Bilkent University

Department of Computer Engineering

Object Oriented Software Engineering Project

Alien Invasion

Design Report

Berkcan Gürel

Günçe Kalyoncu

Cemal Kılınç

Levent Köksal

Contents

1. Introduction

As group 1, we decided to design and implement a game called “Alien Invasion”. The game is inspired by the famous “Space Invaders”[1]. We chose this game because it does not require complex algorithms or network or database implementations. Instead, all it requires is to manage a cluster of classes that are in different hierarchy levels. For that reason, we think that this adaptation of Space Invaders is a very suitable choice for the CS319 course.

In the requirement analysis section, an overview as well as functional and non-functional requirements of the game will be introduced. Scenarios, use case models and information about the user interface will follow. A class diagram in addition to state chart diagrams and sequence diagrams will be the content of the analysis section. Conclusion and references sections will finalize the report.

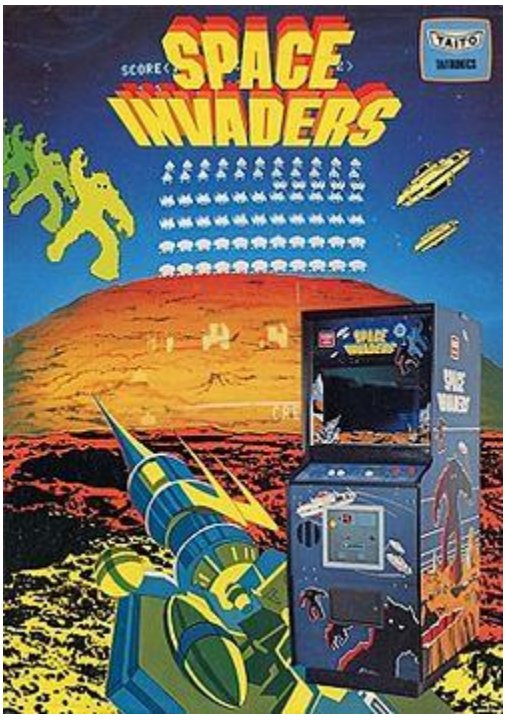


Figure 1: The cover of the original Space Invaders (1978)

2. Requirement Analysis

2.1. Overview

The player controls the laser gun that is located at the bottom of the screen. The gun can only be moved in the x-axis by using the arrow keys in the keyboard. The player can use the space button in the keyboard to shoot the aliens. The laser shot will be a very small vertical line emerging from the laser gun and moving to the top of the game screen. A shot will hit the target if it collides with the hit box of the alien model. The player will earn coins for each alien destroyed and will lose coins for each unsuccessful shot.

At the start of each level, the aliens will spawn at the top and will move horizontally across the game screen and if they reach to the end of the screen, they will move one level down vertically and will continue to move horizontally but this time in the opposite direction. So if the player fails to destroy all aliens in a specific time period, the aliens will eventually reach the bottom of the screen and the game will be over. If the player succeeds to eliminate all aliens, a new wave of aliens will come. With each wave, the difficulty of the game will gradually increase.

There will be an inter-level menu that will come up between each wave where the player can choose from three different options to increase the power of their laser gun permanently. The player will spend the coins he has earned to unlock those power ups.

There will also be a high-score window that will keep track of the ten players with highest points. The points will be calculated when the game is over by multiplying the number of levels the player has successfully passed by 1000 and adding the amount of unspent coins to that number.

The player will be able to pause the game at any time and continue afterwards from where he left. The player can also enable/disable the sound of the game from the pause menu. There will also be an option where the player can exit the game.

2.1.1. Power-ups

Attack damage: This option increases the amount of damage each shot inflicts to the aliens.

Attack speed: This option reduces the time between each consecutive shot.

Speed: This option increases the horizontal speed of the laser gun.

Figure 2: The model of the laser gun

Figure 3: The model of the aliens

2.2. Functional Requirements

2.2.1. Play Game

The game will start from wave 1 with default properties and will continue progressively until the aliens reach the surface. If the player succeeds, a new and more difficult wave will show up.

2.2.1.1. Purchase Power-ups

In the inter-level menu, the player can spend the coins he has earned to purchase power ups that are mentioned in the section 2.1.1.

2.2.1.2. Move the laser-gun

The player can move the laser gun horizontally between the left and right boundaries of the screen.

2.2.1.3. Shoot

The player can shoot the aliens.

2.2.2. Change Settings

The player can enable or disable the sound effects from either the main menu or the pause menu.

2.2.3. Pause Game

The player can pause and continue the game whenever he wants. Pausing the game will also enable the pause menu.

2.2.4. Exit Game

The player can exit the game whenever he wants from the pause menu.

2.2.5. View Highscores

The player can see the ten top players with highest points in the highscore table.

2.2.6. Display Help

The player can read a tutorial about the basic concepts of the game.

2.2.7. View Credits

The player can see information about the designers of the game.

2.3. Non-Functional Requirements

- The transitions between levels should be immediate. There should be no loading time.
- Application should not take more than 100 MB of space.
- The application should run at minimum 30 FPS at all times.
- The power-up menu should ask for a confirmation before applying the upgrades.
- There should be an option in “Change Settings” menu where the user can go back to default settings.
- The destruction of the aliens should be visualized with an explosion effect.

2.4. Constraints

- The implementation language must be Java.
- The project will be a swing application for desktop.

2.5. Scenarios

Scenario 1:

- The player wants to learn how to play the game. In order to learn the game, player chooses how to play option of the main menu of the game.

- The system displays the game manual.

- The player wants to exit the game manual. Player chooses the back option of the game manual to return to the main menu.

Scenario 2:

- The player wants to learn the high scores made. Player chooses view high scores option to see the top ten high scores made.

- The system displays the top ten high scores to the user.

- The user wants to exit the view high scores option. Player selects the back option of the view high scores option to return to main menu.

Scenario 3:

- The user selects the play game option of the main menu.

- The system loads the game according to the game settings.

- Player starts to play the game from the first level of the game. Player shoots to

kill aliens. Player uses a laser gun to shoot the aliens. If player's shot hits an alien, that alien gets killed.

- Aliens try to shoot the spaceship of the player. If an alien's shot hits the spaceship, player loses the game.

- Player earns points when all aliens get killed. Points earned are proportional to the aliens killed and inversely proportional to the time spent to kill all the aliens. System calculates the points earned.

- The system updates player's score. If the player's score is in the range of the top ten highest scores list, the system updates the high scores list. Otherwise, the system displays the high scores list.

- When all the aliens die, the system displays the upgrade player option before passing to the next level.

Scenario 4:

- Player wants to see credits. Player uses view credits option of the main menu. The system displays the credits.

- Player wants to return to the main menu. Player selects the back option of the view credits. The system returns the main menu.

Scenario 5:

- Player wants to change the default settings. Player selects the change settings option of the main menu or pause menu.

- Player can turn the sound of the game and music of the game off using change settings menu. System updates the game settings according to the specifications made by the player.

- Player uses the back menu to return to main menu or pause menu from where change settings menu is reached.

Scenario 6:

- The system activates the upgrade player function of the game when the player completes a level.

- The player either continues to play the game without spending coins or selects

power-ups to upgrade the spaceship. The system updates the coins earned according to the player's selection.

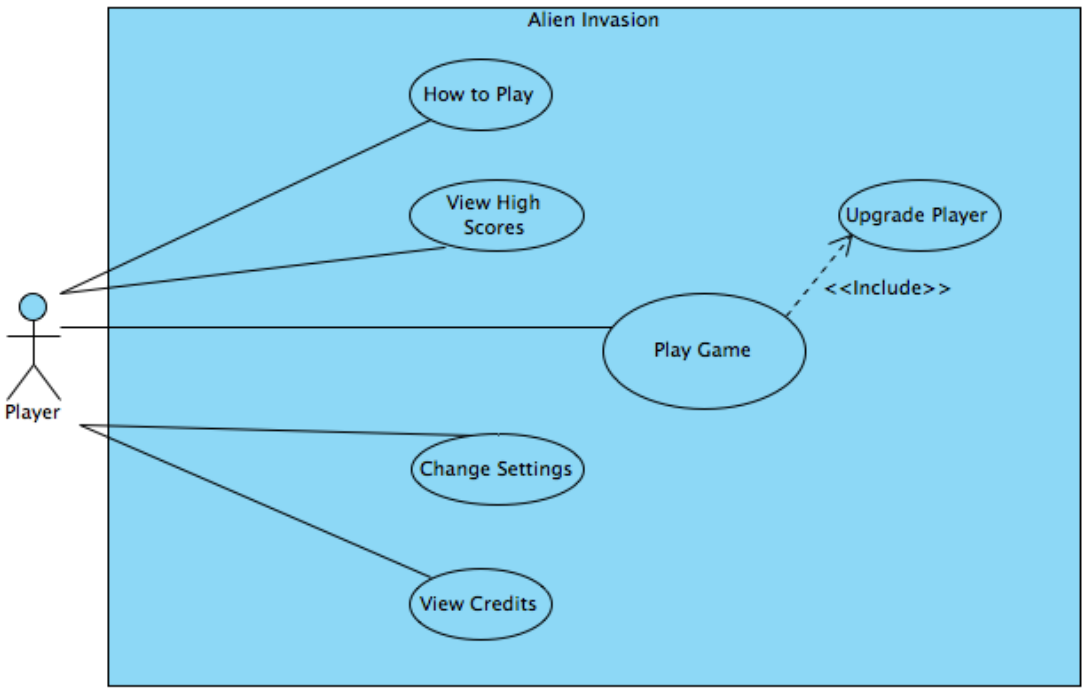
Scenario 7:

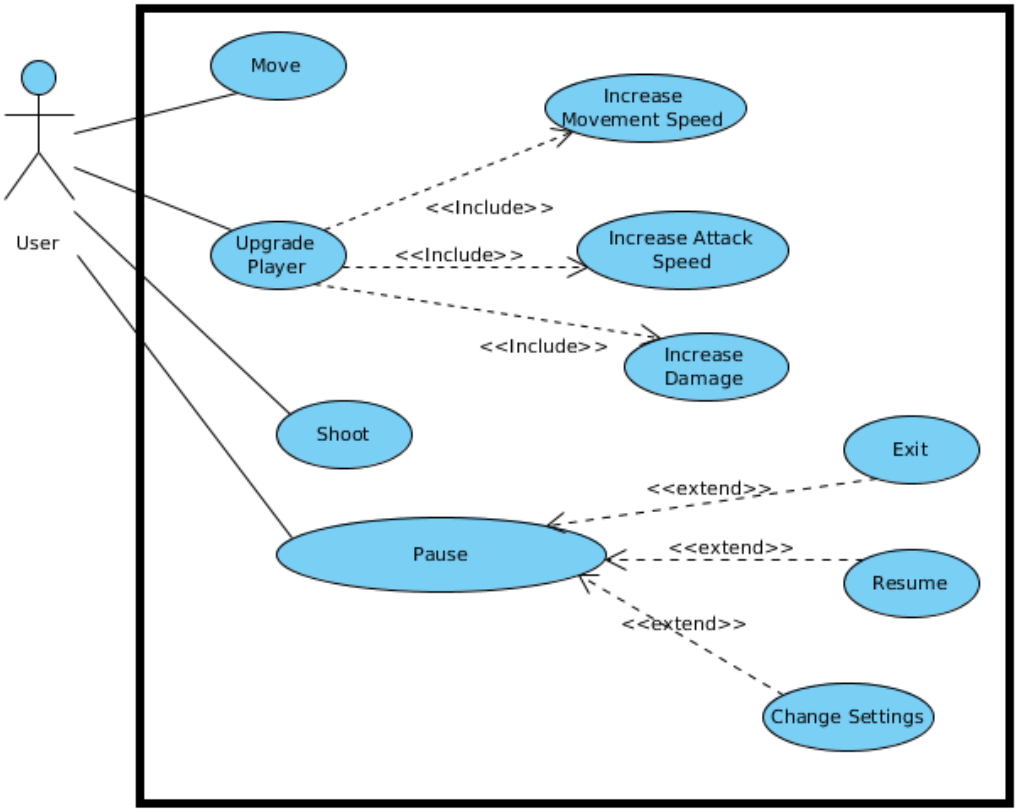
- The system initiates the emergence of the aliens with the beginning of each level. The aliens move horizontally and get closer to the user by one level each time the leftmost or rightmost alien reaches left edge or right edge respectively. The game is over when an alien reaches the bottom of the screen. System initiates the emergence of a new group of aliens if the user manages to kill all the aliens and not to get hit. The system increases the level of difficulty in each upcoming level.

2.6. Use Case Models

2.6.1. Use Case Model

In this section, we will provide use case diagram and models of Alien Invasion game to express it more clearly.





2.6.2. How to Play

Use Case ID:	UC_HP_001		
Use Case Name:	How to Play		
Created By:	GncKlync	**Last Updated By:**	-
Date Created:	26/10/2015	**Last Revision Date:**	-
Actors:	Player		
Description:	Player could read the manual of the game to learn how to play the game.		
Trigger:	Player selects “How to Play” from Main Menu.		
Preconditions:	-		
Postconditions:	-		
Normal Flow:	1. Player selects “How to Play” option from the Main Menu. 2. How to Play will be displayed to the Player on screen.		

Alternative Flows:	A. If Player wants to exit from the How to Play screen A.1. Player select “Back” button to return “Main Menu” A.2. System displays “Main Menu”
Exceptions:	-
Includes:	-
Frequency of Use:	-
Special Requirements:	-
Assumptions:	Player will read the manual and ACTUALLY learn the game and buttons to play it.
Notes and Issues:	-

2.6.3. View High Scores

Use Case ID:	UC_HS_001		
Use Case Name:	View High Scores		
Created By:	GncKlync	**Last Updated By:**	-
Date Created:	26/10/2015	**Last Revision Date:**	-
Actors:	Player		
Description:	This event will display “High Scores” menu when its triggered by the user. Also makes the transition between “High Scores” and “Main Menu”.		
Trigger:	Player select “View High Scores” from Main Menu.		
Preconditions:	**System should keep a record of top ten scores.**		
Postconditions:	-		
Normal Flow:	1. Display top ten high scores with player names.		
Alternative Flows:	A. To go back to main menu at any time: A.1. Player select “Back” button to return “Main Menu” A.2. System displays “Main Menu”		
Exceptions:			
Includes:	HighScoreChart		
Frequency of Use:	-		
Special Requirements:	-		
Assumptions:	If there is not enough high scores saved by the player(less than 10), System will fill out the chart with player name “XXX” and score of “0”.		
Notes and Issues:	-		

2.6.4. Play Game

Use Case ID:	UC_PG_001		
Use Case Name:	Play Game		
Created By:	GncKlync	**Last Updated By:**	-
Date Created:	26/10/2015	**Last Revision Date:**	-
Actors:	Player		
Description:	Player tries to achieve highest score by completing levels. System will keep the score, with the provided name from the Player at the beginning of the game if Player gets a high score.		
Trigger:	Player select “Play Game” from Main Menu.		
Preconditions:	**Game loaded according to the game settings. At start, if Player did not change game settings, game will begin with default settings.**		
Postconditions:	If score is greater than the top 10 high scores, system will update the high score list with this new score by the name user entered at the beginning of the game.		
Normal Flow:	1. System starts the game. 2. Player will start from the first level. 3. Player will kill all the aliens in the screen and earns credits for every alien he killed and also from the time used to kill them. 4. System will call “UC_UP_001” use case. 5. Play will spend or not his scredits to upgrade his spaceship for the next round. 6. User then selects 'Continue' to start the next level. - Steps 3-6 will be repetitive for the all levels of the game. 7. If Player scores higher than a value in the high score chart,		

	System will store his score to the high score list. Otherwise System will show the high score list only. 8. System returns to Main Menu.
Alternative Flows:	3A. Player tries to finish the level by killing all the enemies while trying to avoid getting hit from aliens. 3A.1. Player presses 'Shoot' button to shoot enemies. 3A.2. Enemies are dead and removed from the screen if they get hit. 3A.3. System will update Player's score. 3A.4. System will check if all the enemies are killed. -If not player will continue again from the 3A.1.-3A.4. 3A.6. If all the enemies killed by the Player, level ends and System will call "UC_UP_001" use case. 3B. Player activates a power-up. 3B.1. Player will buy power-ups between levels, during "UC_UP_001" use case. 3B.2. These power-ups are added to Player's spaceship. 3B.3. Player will activate a power-up. System will do the necessary changes over the game accordingly. 3B.4. System will clear the power-up from the inventory of power-ups. 3B.5. At the end of power-up time, System will return back to its normal settings. -This process is same for every time a power-up is used. A. If Player pauses the game.

	A.1. Player press the proper key to pause the game. A.2. System will pause the game. A.3. System will show the “Pause Menu” A.3.1. If Player selects “Resume”, system will return to game. A.3.2. If Player selects “Return to Main Menu”, System will close the current game and opens the “Main Menu” A.3.3. If Player selects “Change Settings”, System will call “UC_CS_001” use case. A.3.4. If Player selects “How to Play”, System will call “UC_HP_001” use case. A.3.5. If Player selects “Exit Game”’, System will return to the desktop by closing the game.
Exceptions:	-
Includes:	UC_UP_001
Frequency of Use:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

2.6.5. View Credits

Use Case ID:	UC_CR_001		
Use Case Name:	View Credits		
Created By:	GncKlync	**Last Updated By:**	-
Date Created:	26/10/2015	**Last Revision Date:**	-
Actors:	Player		
Description:	This event will display "Credits" screen when its triggered by the user. Also makes the transition between "Credits" and "Main Menu".		
Trigger:	Player select "View Credits" from Main Menu.		
Preconditions:	-		
Postconditions:	-		
Normal Flow:	1. Display information about developers and their messages about the game.		
Alternative Flows:	A. To go back to main menu at any time: A.1. Player select "Back" button to return "Main Menu" A.2. System displays "Main Menu"		
Exceptions:	-		
Includes:	-		
Frequency of Use:	-		
Special Requirements:	-		
Assumptions:	-		
Notes and Issues:	-		

2.6.6. Change Settings

Use Case ID:	UC_CS_001		
Use Case Name:	Change Settings		
Created By:	GncKlync	**Last Updated By:**	-
Date Created:	26/10/2015	**Last Revision Date:**	-
Actors:	Player		
Description:	Player could change the default game settings such as “sound on/off”, “music on/off”, "background" from the “Change Settings” screen.		
Trigger:	1. Player selects “Change Settings” from Main Menu. 2. Player selects “Change Settings” from Pause Menu.		
Preconditions:	**Current – if not changed by the player, default – settings will be shown to the player.**		
Postconditions:	Game settings are updated.		
Normal Flow:	1. Player selects “Change Settings” option from the main menu or Pause Menu. 2. Game Settings will be displayed to the Player on screen. 3. Player will change/update game settings. 4. System will update the game accordingly.		
Alternative Flows:	A. If Player wants to exit from the Game Settings screen A.1. Player select “Back” button to return previous menu A.2. System displays “Main Menu” or “Pause Menu” accordingly		
Exceptions:	-		
Includes:	-		
Frequency of Use:	-		

Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

2.6.7. Upgrade Player

Use Case ID:	UC_UP_001		
Use Case Name:	Upgrade Player		
Created By:	GncKlync	**Last Updated By:**	-
Date Created:	26/10/2015	**Last Revision Date:**	-
Actors:	Player		
Description:	Player could upgrade his spaceship and buy power-ups to use druing a level with his credits he earned during the levels.		
Trigger:	System will automatically call this at the end of each level's completion.		
Preconditions:	**Player should have completed a level by killing all the enemies.**		
Postconditions:	Player's spaceship is upgraded accordingly.		
Normal Flow:	1. System pops up the "Upgrade Player" screen after end of a level. 2. Different options to upgrade Player's spaceship will be displayed to the Player on screen. 3. Player will spend credit he earned during levels to buy power-ups or upgrade his spaceship. 4. System will make the necessary changes. 5. Player clicks "Continue" button to return back the game from		

	the next level.
Alternative Flows:	A. If Player did not want to spend his credits to buy power-ups or upgrade his spaceship. A.1. Player select “Continue” button to return back to game without buying anything. A.2. System will call "UC_PG_001" use case.
Exceptions:	This use case might never be called if user could not complete the first level of the game.
Includes:	-
Frequency of Use:	After successfully end of every level
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

2.7. User Interface

2.7.1. Main Menu

When the game starts, user will be welcomed with the 'Main Menu'. On this screen, user could choose from six different options that are 'Play Game', 'Highscores', 'Credits', 'Change Settings', 'Help' and 'Exit'.

Figure 4: 'Main Menu' screen

2.7.2. Play

When the players click on 'Play Game' button, the play screen shows up and player enters a username than a new game starts. It could be easily followed that how many lives and points the player has from screen.

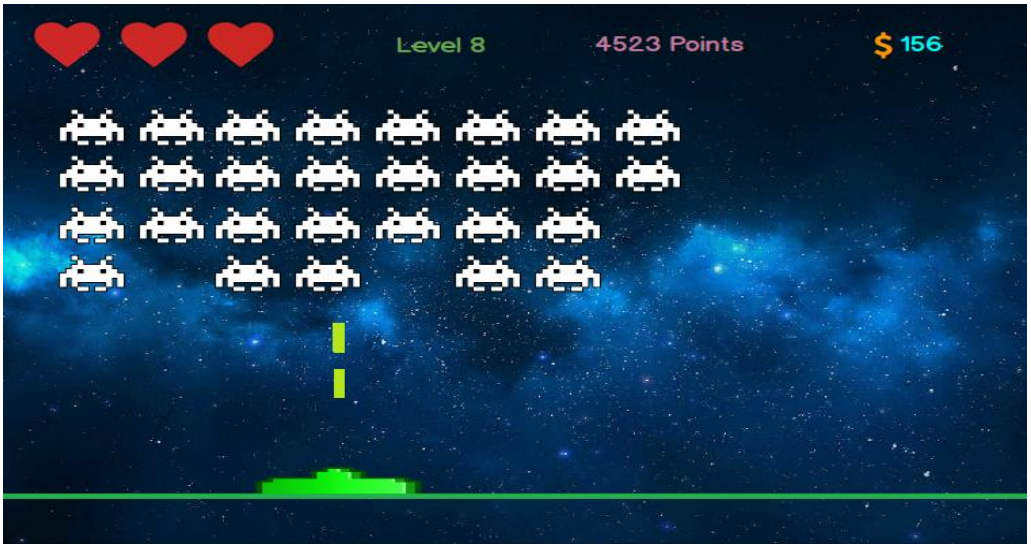


Figure 5: 'Gameplay' screen

2.7.3. Pause Menu

When “ESC” button is pressed during game, the 'Pause' menu will be displayed to the use r and he could select options of 'Resume', 'Change Settings' and 'Exit'.

Figure 6: 'Pause Menu' screen

2.7.4. Upgrade Player

After completing every round, user will be directed to this page. He could spend his credits earned during levels to upgrade his spaceship to and prepared for the next and harder levels.

Figure 7: 'Upgrade Player' screen

2.7.5. Highscores

When the game has ended, system checks if user's score is higher than any of the scores in the current list. If it is the case, then system automatically adds user's score with the username entered at the beginning of the game to the high score list. This list can be checked from the 'Highscores', which can be reached from 'Main Menu'.

Figure 8: 'Highscores' screen

2.7.6. Change Settings

The players could change audio setting by clicking on 'Settings' button.

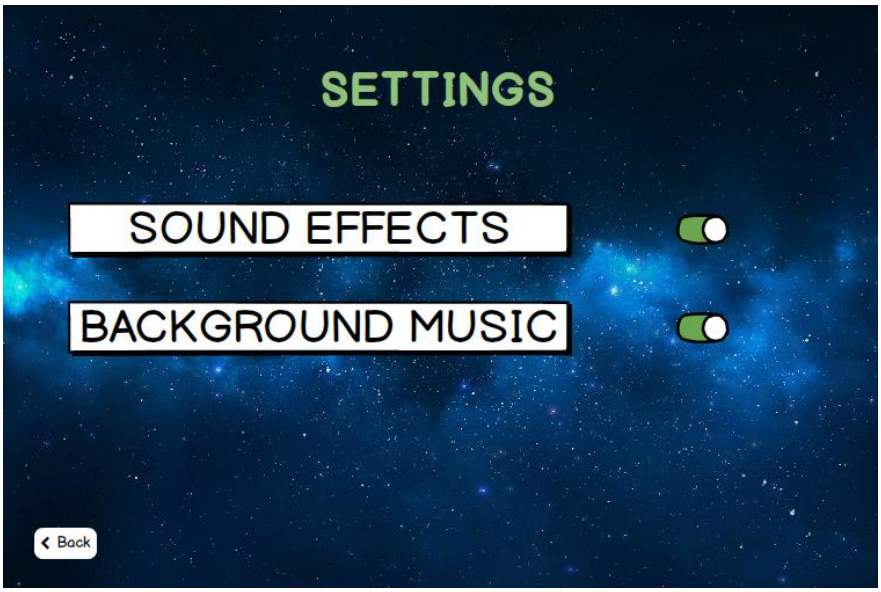


Figure 10: 'Change Settings' screen

2.7.7. Help

Players could learn what the aim of the game is and how to control spaceship during the game from this screen by pressing 'Help' button.

Figure 11: 'Help' screen

2.7.8. Credits

Players could reach the name of the game's developers and their contact information by clicking on 'Credits' button.

Figure 12: 'Credits' screen

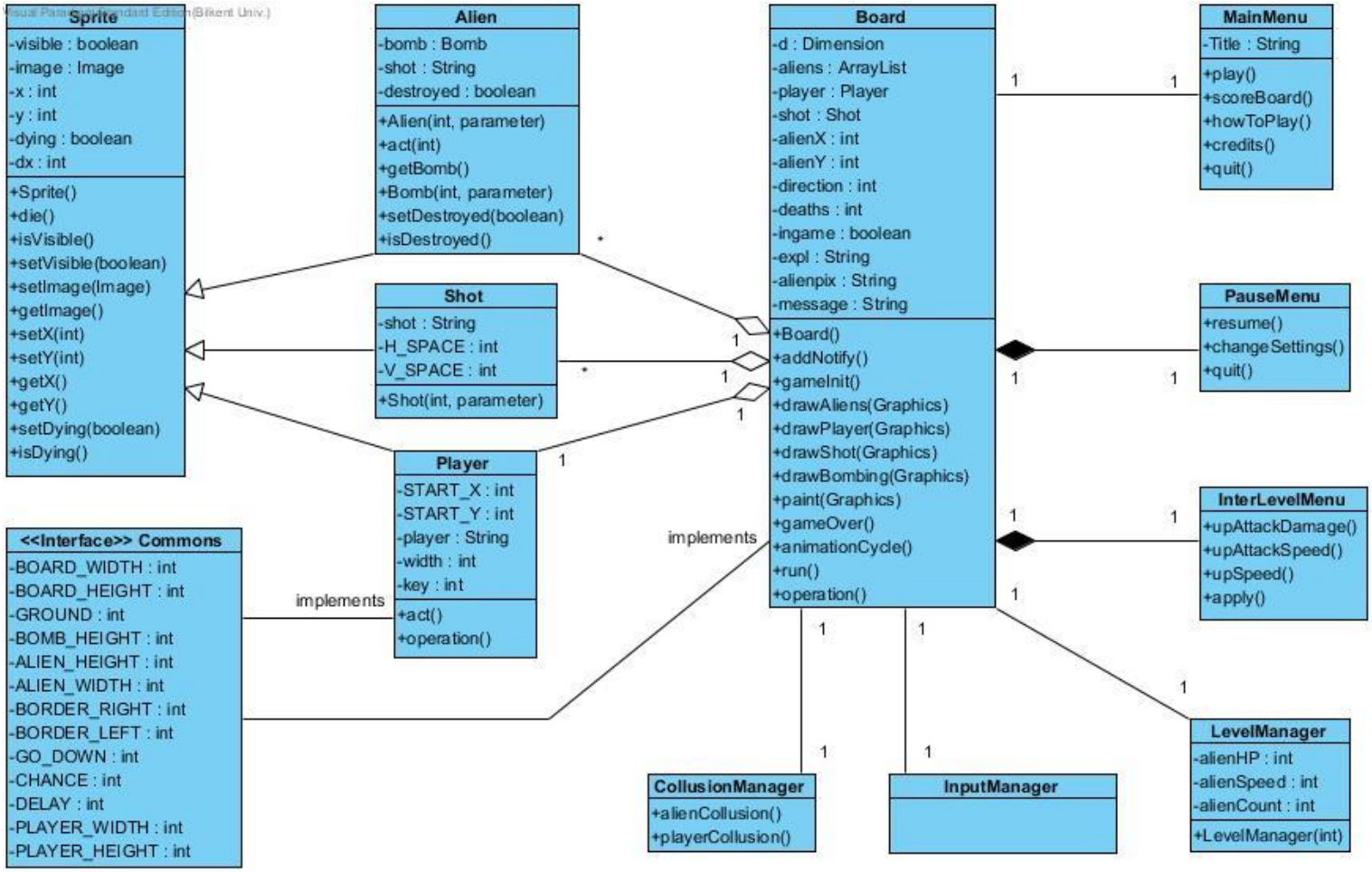
3. Analysis

3.1. Object Model

3.1.1. Domain Lexicon

- Laser gun: The weapon that the player controls. It can fire a laser shot in vertical axis.
- Aliens: The enemies of the player that are approaching the laser gun. They need to be destroyed to continue the game.
- Attributes: Properties of the laser gun that can be upgraded in the inter-level menu.
    - *Attack Damage:* The damage that is inflicted with a single shot of laser gun.
    - *Attack Speed:* The rate at which the laser gun can shoot.
    - *Speed:* The horizontal speed of the laser gun.
- Coins: Currency that the player earns or loses depending on their performance. Coins can be spent at the inter-level menu to upgrade the attributes.
- Level: The current stage of the game. Each level is more difficult than the previous one.

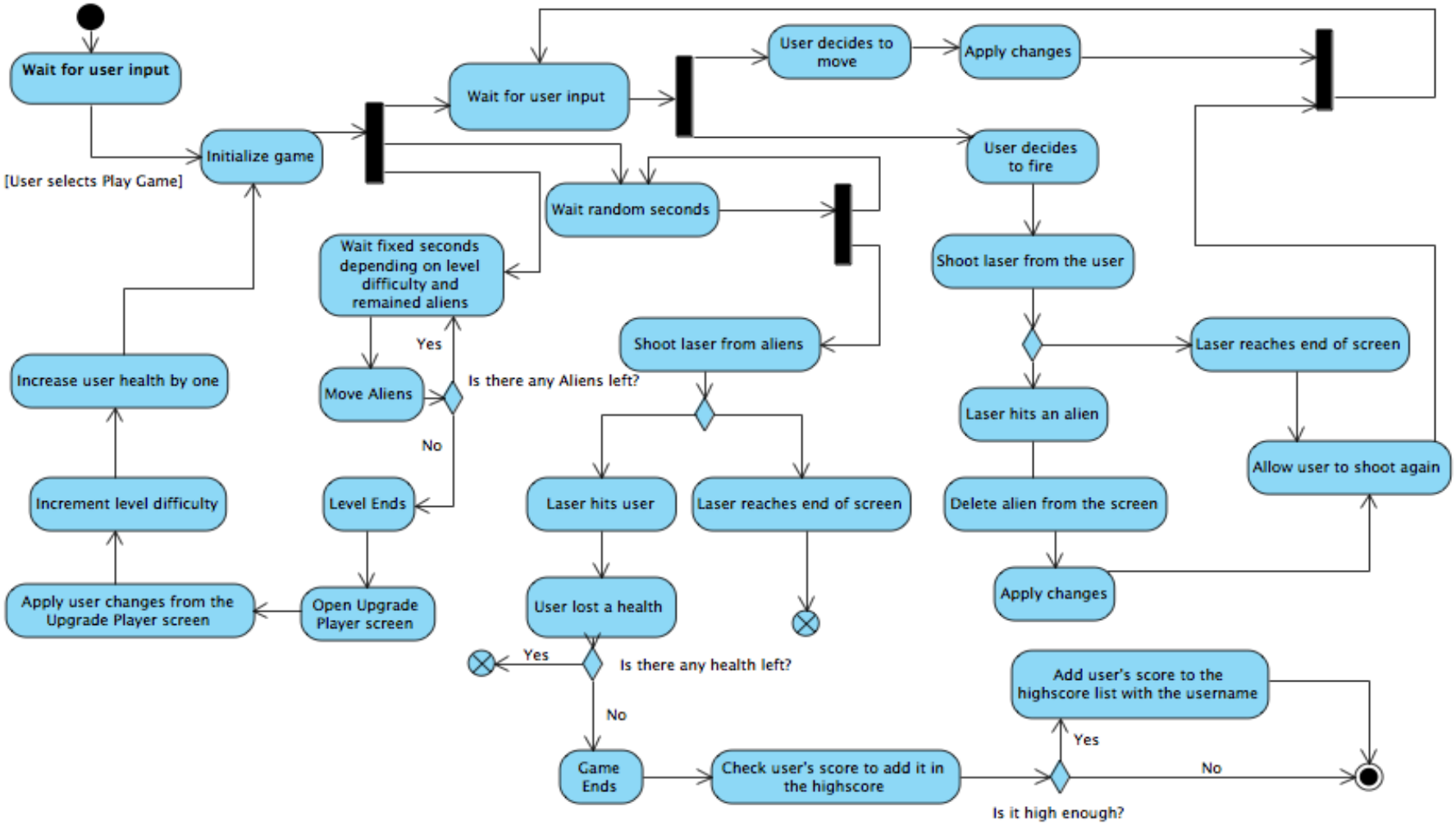
3.1.2. Class Diagram



MainMenu, PauseMenu and InterLevelMenu are the boundary classes that provide the user interaction. Menu class will be instantiated in the main method of the program but the PauseMenu and interLevelMenu will be instantiated during the game by the Board class. Board class is the control class of the game. It can be described as the brain of the game. It provides a connection between the user and game entities and maintains the game by continuous checks and loops. There are three entities in the game that are Player, Alien and Shot. They extend the Sprite class and are instantiated by the Board class during gameplay. The Commons class is an interface and its only job is to keep the constants.

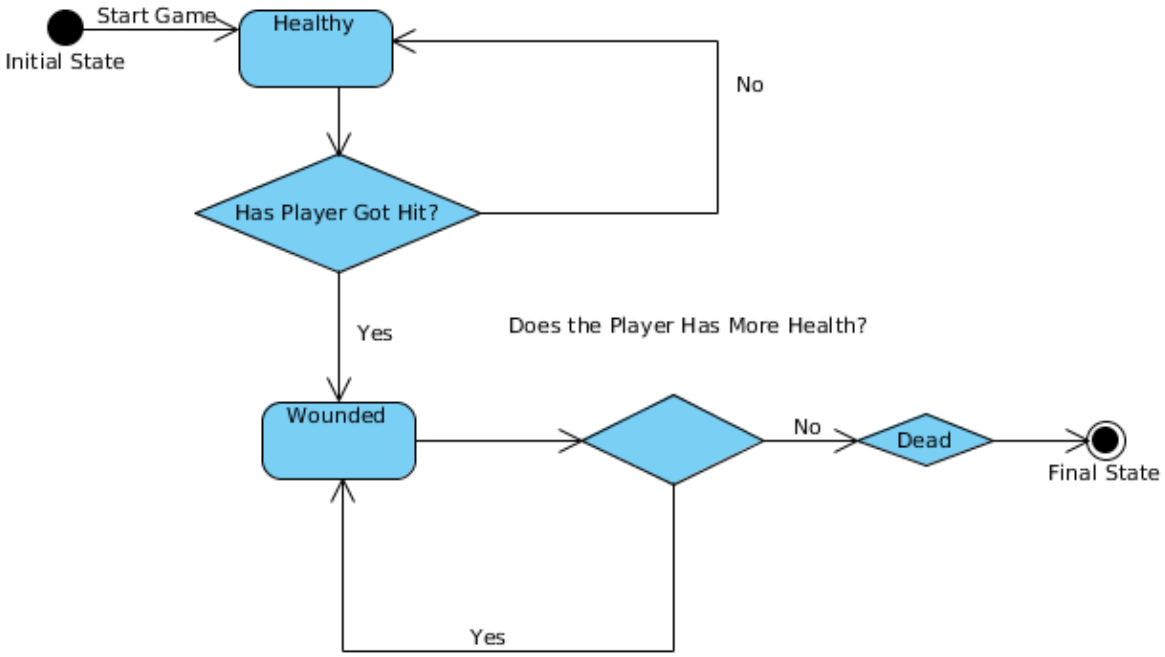
3.2. Dynamic Models

3.2.1. Activity Diagram

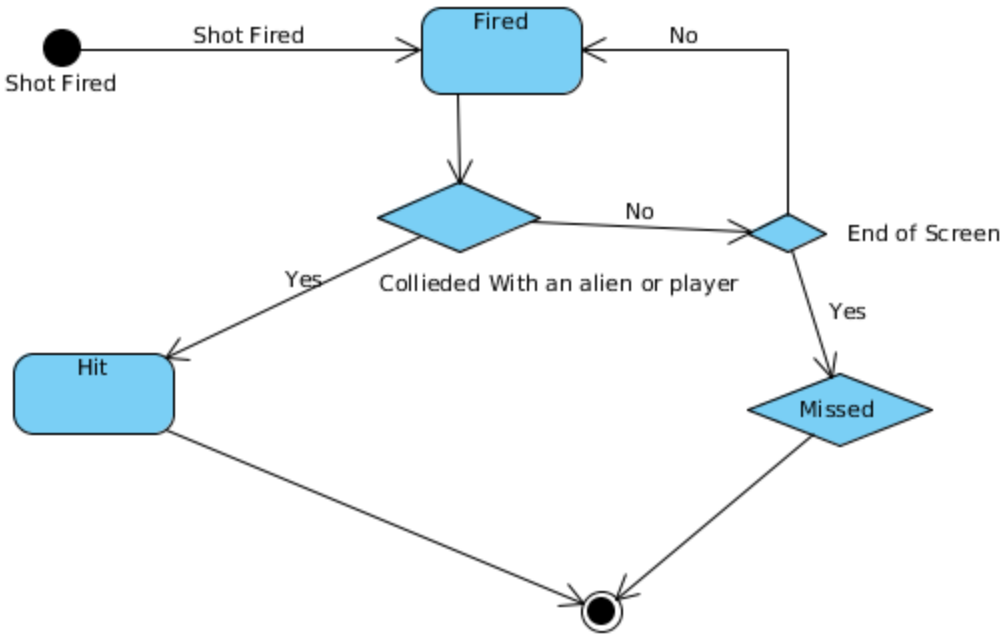


3.2.2. State Chart Diagrams

3.2.2.1. Player

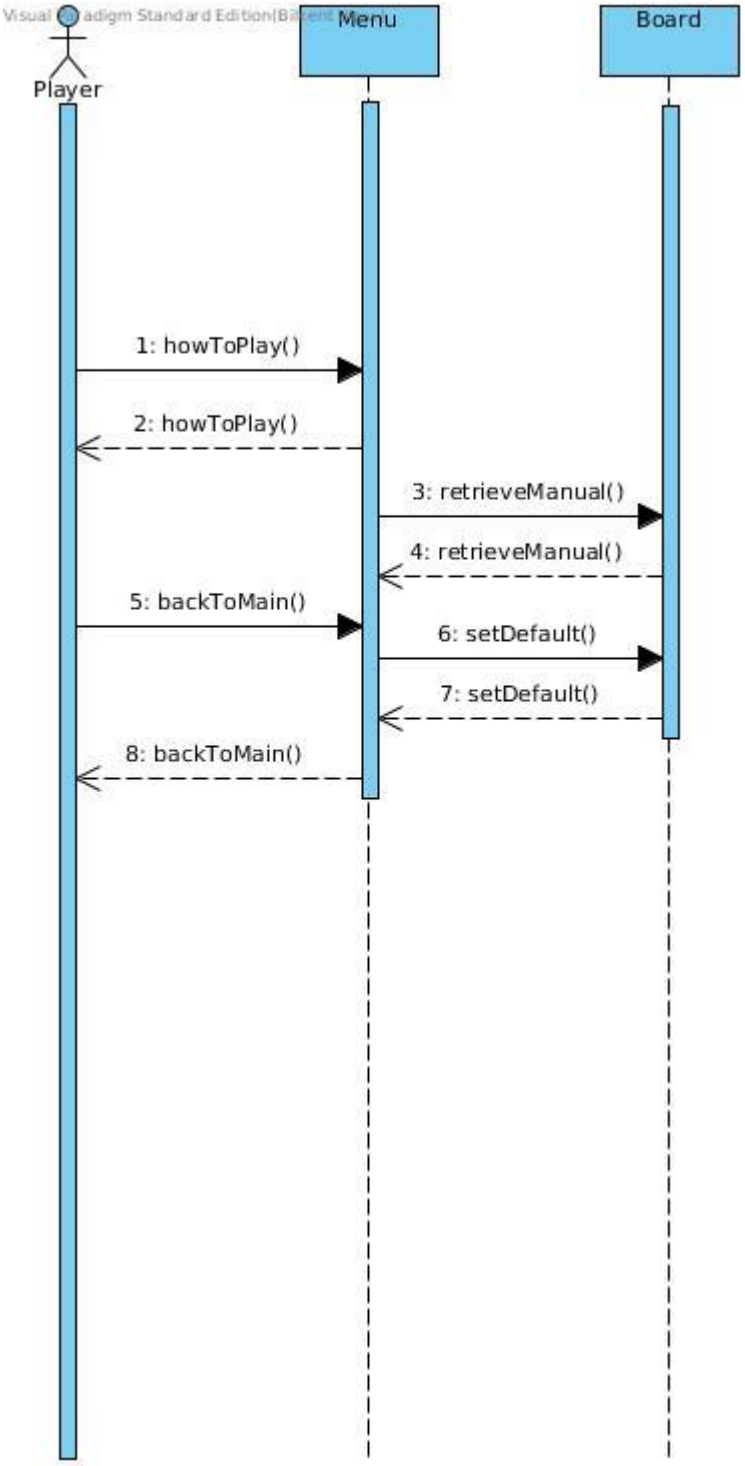


3.2.2.2. Shot



3.2.3. Sequence Diagrams

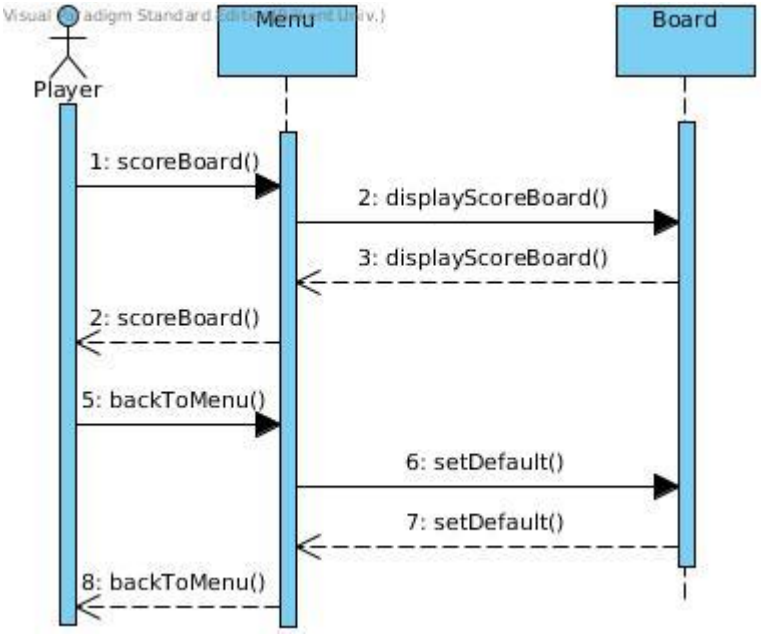
Sequence Diagram 1:



The player chooses how to play option of the main menu of the game. This trigger calls the “howToPlay()” method of the Menu object. Call of the “”howToPlay() function triggers the call of the “retrieveManual()” method through the Board object.

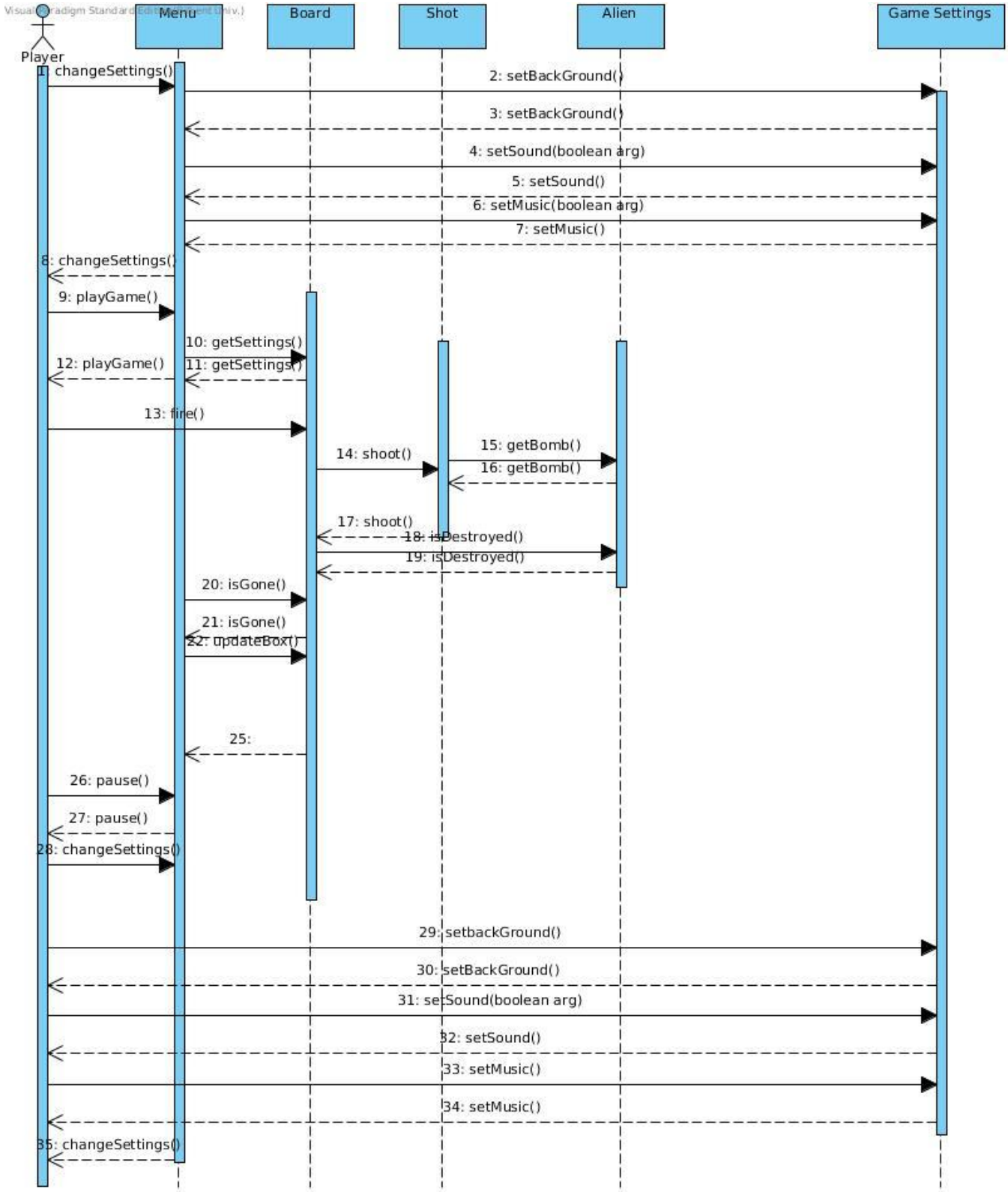
"retrieveManual()" method displays the playing instructions. User triggers the call to the "backToMain()" method which causes the transition to the main menu that is done through the "setDefault()" method of the Board object.

Sequence Diagram 2:



The player chooses view high scores option of the main menu. This triggers the call of the "scoreBoard()" method of the Menu object. Call to the "scoreBoard()" option triggers the call to the "displayScoreBoard()" method which updates the screen and displays the high scores. User triggers the call to the "backToMenu()" method by interacting with the back to main menu option of the display high scores. This interaction triggers the call to the "setDefault()" method of the Board object which in turn updates the user interface and displays the main menu.

Sequence Diagram 3:



The user chooses change settings option of the main menu. This interaction triggers the call to the "changeSettings" method which causes the transmission of the display

menu to change settings mode. The user may cause the call to the “setBackGround()”, “setSound()” and “setMusic()” methods by interacting with the control buttons of the change settings mode. After the user confirms the change of settings main menu is displayed.

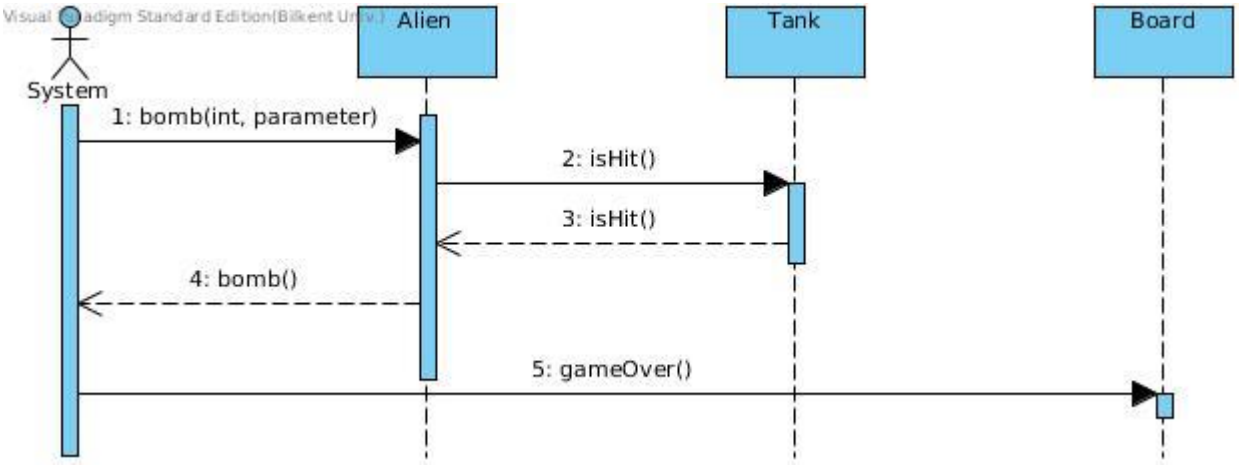
This process of change of settings is optional. If the user does not enter this mode the game starts with default settings.

Player triggers call to the “playGame()” method by interacting with play game option of the main menu. This event triggers the call to the “getSettings()” method of the Board object which takes the current settings and loads the game.

In the game mode, the player uses the fire button to call the “fire()” method of the Board object which in turn calls the “shoot()” method of the Shot object. Interaction between the bomb and alien is formed through the “getBomb()” method of an Alien object. If the shoot is successful it is announced through the “isReturned()” method of the Alien object to inform the Board object. The existence of the remaining Aliens is checked through the “isGone()” method of the Board object. If all of the Aliens die, credits that the user earns are calculated and score of the user is recalculated. This causes the call to the “updateBox()” which updates the top 10 scores made.

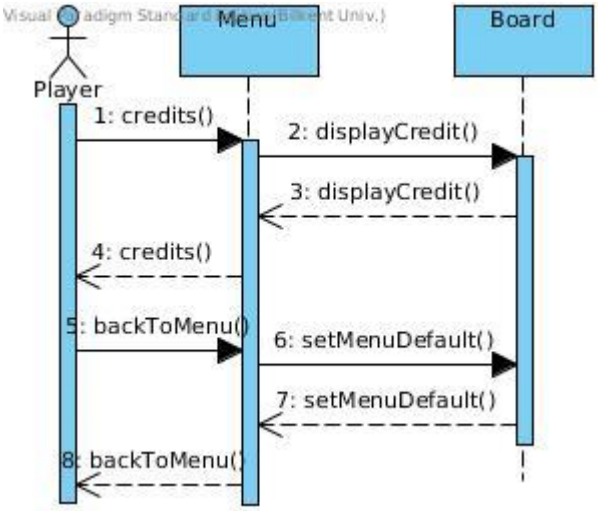
During the game mode, player has the option to pause. In order to do so, the player uses the pause button to call the “pause()” method of the Menu object. In the pause mode the player has the option to change the settings as in the beginning of the game.

Sequence Diagram 4:



The system causes the call to the "bomb(int, parameter)" method of the Alien game. "bomb()" method triggers the call to the isHit() method of the tank of the player. If the tank is hit "isHit()" method returns true and system triggers the call to the "gameOver()"method of the Board object.

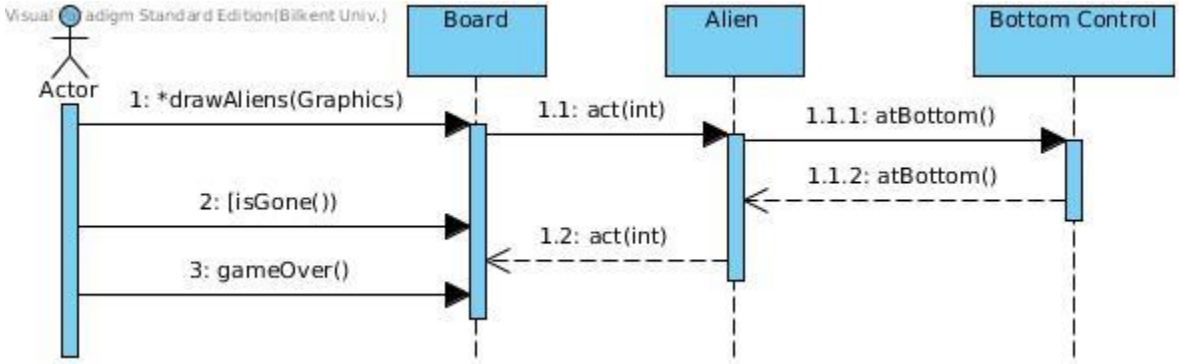
Sequence Diagram 5:



The player selects the view credits option of the main menu which triggers the "credits()" method of the Menu object which in turn calls the displayCredit() method of the Board object. Board updated the screen and displays the credits earned. Player triggers "backToMenu()" method of the Menu object which in turn calls the

"setMenuDefault()" method of the Board object. Board updated the display and retrieves the main menu mode.

Sequence Diagram 6:

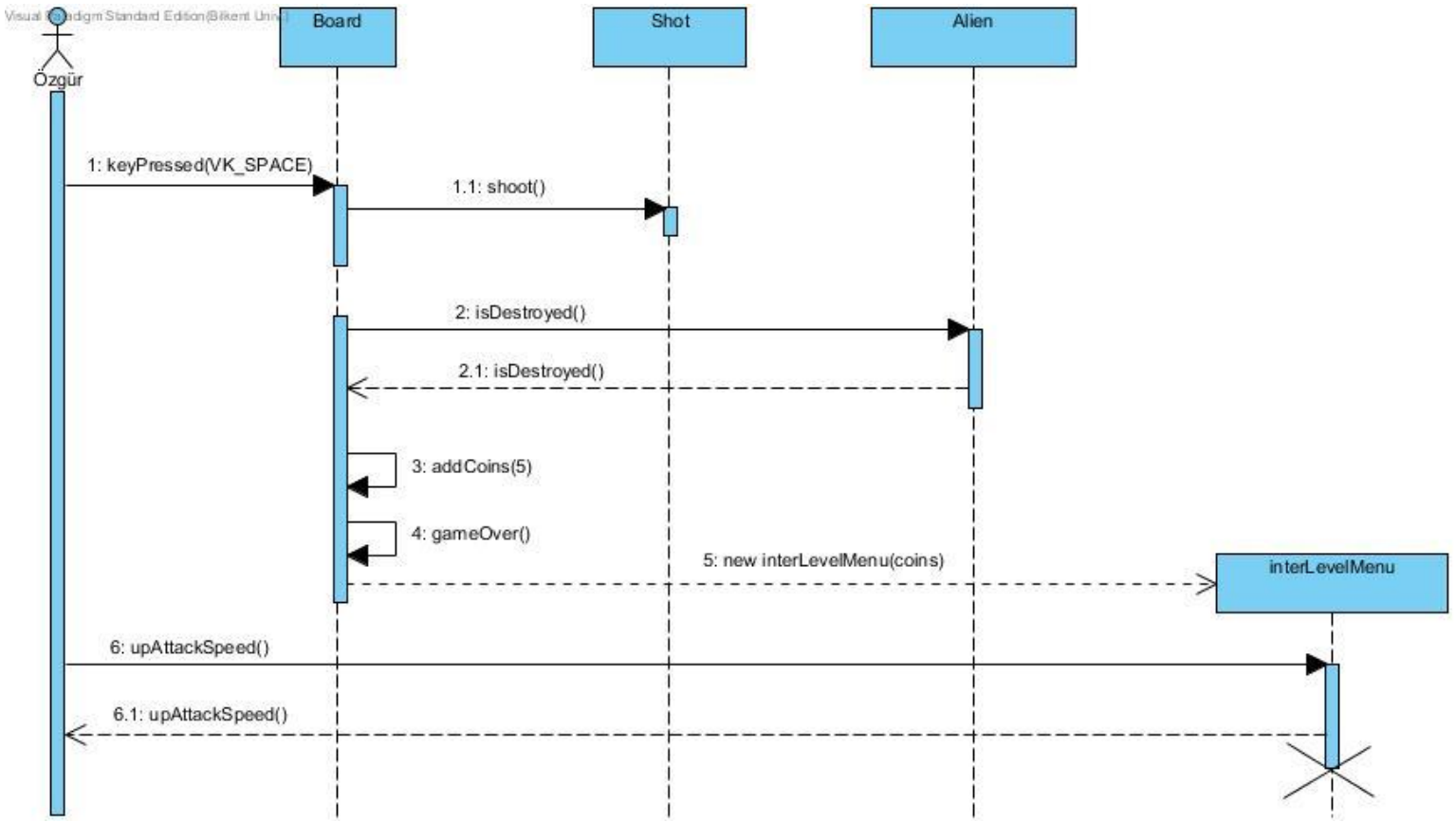


The system triggers the creation of the Aliens through the call to the "drawAlliens(Graphics)" method of the Board object. Creation of the Alien objects triggers the "act(int)" method which causes persistent movement from left to right and right to left. Act method calls the "atBottom()" method of the BottomControl object which returns true or false on the condition that an Alien object reaches the bottom of the screen which causes the system to call the "gameOver()" method of the Board object.

The system repeats this process on the condition that "isGone()" method of the Board object returns true.

Sequence Diagram 7:

Özgür is playing the game. There is only one alien left in the current level. He presses the shoot button and successfully hits the last alien. He is awarded 5 coins for destroying an alien. Since all aliens are destroyed, Özgür successfully passes this level and the inter-level menu appears. He chooses to upgrade the laser gun's attack speed attribute so his characters attack speed will be increased for the rest of the game.



4. Design

4.1. Design Goals

We aim to develop a robust, user-friendly, maintainable, portable and well-documented software through object oriented analysis and design. We aim to clarify each aspect of our system through modules with well-defined interfaces to motivate modularity and hierarchy principles and to realize object-oriented design approach. In addition to this, we are determined to eliminate the impact of changes in such a way that modules or subsystems interact with each other through well-defined interfaces only. Finally, we aim to break the modules of our project into understandable and easy to handle elements in order to realize the system on time.

Robustness

We want our program to be stable, clever on handling problems and not to crash on exceptional situations.

User-friendly

We want our program to be easily learned and played so that it can be entertaining. Therefore the game must be user-friendly.

Maintainable

We want our program to be maintainable so that possible bugs and errors can be eliminated, and program can be compatible with the fast changing technologies.

Portability

We want our program to be executed on several different platforms so that end users using various platforms can enjoy our game. Therefore we decided to use Java programming language so that the game can be played on different types of platforms.

Well Documented

We want our program to be well documented so that its maintenance could be performed.

4.1.1. Possible Trade-offs

Robustness vs. Cost

In order to have a robust system, a longer design and testing period is needed. Therefore, robustness may increase the cost of the software.

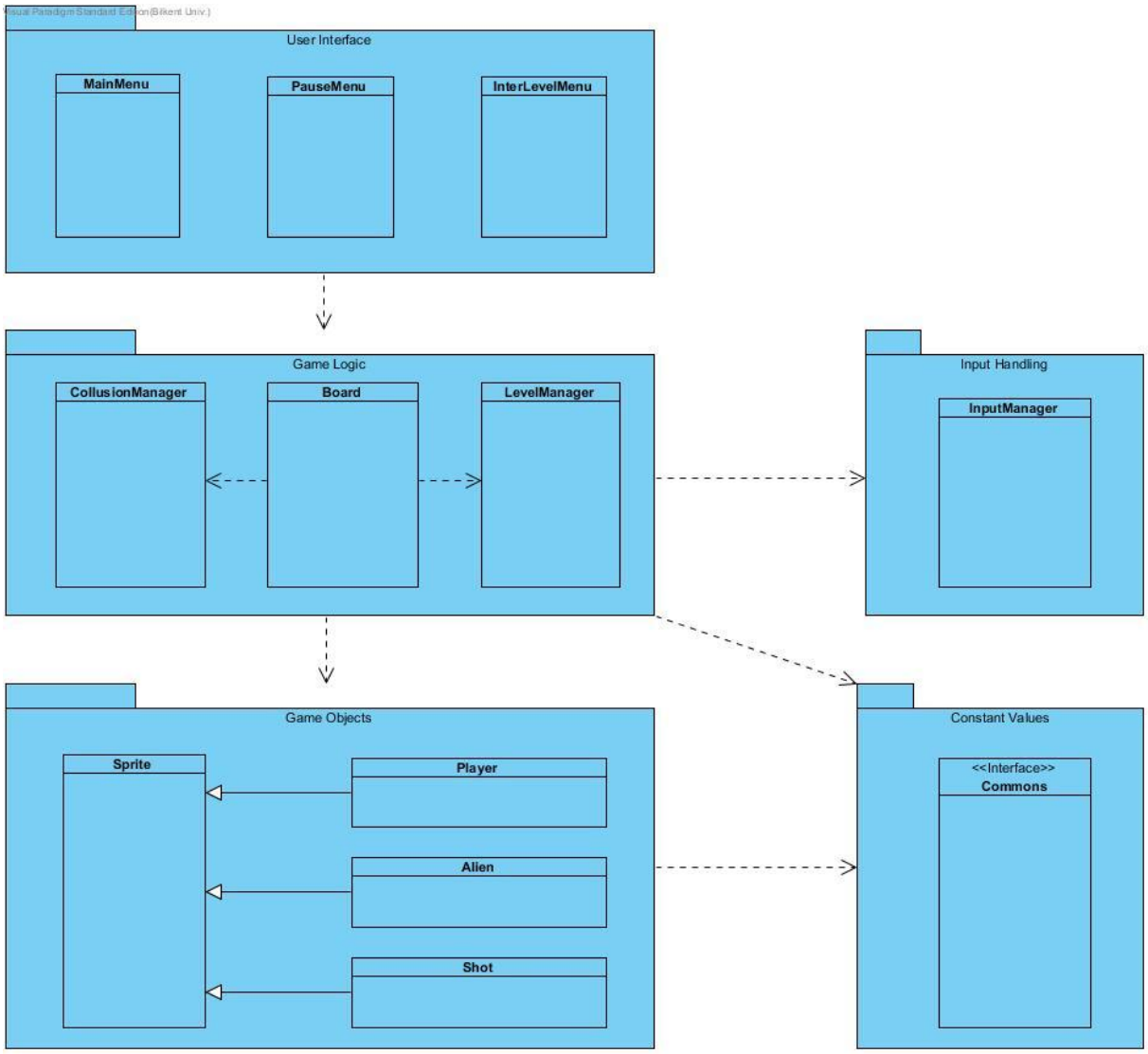
User Friendliness vs. Cost

In order to have a user-friendly system, analysis period gets longer. Therefore, user-friendliness increases the cost of the software.

Portability vs. Efficiency

In order to have a portable system we need to use Java programming language which runs on a virtual machine and platform independent. However, using a platform dependent programming language, a better runtime performance can be acquired.

4.2. Subsystem Decomposition



We divided the classes into 5 packages according to their purposes and relationships. The top layer has the user interface subsystem that includes the menu classes which provide the user interaction. Those are MainMenu, PauseMenu and InterLevelMenu. MainMenu provides the main menu, PauseMenu provides the pause menu when the user pauses the game and lastly InterLevelMenu provides an upgrade menu where the player can upgrade his attributes between each level.

The user interface layer depends on the game logic subsystem. The Board class inside the game logic subsytem is the core class of the project. It controls the game loop and the interaction between the game map and the game objects. However, it has a partition relationship with level manager and collusion manager inside the same subsytem to keep the game loop running. The game logic subsytem has a partition relationship with the input handling subystem that processes the user input during the gameplay. The game logic subsytem is also dependent on the game objects and constant values subsytems that are at the lowest layer.

In the lowest layer, there are game objects subsytem and the constant values subsytem. In the game objects subsytem, there are three classes that are Player, Alien and Shot that are extending their parent class Sprite. These game objects constitute the visible part of the game and are manipulated by the game logic subsytem according to the actions of the user. The game objects subsytem has a partition relationship with the constant values subsytem that includes an interface which holds the constant values that are necessary.

During the decomposition process, our main concern was to achieve high coherence and low coupling. In our decomposition, the calls are made from top to bottom so the lower classes do not need to know anything about higher level classes. In the case of partition relationships, the called classes are generally manager classes or adapters so they do not also need to know the details of the calling class. In that sense, our design can be considered as a low coupled one. Moreover, the classes in a particular subsytem perform similar tasks so that our design can also be considered as a high coherent one.

4.3. Architectural Patterns

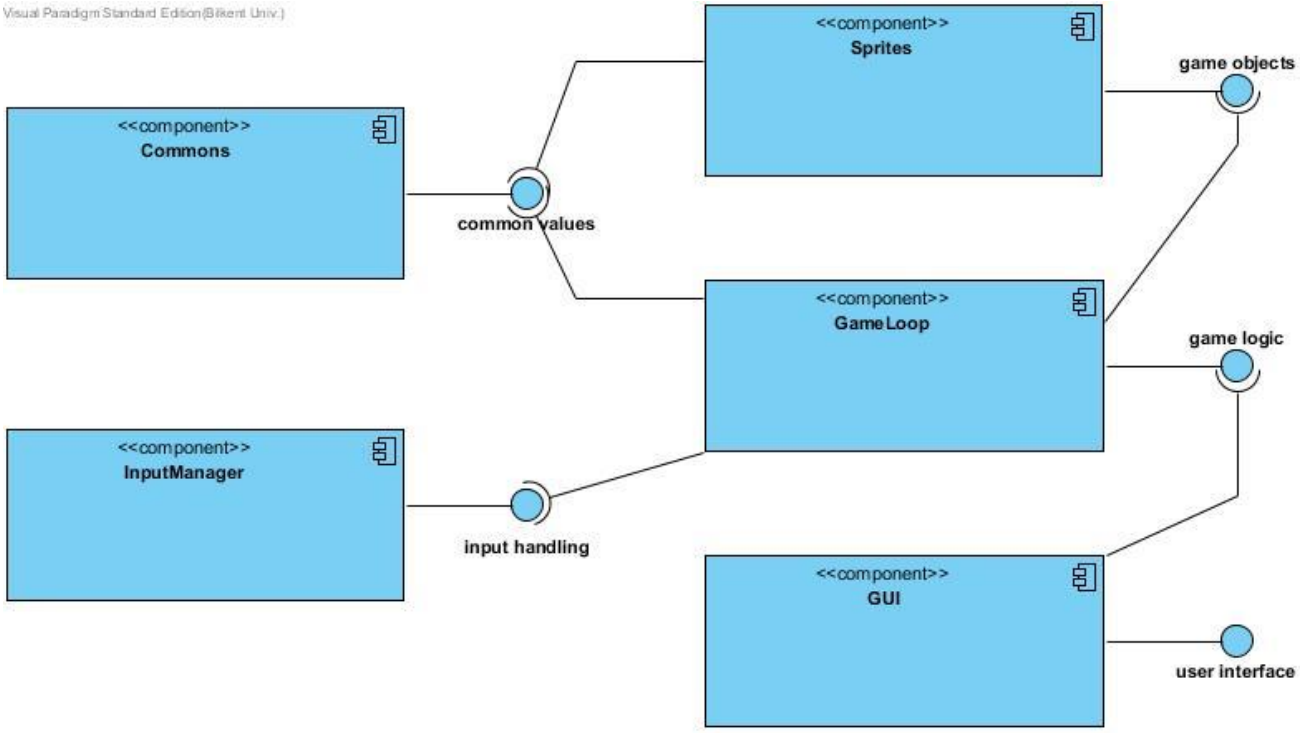
While designing 'Space Invaders' we decided to use three-layer architecture, as we have independent three modules that are developed and maintained separately. On the presentation level, we have user interface, game logic in the logic tier between presentation and data tiers. We have game object at the bottom as data tiers. These layers are explained in the 'Sub-System Decomposition' section with details.

In this model we also planned to keep maintaining easy and open for future renovations, like on the user interface without affecting the game logic and objects code. Opaque layering provides it for us. Also it helps us to plan three different work groups in the project group, and to maintain the implementation easier with lesser errors.

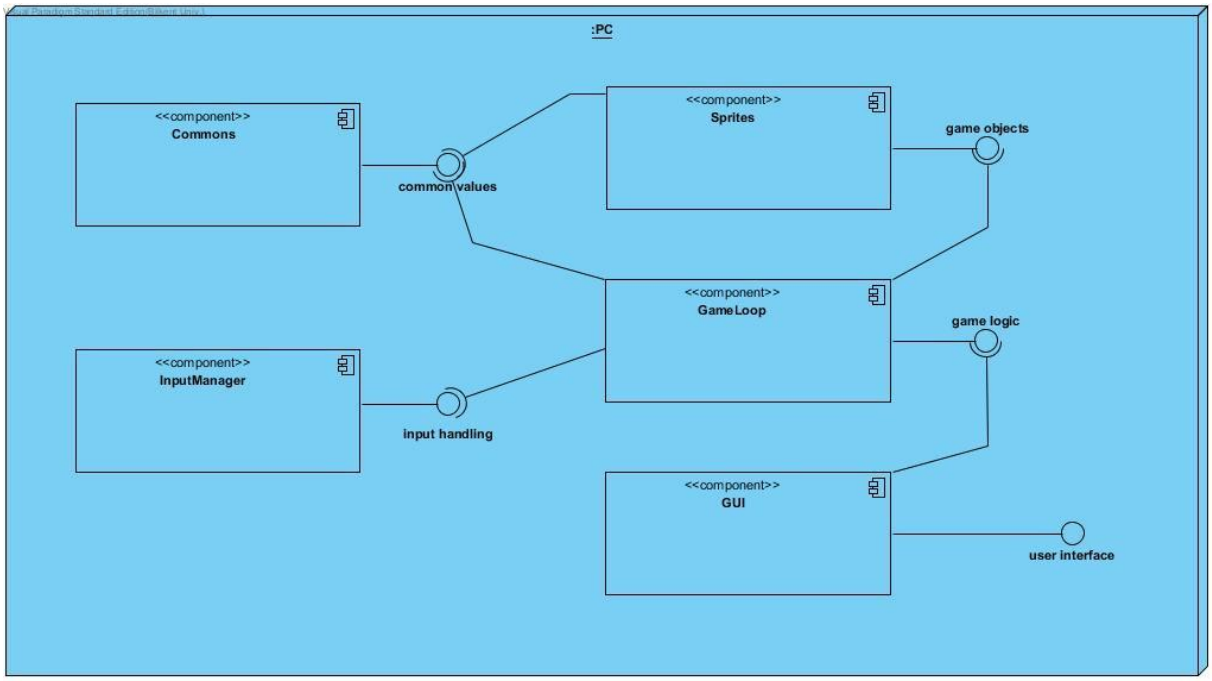
4.4. Hardware/Software Mapping

The game will require Java Runtime Environment to run since it is written in Java. The game will run on any operating system that supports Java and its components. The hardware requirements are a mouse and a keyboard. Online multiplayer or local multiplayer is not supported. The game is played offline therefore there is no need for a remote server. The game uses Java 2D graphics so a low end PC with Java Runtime Environment is sufficient to run the game properly.

4.4.1. Component Diagram



4.4.2. Deployment Diagram



Since the game runs on a single PC in offline mode, deployment diagram is pretty much the same with the component diagram.

4.5. Addressing Key Concerns

4.5.1. Persistent Data Management

'Alien Invasion' 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 encrypted in the '.jar' file, so that user could not modify the files. In the future, we also plan to provide another platform for the users where they can upload their own images to the system, and it will give them an executable '.jar' file with their selection of images and game elements.

4.5.2. Access Control and Security

'Alien Invasion' will not require any network connections, not have any authentication system and not store any user credentials. Game could be played by simply running the '.jar' file. For the security of game logic, classes like 'Board' and 'Sprite' includes private variables and also we have constant variables to ensure we have a secure information flow.

4.5.3. Global Software Control

Game has 'Manager' classes to handle with basic tasks (like 'LevelManager' handles the difficulty of each level) to reduce the complexity by decentralizing the game logic. Each object delegates some responsibility to another object, which is the main idea behind decentralization.

4.5.4. Boundary Conditions

- Initialization

'Alien Invasion' does not require a setup, but game could be opened through an executable '.jar' file.

- In-game Handling

Game has infinite number of levels, but it becomes impossible to win a round, because enemies are getting faster and faster. If user loses all of his health, game will return to the main menu unless user has a high score. In case of a high score, high scores will be updated with the name user provided at the start of the game.

- Termination

'Alien Invasion' 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 'Exit' button and terminated the game like the first case. In case of 'Alien Invasion' opened second time, first game will be terminated and user could lose all the current data.

- Error Handling

If game files without effect to the game logic is corrupted (like sound files), game will inform the user about the problem and than continue to run. If files that are corrupted are crucial for game logic, than game will open up just to inform user about downloading the executable '.jar' file again to play the game.

4.5.5. Concurrency

The game does not support multiplayer. Also there is only one controller class named Board that can access to the data layer. Therefore there is no risk of a concurrency error.

5. Conclusion

Our game “Alien Invasion” is inspired by the famous arcade game “Space Invaders”. Our game is not a subset of the original game but rather an expansion to it. We will adapt every major feature of the original game and will add new functionalities to make it more interesting.

While preparing the diagrams, we used the Visual Paradigm tool to help us. After creating diagrams and addressing the design goals, now we have a better understanding of the overall design of the project. With no doubt, this design report will contribute to us greatly during the implementation stage of the project.

6. References

[1] https://en.wikipedia.org/wiki/Space_Invaders