Bilkent University

Department of Computer Engineering

CS 319
Object Oriented Software Engineering

Design Report

Space Invaders

Group10 Members:

Ayşe Öykü Özer

Turgay Arda Usman

Yonca Yunatcı

Table of Contents

1. Introduction

1.1 Purpose of the System

Space Invaders is a 2D game which is designed to entertain the user and provide a better experience compared to the original game. In order to achieve this aim, we will provide a user-friendly interface with no significant hesitations during animation. We hope that splitting the game into 2 levels and adding new power-ups will attract more user. Therefore, our purpose is to make an enjoyable and easily understandable game for everyone.

1.2 Design Goals

Goals of the system are providing a robust, user-friendly, portable and fast software through object oriented analysis and design.

User-friendly

The system is a game which should be entertaining for the player. In order to entertain the player the system should be user-friendly. In this respect, system will provide player friendly interfaces for menus. Furthermore, the logic and the control of the game should be also very simple that user can easily understand intuitively or by reading the help document.

Response Time

For all games response time is really important in order not to lose player's interest. Our system will respond player's actions less than half a second which makes player focus on to the game only.

Robustness

The game should easily handle the problems so that it won't crush on exceptional situations, also it should be stable.

Portability

The game should be executable on several different platforms so that various platforms users (players) can play the game. In order to provide this, Java programming language is used.

Possible Trade-offs:

Robustness vs. Cost

Providing a robust system need longer design and testing period. Therefore, robustness may increase the cost of the software.

User Friendliness vs. Cost

As for providing more user-friendly system, analysis period may get longer. Cost of the software increases as the system gets more user friendly.

Portability vs. Efficiency

Having a portable system needs to use Java programming language which is platform independent. On the other hand, platform dependent programming language satisfies better run-time performance than independent systems.

2. Software Architecture

2.1 Subsystem Decomposition

We decided that three-tier architecture is the best fit for our project since it does not require any web based servers and it is sufficient to have 3 layers which are: Interface layer, application layer and data layer. The high level design is shown in figure 1:

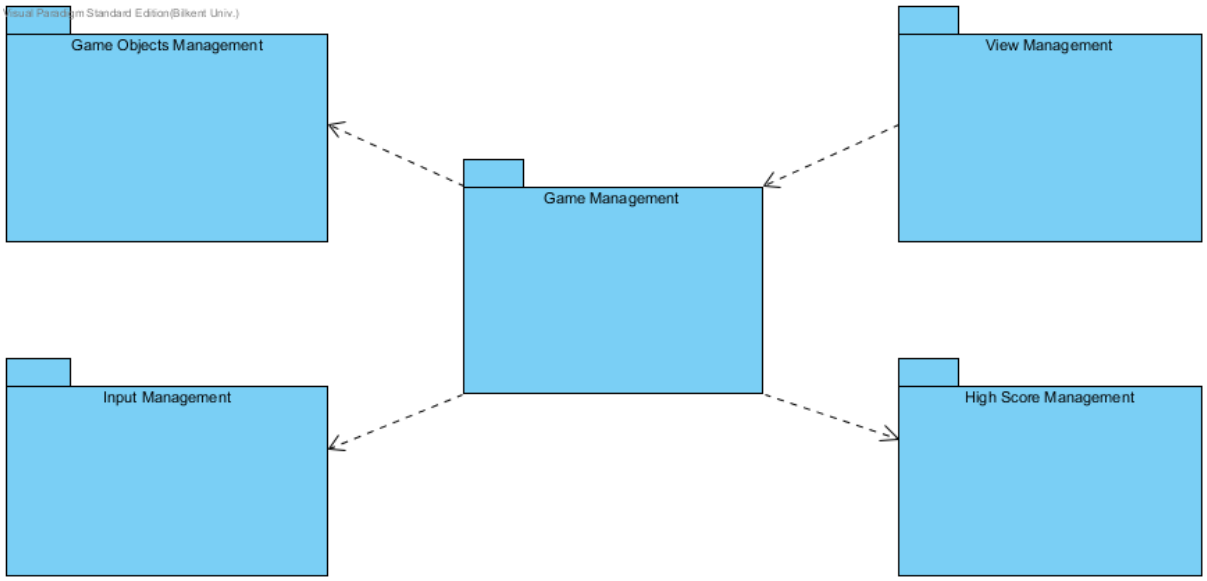

Figure 1: High Level Subsystem Decomposition

All the menu objects and play screen class are interfaces; therefore, they belong to "view management" package which presents the first layer of the system. From these interfaces, the system decides which operations to make. These management classes (MainManagement, PowerUpManagement, SoundManager and CollisionManager) form the "Game Management" package. Interface interacts directly with MainManagement class; thus, MainManagement is the façade class of "Game Management". Moreover, MainManagement interacts with InputManager in order to be able to sustain the game. These propagations are shown in the figure 2:

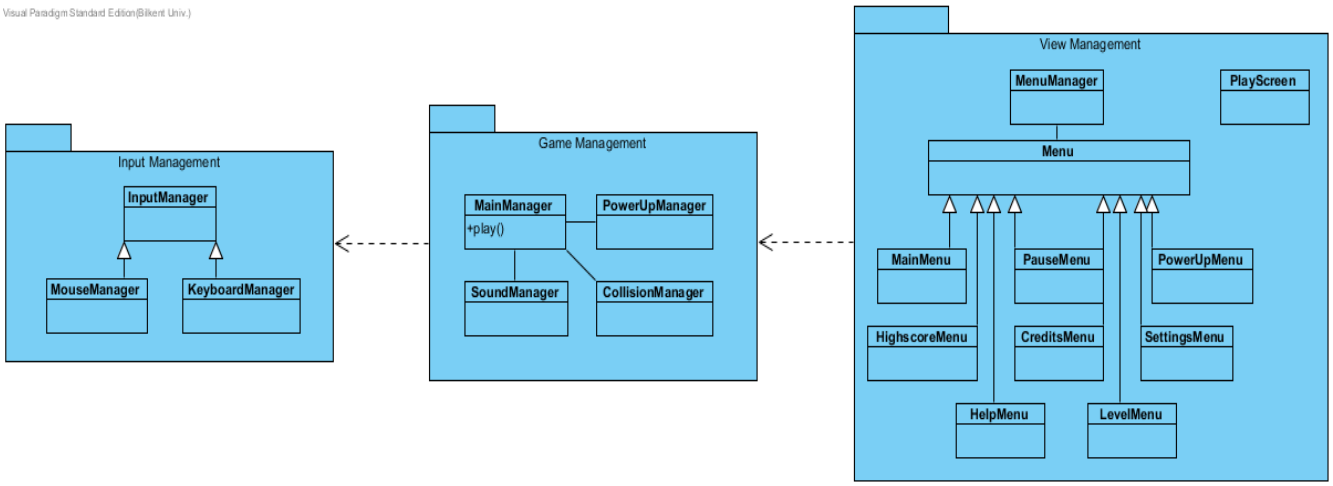

Figure 2: The Propagation of Layer 2 by Layer 1

The last layer consists of the data of game objects and high score data. The spaceship, bomb, alien and beam classes are in the package of "Game Objects Management" which is controlled by the "Game Management" package. High scores are also kept in a text file which is modified by the HighscoreManager class in the "High Score Management" package. The complete system decomposition with the last layer is demonstrated in Figure 3:

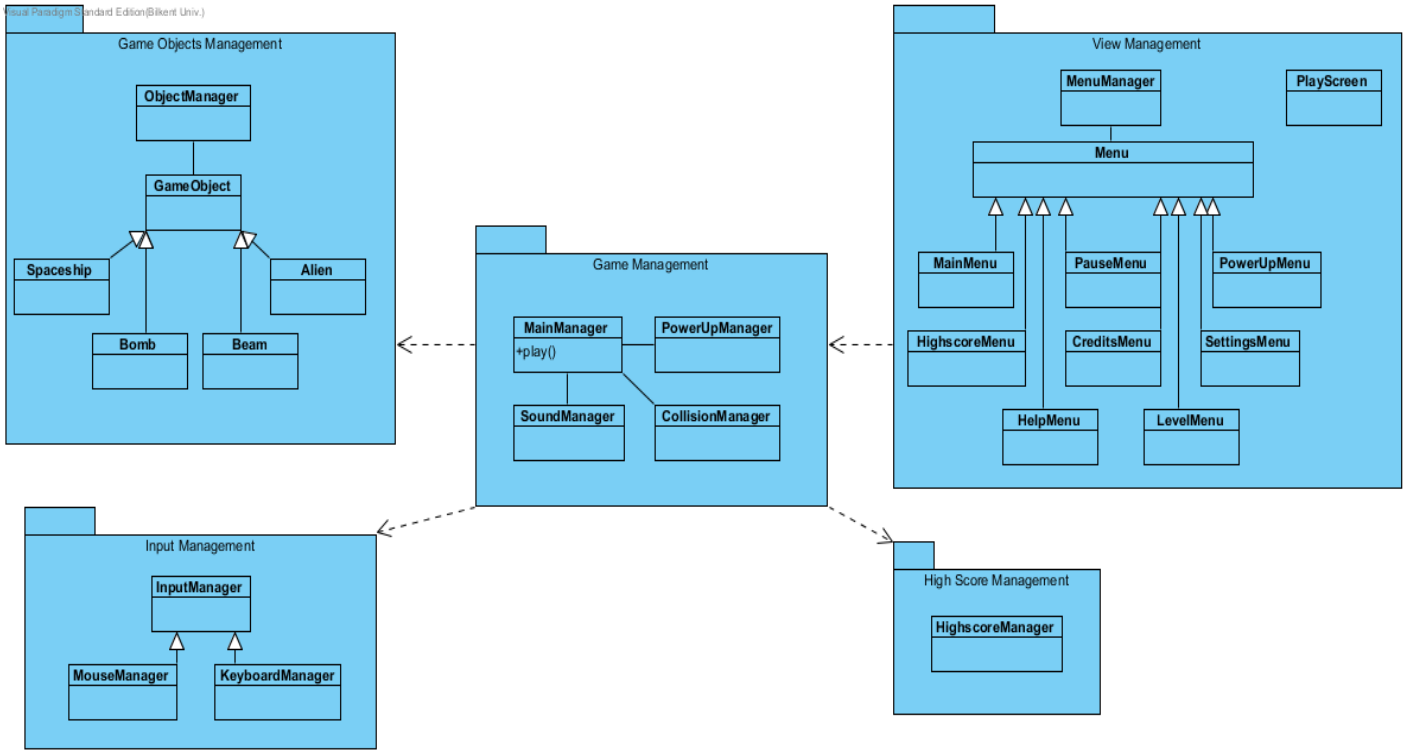

Figure 3: Data Layer Added to Layer 2

2.2 Hardware/software mapping

For hardware configuration, a keyboard for moving spaceship left to right, shooting aliens and typing names on high score list is needed. It also needs mouse for users to give input to the system.

For software configuration, a basic computer which has basic software like operating system and a java compiler to run the .java file will be enough. Because the game will be implemented in java and system requirements are minimized.

The game will be played offline and it won't support online multiplayer or local multiplayer. Therefore, there is no need for a remote server. The game uses Java 2D graphics so a low end PC with Java Runtime Environment will be enough to run the game.

2.3 Persistent data management

'Space Invaders' does not require a database system to hold game and user data; therefore, all the required information will be loaded to memory to access them in real-time. Files like background images, game elements and other kind of graphical user interfaces will be stored in the '.jar' file.

2.4 Access control and security

The game won’t require any network connections nor any authentication system. Game could be played by simply running the '.jar' file. Therefore, security is not a problem for those issues. However, for the security of game logic, some classes include private variables and also we have constant variables to ensure that user cannot change those values intentionally or unintentionally.

2.5 Boundary Conditions

'Space Invaders' does not require a setup; game can be opened with executable '.jar' file.

There is infinite number of levels in the game, as each level is more difficult because the enemies namely aliens move faster in each passing level, it is not possible to win every round. In case of losing life without making a high score, game will return to the main menu. When high score is made, high scores will be updated with the user name provided at the start of the game.

'Space Invaders' could be terminated from the main menu by clicking the 'X' on the upper right side of the screen. While in-game, user should first

pause the game and return to main menu by clicking 'Main Menu' button and terminated the game like the first case. In case of 'Space Invaders' opened second time, first game will be terminated and user could lose all the current data.

3. Subsystem Services

View Management Subsystem

View Management subsystem has eight main classes that provides an interface between the user and the system. MenuManager controls all the menu objects and determines which one is going to be displayed.

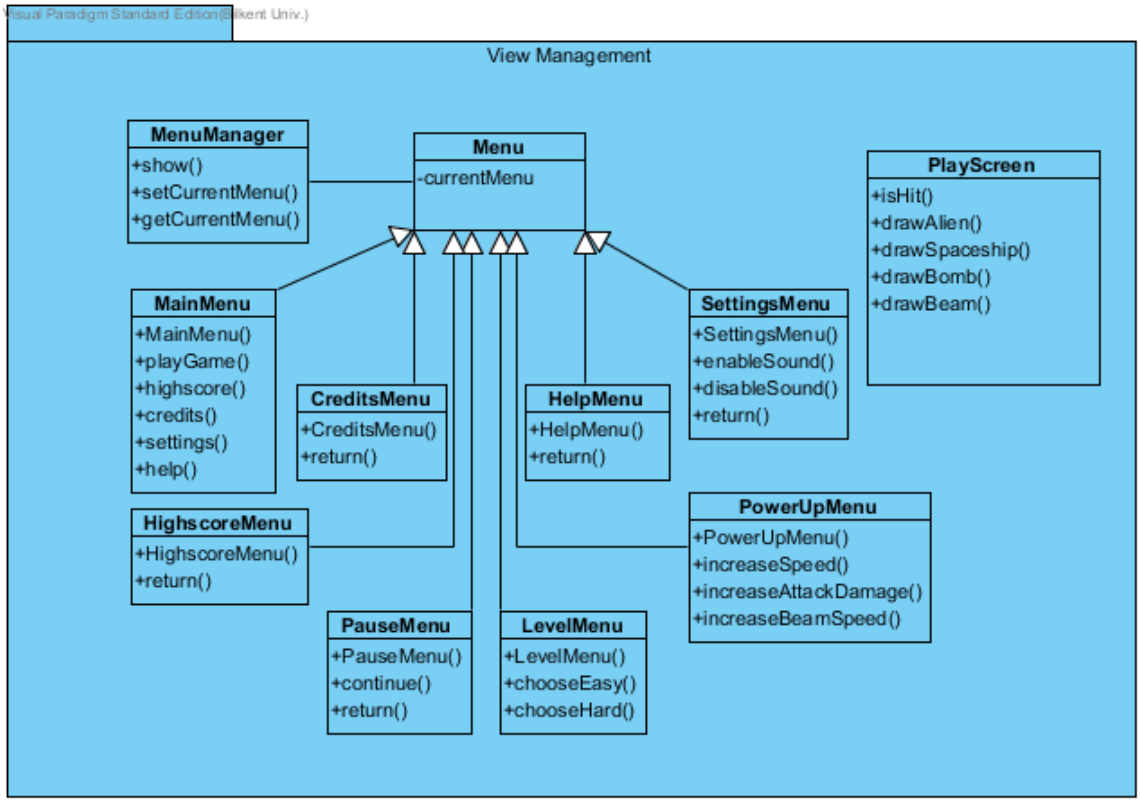

Figure 4: View Management Subsystem

PlayScreen class: provides the window in which the game will be played.

MainMenu class: enables user to select whichever operation they want among these options: Play game, view high scores, show credits, show help and change settings.

LevelMenu: at the very beginning of the game, this menu will appear in order to choose the easy or hard game. According to the choose of level, game will be loaded.

PowerUpMenu: provides upgrade among three power-ups namely speed, beam speed and attack damage. User can choose not to purchase an upgrade go to the next level by clicking continue button.

PauseMenu: enables user to return the main menu or disable/enable sound or continue from where the user left.

SettingsMenu: user can enable or disable the sound from this menu.

HighscoreMenu: shows the list of the high scores with names.

CreditsMenu: displays the contributers.

Game Management Subsystem

This subsystem contains the main controllers of the system. Game logic will be implemented in those classes. Upgrades of power-ups, managing collisions and managing sound will be the main responsibilities of this subsystem.

MainManager: is responsible for controlling all other objects. It gives order to specific manager classes to make sure that the system runs correctly. It directly interacts with user interface; therefore, it is the façade class of this subsystem.

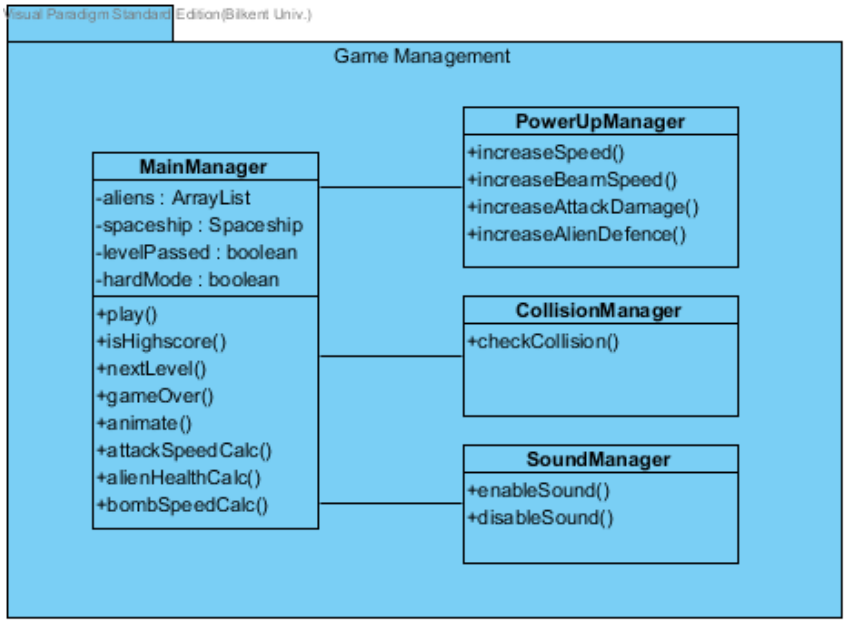

Figure 5: Game Management Subsystem

Input Management Subsystem

InputManager process all the inputs coming from both keyboard and mouse to make the correct moves on the screen.

KeyboardManager: takes the inputs from the keyboard and this allows player to move his/her spaceship horizontally in the game.

MouseManager: takes inputs from the mouse, this allows player to make selections through the menu and select its power-ups.

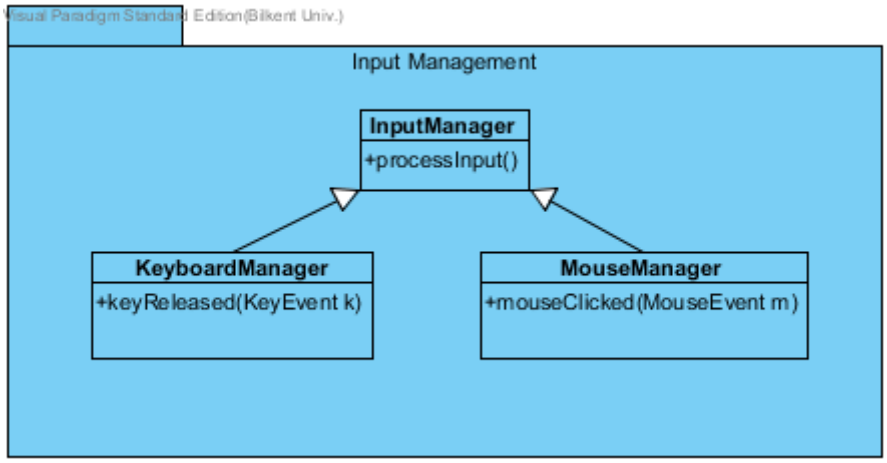

Figure 6: Input Management Subsystem

High Score Management Subsystem

This subsystem contains only one class which is HighscoreManager. This class contains the data of both current score and top 10 scores in an array. At the end of the game, if the score is greater than one of the scores in the high scores array, the array will be updated with the current score and with a name. After typing name, highscoreMenu will be displayed according to the new high score entry.

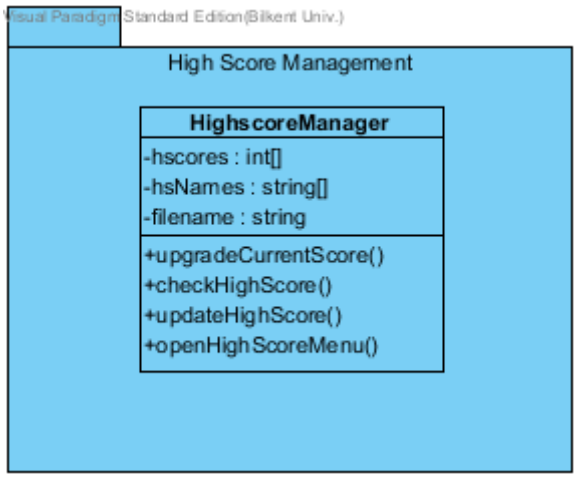

Figure 7: High Score Management Subsystem

Game Objects Management Subsystem

This subsystem is responsible for providing the main objects shown in the screen. ObjectManager updates the game objects according to the order from the MainManager. All those objects have in common attributes such as speedX and speedY.

Spaceship has the ability to throw beams in order to destroy alien objects. The attribute attackDamage is updated according to the purchase of the power-ups. Alien object can throw bombs downwards to the spaceship. However, it can get hit by the beam. In this case, the alien object will no longer be displayed in the screen. Beam object is created by the spaceship object and bomb object is created by an alien object.

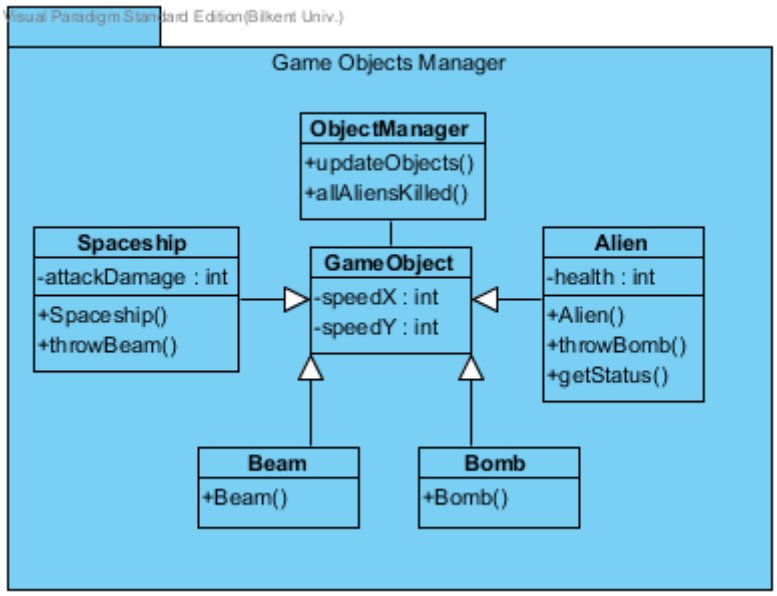

Figure 8: Game Objects Management Subsystem