CS 319 - Object-Oriented Software Engineering

Design Report

Colony Wars

Supervisor: Bora Güngören

İbrahim Taha Aksu

Ahmet Emre Nas

İzel Gürbüz

Kaan Çakmak

Table of Contents

1. Introduction

1.1 Purpose of the system

Colony Wars is a time based strategy game that requires usage of a considerable amount of strategical thinking. The game aims to provide the users with a leisure time activity that is both challenging and fun to spend time on. The user interface of the game is quite simple which enables even the user who is not familiar with the common flaw of strategy games to get use to it easily. Also a tutorial option is provided to the user where the basic mechanics and the main purpose of the game is explained. Different playing modes such as campaign and skirmish are added to gameplay to keep the player interested and busy.

1.2 Design goals

1.2.1 Ease of Learn

Our aim for this game is easy to learn of game mechanics with the game tutorials. Users don't have to know playing of game. Aim of tutorials, to teach the game mechanics and rules to user easily. Therefore, user can understand game mechanics and aim rapidly.

1.2.2 Usability

Aiming a user scope that is so broadly open from children to adults our program should have a user interface that is clearly understandable regardless of the users experience with any software applications or games. Besides having a simple interaction for user the tutorial/help part should be explaining basics and usage of game's features neat enough so that nearly no one would not have any problem using it.

1.2.3 Reliability

Provided with any input regardless of valid or not the game should keep working without any crash and should be well prepared for such cases so that it gives the best service possible to the user.

1.2.4 Performance

The game is going to be refreshing frame and doing calculations constantly since it's a real time dynamic strategy game. Thus it should have a fast enough response time and should be stable in spite of the length of the game which comes with more workload by time passing.

1.2.5 Extendibility

Regarding the limited time we as developers have we are not including some of the functions although we are eager to have them. There are also many possible game future such as multiplayer playing or new arcade game modes. Thus the game should be extendible further in the future.

1.2.6 Memory Usage

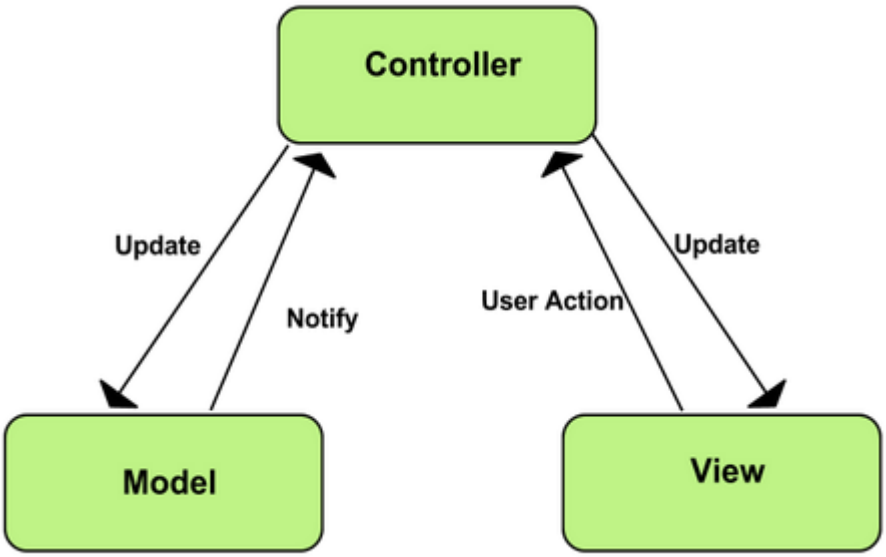
According to game mechanics, there are a few maps, basic game animations and collection of integer game datas. User also can save his/her proses. System saves proses by using map name and integer values of his/her last game. When user wants to save the proses again, old saved data will be ereased and new game data will be saved. According to these proporties, memory usage of the game is not high.

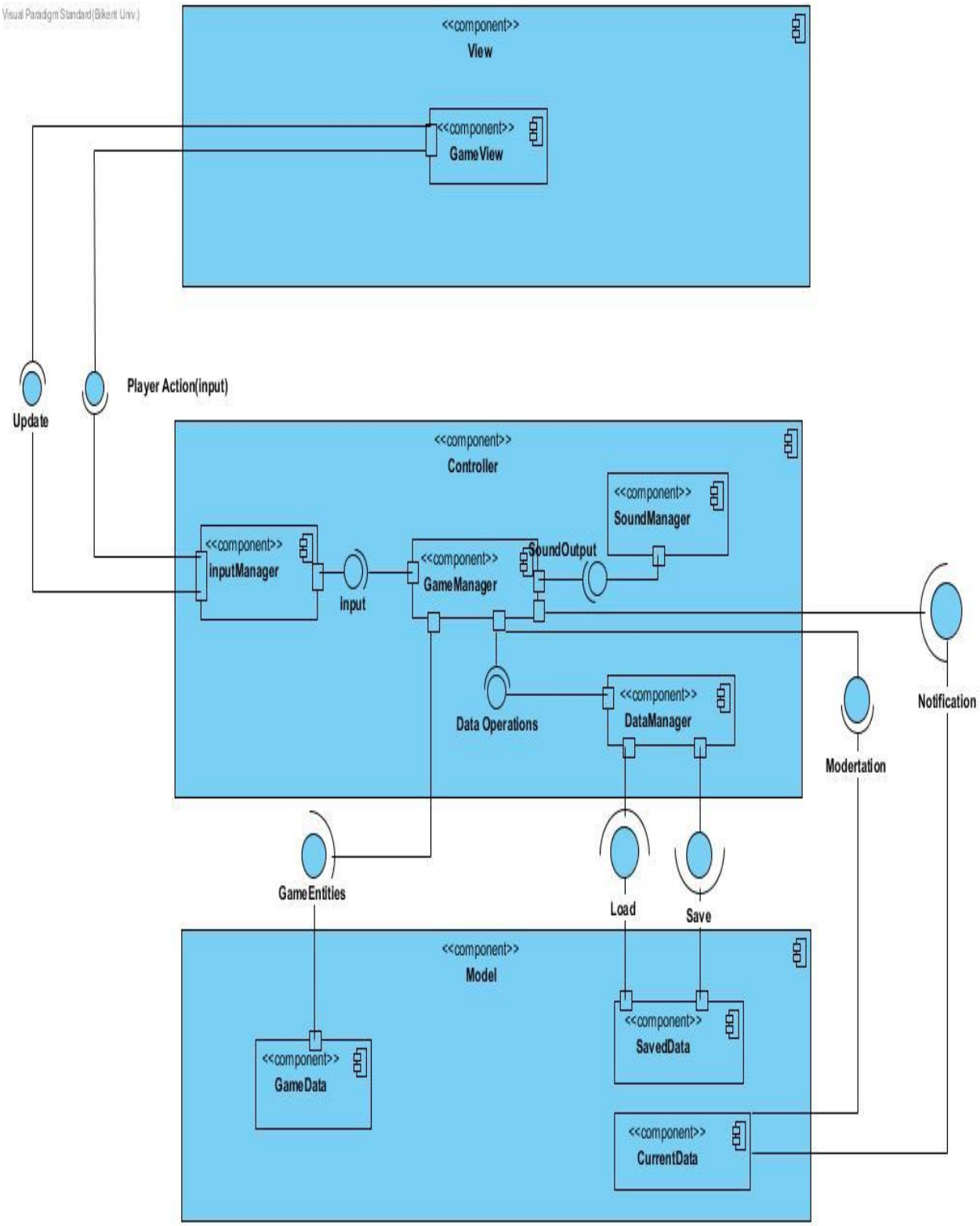
2.Software Architecture

2.1 Subsystem decomposition

Aiming to build a strategy game in which visualizations take a considerable part in the design as much as computations do we decided that the best architectural style to use for modeling the subsystems is MVC. Where we can handle all user interface related issues in the view subsystem while doing the computations that needs to be done behind the scenes in

the controller subsystem. Controller subsystem is going to handle the user input make the necessary computations and transmit the new data to model subsystem. Controller subsystem would also notify view subsystem which will update the view accordingly.





2.2 Hardware/software mapping

The game will be developed using Java so Java Runtime Environment is needed to execute the game. In terms of hardware requirements, our game is played with a mouse.The game requires a standard computer that could rely on today's technology. Also the game will use swing awt library for graphics.

2.3 Persistent data management

The data will be stored in the user hard disk and the game doesn't require a database . The data will be accessible at real time so, documents of the game will be stored in the memory to be accessible when the game engine works. Other components of the game such as images are stored as to allow user to make changes to create the game according to personal choices.

2.4 Access control and security

All users who play at the same computer, will have a uniqe id and non-uniqe password. These id and password will be registered at the beginning of the game. Therefore, every player will use only their account. When user entered the his/her account, user can only load his/her saved datas which he did previous game. By this future, users can only access to their game datas and according to these, game security and user data protection will be provided.

2.5 Boundary conditions

2.5.1 Initialization

System will be an executable jar file. It will not have .exe extention. That's why, there is not need to install the game.

2.5.2 Termination

User can use the “x” button which is located at the top-right of the frame or user can can go to main menu by using the pause menu and then click to quit button for termination.

2.5.2 Starting

User can load his/her last game by using the load button from the menu or can start a new game. User can only load his/her datas.

3.Subsystem services

3.1 View Subsystem

The classes included in this subsystem are basically to handle the graphical user interface of the program.

3.1.1 Game View

This subsystem is to handle the user interface of the software system. It basically includes all the classes related to the topic. GUI classes thats aim is to visualize main menu and other sub menus the main game screen are builded in this subsystem. This subsystem is also responsible for realizing the user input via interactivities between the user and the interface such as a button click or a mouse drag. Sending those realizations to the controller component it eventually gets updates from it and updates the game view accordingly.

3.2Controller Subsystem

This subsystem is for realising the interactaion between the user and the interface. It realizes the input provided by view manager and computes/changes game entities accordingly. Computing new entities the controller subsystem stores new data to model subsystem . The system also handles the sound that is played during the game and also handles the issues related to saving and loading the game.

3.2.1 Input Manager

Input manager is responsible for realising the interactaion of the user with user interface. Provided with the action of user from the view model the input manager transmits this information to the game manager.

3.2.2 Game Manager

There are various duties that are handled by the game manager. Its basic purpose is to manage the main game logic.

It is responsible for calling the operations that correspond to saving and loading the game, managing the sound functionality of the game via the sound manager. It also updates and realizes the current data by its relationship with the model subsystem through data manager subclass.

3.2.3 Sound Manager

This subclass is responsible for handling the sound output from the system. It provides the gama manager with the corresponding output which eventually play the sound.

3.2.4 Data Manager

Data manager is the subclass which game manager calls upon a request of saving or loading a game that is given by the user. It updates the saved data component if the user asks for saving the game and realizes the saved game if the user requests to load a game.

3.3 Model Subsystem

This Subsystem is the collection of subclasses which are responsible for managing the data related to game entities which are stable information, the data related to the current game which the controller queries to build the current game and provide its continuity and data related to saved games for the user to be able to continue an early saved game by loading it back.

3.3.1 Game Data

This subclass is to hold all the information related to the game that is: Game entities. It holds all the information from maps to constants to be used in the game. When game is built the game manager uses this subclass to access the game entity values.

3.3.2 Saved Data

This subclass is to store the values that are related to a specific game session such as map, base and army informations. In case of load request it communicates with the data manager from where data eventually reaches the game manager leading to the rebuilding of the game.

3.3.3 Current Data

Current data subclass is where all the information related to current game session is held at. Game manager communicates with this very subclass to access to entities belonging to the particular game session that is currently going on. Game manager updates any kind of change to the game entities again in to this subclass.