Bilkent University

Department of Computer Engineering

CS319 Term Project

Defender: Fiyuv++

Final Report

Doğukan KÖSE, Hamza PEHLİVAN, Musab OKŞAŞ, Meryem EFE, Aybüke ERTEKİN

Instructor: Eray TÜZÜN

Teaching Assistants: Barış ARDIÇ, Alperen ÇETİN, Hasan BALCI

Contents

Final Report

Defender: Fiyuv ++

1. Introduction

For menu part of the game, we have decided to code it for the second demo and iteration, because we thought that the menu is not so much related to game's logic, that's why we will make it later.

In pre-boss scene, firstly, we tried to add main components of game such as firing, killing enemies, moving enemies, moving spaceship. However, we have not added different enemy types or enemy buildings and meteors to the game yet. Because, we thought that if we can make the game with its main components, then it is easy to add new extra features. Also, in this scene we tried to make our objects by extending ImageView instead of extending Rectangle class so that we have a scene that is similar to mockup screens of the game. Moreover, we added radar panel at the top of the screen.

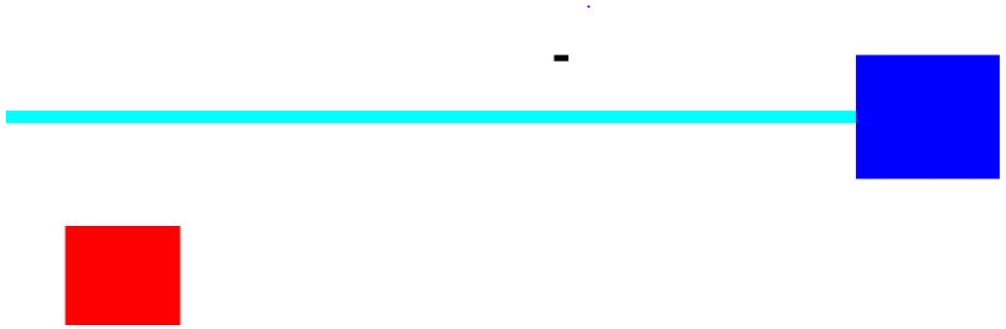
Boss Scenes:

-Functionalities of level one boss is completed. However, other bosses have not been implemented yet.

-There is no good-looking UI for boss scenes yet. Therefore, background of boss scenes and images of bosses will be updated.

-Boss scenes work independent from pre-boss scene. This will be changed to provide coherence.

In figure 1, left rectangle is spaceCraft and right rectangle is boss of level 1. The boss currently uses its laser ability and fires a bullet.



In figure 2, player has moved right (the gap between spaceCraft and boss is decreased), and the player is shooting and damaging the boss.

2. Design Changes

At the start, we hypothetically constructed our system models such as object class diagrams, sequence diagrams according to the functional requirements. However, as we started to implement the game: Fiyuv++, we made both minor and major changes in our system models because there were some inconsistencies and poorly chosen design features.

First and the last situation of the object class diagram are shown in the figures below. The biggest major change in between is that at the first diagram we tried to implement boss scene and pre boss scene together, hence, both of the entity classes of pre boss scene and boss scene were using same map class and controller class. Since these two are separated considering most of the logic and classes such as Enemy, Building, Meteor, Buff, Boss, Laser, Rocket we decided to divide the class diagrams of these scenes into two and implement them by using different maps and controllers. Commonly used objects such as LocatableObject, Spacecraft, Bullet are shown in the both diagrams. We had also minor changes mostly in classes we defined at first diagram. The common biggest reason of these changes is at the first design we just poorly placed functions, attributes and absent functions, attributes to do the business logic. When we start to implement the project, we figured out better places for some functions. For example, the functions in SimpleEnemy interface in the first design are moved to other locations such as PreBossMap for wandering around and radar search or EnemyBuilding2 for evolving. Thus, we moved SimpleEnemy interface from the design. Another example is combining health attribute of different object in LocatableObject class in order to manipulate health and isDead attribute together. Furthermore, we detected that we need attributes to handle relative layout and time related business logic, so, we added new attributes and functions related to these concepts. Lastly, we tried to design PreBossGameController and PreBossMap more realistically considering how they will handle business logic such as spacecraft's, enemies', bullets' motions or game states for pause, resume, winning, losing. Desing changes considering the object class diagrams of fiyuv++ can be shown at the figures below.

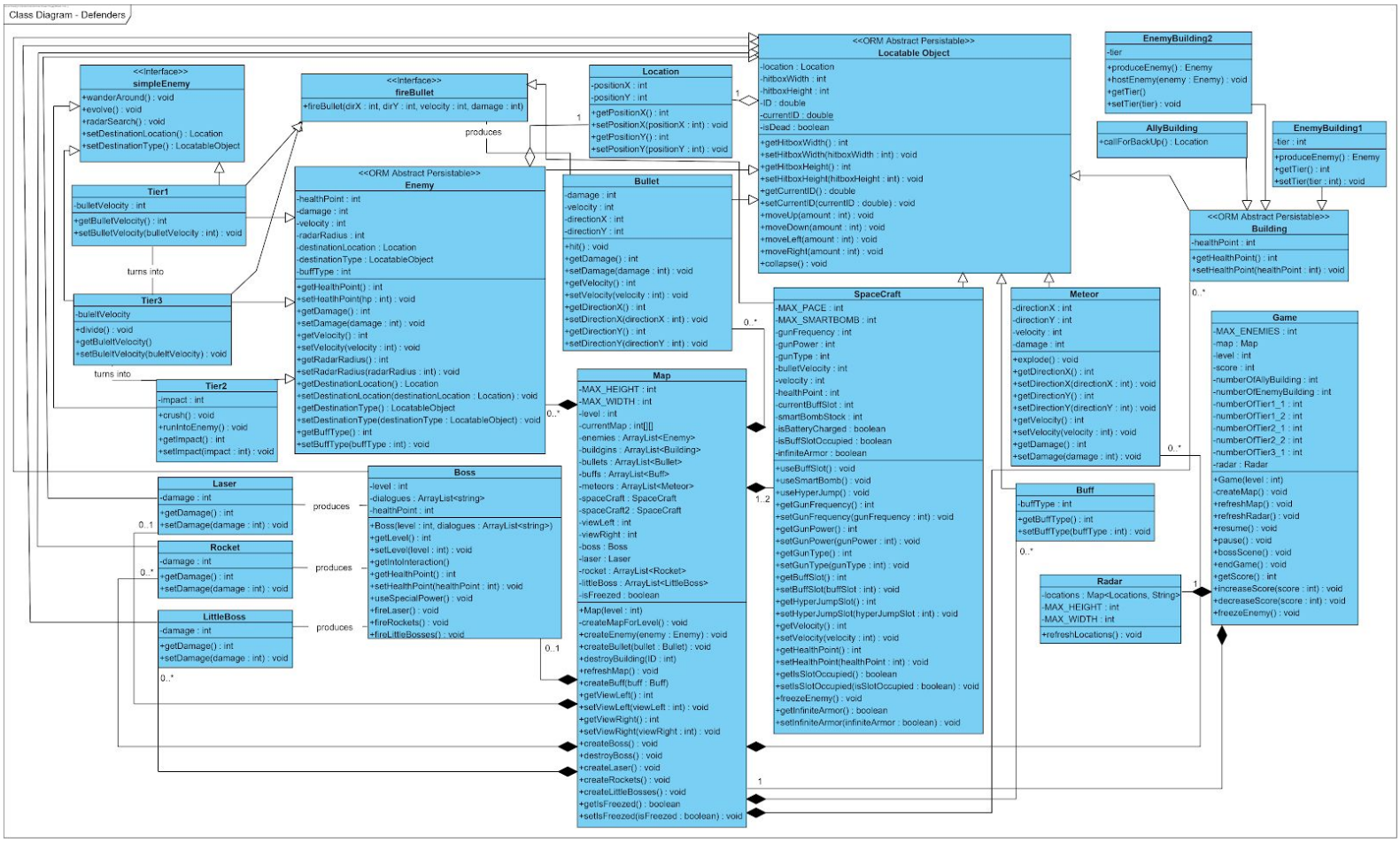


Figure 1: First Object Class Diagram

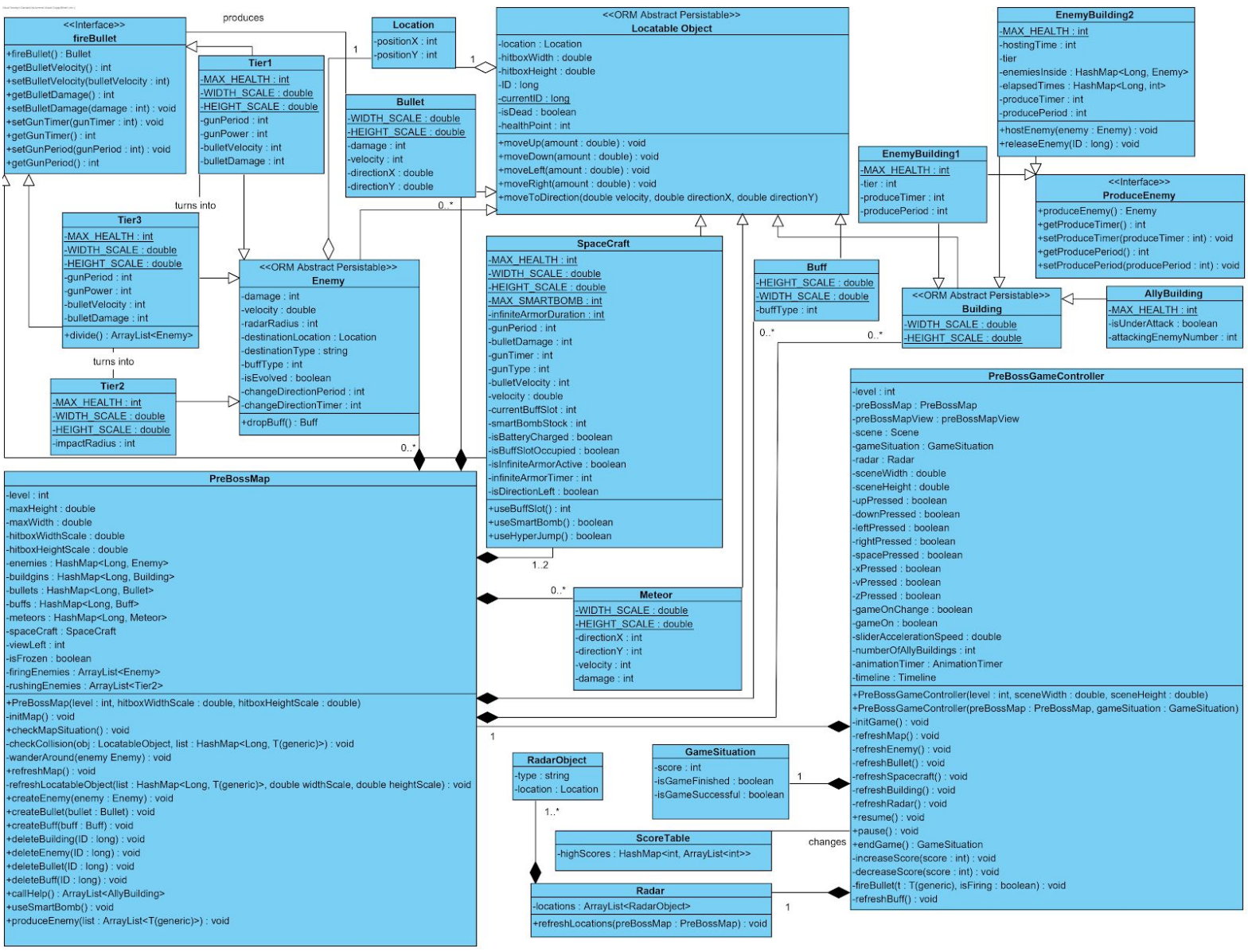


Figure 2: Changed Object Class Diagram Part 1

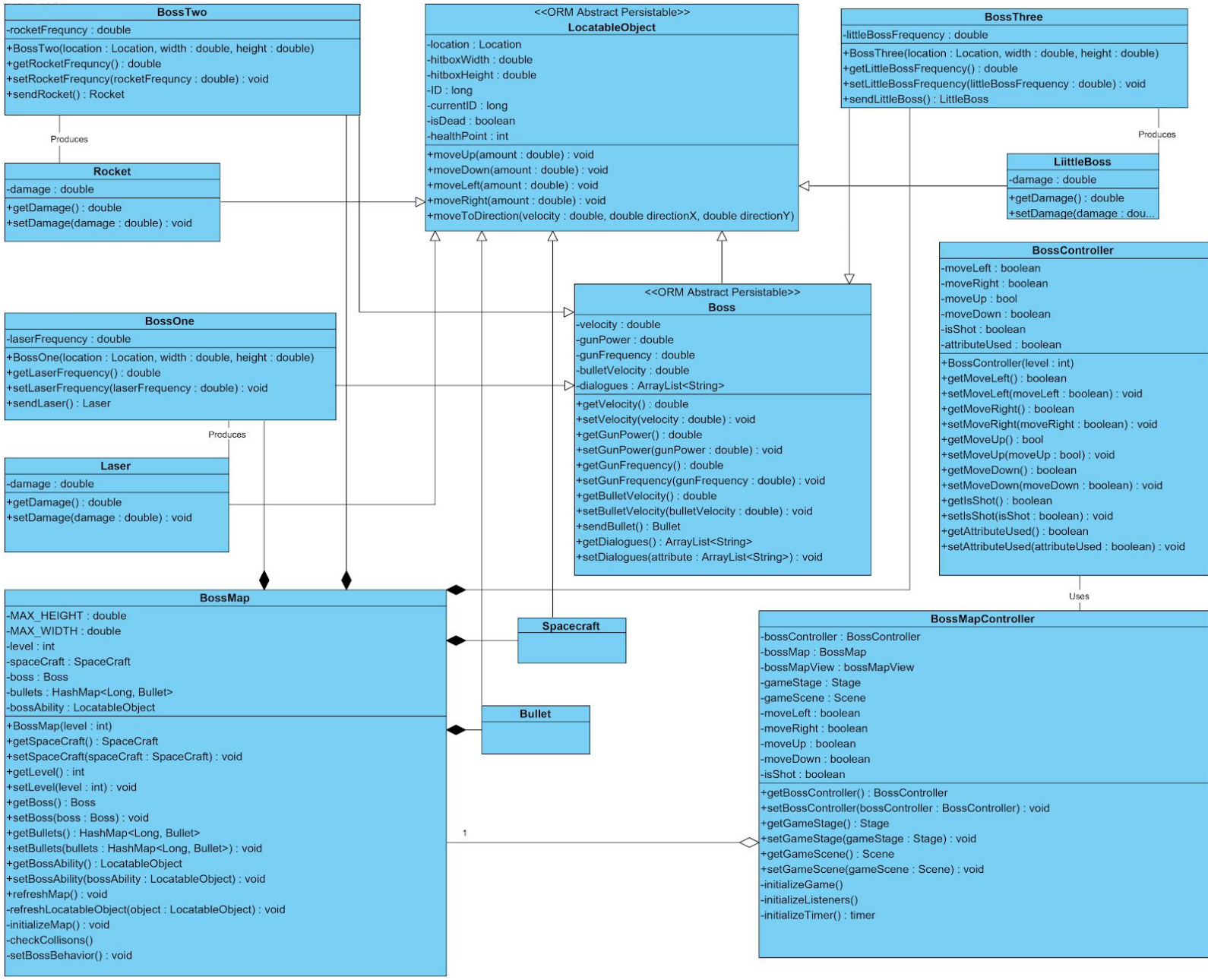


Figure 3: Changed Object Class Diagram Part 2

3. Lessons Learnt

We can divide lessons that we learnt into two parts. First part is about technical issues related to implementation of the project which includes concepts about IntelliJ, Java, Javafx, GitHub, Maven. Second part is about less technical more documentation and design related issues about analysis and design report.

To begin with, none of us had any experience with Javafx and some of the group members didn't have any experience with IntelliJ, GitHub, Maven. Moreover, we needed to brush up our memories with Java to remember what Java offers to developers such as inheritance, polymorphism, encapsulation, abstraction, generics, serialization, collections. Therefore, all members learned what is necessary for implementation of the project and during this process we gain many experiences. For example, considering Javafx, we learned how to control classes and object by using AnimationTimer, TimeLine, or ChangeListener. As we spent time using and learning Javafx, we noticed some utilization that offers such as Javafx properties or Javafx Collections, i.e. instead of using Boolean we used BooleanProperty or instead of using ArrayList we could use ObservableList. These utilizations use Java, but in an optimized way. Some of these utilizations, we figured earlier in implementation process and used. For some others, we figured a little later, so we needed to evaluate that how much we needed to spent to change our implementation. According to the evaluation we decide whether we will use the utilization or not. Lesson that we learn from this could be stated as before jumping into implementation, comprehensively investigating the tools for usage would be better approach. Another example of this lesson is when we spend almost 2 hours to understand why we cannot access to the classes imported by Maven, which was because we needed to require the classes for configuration after importing by Maven.

Secondly, we also didn't have any experience with design and documentation of a software. However, we needed to immediately jump into design and documentation

processes. Thus, we made a lot of mistakes at the beginning and we still continuing to do them. As we learn from and study to the lectures, we try to correct these mistakes. One example of this mistakes was to ignore talking about small details such as class names, as if we already had consensus on them or we could discuss them later. Ignoring the small details lead to inconsistency in both documentation of reports and designing the diagrams. We didn’t even had consistency about our game name. Hence, the decisions about small details of the software shouldn’t be ignored or postponed.

4. User’s Guide

4.1. System Requirements & Installation

Since Fiyuv++ is a simple Java game, it doesn't need too many system requirements. The only things you need are JDK and preferably IntelliJ IDE and source code will be available on GitHub.

4.2. How to Use

When Fiyuv++ game is opened, the player will see the main menu which includes "Single Player Game", "Two Player Game", "How to Play", "Settings", "Credits", and "Quit" options.

If the player clicks on How to Play button, he/she can see details of the game such as aim of the game, enemy types, bonuses, getting score etc.

In order to change volume and keys settings, the player can go to Settings. In Settings part, volume level can be arranged. Also, the player can change keys which are used for moving and firing. For a single player, as default, arrow keys are assigned for left, right, up, bottom moves and space key is assigned for firing.

Besides, player can see contributors to development of the game in Credits part or exit the game by clicking Exit button.

If player wants to start game, there are two options which are Single Player and Two Player. When one of these two options are selected, player or players will choose

level. After that, game will start and player should try to defeat all enemies and get high score.

*** The menu part has not been implemented yet.