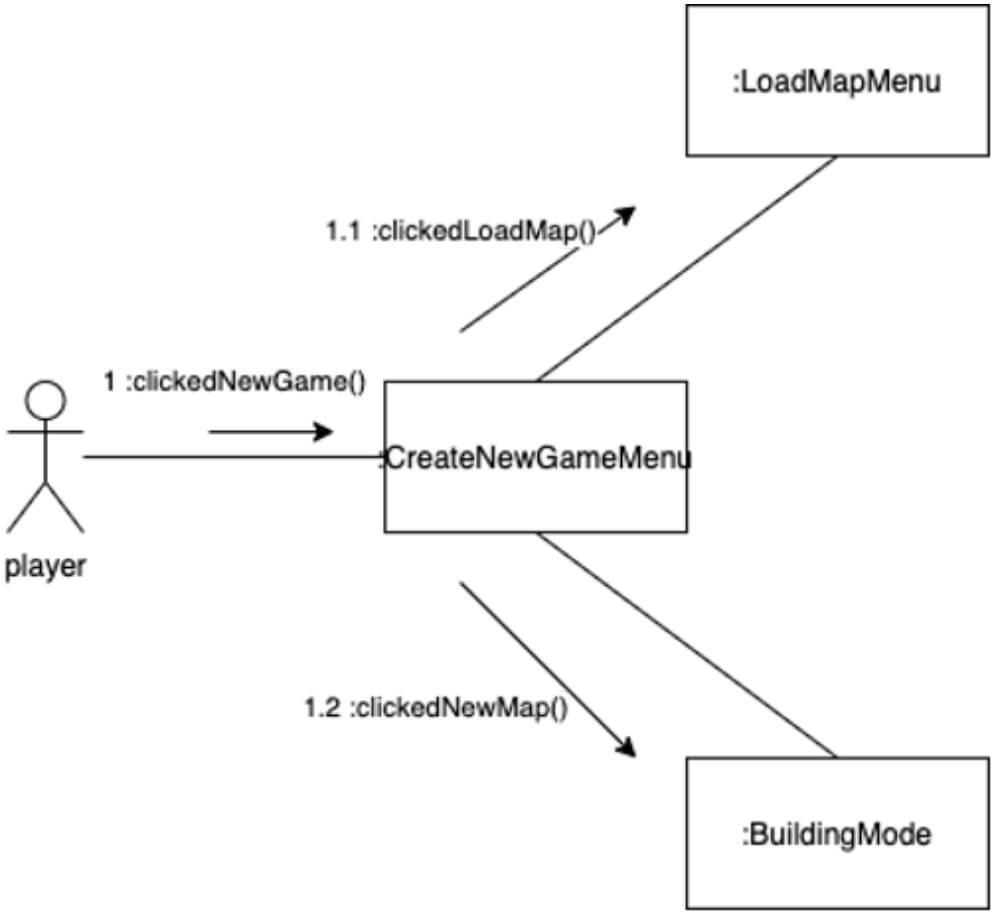
Operation	EnterParameters()

Cross Reference	Use Case: Enter Parameters
Precondition	User runs the application The GUI is rendered Player chooses “create game” Player enters number of obstacles he want to see on the game field for each obstacle
Postcondition	createObject(n) function takes place. N is the number added by the user. obstacles.add(n) adds n object instances to the <obstacles> list.

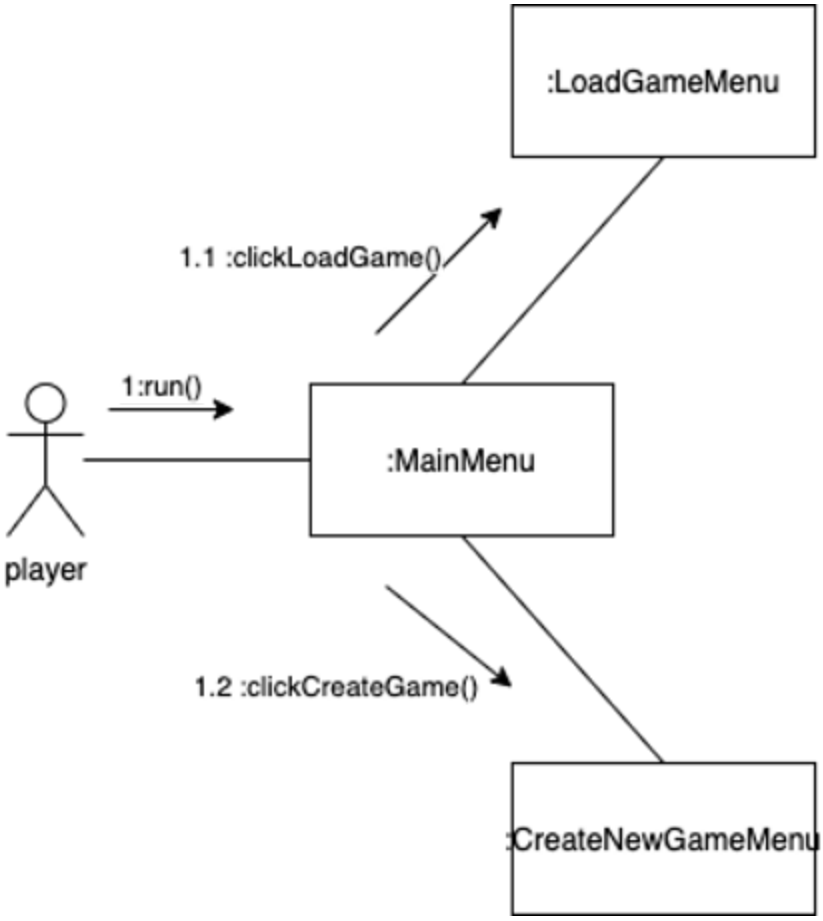
Sequence/Communication Diagrams

Communication Diagrams

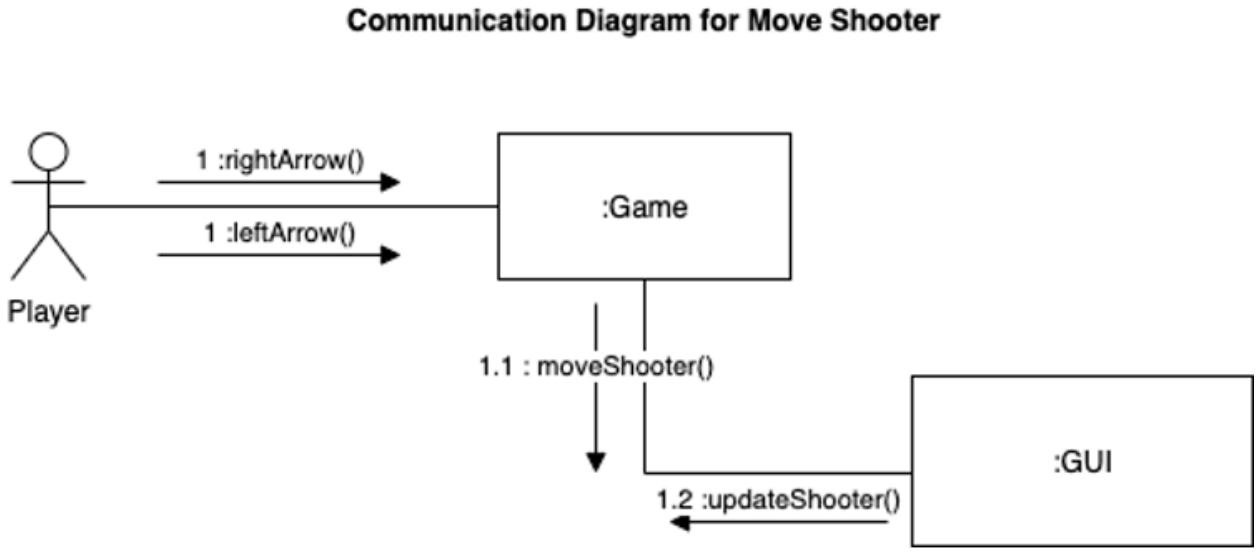
1) CreateGame Communication Diagram



2) RunApplication Communication Diagram



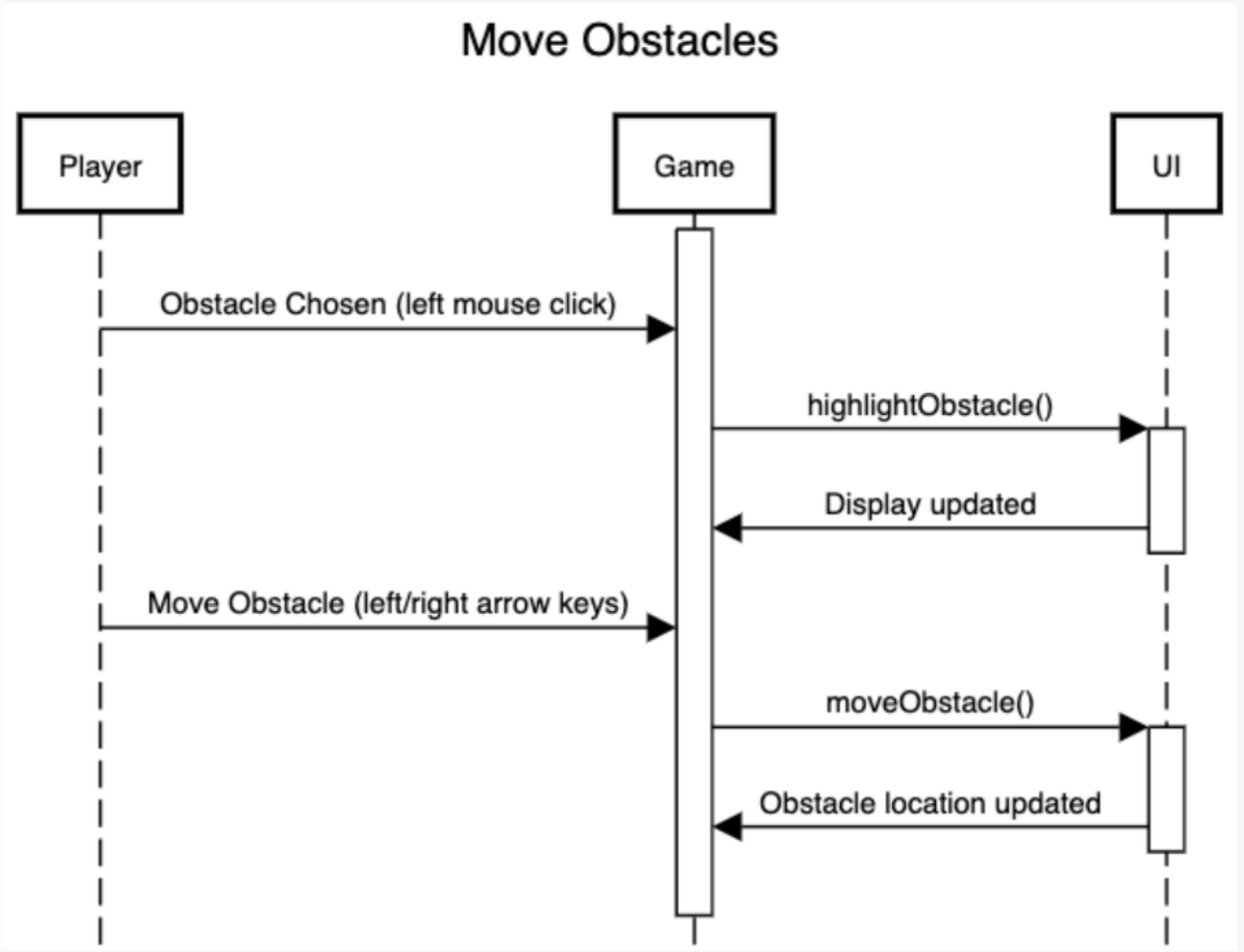
3) Move Communication Diagram



Sequence Diagrams

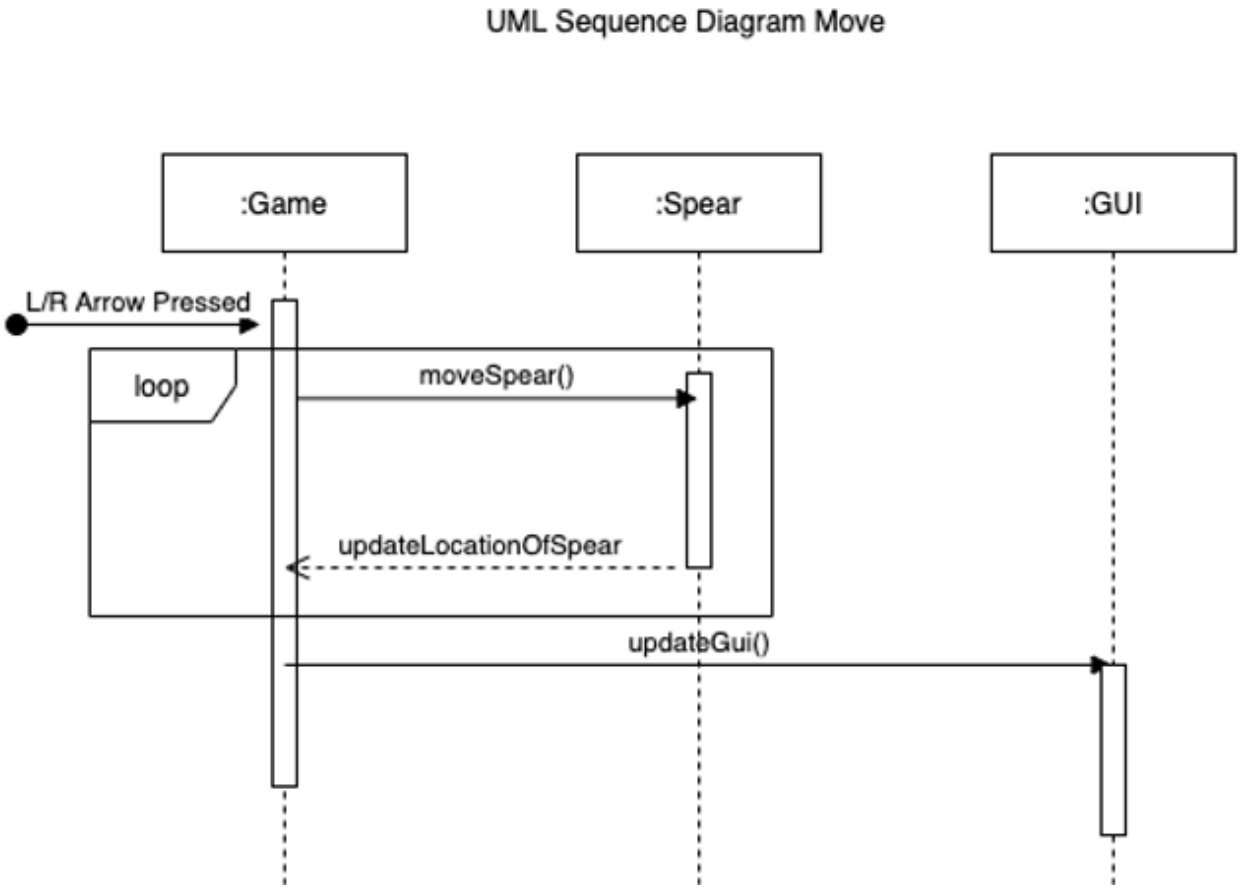
1) Move Obstacle Sequence Diagram

Our first design thought was this but then we converted mouse click to space button and used arrow keys to move at building mode.

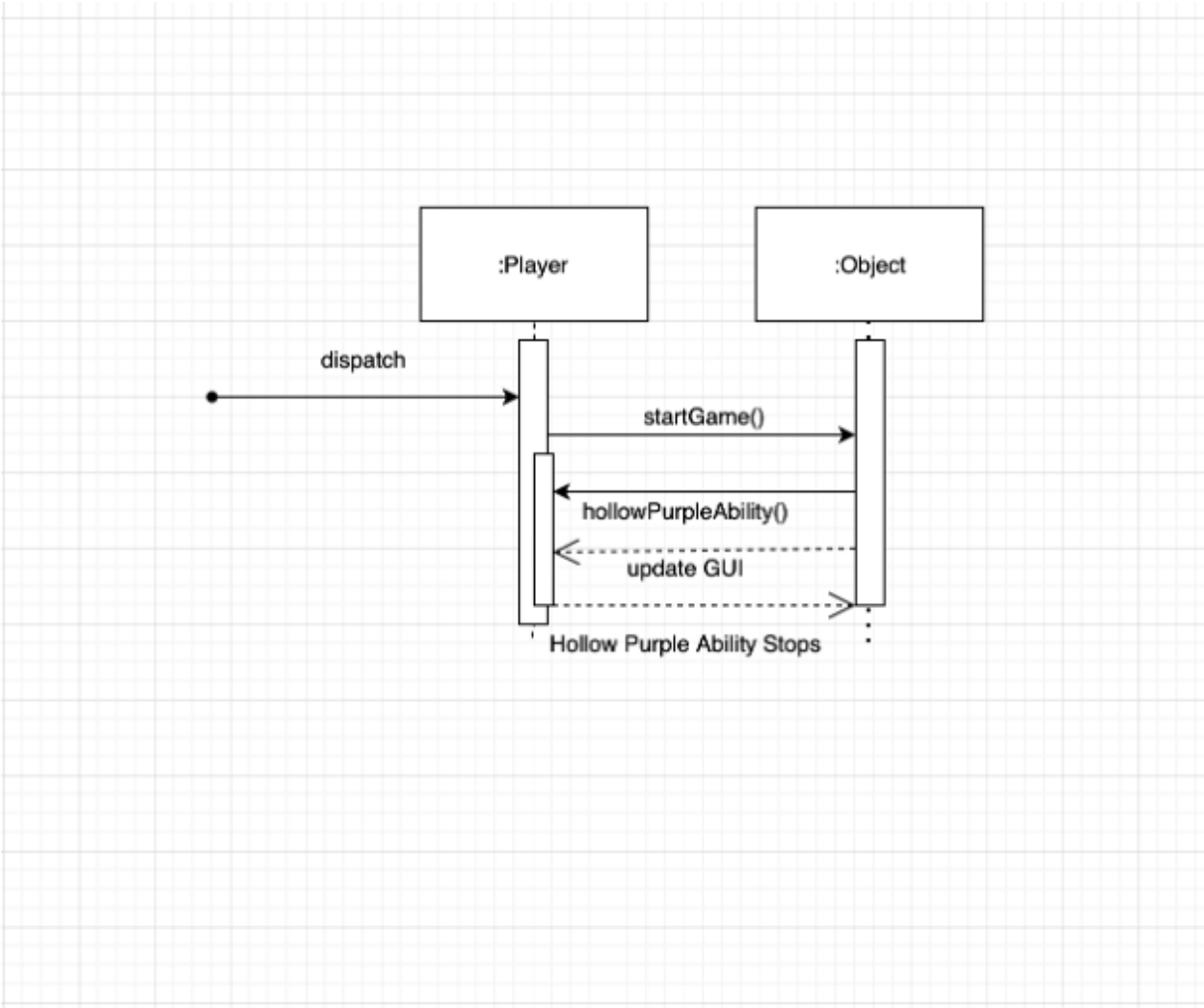


2) Move Sequence Diagram

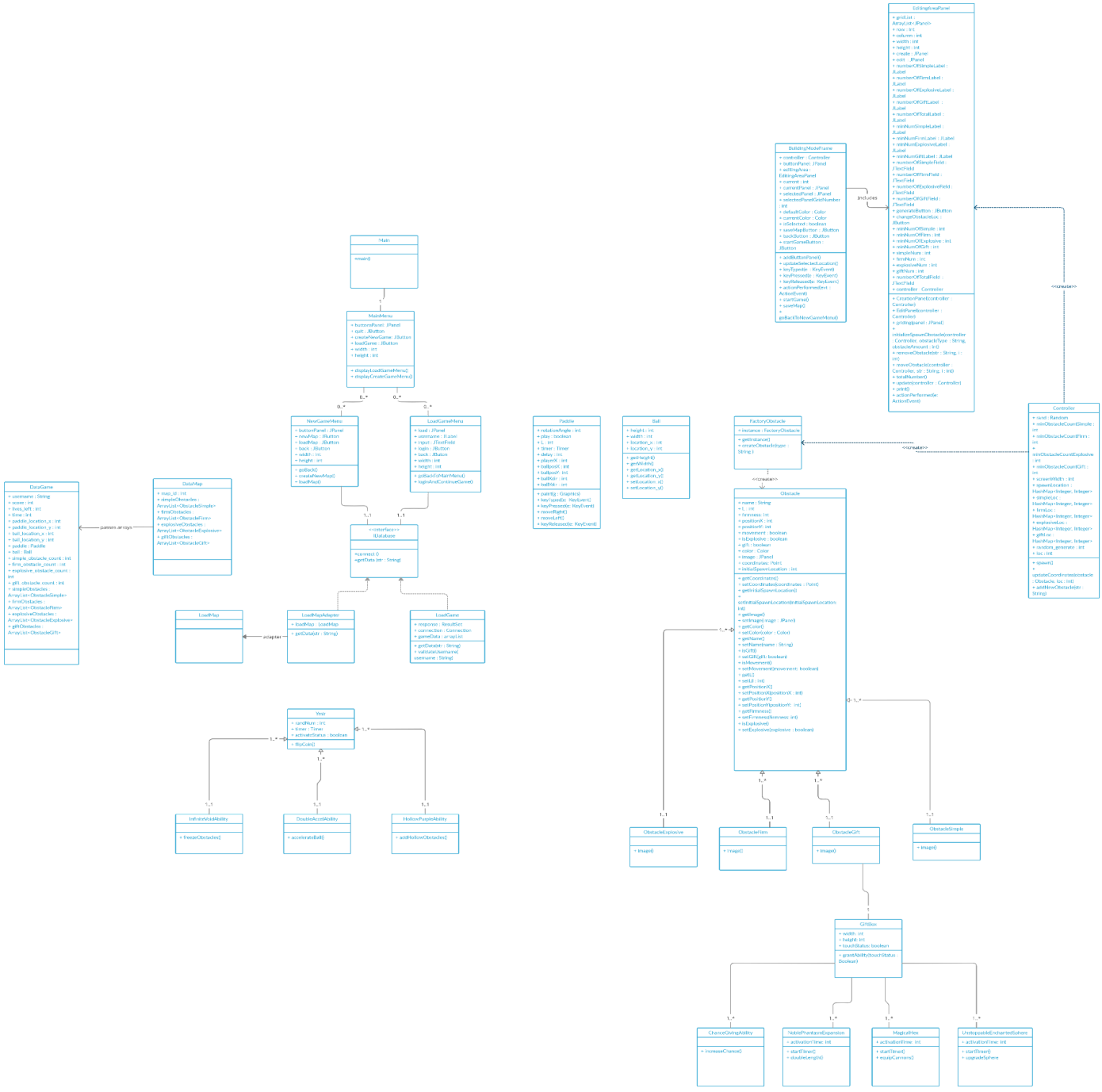
Our initial goal was the create move as one method but then we converted this method to two different method as moveRight() and moveLeft() but the principal stayed same in both of the methods and we updated GUI in GamePanel instead of GUI class and we decided to rename the spear class as paddle.



3) HallowPurpleAbility Sequence Diagram



Class Diagrams



Discussion of design alternatives and design patterns

During the project, we understood the importance of design patterns and principles. Before taking this course, as we have not worked in a team or created a project this big, we have not experienced the difficulties of editing an existing code. After the mid-demo period, which corresponds to a level where everything was initialized but most of the things are not fully done, every group member understood the importance of writing a flexible, readable, sensible code as well as agile modeling which was already done because of the course requirements. For this manner, we had to follow the GRASP principles. With these principles our program ended up to be a one that was easy-to-understand and easy-to-edit. We did not violate the model-view separation thus our program does not face any errors when a part is removed or modified in the UI with the controller pattern. As we followed GRASP principles, our program supported low coupling and high cohesion which made our further additions and implementations in the phase 2 much easier because according to these principles the dependency of classes with each other is lowered in the purpose of lowering the impact when making changes. In our project, we implemented singleton and factory design patterns in terms of GoF design patterns.

Singleton Design Pattern

The singleton pattern was used in the paddle class. This is because we had only one paddle while we were in the running mode and further changes in the paddle, for example the “paddle expansion” ability, should have been done on the current instance that is in the building mode. Furthermore, the controller class should have been initialized only once and the one that is created should be used in the building and running mode. Because of this using singleton was crucial as it restricts the instantiation of a class.

Simple Factory Design Pattern

We used the simple factory pattern in creating the obstacles. Our obstacle factory gave us the advantage of initializing the obstacles dynamically as well as making our obstacle additions clearer. With the offerings of this pattern the program ran with a better performance and became smoother in terms of readability and functionality. Also we created the instance of the obstacle factory with the singleton pattern so that no other factories would be created and the obstacles that are added in our array lists originated from different sources of factories.

Strategy Design Pattern

During our work we have thought that it can work well with our obstacle creation methodology but later we thought as there are few types of obstacles, simple factory pattern would be the better option with a super class “obstacle” containing some method that can override the shape which can later be transformed into the “explosive obstacle”. As we found

out that we can be more efficient with simple factory in the context of obstacles, we decided not to increase the number of classes for obstacles and over complicate things.

Adapter Design Pattern

At first, we designed our program to use the adapter pattern in the database package and adapt the save and load classes with an interface so that we can modify the according methods and values. We thought of creating three tables: load_game, obstacles, and load_map.

Load_game and obstacles would be joint tables. Load game will contain; username, time, score, lives left, simple obstacle count, firm obstacle count, explosive obstacle count, gift obstacle count, magical abilities, paddle location, ball location. Obstacle table would contain username, obstacle type, the locations of the obstacles, firmness of the obstacles etc. On the other hand, the load map table will only contain map id and the locations of the obstacles. For this reason we wanted to use the adapter design pattern and differ the getData() method in it as they will get different types of values from different tables.

Observer Design Pattern

Observer design pattern would have been a useful pattern to connect the user interface package and the domain package. One of its pros is that it can be added or removed at any point of the project. One of the cons of this pattern is that it involves inheritance which will be hard to handle in the introductory level of the project. Our controller class has done the work as an implementation of the observer pattern. Thus we did not create an extra class for the pattern.

Supplementary Specifications

Functionality:
Most of the action is produced in the game time (enchanted sphere hitting an obstacle, magical abilities, etc.).

Usability:

Human Factors
The Player should be easily able to navigate through the game menu.
The Player should be easily able to distinguish between the colors present in the game.
The Player should be able to control the Noble Phantasm as the game is being played.

Reliability:

Recoverability
In case of a crash or a lost game, the player should be able to restart the game.

Performance:
The game should be able to respond and be played according to the actions produced by the player and the game reaction time should be quick.

Supportability:

Adaptability
The game should be able to react, variate and adapt as the game is played.
Configurability
In order to create a game environment, the user interacts with the system to specify the number of each obstacle type (top right box). When a user enters the number of obstacles, the system puts these obstacles in random places. Then, users can change their places, remove some of them and/or add more obstacles by mouse clicks. System always places the obstacles so that none of them overlaps with the others. Also, the system should reject the user's attempt if the user tries to place overlapping obstacles.

Implementation Constraints
Our team Robot uses Java to implement the features into the game.

Application-Specific Domain Rules

General Game Rules:

- Noble Phantasm moves horizontally, and can be rotated by up to 45 or 135 degrees.
    - The rotation angle is going to be changed by a rate of 20 degrees/second until reaching 45 degrees if the key A is pressed, or 135 if D is pressed.
    - Once the key is released the noble phantasm will go back to its horizontal state, and the rotation angle is going to be changed by a rate of 45 degrees/second.
- The noble phantasm length L is 10% of the screen width.
    - This L is going to be used as bases to measure other variables in the rest of this document.
- The noble phantasm thickness T is 20px.
- Movement speed is:
    - If the left or right arrow is pressed and released: The noble phantasm should move by an offset equal to L/2 with a speed of L/second.
    - If the button is down, it should move with the speed of 2*L/second.
- When the enchanted sphere hits a moving object:
    - If the direction of the movement of the object is the same as the direction of the component of the enchanted sphere velocity that is parallel to that movement direction, the enchanted sphere will reflect according to the roles described in the non-moving object case. However, the speed of the enchanted sphere will increase by 5px/second.
    - If the direction of the movement of the object is the opposite of the direction of the component of the enchanted sphere velocity that is parallel to that movement direction, the enchanted sphere will reflect with an angle of 180 degrees relative to the line of the original enchanted sphere movement direction and the speed of the enchanted sphere stays the same.
    - If the direction of the movement of the object is perpendicular to the direction of the enchanted sphere velocity, the enchanted sphere will reflect with an angle of 45 degrees relative to the line of the moving object movement direction and the speed of the enchanted sphere stays the same

Building Mode Rules:

- There has to be at least 75 simple obstacles,
- At least 10 firm obstacles,
- At least 5 explosive-obstacles, and
- At least 10 gift obstacles (we have in total 4 magical abilities, so each magical ability should appear at least once in a gift obstacle, the rest of the gift obstacles are assigned magical abilities randomly).

Once the minimum criteria have been satisfied, the user can save the game to play later or he/she can immediately play it.

Rotate Shooter:
Shooters will be rotated if the player presses the “A” or “D” button on the keyboard.
Player presses the “A” button and the shooter will be rotated 10 degrees to the left.
Player presses the “D” button on the keyboard and the shooter will be rotated to the right.
If the shooter is already 90 degrees rotated to the left and the player presses “A”, the shooter will not be rotated.
If the shooter is already 90 degrees rotated to the right and the player presses “D”, the shooter will not be rotated.
The rotation speed of the shooter is 90 degrees/sec.
This process will only work for “A” and “D” buttons on the keyboard, the shooter will not rotate if the player presses any other arbitrary button.

Move Shooter:
Shooter will be moved left or right as the player presses the “Left Arrow” or “Right Arrow” buttons on the keyboard.
Player presses the “Left Arrow” key and the shooter will be moved to the left as long as the key is pressed.
Player presses the “Right Arrow” key and the shooter will be moved to the right as long as the key is pressed at the speed of L/sec.
If the shooter is at most left of the game screen and the player presses the “Left Arrow”, the shooter will not be moved.
If the shooter is at most right of the game screen and the player presses the “Right Arrow”, the shooter will not be moved.
The moving speed of the shooter is L/sec.
This process will only work for “Left Arrow” and “Right Arrow” buttons on the keyboard, the shooter will not move if the player presses any other arbitrary button.

Magical Abilities:

Change Giving Ability

This ability increments the chances of the player

Noble Phantasm Expansion

This ability doubles the length of the noble phantasm. It can be activated by either pressing the button “T” or clicking on its icon on screen. Once it is activated, it lasts for 30 seconds.

Magical Hex

This ability equips the noble phantasm with two magical canons on both of its ends. The canons should point upwards and they rotate as the noble phantasm rotates. They can fire magical hexes which have the same damage effect as the hit of the enchanted sphere. It can be activated by either pressing the button “H” or clicking on its icon on screen. Once it is activated, it lasts for 30 seconds.

Unstoppable Enchanted Sphere

This ability upgrades the enchanted sphere and makes it much more powerful, such that if it hits any obstacles, it destroys it and passes through it regardless of its type. This upgrade only lasts 30 seconds after it is activated.

Controls

Controls in Building Mode

Left Mouse Button	Choosing an obstacle (while the cursor is on an obstacle)
Arrow Right	Move the obstacle to right (once the obstacle is chosen)
Arrow Left	Move the obstacle to left (once the obstacle is chosen)
Arrow Up	Move the obstacle to left (once the obstacle is chosen)
Arrow Down	Move the obstacle to left (once the obstacle is chosen)
Backspace	Removes obstacle
Space	Adds obstacle
Enter	Confirms obstacle’s location

Controls in Running Mode

W or Left Mouse Button	Shoots the Enchanted Sphere
Arrow Left	Moves the Paddle to left
Arrow Right	Moves the Paddle to right
A	Rotates the Paddle clockwise
D	Rotates the Paddle counter clockwise
T	Activates Noble Phantasm Extension
H	Activates Magical Hex

Glossary

TERM	DESCRIPTION	ALIASES
Enchanted Sphere	The enchanted sphere has the dimensions of 16x16 pixels. The enchanted sphere bounces between the obstacles and the noble phantasm. If the enchanted sphere hits an obstacle it will have a certain effect depending on the type of the obstacle or the currently activated magical abilities. When the enchanted sphere is going down, if the player does not move the noble phantasm to hit the enchanted sphere, it will be lost and the player would lose a chance. The player normally has three chances at the beginning of a game. When losing the three chances, the game is over, and the player loses the game.	
Noble Phantasm	The player has control over the noble phantasm (paddle), he/she should use the noble phantasm (paddle) to direct the enchanted sphere (ball) to destroy as many obstacles as possible and at the same time protect the enchanted sphere (ball) from falling.	
Simple Obstacle	Can be broken in one hit. When broken, it disappears	Wall Maria
Firm Obstacle	These obstacles are more difficult to destroy. Each one contains a number written on it, which corresponds to the number of hits it requires to be destroyed. After every hit it receives, the number decreases by 1, and the obstacle disappears once the number reaches zero.	Steins Gate
Explosive Obstacle	This obstacle has a circular shape and it explodes once it is hit. Once exploded, its remains fall downwards towards the noble phantasm. If the remains touch the noble phantasm, the player loses a chance.	Pandora’s Box
Gift Obstacle	This obstacle can be destroyed in one hit like the simple one. Once destroyed, it drops a box	Gift of Uranus

	downwards towards the noble phantasm. If the noble phantasm touches the box, then the box opens and rewards the warrior with a magical ability that can be either used to support the warrior, or to create more challenges and obstacles for the other player.	
Chance Giving Ability	This ability increases the player's chances by 1.	
Noble Phantasm Expansion	This ability doubles the length of the noble phantasm. It is not necessarily activated once it is received. The player can choose to activate it whenever they want by either pressing the button T, or pressing its icon on the screen. Once activated, it lasts for only 30 seconds, after which the noble phantasm returns to its original state.	
Magical Hex	This ability equips the noble phantasm with two magical canons on both of its ends. The canons should point upwards and they rotate as the noble phantasm rotates. They can fire magical hexes that can hit the obstacles. A hex hit has the same effect as the hit of an enchanted sphere. It does not activate immediately, but can be activated by pressing H or pressing its icon on the screen. Once activated it remains active for only 30 seconds and then disappears afterwards.	
Unstoppable Enchanted Sphere	This ability upgrades the enchanted sphere and makes it much more powerful, such that if it hits any obstacles, it destroys it and passes through it regardless of its type (even for the firm obstacles). This upgrade only lasts 30 seconds after it is activated.	