

CS 319

Object-Oriented Software Engineering  
Fall 2014

**Conquerors**

Final Report

Group H  
Çağla Arıkan

Metin Enes Uçan

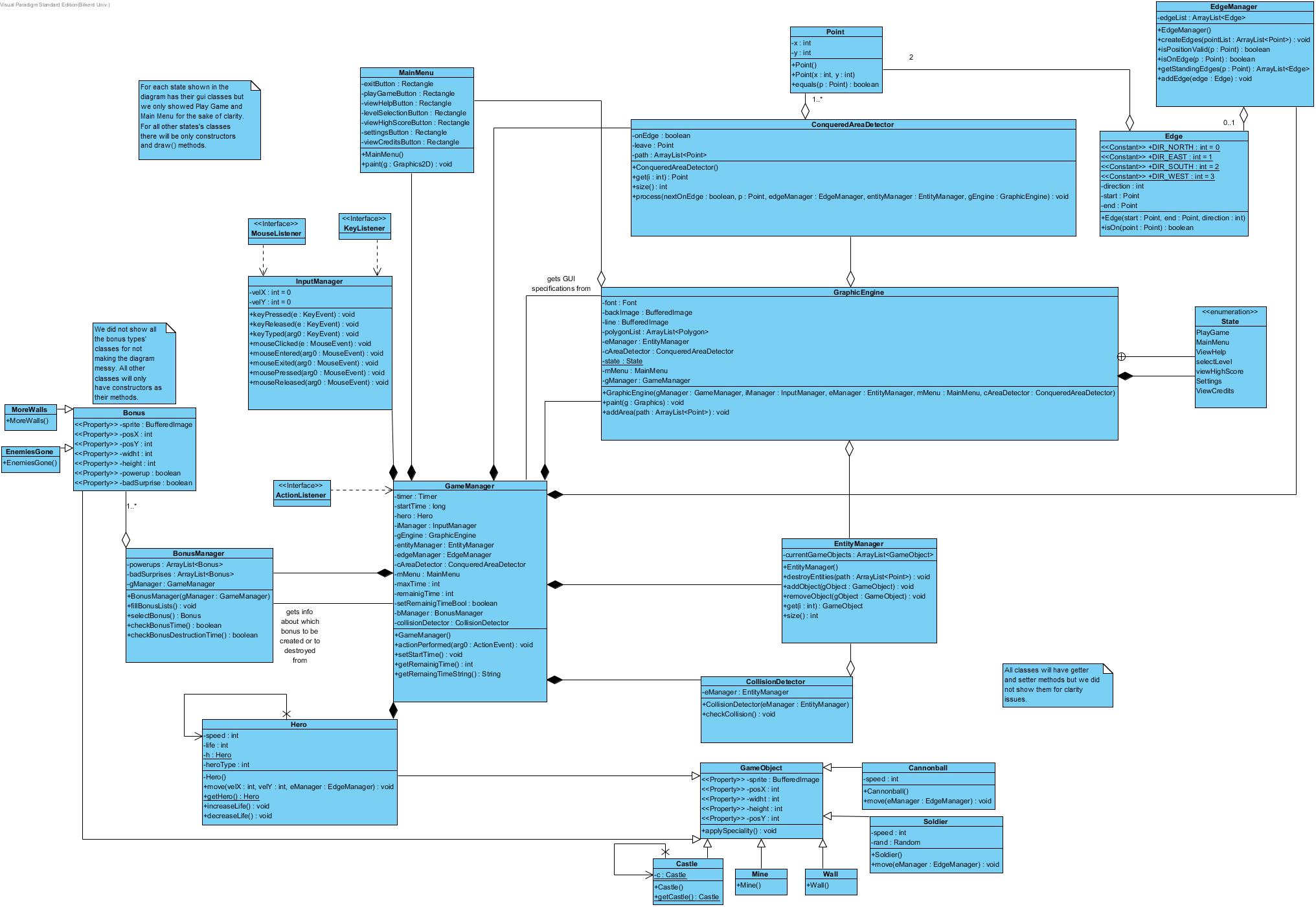
Can Akgün

İsmail Can Dereli

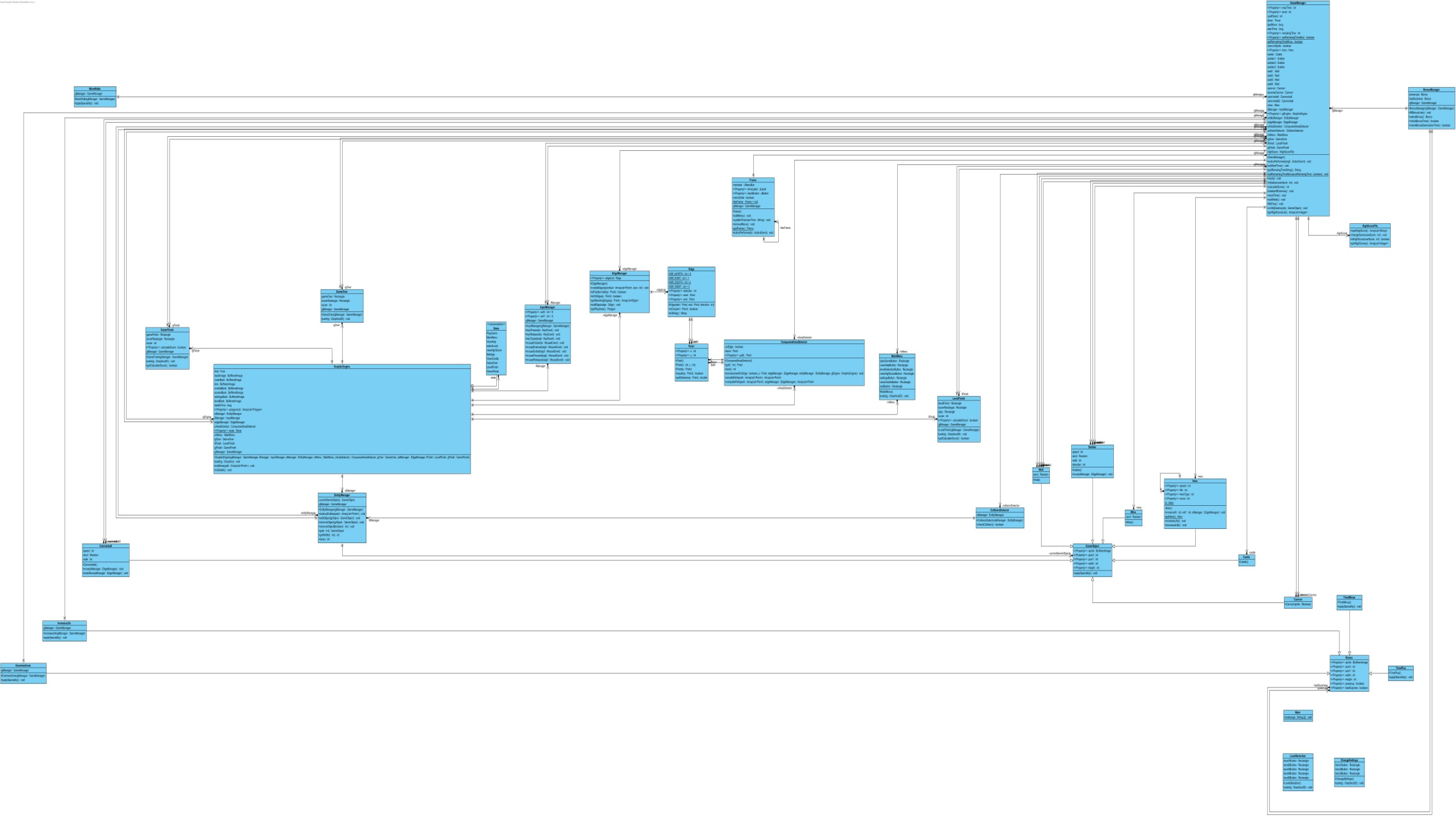
In this report we will discuss how the implementation process went, what changes we made to our initial design and why we did those.

The implementation stage of our program went easier that we expected. However we faced many challenges too. For some of us it was the first time that they implement a game so it was a new era for them. On the other hand we have people who coded game before so those situations balanced one another. We all had to do many researches to reach our requirements. Our goal was to create a game which is inspired by the Atari game “Volfied”. The game was designed differently. In our version of game, there is a *hero* which conquers lands by drawing lines. In order the *hero* to win the land which the *castle* lays on should be conquered. The game sounds very easy like that but we decided make it more enjoyable and hard to master. Therefore in the game there are different obstacles that are called *enemies* to make the game as we wished to be. Moreover we decided to have *bonuses* which can be both positive and negative in our game to make the game better. To make the game more enjoyable we decided to make 5 different levels too. We made many promises for our game. Therefore it made our work harder. Our first goal was to create a *hero* which can move and while moving it will draw lines. After managing that we created *enemies* and made a *collision detector* so when the *hero* collides with any of them the game is over. Making those was easier than expected. Then we created the castle however we could not manage the idea of *conquered land* yet. It was a big issue and very hard for us. As lands conquered the area under that land should act like edges and the movement of objects must be limited according to that. Moreover we had to find the objects underneath that area to check if they are either a bonus object or the castle or an enemy. That was the hardest thing we faced during our implementation and it was the main thing that forms our game. Fortunately we managed to that too. Finally we had a game. At the requirement elicitation stage we promised to make *faster, smaller hero, bigger hero bonuses.* However we did not implement them in our code. Other bonuses are implemented but we could not add them to our game because of time constraints. Another thing we could not manage to do is the *level selection by giving a given code*. We do not have that feature in our game. However the player is able to choose which level to play without giving a code in our game. Thus it can be said that this feature is modified. We could not make our hero’s type selectable because of time constraints too. Other than those we managed to do everything that we promised. Unfortunately the smoothness of our game is not perfect as we wanted but we made it as smooth as possible. As we promised we made our game very easy to use and easy to learn. It was an important requirement for us. Furthermore the response time is very quick which was vital for our game. As stated in the design report we had to compromise the memory and functionality to reach our main goals.

We are proud to say that we did not make any major changes for our design to make implementation. We used all the design patterns that are stated in the design report. Those patterns were Façade and singleton design pattern. Moreover we updated our game as promised in the design report. We used timer and at every 5 milliseconds the action performer will be called. Inside it will check all the necessities to keep playing the game. If there is anything wrong with those then it will call repaint() method which will call the paint() method which is overridden in the GraphicEngine class. This method will create all the objects which should be on the screen at the desired positions. Thus game will be updated. That method helped us a lot and after implementing that we saw that we made the right decision. This made the game graphics smoother as we expected. On the other hand we added some classes which are not decided in the design phase. While implementing we realized that we need another classes for our game. For instance we added a Frame class which initializes the frame for our game. We added the changes can be seen by comparing the detailed object diagrams one of which is created while design stage and another after the implementation stage Visual Paradigm by reverse engineering method. That’s why the arrows are different then the first time since it does not show composition or aggregation arrows etc. Since we did reverse engineering it looked very complex since all bonus classes are shown too.



**Figure 1:** Detailed Object Design created in the Design Report



**Figure 2:** Detailed Object Design created by Reverse Engineering