Bilkent University

Department of Computer Engineering

Object Oriented Software Engineering

CS-319: Kill The Bugs

Design Report

Anisa Llaveshi, Cüneyt Erem, Ertunç Efe, Mert Gürcan

Course Instructor: Uğur Doğrusöz

Progress / System Design
March 27, 2016

Table of Contents

1.) Introduction....2
1.1 Purpose of the System....2
1.2 Design Goals....2
End User Criteria....2
Performance Criteria....2
Maintenance Criteria....3
2.) Software Architecture....3
....4
2.1 Subsystem Decomposition....4
2.2 Harware / Software Mapping....5
2.3 Persistent Data Management....5
2.4 Access Control and Security....5
2.5 Boundary Conditions....5
3.) Subsystem Services....6
Services....8
4.) References....10

1.) Introduction

1.1 Purpose of the System

The purpose of this system is to provide an entertaining game for people who enjoy playing games for all ages. The game consists in a set of bugs and a number of different food products initially placed in opposing sides of a field. The bugs aim to eat the food and the player's aim is to not let the bugs reach them. The player can buy different weapons by collecting coins and place them in the field to kill the bugs. The game ends when a bug reaches one food product. When the game end, if the score that player earn is in the top ten highest score, the score will be recorded locally in order to show in high scores page. In addition to the basic gameplay the user will be offered with additional features like changing the game theme, changing game volume, and changing the background music volume.

1.2 Design Goals

End User Criteria

Ease of Learning

The game will be significantly easy to be learned to play. There will be a help section which can be reached from the main menu which will guide the user through the basics of the gameplay and of the features that it provides. In addition, the object images will be self-explanatory and intuitive for beginner players to understand their functionality.

Ease of Use

Any person will be able to play the game regardless of previous experience in gaming. The game will not be very difficult to play. A reasonable number of features will be available for the player to use so that the game does not become very complex in terms of difficulty to remember everything and it still remains manageable for the player to play and win high scores.

User Friendliness

The game is designed to be user-friendly.The game does not require any setup, it will be provided as a .jar file which can be executable within the java runtime environment (JRE) [1]. In that way, game can be played among different operating systems.Moreover, its main menu design and the game screen is easy to understand and easy to use which are also important aspects of user friendliness criteria.

Performance Criteria

Run–Time Efficiency

Another important issue is the runtime efficiency of the game, player will not be affected negatively from the gameplay screen while s/he is playing the game. The functions that are provided to the player in the gameplay screen will perform reasonable time, such as buy weapon buttons, placing the weapons to the field etc

Optimized Memory Usage

Memory usage is one of important performance criteria which depends on speed also. In this project, there are lots of animation tools which are bugs, weapons, bullets, but also their explosion animations and updates all images and scores. Therefore, the game cannot have less memory usage and need to have enough space to improve performance and speed, java virtual machine (JVM) will delete destroyed objects such as bullets, bugs etc [2].

Maintenance Criteria

Readability

Readability is an important thing that engineers always should write their code in readable because when other engineers look at the codes, they should understand the flow of methods and operations in codes fast. This provides maintenance of the code, time, efficiency. In this project, this design will based on maximum readability first, after optimizing maximum performance criterias, maximum performance and readability will be combined and have maximum readability and performance.

Modifiability

Modifiability is an important criteria for mordern systems to keep cost of the system at a reasonable level and also to gain time when updating the system.In Kill the Bugs game, MVC (Model-View-Controller) pattern will be used to maintain modifiability [3]. In that way, the game's graphical objects, algorithm and specialities of the objects can be updateable separately. Moreover, in the testing phase, fixing the problems will not be take so much time, since the causes can be easily find. For instance, if the problem will occur about the control of the game, it will be searched into the control objects and corrected as soon as possible.

Good Documentation

Good documentation is one of the key features of maintenance criteria. Comments will be written in order to make the coder more familiar to project. Also methods and variable names will be self describing. And explanation will be made how code fits into the system process. So, code should be proper for the documents and its purposes. The code may be more elegant than clever to keep it simple and understandable.

2.) Software Architecture

Model-View-Controller (MVC) architecture has been chosen for the design of the game, because it is time-efficient and it fits the game's structure [4]. In figure 1, the swimming lanes show the corresponding parts of the MVC pattern. Inside the lanes it can be seen that several components have designed for the sake of understandability and ease of implementation.

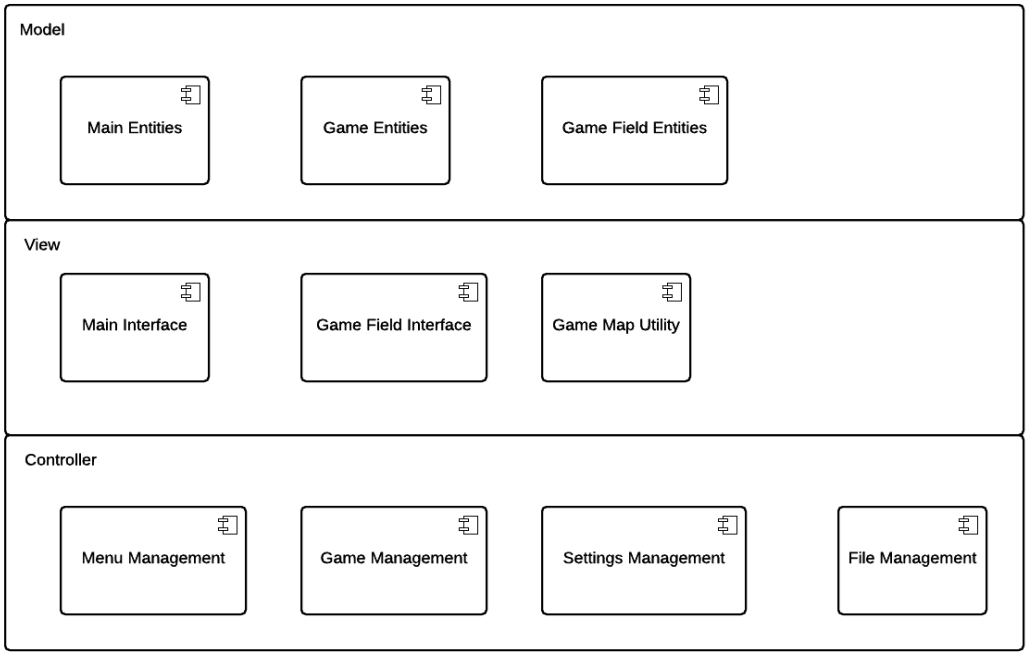

Figure 1 : Subsystem Components with Swimming Lanes for MVC pattern [5]

2.1 Subsystem Decomposition

Main Entities component is composed of the models which keep the data for the background music and similar game features needed during the whole gameplay. Such model classes are Sound, Theme, Button and User class.

Game Entities component has the game objects' models that keep game tool data which are not relevant to the actual game field but which the user needs for the gameplay. Such game tools classes are Score class, MoneyAccount, BuyButton class etc.

Game Field Entities component has the actual game field objects' models that keep objects' data which the user uses during game. Some of them are Weapon class, Bug class, Bullet class, Grenade class etc.

Main Interface component has the main view objects of the program. It displays the menu and its submenu options to the user like CreditsView, SettingsView, HelpView, etc.

Game Field Interface component has the actual game screen view objects of the program which consists of WeaponView, BugView, CoinView, GroceryView, ScoreTable, MoneyAccountTable, BulletView, and BitsNPiecesView. It displays the game play screen to the user.

Game Map Utility component has the view objects that actually need to be inherited from the Game Field Interface. It provides the methods that are needed from the game field objects to access their coordinates and to mutate their coordinates into the game field.

Settings Management component has the control classes which controls the sound and theme by using the provided services from the Model and View components.

Menu Management component has the control objects which controls the all main menu selection and also other menu selections such as, newGameButton action listener, creditsButton action

listener, backButton action listener etc.

File Management component has the control objects of the data management that provides taking data from the relevant entity object to be held into .txt file locally which are user name, high score, and game duration.

Game management component has the control classes for managing the game playing stages like, GameManager class.

These components' services will be explained in more detail at section 3.Subsystem Services part.

2.2 Harware / Software Mapping

Our game will be a Desktop based game and a user will only need a computer in order to be able to play; no internet connection is needed. We are going to develop this software in Java Programming language thus, a user must have Java Runtime Environment (JRE) installed in his/her PC [6]. In addition, a mouse is needed in order to be able to interact with the game. A keyboard is optional as it will have shortcuts to pause the game and mute the menu.

2.3 Persistent Data Management

Kill the bugs game does not consist a high-level database structure. It consists of simple text file which is used for high score list. Text file is kept inside the kill the bugs game’s jar file. Also game sounds and game images are stored inside the jar file.

2.4 Access Control and Security

There will be no restrictions on control access of the game. Anybody who possesses the executable file of the game can have access to it. In addition we do not manage a user profile system. As a result, there will be no security provisions.

2.5 Boundary Conditions

The system will be initialized when a player will open the executable file (.jar). The system shuts down when the user will click the quit button in the game, the exit button of the window or in case the games shut down abruptly due to errors. A possible error that might occur is corruption of the data file holding the high scores of the users who have played in that machine. In such a case, the saved data will be lost and a new file holding the highest scores from that moment will be created. Another possible error is an unexpected brute shut down of the game while a player is playing. In such a case, the score of the current player will not be saved or evaluated to be put in the high score table.

3.) Subsystem Services

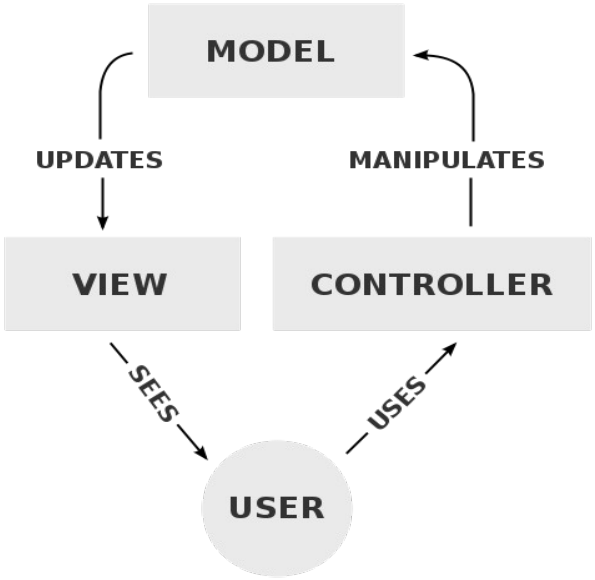

Figure 2 : MVC Pattern Usage [7]

This project will follow the MVC pattern [8].

There will be four control components which make possible the settings, menu, file and game management which are Settings Management, Menu management, File management and game management components respectively.

The view components are the Game Field Interface, Main Interface and the Game Map Utility. The Game Field Interface will provide the graphical interface for all the gaming tools and Main Interface for the main menu. Game map utility component is also a view component but this component will be inherited by the game field interface subsystem.

Model system keeps the data of the objects which are main, game and gamefield entities to calculate and update logic of the game. Model, view and controller systems are connected to each other separately to regulate the system but these connections will not affect MVC pattern negatively, all connections have been made appropriately according to the MVC pattern which can be seen in figure 2.

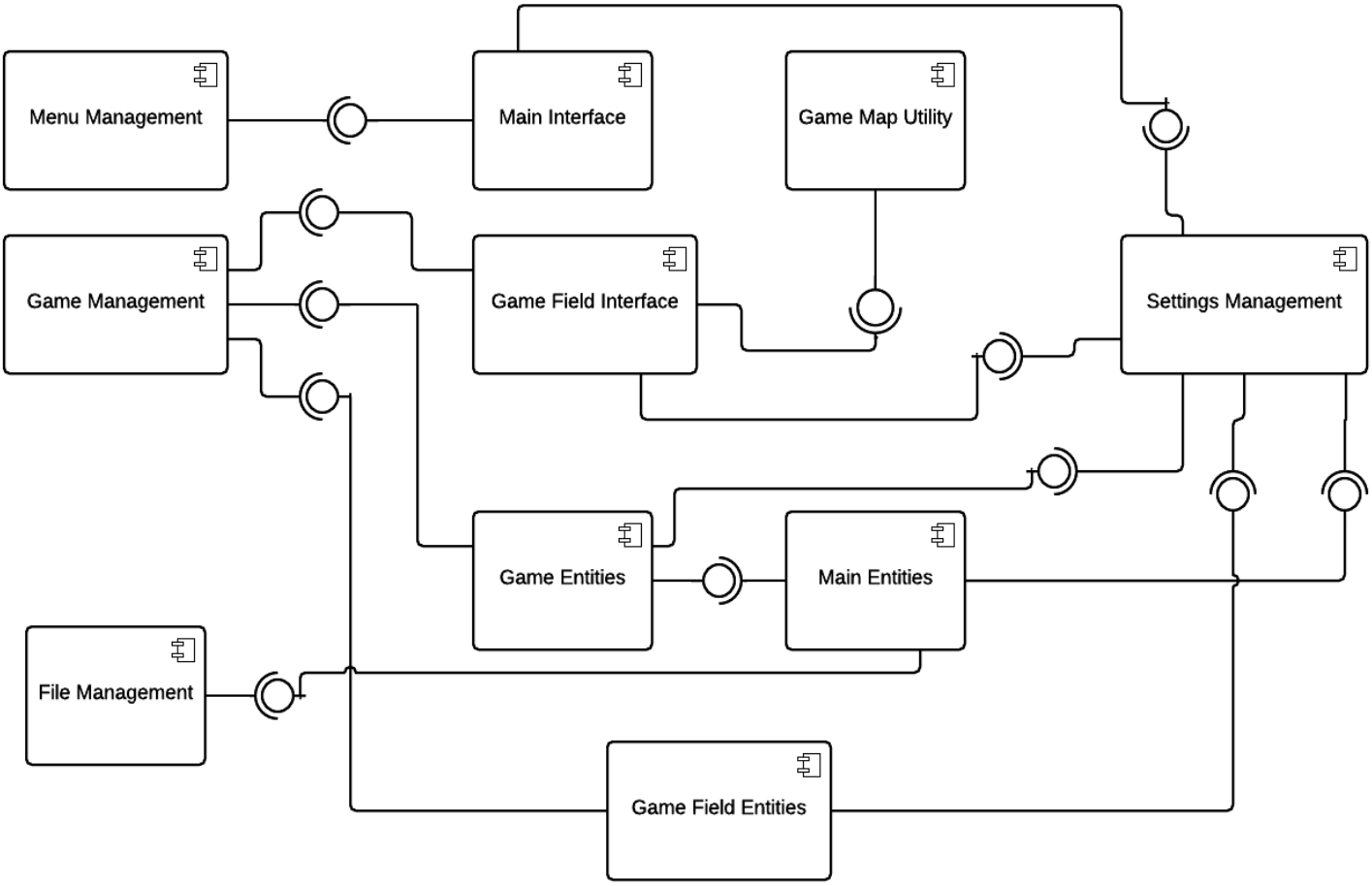

Figure 3: Component Diagram of the Project [9]

Settings Management subsystem takes input from Main Interface and Game Field Interface, which are subsystems of the view component and holds the data to the relevant Entity objects by using the provided services. Moreover, this subsystem has the ability to change the sound of objects, to adjust the selected theme by taking the relevant data from Main Entities, Game Field Entities, Game Entities subsystems and update the Interface subsystems by using their services.

Menu Management subsystem controls the Main interface by using the services that are provided from Main Interface subsystem like, changing menus using buttons.

File Management subsystem interacts Main Entity subsystem which have game entities' data to getting username, score and duration by using Main Entity subsystem services.

Game Management subsystem interact with Game field entities by manipulating the data of objects of game like, bugs, weapons, etc. Also, it keeps and updates data of Game Entities subsystem which are Score class and Money class by using the provided services from these subsystems. Moreover, it controls the display of Game Field Interface by using the services of Game Field Interface,

some of Game Field Interface services's base operations will be taken from Game Map Utility subsystem's services via inheritance.

In the explanation part of services, management components' sockets (figure 3.2) and the provided services by the other components will be explained under the headers of services separately to maintain the simplicity.

Services

Change Background Sound is a service provided by the Game Entities component through methods to set and/or update the background music volume of the game.

Change Object Sound is a service provided by the GameField Entities component through methods to set and/or update the music and the music volume of different game objects.

Change Theme is a service provided by the Main Entities Component through methods to set and/or update the theme of the game. It is the Settings Management component which uses these services to modify the data of the entities accordingly and makes possible the update of the respective interfaces.

Change Highest Scores service is provided by the Main Entities component through methods to set and/or update the score of a user, to check whether a user's score is among the top 10 highest scores and to update the list of the users with the highest scores. The File Management component uses this service to update the files keeping the high score accordingly.

Change Score service is provided by the Game Entities component with methods to set/update the score. These methods are then used by the Game Management component which updates the score accordingly and also by the Main Entities which updates the score of a certain user.

Update Money Account service is provided similarly as the Change Score service by the Game Entities component. This service is then used by the Game Management Component.

Buy gun service is a service provided by the Game Management component by a method which taking as an input the gun information, checks if the money account has enough available money to be able to buy the gun. In addition, other methods to provide this service is a method to make the certain gun required available and another to update the money account accordingly after the gun has been bought. Game Management then also, updates the view component of the game field accordingly.

Collect coin is a service provided by the Game Management component through methods to update the money account according to the value of the coin taken as an input, and to update the game field view to make the coin disappear.

Check Collision is a service provided by Game Management component through methods to check if two objects have collided with each other. This method takes as an input the location of two objects and decides whether their positions overlap.

Pause/Continue Game service is provided by Game Management component through methods to freeze and unfreeze the current status of the game. Also Game Management updates the

game field interface accordingly.

End Game is a service provided by Game Management component through a method to check if any of the conditions which ends the game are fulfilled and methods to load a new game.

4.) References

[1], [2], [6] Explanation of JRE and JVM, Oracle [Online]. Avaliable: https://docs.oracle.com/cd/E19455-01/806-3461/6jck06gqb/index.html, Accessed on: March 27,2016.

[3], [4], [8] Explanation of MVC Pattern, Wikipedia.[Online]. Avaliable: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller, Accessed on: March 27, 2016.

[5], [9] UML Tool. [Online]. Avaliable: https://www.lucidchart.com/, Accessed on: March 24,2016.

[7] MVC Pattern Diagram. [Online]. Avaliable: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller#/media/File:MVC-Process.svg, Accessed on: March 25, 2016.