 Bilkent University

Department of Computer Engineering

Object Oriented Software Engineering Project

CS 319 Project: Civilizational Wars

Design Report

Project members:

• Fuad Aghazada \_ 21503691

• Seyfullah Yamanoğlu \_ 21400697

• Bayram Muradov \_ 21503664

• Berk Erzin \_ 21201516

Supervisor: Uğur Doğrusöz

Progress: November 18, 2017

Design Report

CS 319 Project: Civilizational Wars

# Introduction

     This project will be a 2D platform game. There will be some main characters, with which the player should be able to pass different levels of the game by controlling them. Different levels will have distinctive maps, where the player will encounter with some different enemies.

   By writing this report as a group we aimed to convey some general and detailed information about the analysis of this game, specifically, functional, non-functional requirements, and system model diagram.

# Purpose of the system

This game will be an adventure game, in which there will be different maps to play in order to make the game interesting to the players. Game interface will be easy to adapt for the players within all groups of ages. The game controls will not only be easy to use as default, but also will be modifiable in the settings of the game. In terms of the interests of the players, the game will have 3 different difficulty types: easy, medium, hard - which could help the player to test his/her gaming skills in different types of difficulties. Another purpose behind the designing this kind of game is also experience the players, especially the ones with same generation with us, the nostalgic Atari games like Contra (in terms of map and fighting).

* 1. Design Goals

**Efficiency -** Every player would like to play the games, which are quite responsive and do not have so many performance issues, which are irritating. Considering this fact, we will make our game sufficiently efficient in order to be able to obtain the attraction of the players. So as to achieve this goal, we will try to handle our limited memory properly so that it is minimally used, and minimize the execution time of the main operations, which play the most crucial roles on the proper flowing of the game. For example, in order to check the collisions between the different game objects, we need to create a class, in which we will have some functions, control all the computations done efficiently, otherwise, it could lead to some bugs prevent the game flowing properly.

**User-friendliness -** In order not to make the player confused when he/she launches the game, we need to design the User Interface easy to control and use. In terms of the comfort of player, some settings, like activating music, changing difficulty modes, adjusting controls and so on will be available. However, comfort in the controls and the interface does mean the game be extremely easy that the player will be bored while playing the game.

**Adaptability -** So as to be able to run the game in different platforms, it could be a proper decision to implement the project in Java programming language, which is also advantageous in terms of the object-oriented approach. The only thing is need to launch the game in a platform is to install proper JRE - Java Runtime Environment, which is compatible with almost all widespread operating systems. That is why we decided to choose Java among the languages like C++, C# and other languages which can provide us some different advantages in terms of performance and security.

**Modifiability** - The very first release of the game will contain only 3 different levels which is not sufficiently enough for a player to entertain, because in term of the time, it would finish early. So that we could need to add new maps, characters and levels to the game. In order to achieve this, with the advantages of the object-oriented approach, our design and implementation should be able to modify and be added some new features and game states. So, our project should also have modifiability as a design goal.

* 1. Design Tradeoffs

**Functionality - Usability -** Complicated and complex game system is not a desirable specification for the comfort of the users, so we decided to design a plain usability, which requires basic functionalities. This tradeoff creates easiness at the adapting of the player to the game system so that the players can enjoy their time by playing the game.

**Time - Memory efficiency -** In order to create a game system with a high performance, we need take efficiency enhancements into account, which is in terms of time and space. However, we will consider that the memory of the today’s average computer is sufficiently enough to utilize, so we will need to sacrifice the memory usage to perform the operations and functions in a considerably less amount of time. Shortly, we will use a time efficient algorithms so as to cover the downsides of the using extra space in the memory for the game objects.

1. **Software Architecture**
   1. Subsystem Decomposition

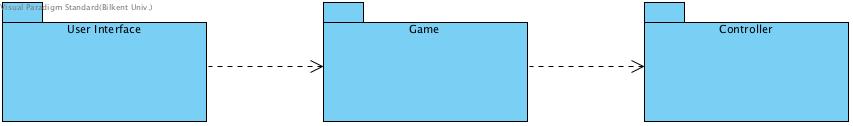


Figure 1. Subsystem decomposition

We made up our mind to use MVC (Model-View-Control) as an architectural style, because it will not be too complex game system. Using this architectural style, we can divide our system to mainly 3 subsystems as described in Figure 1: GUI, in which there will be rendering system for the game states (menu, pause, settings and etc.), Game, will contain the game logic and updates of their properties, Controller, will handle the user input, data management for saving and loading. More details for the subsystem are described in Figure 2 below.

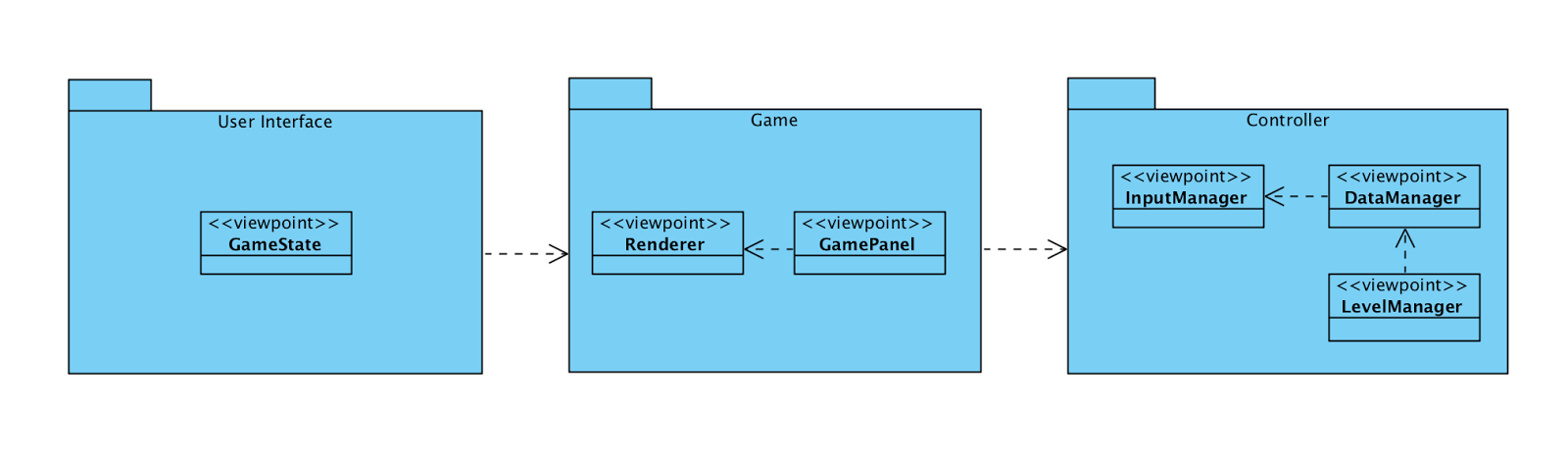


Figure 2. Some details for the subsystems

* 1. Hardware/Software Mapping

As hardware requirements, our game will need a keyboard and mouse in order to create an interaction between the user and the game. The player will use 3-4 keys from the keyboard to move and fight, and mouse for aiming by some weapons. In order to run the game, the machine in which the game is launched and played should not have extremely high requirements rather than nowadays’ standards.

* 1. Persistent Data Management

Since our game will have features for saving and loading the game, we will need to save the current time data for game. Data for the game will not be highly large and will not be concurrently used by different users - the game will be installed to each machine with the necessary files in order to properly run the program. That is the reason why, we decided to use a simple file system so that it will save the data for the game objects, maps and the levels in proper formats to some files, so we can access saved data by loading the game from those files.

* 1. Access Control & Security

Our game will not have a feature of login authentication, because it will be a single player game, and as mentioned above, it will use a simple file system to store the necessary data for the game. Nevertheless, we will need proper access modifiers for the data of the game objects so that it will not affect the security of data flow of the program. Additionally, we need to protect the file, where the saved game data is stored. In order not to be modified manually by the players, we will need to encrypt the game data in a way that the data is not obviously readable by the users.

Object data serialization would also be another solution for this security problem in a way that, when the game is saved, the necessary data, which will be saved, could be represented as sequence of bytes, which contains every information about the object - object type, data type and etc. After writing the serialized data into the file, when the game is loaded it will get the data by deserializing it. In Java Virtual Machine, the data can be serialized in a platform, and can be deserialized in an another platform, which again makes the Java programming language much more adaptable.

* 1. Boundary Conditions

When the game is launched, necessary data will be initialized and be waiting the user input. When the gameplay starts, if the game is loaded, it will initialize the game object with the obtained data coming from the file. However, if there is some format problems in the file, which have been created by the modifications of the players, the program will not allow the user to load the game by throwing an exception. When all the levels have been passed, the player will be congratulated and brought to the main menu to create a chance for trying the levels in different difficulty modes.

**3 .Subsystem services**

**4. Low-level Design**

4.1. Object Design trade-offs

4.2. Final Object Design

4.3. Packages

4.4. Class Interfaces

**5. Glossary & References**