Bilkent University

Department of Computer Engineering

# Object-Oriented Software Engineering Project

*CS 319 Project: Sky Wars*

## Design Report

Ayşe Berceste Dinçer, Beril Başak Tukaç, Dilara Ercan, Sertaç Onur Hakbilen

Course Instructor: Hüseyin Özgür Tan
TA: İstemi Bahçeci

# *Table of Contents*

# *Table of Figures*

# *1. Introduction*

Sky Wars is a game of air battle where the Player is controlling a war plane which can shoot and be hit. Sky Wars is a level-based game. The Player gains access to next level after completing the current level. Along with regular levels, the User can also play Bonus Missions. The aim of the Player is to pass the level by reaching point threshold without depleting his health within limited level time.

In Sky Wars, various dangers and Targets exist. Levels have different themes and various Obstacles appear according to these themes. The game also contains Bonus Packages, some of them helpful for the User and some of them act as traps. Present Bonus Packages include speed or health boosts and time-limited powers such as Obstacle Invisibility and Invincibility. On the other hand, Trap Packages make the game harder for Player by decreasing health or speed. Along with Obstacles and Bonus Packages, various Targets exist. Any object with a certain health which can be shot by the Player is considered as a Target. Different Targets exist with various shapes, powers and movements. Target Planes shoot to Player with various Weapons. During the game, Obstacles, various Targets and Bonus Packages appear on the screen. Moreover, enemy planes fly and shoot to plane of the Player.

The Player tries to avoid colliding with Obstacles, Targets and Trap Packages. Collision with Trap Packages makes the game harder while colliding with Targets and Obstacles depletes the health of User and results in failing the level. Player collects Bonus Packages to use various boosts. The Player shall escape from Target Plane shots as well in order not to lose points. Furthermore, Player shoots Targets with Weapons to earn points. Each level has a certain time period and a point threshold. If the Player reaches the end of the level, his points are transformed to coins. These coins are used for the purchase of items.

Sky Wars has a Store within where the Player can purchase new items. The game offers options for UserPlane, each of them with various properties such as speed or health. There are also different Pilots which the Player can select. Bonus Packages Locks are also sold in the store for users to purchase in order to collect Bonus Packages during levels. Moreover, different Weapons with different damage amounts and Explosives exist. The Player can purchase various weapons and can use these weapons in the game for shooting Targets.

Sky Wars is a desktop application and it is controlled with keyboard.

# 2. Requirements Analysis

## 2.1. Overview

### 2.1.1. Gameplay & Control

In Sky Wars, the Player is controlling a plane, UserPlane. The aim of the game is to escape from colliding with Targets, Obstacles, Trap Packages and Weapons and also shoot to Targets and collect Bonus Packages. In the game, the User will be using up and down keys to get away from the Obstacles, Targets and also to collect Bonus Packages. For shooting, space key; for weapon change 'C' key will be used. In a Bonus Mission the User will use right and left keys instead of regular up and down keys or all four arrow keys according to Bonus Mission type.

### 2.1.2. Levelling

Sky Wars is a level-based game. Each level has a theme, an allocated time period, a point threshold, a coin coefficient and a list of GameObjects in which their coordinates, movements and appearance times are specified. Point threshold is for evaluating Player success. When the User reaches point threshold at the end of the level the level is successful. Coin coefficient on the other hand, is the amount of points necessary to obtain one Coin. For instance, if the coin coefficient is 4, at the end of the level for each 4 points the User will earn a coin. GameObjects within a Level are UserPlane, Pilot and Weapons of UserPlane, various Targets and Bonus Packages. The User purchases a UserPlane and a Pilot from Store and select one of those as preference from the Collection before level. User also purchases Bonus Package keys from Store. If a Bonus Package Lock is purchased, it will appear in the level. However the game decides how many Bonus Packages will appear and when they will appear. Moreover, the game decides which Target, TrapPackage or Obstacle types will appear during the game how many will appear.

There will be a Level Map, showing all the levels completed and uncompleted all through the path. The User will be able to play previous levels. When the game is started, the Player has access to Level 1 only. The User is allowed to access next level only when he successfully completes the current level. While the Player continues passing levels, levels become harder. The point threshold and coin coefficient increase while the levels become harder. Number of Targets, Obstacles and attacks on UserPlane increase and intensify.

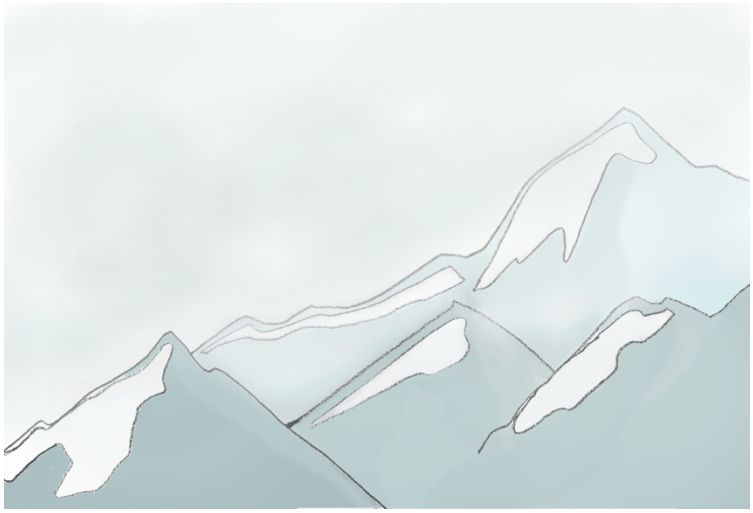

***Figure 1 Example Level Background Level 4 Glacial Area***

The game will contain 5 different levels for now with various themes and increasing difficulties:

- **Level 1** Field Area
- **Level 2** City Area
- **Level 3** Desert Area
- **Level 4** Glacial Area
- **Level 5** Forest Area

### *2.1.3. UserPlane*

UserPlane is the plane the Player is controlling. UserPlane is controlled by keyboard and it can fire Weapons. The UserPlane has below listed properties:

- **Speed** which corresponds to vertical speed in regular levels and horizontal or/and vertical speed in Bonus Missions. Basically speed corresponds to how fast the UserPlane reacts to keyboard input.
- **Health** represents the living power of the UserPlane. When health of the UserPlane is depleted, the game is over and the Player fails.
- **Shoot Damage** is the amount of damage the UserPlane can give to Target. When a Target is shot the total damage given to Target is the sum of the damage value of the Weapon and the Shoot Damage of the UserPlane.
- **Shooting Type** indicates whether the UserPlane has singular or double shoot which means that the UserPlane can send one or two Weapon at one time, when 'Space' key is pressed.

Sky Wars will include several UserPlanes. The UserPlanes differ in terms of above listed properties. The User will be able to use different UserPlanes in Levels as long as he purchased it previously from the Store and specify it as current selection from the Collection. Furthermore, the UserPlanes can be upgraded or weakened in terms of its speed, health, damage etc. when a BonusPackage is collected. UserPlanes do not have a specific Weapon type, they can shoot any Weapon the Player has purchased and selected. The UserPlane has a certain health and its health is decreased when UserPlane collides with a Weapon that is sent by TargetPlanes. The health of the UserPlane is depleted when it collides with an Obstacle or Target. There will be 5 different types of planes which are;

- **Alderaan Cruiser** which is the standard plane, having the standard health, speed and damage.

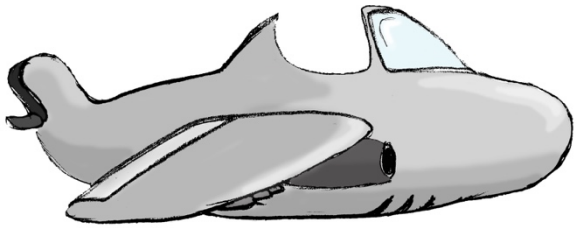

***Figure 2 Alderaan Cruiser UserPlane***

- **Tomcat** is faster than Alderaan Cruiser and its health is much more.

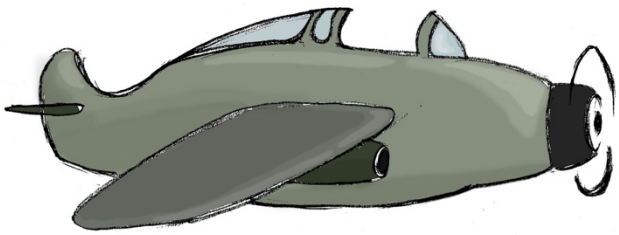

*Figure 3 Tomcat UserPlane*

- **F-22 Raptor** has the standard speed but it has 2 locations for shooting, has double shoot as shoot type, and the plane's default damage is higher than Alderaan Cruiser and Tomcat.

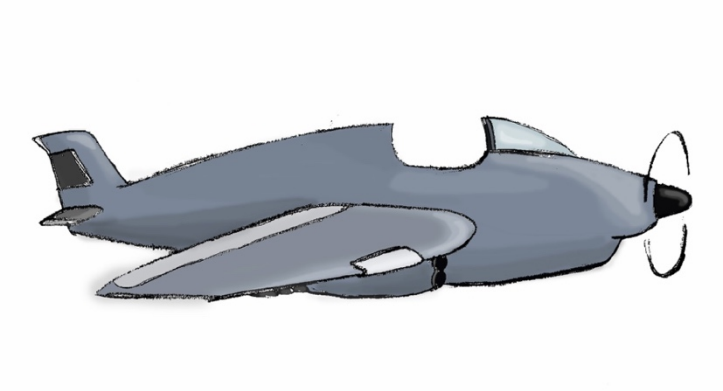

*Figure 4 F22-Raptor UserPlane*

- **Saab Gripen** is faster than F-22 Raptor, has a higher level of health and default damage compared to the previous planes and also possesses 2 shooting locations for shooting.

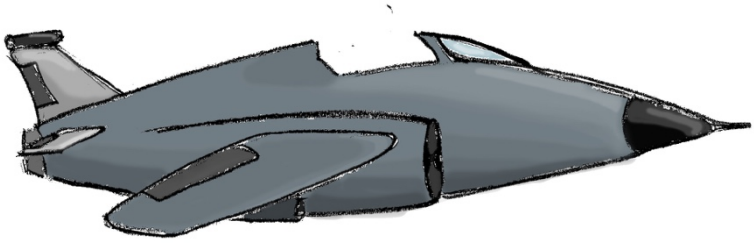

*Figure 5 Saab Gripen UserPlane*

- **Wunderwaffe** has the maximum capacity in all of the domains, rendering itself the most powerful UserPlane in the game.

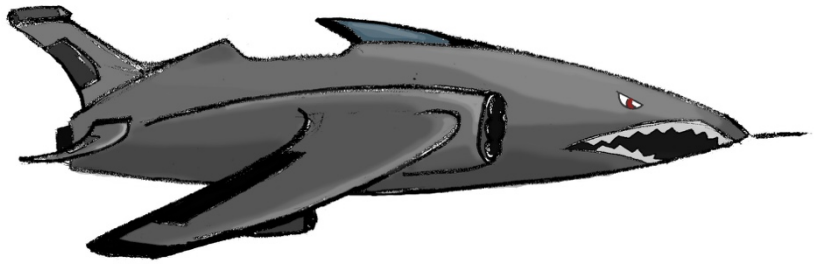

*Figure 6 Wunderwaffe UserPlane*

### *2.4.1. Target*

The Targets are the destructible objects in the game. They have a certain health value and can be shot by Player Weapons. When shot, Targets earn the User points. However, some kind of Targets is not supposed to be damaged and if they are shot, the User will lose points. Besides, Bonus Mission has also a corresponding Target which when shot, earns the Player the access to Bonus Mission. If UserPlane collides with any Target, the game is over and the Player fails.

#### 2.1.4.1. *TargetPlane*

TargetPlane is a Target type which can shoot to UserPlane and represented as enemy planes. TargetPlane has below listed properties:

- **Health** represents the living power of the TargetPlane. The health of the TargetPlane decreases when the Player shoots it with Weapons.
- **Shooting Time Period** represents the shooting speed or intensity of the TargetPlane. This property can be explained as the amount of seconds between consecutive shoots of the TargetPlane.
- **Weapon Type** is the Weapon that is associated with the TargetPlane. Contrary to UserPlane, TargetPlanes have a constant Weapon type which cannot be changed by the User and determined by the game itself.

TargetPlanes differ by these properties. Levels contain many TargetPlanes with various properties. Their paths (movement route), coordinates and appearance in the level will be specified beforehand for a level. If they get shot their health will be decreased and additional points will be given to the User. Sky Wars contains 5 types of TargetPlanes:

- **F-16** has low health and its shooting period is long. Its Weapon type is Bullet.

*Figure 7 F-16 TargetPlane*

- **Republic Attack Cruiser** has medium health and long shooting period. Its Weapon type is Metal Ball.

*Figure 8 Republic Attack TargetPlane*

- **Imperial Shuttle** has medium health and medium shooting period time. Its Weapon type is Flame Gun.

*Figure 9 Imperial Shuttle TargetPlane*

- **Havoc Marauder** has the maximum health with low shooting period time. Its weapon is Frost Laser.

*Figure 10 Havoc Marauder TargetPlane*

- **BOSS** has the maximum health and minimum shooting period. Its Weapon type is Laser.

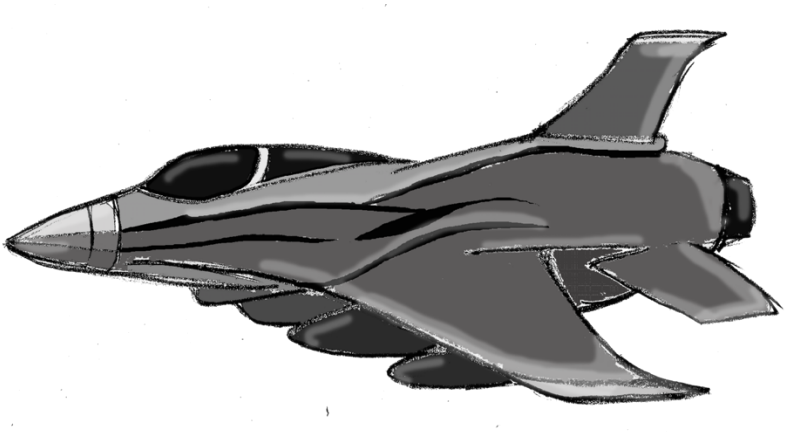

*Figure 11 BOSS TargetPlane*

#### 2.1.4.2. *Rocket*

Rocket is a different type of Target which explodes when its health is depleted. Rockets create an explosion within a specified area and gives damage to all GameObjects within the area. However, if the UserPlane in that area its health is also decreased or even depleted sometimes. Hence, Rocket should be destroyed by the Player when it is far from the UserPlane to earn maximum possible points. Rocket has two properties which are health and damage area that specifies the explosion area.

*Figure 12 Rocket*

#### 2.1.4.3. *Carriage*

Carriage is a Target which does not cause any damage to UserPlane, meaning it cannot shoot. When User plane destroys carriage it gains the Player points. The only property of carriage is Health.

#### 2.1.4.4. *Ally*

Ally is a harmless plane in Sky Wars. It only passes from aerospace of UserPlane. The only thing that User should do is not to shoot the Ally Plane. If User hits an Ally, User's points decrease according to the Ally type. The Ally Planes are the same as UserPlanes and they are driven by Pilots other than the current Pilot of UserPlane. There are 5 different Ally types. The health of Ally planes increase gradually in below list:

- **Alderaan Cruiser Ally**

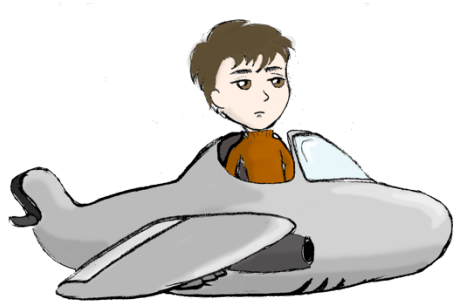

*Figure 13 Alderaan Cruiser Ally*

- **Tomcat Ally**

*Figure 14 Tomcat Ally*

- **F-22 Raptor Ally**

*Figure 15 F22-Raptor Ally*

- **Saab Gripen Ally**

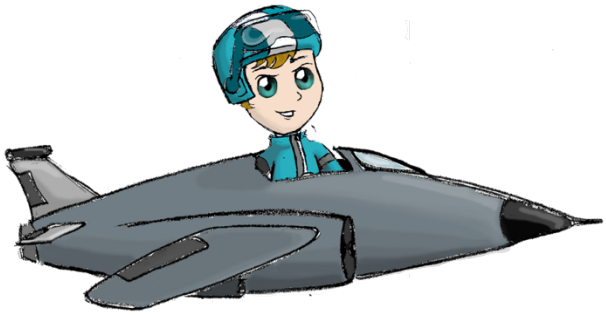

*Figure 16 Saab Gripen Ally*

- **Wunderwaffe Ally**

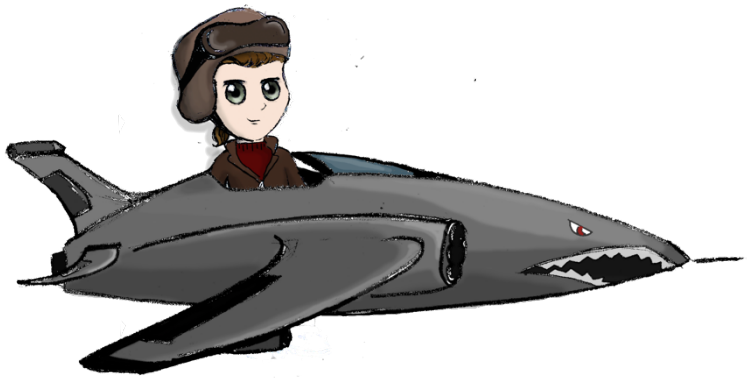

*Figure 17 Wunderwaffe Ally*

### 2.1.4.5. *Bonus Mission Target*

Bonus Mission Target is used for unlocking the Bonus Mission in the Level Map. In every chapter there is only one Bonus Mission Target and User should destroy this Target to open access to Bonus Mission. The only property of Bonus Mission Target is health. However relative to its rareness it has a high health value.

### 2.1.4.6. *Ship*

Ship has a no fire power and it only appears in Bonus Missions. User plane can damage Ship by throwing Torpedos from above. Ships are naturally in water, ground level.

## 2.1.5. *Bonus Package*

Bonus Packages are exceptions to regular game flow. When UserPlane collides with a Bonus Package, the collided bonus is applied. There are two types of Bonus Packages. Present Bonus Packages enhance the game while Trap Bonus Packages make the game harder. The Player has to purchase keys to Present Bonus Packages in order for them to appear during level. There are three levels of Present Bonus Package Locks in Store. The higher level keys create Packages with higher power during level. Trap Bonus Packages are not sold in Store and they can appear in any Level. The appearance time, number and coordinates of the Bonus Packages are determined by the game itself. However, whether a PresentBonusPackage will occur during

levels and how powerful it will be depend on whether the Package Lock is purchased by the user and the level of the Bonus Package Lock.

### 2.1.5.1. *Present Bonus Packages*

*Figure 18 Present Bonus Package*

- **Health Bonus Package**

When UserPlane collects these Packages, the health of the UserPlane is increased by the amount specified in the Bonus Package. There are three type of health Packages, each of them contribute to health of the UserPlane with different values.

- **Speed Bonus Package**

Speed Bonus Packages increase UserPlane speed by the amount specified in the Bonus Package. However, these Bonus Packages have a time limit. When the time period ends UserPlane's speed get back to its original level. There are three types of speed Packages and these Bonus Packages add different amounts of speed to UserPlane.

- **Shoot Damage Bonus Package**

When UserPlane collects Shoot Damage Packages, these Packages boost UserPlane's damage points according to corresponding Package value. These Packages have a time period and when it finishes the User plane's damage returns to its default value. There are three types of Packages and each of them increases the Shoot Damage by different amounts.

- **Time Bonus Package**

Time Bonus Package increases remaining time allocated for the level or Bonus Mission. It also has 3 types and these Packages add different amounts of time to level time. These Bonus Packages can help User collect more points until the end of the mission.

- **Obstacle Invisibility Package**

Obstacle Invisibility Package helps User to avoid from Obstacles in a time period. Obstacles become invisible and User plane can move more easily. This bonus also has a certain time period and until the end of this period the health of the UserPlane is not changed when it hits an Obstacle. There are three types of Obstacle Invisibility Package with varying effect periods.

- **Invincibility Package**

UserPlane does not get any damage when it collects Invincibility Package. This Package also has a certain time period and during this period the health of the UserPlane stays the same. When the time period finishes the UserPlane returns to its default state. There are three types of Invisibility Package with varying effect periods.

- **Coin Package**

UserPlane can collect this Package to increase the amount of Player coins. There are three types of Coin Packages, each of them earning User different amount of coins. Coin Package Locks are not solved in Store. They only appear in Bonus Missions.

### 2.1.5.2. *Trap Bonus Packages*

*Figure 19 Trap Bonus Package*

- **Plane Enlarge Package**

This Package increases the area of the UserPlane. If the UserPlane collides with this Package it can get damage easily because of its surface area. UserPlane returns its original state after the time period of this Package finishes. Plane Enlarge Package has only one type.

- **Health Trap Package**

These Packages decrease the health of UserPlane when it is collected. It also has three types like Health Bonus Packages.

- **Speed Trap Package**

These Packages decrease the speed of UserPlane when it is collected. UserPlane returns to its original speed when the time period of Speed Trap Package finishes. These Packages have also three types each of them decreasing speed by a certain amount just like Speed Bonus Packages.

- **Damage Trap Package**

These Packages decrease UserPlane's damage according to its type. There are three types of Damage Trap Packages with various levels of damage decrease. Similar to Damage Bonus Packages, they have a certain effect period. At the end of this period, the damage of the UserPlane returns to default value.

### 2.1.5.2. *Present Bonus Package Locks*

In order for a Present Bonus Package to appear during level, the Player has to purchase Bonus Package Lock from Store. Bonus Package Locks, when purchased, allow Present Bonus Package to appear during level. There are corresponding Bonus Package Locks for Health, Speed, Shoot Damage, Time Bonus, Obstacle Invisibility and Invincibility Bonus Packages. Package Locks have three levels. When the Player purchases a Lock for the first time, the level of the Lock becomes one and the level increases up to three with each purchase. According to the level of the lock, the power of the Bonus Package in level increase. There are three variations to all Present Bonus Packages and three levels of Locks corresponding to these variations. For instance if the Player has purchased Health Bonus Package Lock Level 2, then Health Bonus Packages with medium health value appear during levels.

## 2.1.6. *Obstacle*

Obstacles are bumps that appear in the Level View. They do not move. When UserPlane collides with an Obstacle its health is depleted and the game is over. The appearance times and coordinates of Obstacles are determined by the System.

*Figure 20 Obstacle Example Iceberg*

There are six different Obstacles:

- **Cloud**
- **Tree**
- **Iceberg**
- **Sand Hill**
- **Pyramid**
- **Building**

These Obstacles are adapted to Level themes. For instance, in City themed Level, Buildings appear as Obstacles.

## 2.1.7. *Pilot*

Pilots are the characters that represent the Player. Pilot drives the UserPlane. It has speed property. Different characters have different speed capabilities. The total speed amount of UserPlane is calculated as the sum of the speed of UserPlane and speed of Pilot. Pilots can also be purchased from Store and selected as the current choice from Collection before the level. There are 5 pilots in Sky Wars, 2 girls and 3 boys.

- **Nick** is a boy. He is the default Pilot and has 0 speed

*Figure 21 Pilot Nick*

- **Penny** is a girl and has low speed

*Figure 22 Pilot Penny*

- **Mike** is a boy and has low speed

*Figure 23 Pilot Mike*

- **Eva** is a girl and has high speed

*Figure 24 Pilot Eva*

- **Neo** is a boy and has high speed

*Figure 25 Pilot Neo*

## 2.1.8. *Bonus Mission*

If the Player shoots Bonus Mission Target in a Level, Bonus Mission is opened in Level Map. In Bonus Missions, there are not any Obstacles and the UserPlane is not attacked. The Player is allowed to play a Bonus Mission only once and then the Bonus Mission button is removed from the Level Map until the Player shoots and destroys another Bonus Mission Target. The health of UserPlane cannot be depleted during Bonus Mission since the Player is not attacked. Hence, there is no concept of success or fail for a Bonus Mission. The Player only aims to earn as many points as he can. There are 2 types of Bonus Missions:

- **Shooting Warships**

In this Bonus Mission Player tries to destroy Ship objects by shooting Torpedos from above. In this Bonus Mission UserPlane can only move to left or right. The Player can shoot only Torpedos and cannot change Weapon type.

- **Coin Heaven**

In Coin Heaven Player tries to collect as many Coin Packages as possible until the end of the mission. UserPlane can be directed to all four directions within this mission. The Player cannot shoot hence cannot change Weapon.

## 2.1.9. *Weapon*

Planes, both UserPlane and TargetPlane use Weapons to give damage to the GameObjects. Weapons cannot exist on their own, they are either send by the TargetPlanes or the UserPlane. User must purchase Weapons from store in order to use it within levels. Weapon purchase is done with amounts; Player specifies the amount of a Weapon type when he tries to purchase it. The Player has infinitely many standard type Weapon, Bullet and he does not need to purchase it. During Level, Player can change Weapon with ‘C’ key and he can use different Weapon types as long as the Weapon is left. There are basically two types of Weapons, Shoot and Explosive.

### 2.1.9.1. *Shoot*

Shoot is a type of Weapon which gives damage to only the GameObject it collides with. Shoot has damage property which represents the amount of health it deduces from GameObjects. There are 5 different Shoot types with different damage values. They are listed from lowest to highest damage amount:

- **Bullet**

*Figure 26 Bullet*

- **Metal Ball**

*Figure 27 Metal Ball*

- **Flame-gun**

*Figure 28 Flame-gun*

- **Frost-laser**

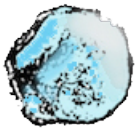

*Figure 29 Frost-laser*

- **Laser**

*Figure 30 Laser*

- **Torpedo** is an additional type of Shoot which is not sold in the Store and can only be used in Bonus Missions.

*Figure 31 Torpedo*

### 2.1.9.2. *Explosive*

Explosives create explosion when they collide with a Target. They have damage and damage area properties. They decrease the health of all GameObjects within their damage area. Explosives have 2 types:

- **Bomb** has small damage value and small damage area

*Figure 32 Bomb*

- **Missile** has larger damage value and larger damage area

*Figure 33 Missile*

## 2.1.10. *Store*

User can use coins, which are collected in levels and Bonus Missions in Sky Wars' Store. In Store, User can purchase UserPlanes, Weapons, Present Bonus Package Locks and Pilots if he has sufficient amount of coins. The Store locks the access to GameObjects the Player cannot afford. Moreover, when the User tries to purchase a Weapon he is required to specify the amount of the Weapon. When the User cannot afford specified amount of Weapons he is notified. After purchase is completed, the User can view all purchased items in Collection. Purchased Bonus Packages can be seen in levels after purchase. All purchased Weapons can be used within levels as well.

## 2.1.11. Collection

In Collection, the User will be able to see all purchased items UserPlane, Pilot, Weapons and Bonus Package Locks. The Player is also able to select a UserPlane and Pilot from Collection to use during level. When the Player has not purchased any items, default UserPlane and Pilot is used in levels.

## 2.1.12. Scoring

The scoring system is used to measure the performance of the Player. When a Weapon hits UserPlane, the health of UserPlane is decreased by the amount of Weapon damage. The same amount of points is deduced from total amount of points within level. When the Player sends a Weapon and this Weapon collides with an object (or objects if the Weapon is an Explosive) the health of the Target is decreased by the sum of Weapon damage and UserPlane shoot damage. The same amount of points is added to total amount of points. The point for a level starts from 0 and cannot drop below 0. The level may end with 3 different situations:

- **Level Failed with Depleted Health:** When the UserPlane collides with a Target or Obstacle or its health is depleted because of multiple shots, the game is over. The Level is unsuccessful. However, the amount of points collected so far is turned into Coins according to Level coefficient.

- **Level Failed with Threshold Unreached:** When the Player completes the level but cannot reach threshold points for the level, the level is unsuccessful. Still, the amount of points collected so far is turned into Coins according to Level coefficient.

- **Level Passed:** When the Player completes the level and reaches threshold points for the level, the level is passed. The amount of points collected so far is turned into Coins according to Level coefficient. The Player gains access to next level.

## *2.2. Functional Requirements*

### *2.2.1. Playing Level*

- The Player should be able to move UserPlane up or down
- The Player must be able to shoot, fire Weapons
- The Player should be able to change Weapon type
- The System should notify the Player when a certain Weapon type is depleted and UserPlane cannot shoot
- The System should decide the background and map of game objects according to the selected level
- The System should place Obstacles to certain locations in the screen
- The System should place Trap Bonus Packages to certain locations in the screen
- The System should place Present Bonus Packages that the Player has purchased to certain locations in the screen
- The System should display and move various Targets according to the level the Player is playing
- The System must send Weapons to UserPlane via Target Planes
- The System must display the time left on the screen
- The System must display the health left on the screen
- The System should display the weapon types and the number of the left weapons relatively on the screen
- The system should end the game and notify the Player when a collision occurs between the UserPlane and an Obstacle or a Target.
- The System should create an explosion animation when an Explosive Weapon is used or a rocket is destroyed
- The System should update the health of a Target when it is hit
- The System must update the Player health when the UserPlane is hit
- The System should create a destruction animation when health of a Target is depleted
- The System should update the time each second
- The System must update the points after each collision
- The System should notify the User about the change in points or health after each collision
- The System should apply the respective bonus when a Bonus Package is collected
- The System must notify the Player about the current Bonus Package
- The System should notify the Player if the level is failed
- The System should notify the Player if he has completed the level
- The System should check the total amount of points collected during level and decide whether the level is successful or not
- The System should notify the Player whether the level is successful or not
- The System should turn the points into coins according to the level coefficient at the end of the level
- The System must notify the Player about the total points and coins earned at the end of the level

### 2.2.2. *Playing Bonus Mission*

- The Player should be able to move the UserPlane left-right and/or up-down
- The Player should be able to collect Coin Packages or shoot Ships with Torpedoes, according to the Bonus Mission definition
- The System should load the background image and game objects according to the Bonus Mission theme
- The System should not allow the Player to play the same Bonus Mission more than once
- The System should update the time each second
- The System should update points after each collision
- The System must not consider or update health of the Player during Bonus Mission
- The System should turn the total points earned during Bonus Mission into coins
- The System should upload coins earned during Bonus Mission directly to Player account
- The System should notify the Player about point updates after each collision
- The System should notify the Player about total points and coins earned after the Bonus Mission completion
- The System should open access to Bonus Mission only if the Bonus Mission Target is destroyed during levels
- The System should lock access to Bonus Mission after it is played once until the access is earned again

### 2.2.3. *Pausing The Game*

- The Player should be able to pause the game while playing any level or Bonus Mission
- The System should pause the game, stop all moving game objects and time when relative button is clicked
- When the game is paused the System must display pause menu
- The Player should be able to continue to play the game by pressing relative button in the Pause Menu

### 2.2.4. *Viewing Store and Purchasing Items*

- The Player must be able to view Store
- The System should display all User Planes, Pilots, Bonus Package Locks and Weapons present in the game
- The System should lock the items the Player cannot afford
- The System should highlight the items the Player can afford
- The System should indicate an already purchased item and should not allow it to be purchased again
- The Player must be able to view details of an available item
- The Player should be able to select an item to purchase
- If the Player selects a Weapon to purchase the System should obtain the amount from the Player
- The Player should be able to specify Weapon amount if he wants to purchase a Weapon
- The System should calculate the total price of selected items
- The System must notify the Player if the total amount exceeds the number of Player coins
- The System should notify the Player when purchase is completed
- The System should update Player Collection after a purchase
- The System should update the Store view after a purchase

### 2.2.5. *Viewing Collection and Changing Preferences*

- The Player must be able to display Collection
- The System should display all purchased items in the Collection
- The System should specify the amount of existing Weapons
- The Player should be able to change UserPlane or Pilot selections
- The System should update the game if the Player changes item selections

### 2.2.6. *Viewing Level Map*

- The System should display all levels the User has completed
- The System must highlight the current level, the level the User is required to pass in order to open access to next level
- The System should display Bonus Mission if the Player has opened access to it
- The System has to disable access to Bonus Mission after the Player plays it once
- The Player should be able to select any level on Level Map to play
- The System should start the selected level or Bonus Mission
- The System should update the Level Map after completion of a level or Bonus Mission
- The System should update the Level Map after the User destroys a Bonus Mission Target in a level

### 2.2.7. *Changing Settings*

- The System should display the current settings to User
- The Player must be able to turn the music on or off
- The Player must be able to alter volume level
- The System should record and update changed settings

### 2.2.8. *Viewing Help*

- The System should display Help page and provide tutorials to the Player
- The System should specify basic concepts of the game
- The System must demonstrate User controls
- The System should explain Store and Collection operations

### 2.2.9. *Viewing Credits*

- The System should display credits

### 2.2.10. *Playing Sound*

- The System should play music during levels and Bonus Missions
- The System system should play music while the Player is on Menus and other pages
- The System should play animation sounds during gameplay when collision occurs

## *2.3. Non-Functional Requirements*

### 2.3.1. *Usability*

**User-friendliness:** The System should be simple and aim user satisfaction

- The System should use meaningful names for buttons and icons in order to ease navigation

**Ease of Use:** The system should be easy to understand and use
**Understandability:** The game itself and its documentation should be understandable

### 2.3.2. *Performance*

**High Performance:** The system should respond to user input immediately
**Rapid Development:** The system should be quickly developed
**Efficiency:** The system should be efficient, operations should be implemented to maximize usage and minimize time cost
**Functionality:** The system should include many functionalities to satisfy user

### 2.3.3. *Reliability*

- The System should automatically save the progress of the Player after the end of each level
- The System should not loose the Player progress in a power-loss situation
- The System should not loose the Player progress if the System crushes

**Traceability of Requirements:** The System development should be coherent with the requirements

### 2.3.4. *Supportability*

**Modifiability:** The System should be open to development, it should be possible to add new features to the game
**Flexibility:** The System should be flexible to adaptations
**Maintainability:** The System can be changed to adapt to new technology
**Adaptability:** The System should be able to deal with additional application domain concepts

## 2.4. *Constraints*

- The System shall be implemented in Java
- Adobe Photoshop shall be used in the design of game graphics
- Bohemian Coding Sketch shall be used in the design of game screens

## 2.5. *Scenarios*

***Scenario 1***

**Use Case Name:** OpenLevelMapToPlayLevel1
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Main Menu

**Exit Conditions:**

- Player Ali is playing Level 1

**Main Flow of Events:**

1. Player Ali presses Level Map button
2. Sky Wars displays Level Map page
3. Player Ali presses Level 1 button
4. Sky Wars starts Level 1

***Scenario 2***

**Use Case Name:** CompleteLevel1
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Level Map page

**Exit Conditions:**

- Player Ali completes Level 1 AND
- Player Ali is on Level Map page

**Flow of Events:**

1. Player selects Level 1 to play from Level Map
2. Sky Wars initializes Level 1
3. Player Ali plays the game
   **3a.** Player moves the UserPlane up or down

**3b.** Sky Wars handles collisions
**3c.** Player shoots
**3d.** Sky Wars handles shoot
**3e.** Sky Wars controls and updates time, points and health of the Player

4. Player Ali completes the level with 1000 points
5. Sky Wars checks whether Player has reached Level 1 threshold, 800 points
6. Sky Wars declares Level 1 as successful since Player Ali has reached threshold
7. Sky Wars turns points into coins
8. Sky Wars directs Player Ali to Level Map
9. Sky Wars displays Level 2 button as well in Level Map

## ***Scenario 3***

**Use Case Name:** FailLevel2
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Level Map page

**Exit Conditions:**

- Player Ali fails Level 2 AND
- Player Ali is on Level Map

**Flow of Events:**

1. Player selects Level 2 to play
2. Sky Wars initializes Level 2
3. Player Ali plays the game
   **3a.** Player moves the UserPlane up or down
   **3b.** Sky Wars handles collisions
   **3c.** Player shoots
   **3d.** Sky Wars handles shoot
   **3e.** Sky Wars controls and updates time, points and health of the Player
4. Player Ali collides with an Obstacle
5. Sky Wars ends the game
6. Sky Wars declares Level 2 as unsuccessful
7. Sky Wars turns points into coins
8. Sky Wars directs Player Ali to Level Map

## ***Scenario 4***

**Use Case Name:** FailLevel2CannotReachTreshold
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Level Map page

**Exit Conditions:**

- Player Ali fails Level 2 AND
- Player Ali is on Level Map

**Flow of Events:**

1. Player selects Level 2 to play
2. Sky Wars initializes Level 2
3. Player Ali plays the game
   **3a.** Player moves the UserPlane up or down
   **3b.** Sky Wars handles collisions
   **3c.** Player shoots

**3d.** Sky Wars handles shoot
**3e.** Sky Wars controls and updates time, points and health of the Player

4. Player Ali completes the level with 1500 points
5. Sky Wars checks whether Player has reached Level 2 threshold, 2000 points
6. Sky Wars declares Level 2 as unsuccessful since Player Ali has not reached threshold
7. Sky Wars turns points into coins
8. Sky Wars directs Player Ali to Level Map

## ***Scenario 5***

**Use Case Name:** PlayShipBonusMission
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali has opened Bonus Mission in previous levels AND
- Player Ali in on Level Map

**Exit Conditions:**

- Player Ali is on Level Map

**Flow of Events:**

1. Player Ali selects Bonus Mission to play
2. Sky Wars initializes Bonus Level which has a theme of destroying below ships with torpedos
3. Player Ali plays the Bonus Mission
   **3a.** Player Ali moves UserPlane left and right
   **3b.** Player Ali shoots Torpedos
   **3c.** Sky Wars updates points
4. Player Ali completes the Bonus Mission
5. Sky Wars declares Bonus Mission as completed
6. Sky Wars turns points into coins
7. Sky Wars removes Bonus Mission button from Level Map
8. Sky Wars directs User to level map

## ***Scenario 6***

**Use Case Name:** PauseGameFor5MinutesAndContinue
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is playing Level 2

**Exit Conditions:**

- Player Ali is playing Level 2

**Flow of Events:**

1. Player Ali presses Pause button
2. Sky Wars pauses the game screen
3. Sky Wars displays Pause Menu
4. Sky Wars stays in pause mode for 5 minutes
5. Player Ali selects to continue level
6. Sky Wars continues the game, Level 2

## ***Scenario 7***

**Use Case Name:** PauseGameLearnAboutBonusPackagesAndContinue
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is playing Level 2

**Exit Conditions:**

- Player Ali is playing Level 2

**Flow of Events:**

1. Player Ali presses Pause button
2. Sky Wars pauses the game screen
3. Sky Wars displays Pause Menu
4. Player Ali presses Help button from Pause Menu
5. Player Ali views tutorials about Bonus Packages
6. Player Ali selects to return to Pause Menu
7. Player Ali selects to continue level
8. Sky Wars continues the game, Level 2

## ***Scenario 8***

**Use Case Name:** TurnOffMusic
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Main Menu

**Exit Conditions:**

- Player Ali is on Main Menu

**Main Flow of Events:**

1. Player Ali presses Settings button
2. Sky Wars displays Settings page
3. Player Ali views relative buttons for turning the music on or off
4. Player Ali turns music off
5. Player presses Main Menu button
6. Sky Wars updates the settings
7. Sky Wars displays Main Menu or Pause Menu

## ***Scenario 9***

**Use Case Name:** SetVolumeLevelToMaximum
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Main Menu

**Exit Conditions:**

- Player Ali is on Main Menu

**Main Flow of Events:**

1. Player Ali presses Settings button
2. Sky Wars displays Settings page
3. Player Ali views relative buttons for altering volume level
4. Player Ali sets volume to maximum
5. Player presses Main Menu button
6. Sky Wars updates the settings
7. Sky Wars displays Main Menu or Pause Menu

## ***Scenario 10***

**Use Case Name:** LearnHowToHandleStoreOperations
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Main Menu

**Exit Conditions:**

- Player Ali is on Main Menu

**Main Flow of Events:**

1. Player Ali presses Help button
2. Sky Wars displays Help page
3. Player Ali examines tutorials about Store operations
4. Player presses Main Menu button
5. Sky Wars displays Main Menu

## ***Scenario 11***

**Use Case Name:** CheckAffordableWeapons
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Main Menu

**Exit Conditions:**

- Player Ali is on Store page

**Main Flow of Events:**

1. Player Ali presses Store button
2. Sky Wars locks the items on Store that Player cannot afford
3. Sky Wars displays Store page
4. Player Ali views unlocked Weapons and their details

## ***Scenario 12***

**Use Case Name:** PurchaseUserPlaneF22Raptor
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Store page

**Exit Conditions:**

- Player Ali is on Main Menu

**Flow of Events:**

1. Sky Wars displays Store
2. Player Ali views unlocked UserPlanes
3. Player selects F22 Raptor UserPlane
4. Player presses Purchase button to complete purchase
5. Sky Wars handles purchase and updates Player Ali's Collection
6. Player presses Main Menu button
7. Sky Wars displays Main Menu

## ***Scenario 13***

**Use Case Name:** Purchase10FlameGuns
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Store page

**Exit Conditions:**

- Player Ali is on Main Menu

**Flow of Events:**

1. Sky Wars displays Store
2. Player Ali views unlocked Weapons
3. Player selects FlameGun

4. Sky Wars requests from Player Ali to enter number of flameGuns
5. Player Ali enters ‘20’
6. Player presses Purchase button to complete purchase
7. Sky Wars cancel purchase and notify User that there not enough coins
8. Player Ali reenters FlameGun amount, ‘10’ this time
9. Sky Wars handles purchase and updates Player Ali’s Collection
10. Player presses Main Menu button
11. Sky Wars displays Main Menu

## ***Scenario 14***

**Use Case Name:** ViewPurchasedWeapons
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Main Menu

**Exit Conditions:**

- Player Ali is on Main Menu

**Flow of Events:**

1. Player Ali presses Collection button
2. Sky Wars displays Collection page with the purchased items
3. Player Ali views purchased Weapons
4. Player Ali presses Main Menu button
5. Sky Wars displays Main Menu

## ***Scenario 15***

**Use Case Name:** ChangeSelectedPlaneToF22
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Collection page

**Exit Conditions:**

- Player is Ali on Main Menu

**Flow of Events:**

1. Sky Wars displays Collection page with the purchased items
2. Sky Wars highlights current UserPlane and Pilot selections
3. Player Ali changes current UserPlane to F22
4. Sky Wars updates Player selection
5. Player presses Main Menu button
6. Sky Wars displays Main Menu

## ***Scenario 16***

**Use Case Name:** DisplayNamesOfDevelopers
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Main Menu

**Exit Conditions:**

- Player Ali is on Main Menu

**Flow of Events:**

1. Player Ali presses Credits button
2. Sky Wars displays Credits page
3. Player Ali views names of developers

4. Player Ali presses Main Menu button
5. Sky Wars displays Main Menu

***Scenario 17***

**Use Case Name:** QuitGameFromMainMenu
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is on Main Menu

**Exit Conditions:**

- Player Ali exited the game

**Flow of Events:**

1. Player Ali presses Quit button on Main Menu
2. System displays a dialog box to make sure Ali wants to quit
3. Player Ali presses Yes button
4. System is exited

***Scenario 18***

**Use Case Name:** QuitGameDuringLevel
**Actors:** Player Ali
**Entry Conditions:**

- Player Ali is playing Level 2

**Exit Conditions:**

- Player Ali exited the game

**Flow of Events:**

1. Player Ali presses Pause button
2. System displays Pause Menu
3. Player Ali presses Quit button
4. System displays a dialog box to make sure Ali wants to quit
5. Player Ali presses Yes button
6. System is exited

## 2.6. *Use Case Models*

Use cases of Sky Wars are represented with the use case diagram. Verbal descriptions of use cases are included below.

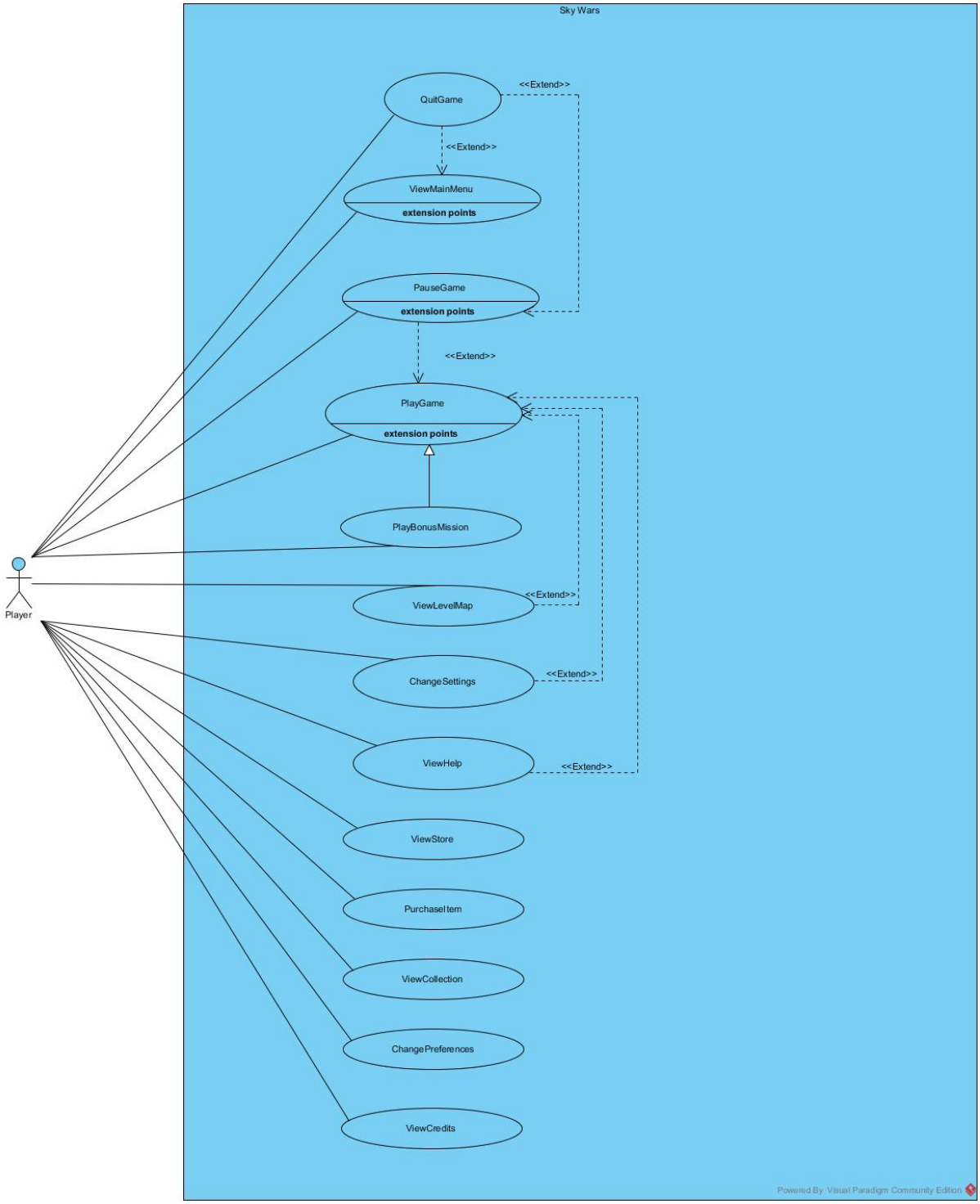


***Figure34 Use Case Diagram of Sky Wars***

**ViewMainMenu:** Player can request to view Main Menu from all screens of Sky Wars
**ViewLevelMap:** Player can request to view Level Map to select a level to play
**ChangeSettings:** Player can view Settings page and change game Settings
**ViewHelp:** Player can view Help page to learn how to play the game
**ViewStore:** Player can view Store page to examine items that are open to purchase
**PurchaseItem:** Player can select an item from Store and purchase it
**ViewCollection:** Player can request to view Collection page and examine list of all purchased items
**ChangePreferences:** Player can change Pilot or UserPlane selections from Collection View.
**ViewCredits:** Player can request to view Credits page to see developer details
**PlayGame:** Player can request to play the game levels by selecting a level from Level Map page
**PauseGame:** Player can request to pause the game while playing the game
**PlayBonusMission:** As a part of the gameplay, Player can play Bonus Mission which appears on Level Map page.
**QuitGame:** Player can request to quit the game from Main Menu or Pause Menu

## 2.6.1. *View Main Menu*

**Use Case Name:** ViewMainMenu
**Actors:** Player
**Entry Conditions:**
- Player is on Store Screen OR
- Player is on Level Map Screen OR
- Player is on Collection Screen OR
- Player is on Settings Screen OR
- Player is on Credits Screen OR
- Player is on Help Screen OR
- Player is on Pause Menu

**Exit Conditions:**
- Player is on Main Menu

**Main Flow of Events:**
1. Player presses Main Menu button
2. Sky Wars displays Main Menu

## 2.6.2. *View Level Map*

**Use Case Name:** ViewLevelMap
**Actors:** Player
**Entry Conditions:**
- Player is on Main Menu OR
- Player is on Pause Menu

**Exit Conditions:**
- Player is on Level Map page OR
- Player is playing Level

**Main Flow of Events:**
1. Player presses Level Map button
2. Sky Wars displays Level Map page
3. Player views relative buttons for completed levels and current level
4. Player views Bonus Mission button

5. Player presses Main Menu button
6. Sky Wars displays Main Menu

**Alternative Flow of Events:**

**5A.** Player selects a level or Bonus Mission to play (Skip Step 5 and 6)
**6A.** Sky Wars starts the game

## 2.6.3. *Change Settings*

**Use Case Name:** ChangeSettings
**Actors:** Player
**Entry Conditions:**

- ChangeSettings use case extends PlayGame use case
- Player is on Main Menu OR
- Player is on Pause Menu

**Exit Conditions:**

- Player is on Main Menu OR
- Player is on Pause Menu

**Main Flow of Events:**

1. Player presses Settings button
2. Sky Wars displays Settings page
3. Player views relative buttons for setting volume level
4. Player views relative buttons for turning the music on or off
5. Player changes settings by turning music on or off
6. Player alters volume level
7. Player presses Main Menu or Pause Menu button
8. Sky Wars updates the settings
9. Sky Wars displays Main Menu or Pause Menu

**Alternative Flow of Events:**

**5A.** Player does not turn music on or off
**6A.** Player does not alter volume level

## 2.6.4. *View Help*

**Use Case Name:** ViewHelp
**Actors:** Player
**Entry Conditions:**

- ViewHelp use case extends PlayGame use case
- Player is on Main Menu OR
- Player is on Pause Menu

**Exit Conditions:**

- Player is on Main Menu OR
- Player is on Pause Menu

**Main Flow of Events:**

1. Player presses Help button
2. Sky Wars displays Help page
3. Player views tutorials to learn how to play Sky Wars
4. Player presses Main Menu button
5. Sky Wars displays Main Menu or Pause Menu

## 2.6.5. *View Store*

**Use Case Name:** ViewStore
**Actors:** Player
**Entry Conditions:**
- Player is on Main Menu

**Exit Conditions:**
- Player is on Main Menu OR
- Player is in Store Page

**Main Flow of Events:**
1. Player presses Store button
2. Sky Wars locks the items on Store that Player cannot afford
3. Sky Wars displays Store page
4. Player views UserPlane, Pilot, Weapon and BonusPackage items, some of them ready to be purchased and some of them locked
5. Player presses Main Menu button
6. Sky Wars displays Main Menu

**Alternative Flow of Events:**
**5A.** Player initiates PurchaseItem use case
**6A.** Sky Wars starts PurchaseItem operations

## 2.6.6. *Purchase Item*

**Use Case Name:** PurchaseItem
**Actors:** Player
**Entry Conditions:**
- Player is on Store page
- PurchaseItem use case extends ViewStore

**Exit Conditions:**
- Player is on Main Menu

**Main Flow of Events:**
1. Sky Wars displays Store
2. Player views UserPlane, Pilot, Weapon and Bonus Package Lock items, some of them ready to be purchased and some of them locked
3. Player selects Weapon item to purchase
4. Sky Wars asks multiplicity of Weapon
5. Player presses Purchase button to complete purchase
6. Sky Wars checks Player coins whether they are enough or not
7. Sky Wars completes purchase and updates Player collection
8. Player presses Main Menu button
9. Sky Wars displays Main Menu

**Alternative Flow of Events:**
**3A.** Player selects UserPlane, Pilot or Bonus Package to purchase (Skip 4,5 and 6)
**6A.** Sky Wars detect that Player coins are not enough
**7A**. Sky Wars indicate that purchase cannot be completed

## 2.6.7. *View Collection*

**Use Case Name:** ViewCollection
**Actors:** Player
**Entry Conditions:**
- Player is on Main Menu

**Exit Conditions:**
- Player is on Main Menu

**Main Flow of Events:**
1. Player presses Collection button
2. Sky Wars displays Collection page with the purchased items
3. Sky Wars highlights current UserWeapon and Pilot selections
4. Player views UserPlane, Pilot, Weapon and BonusPackage items
5. Player views current selections
6. Player presses Main Menu button
7. Sky Wars displays Main Menu

**Alternative Flow of Events:**
**6A.** Player initiates ChangePreferences use case
**7A.** Sky Wars starts ChangePreferences operations

## 2.6.8. *Change Preferences*

**Use Case Name:** ChangePreferences
**Actors:** Player
**Entry Conditions:**
- Player is on Collection page
- ChangePreferences use case extends ViewCollection use case

**Exit Conditions:**
- Player is on Main Menu

**Main Flow of Events:**
1. Sky Wars displays Collection page with the purchased items
2. Sky Wars highlights current UserWeapon and Pilot selections
3. Player views UserPlane, Pilot, Weapon and BonusPackage items
4. Player views current selections
5. Player changes current selection
6. Sky Wars updates Player selections
7. Player presses Main Menu button
8. Sky Wars displays Main Menu

**Alternative Flow of Events:**
**5A.** Player changes UserPlane selection
**5A.** Player changes Pilot selection
**5A.** Player does not chance preferences(Skip 5 and 6)

## 2.6.9. *View Credits*

**Use Case Name:** ViewCredits
**Actors:** Player
**Entry Conditions:**
- Player is on Main Menu

**Exit Conditions:**
- Player is on Main Menu

**Main Flow of Events:**
1. Player presses Credits button
2. Sky Wars displays Credits page
3. Player views credits to learn developer details
4. Player presses Main Menu button
5. Sky Wars displays Main Menu

## 2.6.10. *Play Game*

**Use Case Name:** PlayGame
**Actors:** Player
**Entry Conditions:**
- Player has selected a level from Level Map

**Exit Conditions:**
- Player completed level and returned to Level Map OR
- Player failed the level and returned to Level Map OR
- Player failed the level and requested to play again

**Main Flow of Events:**
1. Player selects a level to play
2. Sky Wars initializes level
3. Player plays the game
    - **3a.** Player moves the UserPlane up or down
    - **3b.** Sky Wars handles collisions
    - **3c.** Player shoots
    - **3d.** Sky Wars handles shoot
    - **3e.** Sky Wars controls and updates time, points and health of the Player
4. Player completes the level
5. Sky Wars checks whether Player has reached level threshold
6. Sky Wars declares level successful
7. Sky Wars turns points into coins
8. Sky Wars directs User to level map
9. Sky Wars opens access to next level

**Alternative Flow of Events:**
**4A.** Player depletes his health and fails the level (Skip 5)
**6A.** Sky Wars declares level unsuccessful
**9A.** Sky Wars does not open access to next level
**6A.** Sky Wars declares level unsuccessful since Player could not reach point threshold
**9A.** Sky Wars does not open access to next level

## 2.6.11. *Play Bonus Mission*

**Use Case Name:** PlayBonusMission
**Actors:** Player
**Entry Conditions:**

- PlayBonusMission use case inherits PlayGame use case
- Player has opened Bonus Mission in previous levels
- Player has selected Bonus Mission from Level Map

**Exit Conditions:**

- Player is on Level Map

**Main Flow of Events:**

1. Player selects Bonus Mission to play
2. Sky Wars initializes level
3. Player plays the Bonus Mission
4. Player completes the Bonus Mission
5. Sky Wars turns points into coins
6. Sky Wars removes Bonus Mission button from Level Map
7. Sky Wars directs User to Level Map

## 2.6.12. *Pause Game*

**Use Case Name:** pauseGame
**Actors:** Player
**Entry Conditions:**

- PauseGame use case extends PlayGame use case
- Player is playing a Level OR
- Player is playing a Bonus Mission

**Exit Conditions:**

- Player is on Level Map OR
- Player is playing the game
- Player changes Settings
- Player views Help page

**Main Flow of Events:**

1. Player presses pause button
2. Sky Wars pauses the game screen
3. Sky Wars displays Pause Menu
4. Player selects to continue level
5. Sky Wars continues the game

**Alternative Flow of Events:**

**4A.** Player presses Settings button
**5A.** Sky Wars displays Settings page
**4A.** Player presses Help button
**5A.** Sky Wars displays Help page

## 2.6.13. *Quit Game*

**Use Case Name:** QuitGame
**Actors:** Player

**Entry Conditions:**

- QuitGame use case extends PauseGame and ViewMainMenu use cases
- Player is on Pause Menu OR
- Player is on Main Menu

**Exit Conditions:**

- Player has exited the game

**Main Flow of Events:**

1. Player presses quit button
2. Sky Wars is exited
3. Player is directed to desktop

## 2.7. *User Interface*

### 2.7.1. *Navigational Path*

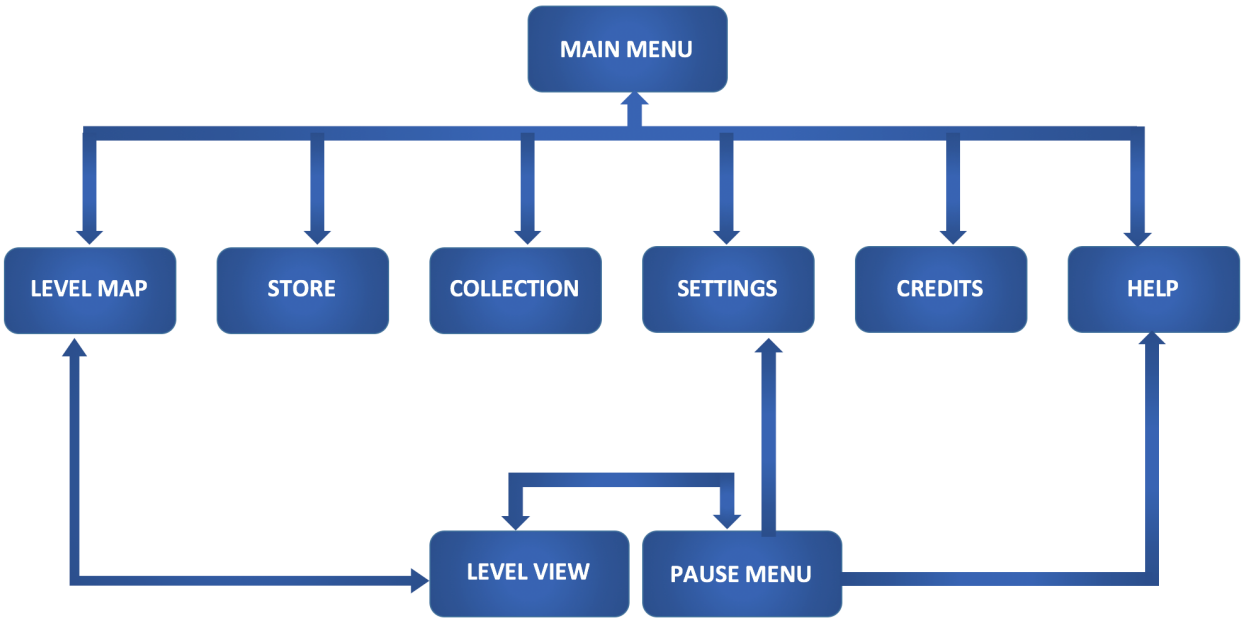


Figure 35 Navigational Path of Sky Wars

### 2.7.2. *Main Menu Screen*

Main Menu screen is the first screen that is displayed when the game is started. Main Menu leads User to ‘Level Map’, ‘Collection’ and ‘Store’ screens when relative buttons represented with clouds are clicked. Moreover, clicking the items in bubbles direct User to ‘Help’, ‘Settings’ and ‘Credits’ pages from top to bottom. The icon on right top of the page is for quitting the game.

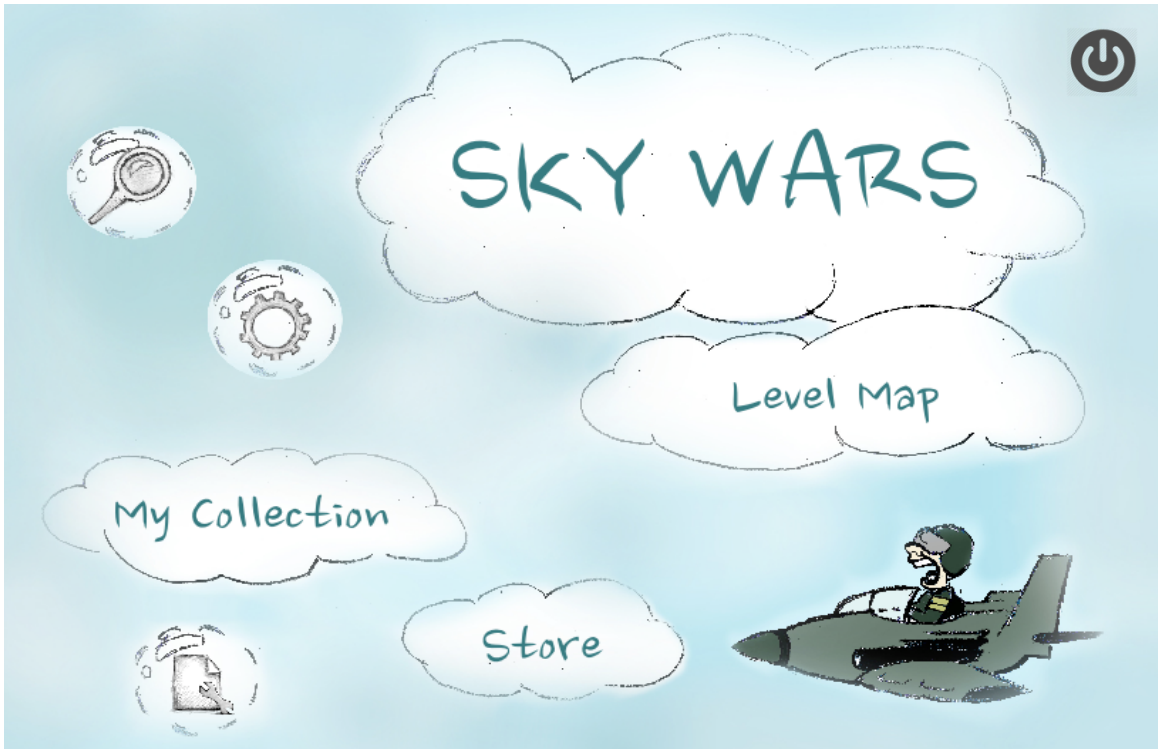


*Figure 36 Main Menu Screen*

## 2.7.3. *Level Map Screen*

Level Map screen displays all levels in Sky Wars. Current level, Level 5 is highlighted. Additionally, present icon can be clicked to play Bonus Mission. Home icon directs user to Main Menu.


*Figure 37 Level Map Screen*

### 2.7.4. *Level Play Screen*

Level Play Screen is an example of how the actual game looks like. A sample screen is demonstrated with Pilot and UserPlane on the left and other GameObjects around the screen. Pause button directs User to Pause Menu.

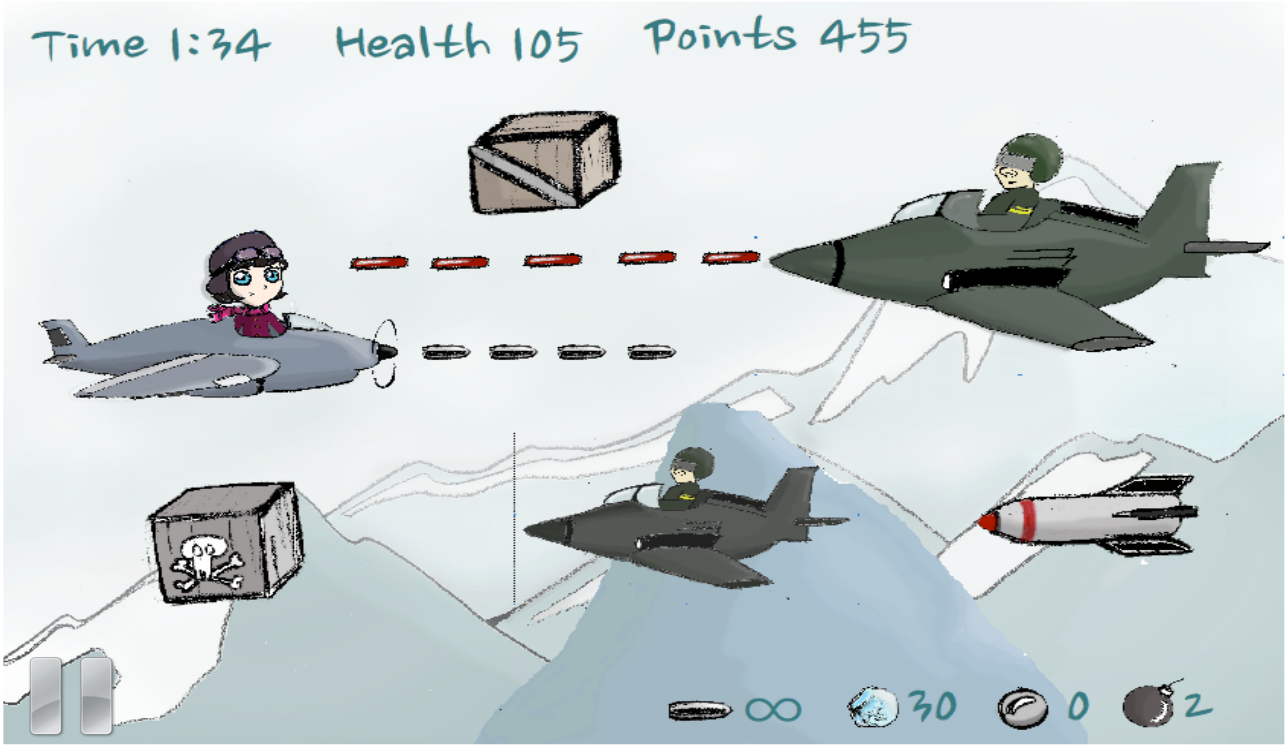


*Figure 38 Level Play Screen*

### 2.7.5. *Store Screen*

Store Screen lists all purchasable GameObjects hence it is a scrollable page. Below scene is the representation of the complete page. The detailed information and price of items are provided. The items Player cannot afford are locked. Moreover, the items that are already purchased are also unavailable. User coins are shown on the upper left corner of the screen. User can return to Main Menu by clicking on Home icon.

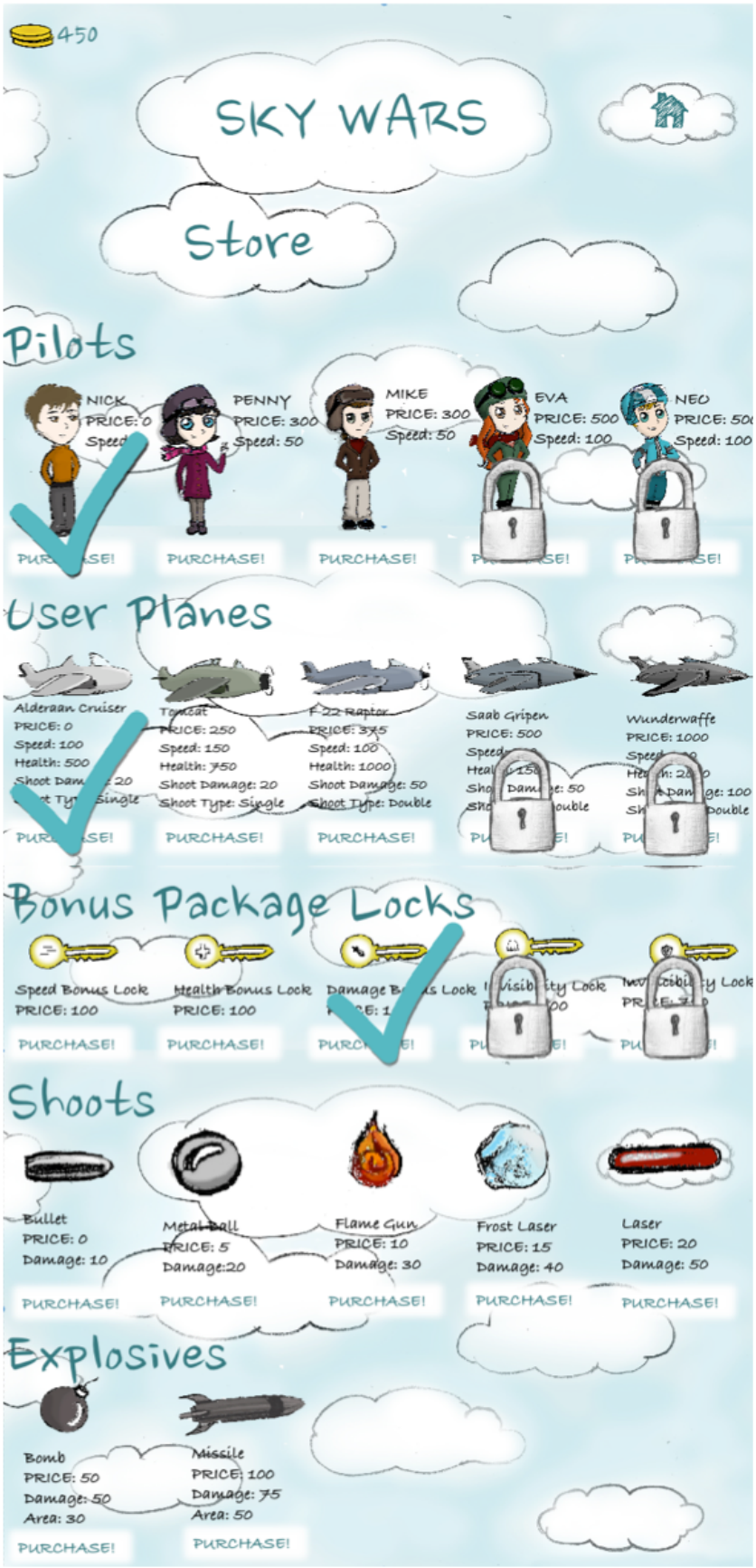


*Figure 39 Store Screen*

## 2.7.6. *Collection Screen*

Collection Screen lists all purchased items hence it is also a scrollable page. Below scene shows entire Collection page. The detailed information of purchased items is provided. The current selection of UserPlane and Pilot is indicated in the left side of the screen. The user can change selection by clicking on any Pilot. User coins are shown on the upper left corner of the screen. User can return to Main Menu by clicking on Home icon.


*Figure 40 Collection Screen*

## 2.7.7. Settings *Screen*

Setting Screen allows user to change volume level by clicking on ‘plus’ and ‘minus’ signs near volume signal and turn music on or off by switching Note icon on or off. User can return to Main Menu by clicking on Home icon.


*Figure 41 Settings Screen*

### 2.7.8. *Help Screen*

Help Screen provides User a video that explains how to play Sky Wars. The User can control video with Video Player icons. User can return to Main Menu by clicking on Home icon.

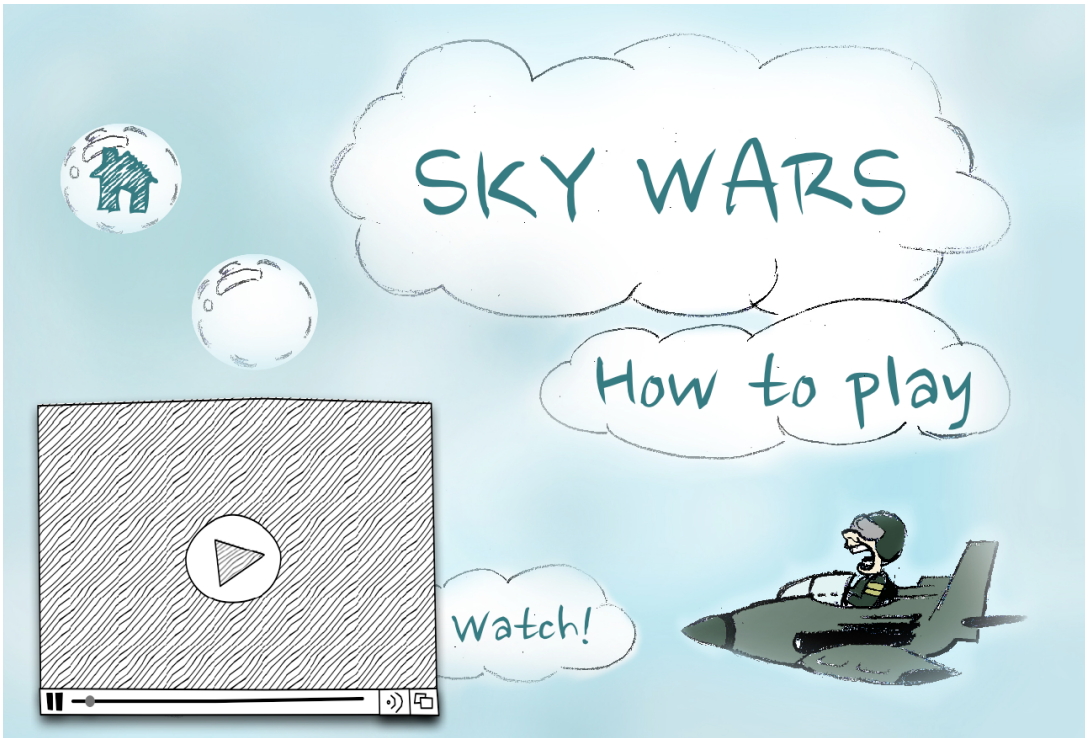


*Figure 42 Help Screen*

### 2.7.9. *Credits Screen*

Credits Screen display User the developer names, publication date and place. User can return to Main Menu by clicking on Home icon.

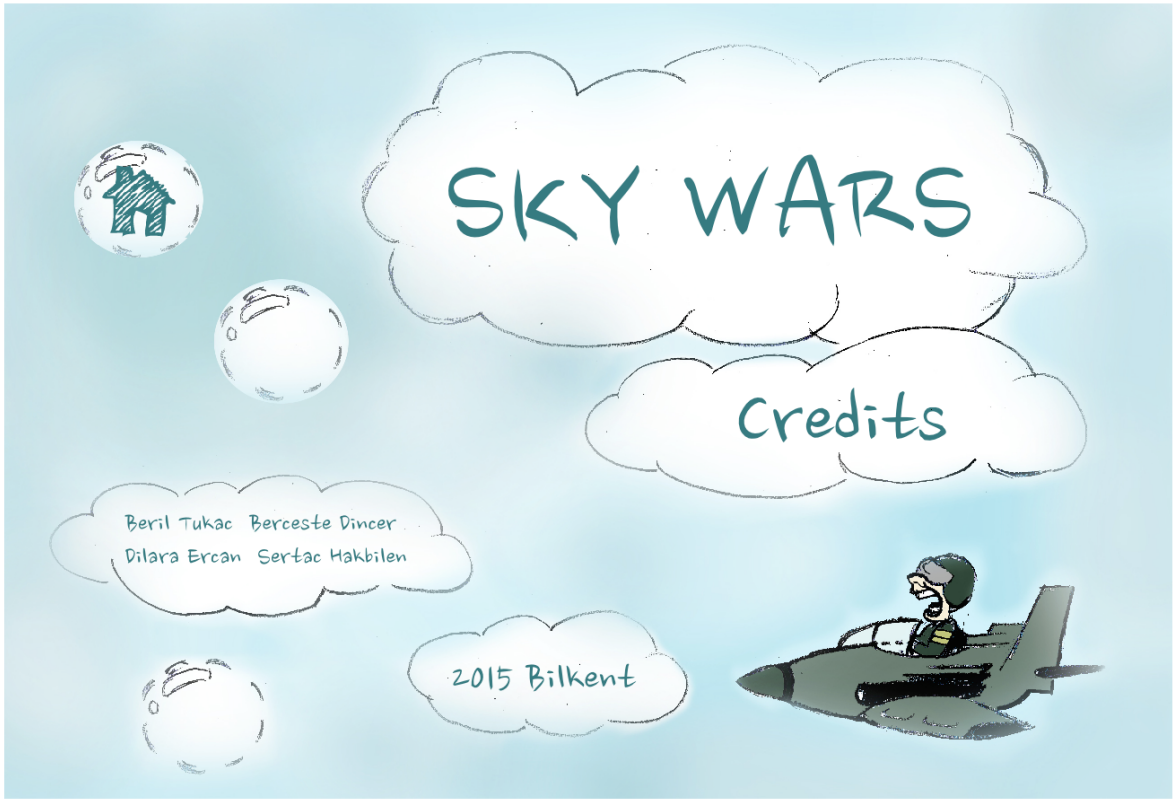


*Figure 43 Credits Screen*

### 2.7.10. *Pause Menu Screen*

Pause Menu Screen is the Menu displayed when the game is paused. The User can click on Continue button to continue playing game or can press Quit button to exit Level and return to Level Map. The icons in clouds direct User to ‘Help’ and ‘Settings’ screens from top to bottom. The icon on right top of the page is for quitting the game.


***Figure 44 Pause Menu Screen***

# 3. Analysis
## 3.1.Object Model
### 3.1.1.Domain Lexicon

**User/Player:** Person who plays and controls Sky Wars

**Game:** Concept of overall system, Sky Wars

**Level:** Small parts of the game which has a time limit and point threshold. A level is opened after previous level is completed.

**Bonus Mission:** A different kind of Level independent from the level flow which Player earns access during regular level and can play once

**GameObject:** Any item visible on the gameplay screen.

**Weapon:** A GameObject which can be shot by Planes, UserPlane or TargetPlane and gives damage to other GameObjects

**Shoot:** A Weapon that can effect only the GameObject it collided with

**Explosive:** A Weapon that creates an explosion and damages nearby objects

**Pilot:** A character figure which represents Player within the game

**UserPlane:** The plane controlled by the Player

**BonusPackage:** A GameObject which creates different bonuses, unexpected variations in the basic game flow. BonusPackages named PresentBonusPackages can help Player, make game easier and help earning point while TrapBonusPackages make the game harder and lead to loss of points.

**Point:** Success unit of the game. Any damage given to enemies, increase Player points while any damage of UserPlane decrease Player points. Points are calculated level based.

**Coin:** The representation of total amount of points gained in all level plays that are kept in Player Account.

**Time:** The amount of minutes specified for each level

**Target:** Any GameObject that has a certain health and can be shot by the User

**Health:** A property Targets and UserPlane has which represents the left damage resistance of a GameObject. Whenever a Target or UserPlane is shot, its health is decreased.

**Collision:** The touch between any two GameObjects

**Level Map:** Map of all Levels and Bonus Missions

**Store:** The place where GameObjects are sold

**Collection:** The list of all items Player has purchased

### 3.1.2.Class Diagrams

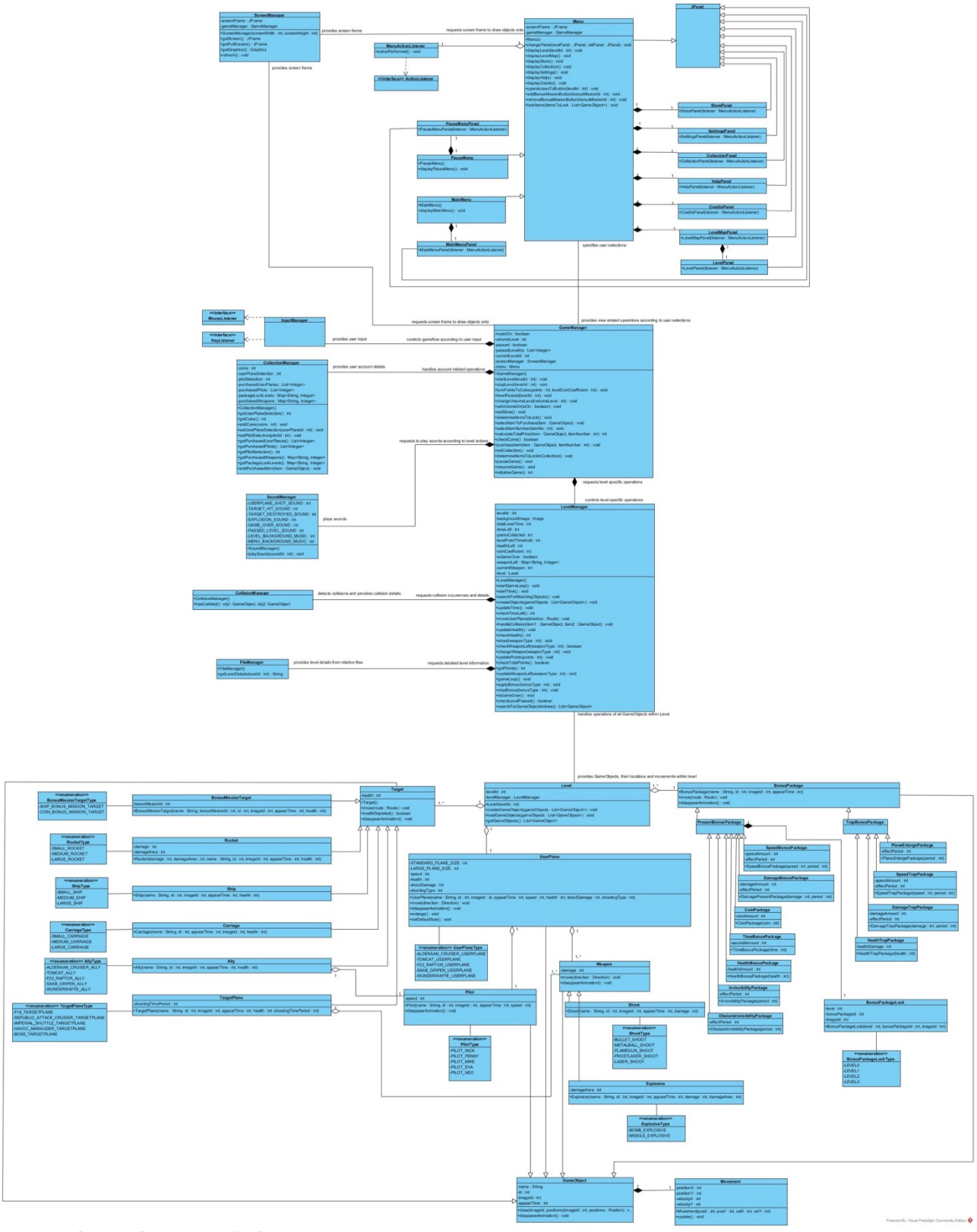

*Figure 45 Class Diagram of Sky Wars*

## Boundary Classes:

**Menu** is the basic Boundary class which allows User to interact with the system.
**MainMenu** is the first screen of Sky Wars
**PauseMenu** is the Menu displayed when the game is paused

**PauseMenuPanel** is the JPanel of Pause Menu
**MainMenuPanel** is the JPanel of Main Menu
**StorePanel** is the JPanel of Store Screen
**SettingsPanel** is the JPanel of Settings Screen
**CollectionPanel** is the JPanel of Collection Screen
**HelpPanel** is the JPanel of Help Screen
**CreditsPanel** is the JPanel of Credits Screen
**LevelMapPanel** is the JPanel of Level Map Screen
**LevelPanel** is the JPanel of Level Screen
**MenuActionListener** implements ActionListener interface and records user inputs

## Controller Classes

**GameManager** is the fundamental Controller class which handles all game operations
**ScreenManager** handles Screen related operations
**FileManager** handles files that specify level details
**InputManager** manages User keyboard inputs. The class implements MouseListener and KeyListener interfaces
**CollisionManager** controls any collision within the game, checks if a collision occurred and specifies the collided objects
**SoundManager** plays game music and sounds
**CollectionManager** handles Collection related operations
**LevelViewManager** controls the GameObjects, their creations, movements and operations within the Level play

## Entity Classes

**Level** holds all GameObjects within level
**Target** is a GameObject which can be shot by the Player
**TargetPlane, Rocket, Ship, Carriage** and **BonusMissionTarget** are different types of Target with various properties. They all have corresponding enumeration classes specifying pre-defined values for different instances of the class
**UserPlane** is the Plane controlled by the Player which can move and shoot. It has a corresponding enumeration class
**Pilot** is the character which controls the UserPlane. It has a corresponding enumeration class
**Weapon** class represents the object that can be shot by planes and damages GameObjects by decreasing their health
**Shoot** is a type of Weapon which only decreases the health of the collided object. It has a corresponding enumeration class
**Explosive** is a type of Weapon that causes an explosion and damages all objects within a certain area. It has a corresponding enumeration class
**BonusPackage** is a GameObject which creates different bonuses
**PresentBonusPackage** is a BonusPackage that helps the Player and boosts the game
**SpeedBonusPackage, DamageBonusPackage, CoinPackage, TimeBonusPackage, HealthBonusPackage, InvincibilityPackage, ObstacleInvisibilityPackage** are classes that represent different PresentBonusPackages
**TrapBonusPackage** is a BonusPackage that makes the game difficult for the User
**PlaneEnlargePackage, SpeedTrapPackage, DamageTrapPackage** and **HealthTrapPackage** are classes that represent various TrapBonusPackages
**BonusPackageLock** represents the objects that are sold in Store which opens access to PresentBonusPackages. It has a corresponding enumeration class

**GameObject** is the basic class which represents objects within Game Level, UserPlane, Pilot, Weapon, BonusPackage and Target classes inherit GameObject class

**Movement** is the class that is responsible from the movement of GameObject

# *3.2.Dynamic Models*

## *3.2.1.State Chart and Activity Diagrams*

### *3.2.1.1.State Chart Diagram of UserPlane*

The below State Chart diagrams demonstrates the dynamic behavior of UserPlane class. State Chart diagram for UserPlane shall be examined in 4 subgroups, damage, speed, size and health. All these activities occur concurrently.

When the game is started, UserPlane is in 'Full Health', 'Default Size', 'Default Speed' and 'Default Damage' states. When the UserPlane collides with a DamageBonusPackage, the UserPlane goes to 'Increased Damage' state, its damage value is increased. UserPlane stays in 'Increased Damage' state until the time period for BonusPackage ends. At the end of the period UserPlane returns to 'Default Damage' state. Similarly, when the Player collides with a DamageTrapPackage, the UserPlane goes to 'Decreased Damage' state. Again at the end of the BonusPackage time, UserPlane goes back to 'Default Damage' state.

Just like damage state flow, when the UserPlane collides with SpeedBonusPackage, the UserPlane goes to 'Increased Speed' state, its speed value is increased. UserPlane stays in 'Increased Speed' state until the time period for BonusPackage ends. At the end of the period UserPlane returns to 'Default Speed' state. Similarly, when the Player collides with a SpeedTrapPackage, the UserPlane goes to 'Decreased Speed' state. Again at the end of the BonusPackage time, UserPlane goes back to 'Default Speed' state.

When the UserPlane collides with a PlaneEnlargePackage, the object passes to 'Enlarged' state. Until the time period for BonusPackage ends the UserPlane states in the same state and then goes back to 'Standard Size' state.

When a Weapon collides with the UserPlane, its health is decreased hence it passes to 'Decreased Health' state. Moreover, when the UserPlane collides with a HealthBonusPackage it goes to 'Increased Health' state, its health value is increased. Similarly, when the Player collects a HealthTrapPackage the UserPlane passes to 'Decreased Health' state. From each health related state, the UserPlane passes to 'Health Depleted' state when the UserPlane collides with an Obstacle or Target. 'Health Depleted' state is the final state since the game is over when the health is depleted.

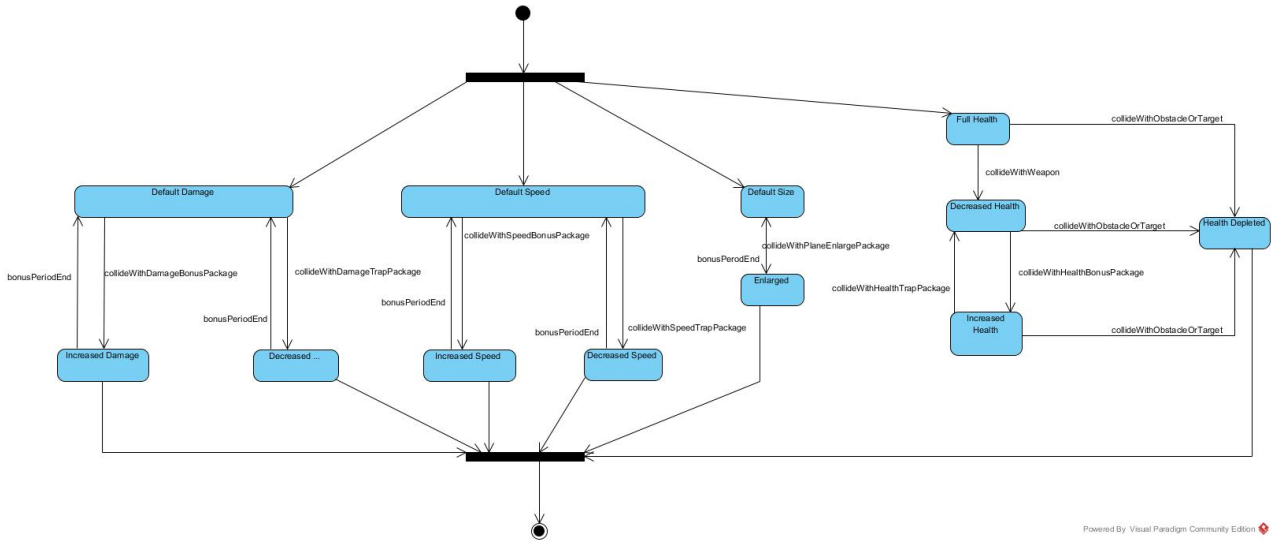


*Figure 46 State Chart Diagram of UserPlane*

### *3.2.1.2. Activity Diagram for Overall Game Flow*

The below diagram represents the overall dynamic behaviour of Sky Wars, game navigations and operations are explained with activity flow.

When the Sky Wars is opened, Main Menu page is displayed. Various activities can be invoked from Main Menu according to Player input. If the Player presses Level Map button Level Map View activity is started. If the Player selects a level or Bonus Mission to play from Level Map LevelPlay activity is invoked. During LevelPlay activity, the Player can select to pause the game, which directs game flow to Pause Menu View activity. The LevelPlay activity continues when the Player selects to continue to play.

When Store button is clicked in the Main Menu View activity, the system moves to Store View activity. Purchase Item activity can be invoked from Store page when the User presses Purchase button.

The Player can press Collection button to move to Collection View activity. From the Collection page, the Player can pass to Change Preferences activity by changing Player item selections.

After Main Menu View activity Settings View activity can be invoked as well if the Player clicks Settings button. When the Player changes Settings game flow moves to Change Settings activity.

When the Player presses Help and Credits buttons, Help View and Credits View activities are invoked respectively. Activity flow passes to Main Menu from all activities when the Player clicks Main Menu button.

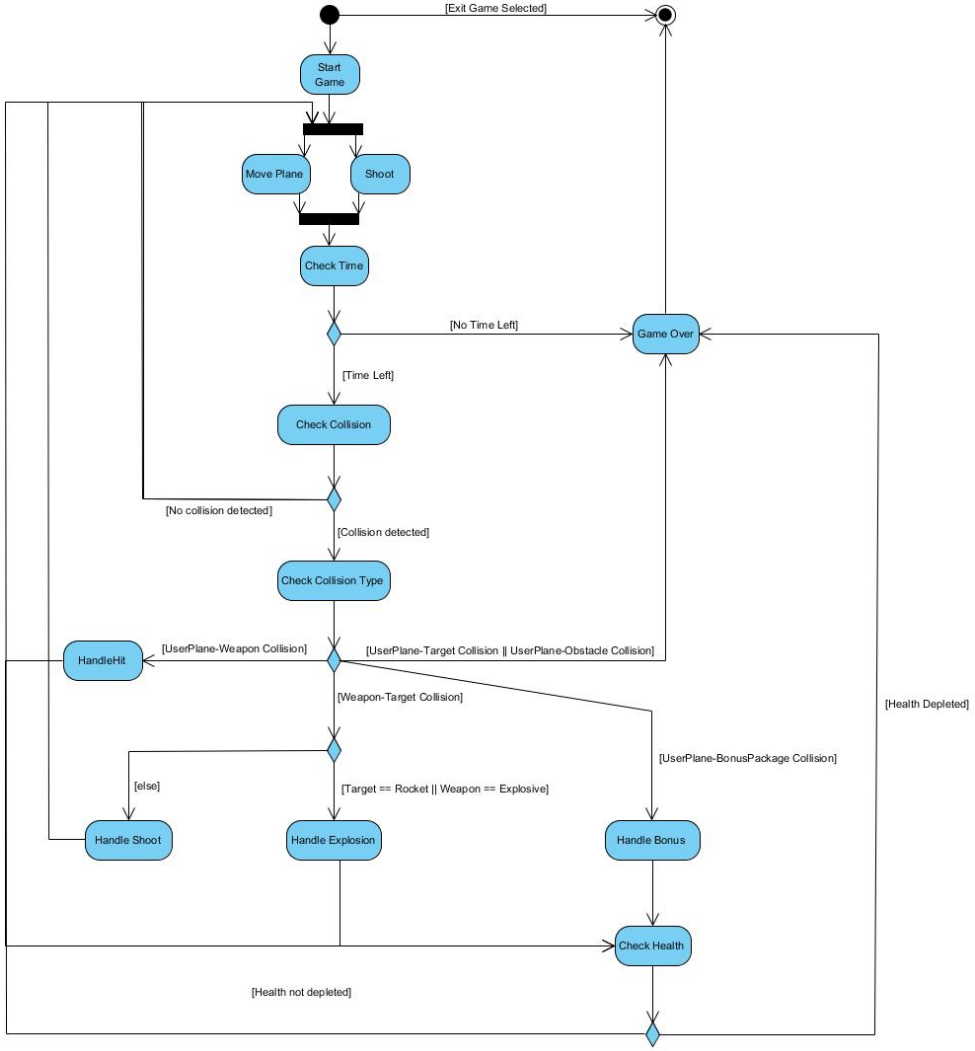


***Figure 47 Activity Diagram of Overall Game Flow***

### *3.2.1.3. Activity Diagram for Game Play*

The above diagram represents the overall dynamic behaviour of Sky Wars game play. The diagram demonstrates activity flow within game play, User actions and system responses.

The level starts with ‘Start Game’ activity. After the level is constructed the Player is given the option to Move Plane or Shoot. These activities occur concurrently. Then the system moves to ‘Check Time’ activity. If the time is over, the flow passes to ‘Game Over’ activity and the flow stops afterwards. If there is time ‘Check Collision’ activity is initialized. If no collision exists then the flow returns to Player activities, ‘Move Plane’ and ‘Shoot’. Otherwise ‘Check Collision Type’ activity is initialized. If the collision has occurred between UserPlane and Obstacle or Target, game is over and the game flow ends. If the collision type is UserPlane-Weapon then the game flow passes to ‘Handle Hit’ activity and necessary operations are performed by the system. Afterwards ‘Check Health’ activity is initiated. If the heath is depleted the game flow ends. Otherwise, the flow is directed back to Player activities. When the collision occurs between UserPlane and BonusPackage the System initializes ‘Handle Bonus’ activity. Afterwards the game flow connects to ‘Check Health’ activity. The fourth possibility is that the Weapon sent by the Player has collided with Target. Then the game flow makes another decision. If the sent Weapon is an Explosive or the Target is Rocket then the system is expected to ‘Handle Explosion’. After this activity the game flow connects to ‘Check Health’ activity. If the Target was not an Explosive ‘Handle Shoot’ activity is started. This activity directs the game flow back to ‘Move Plane’ and ‘Shoot’ activities.

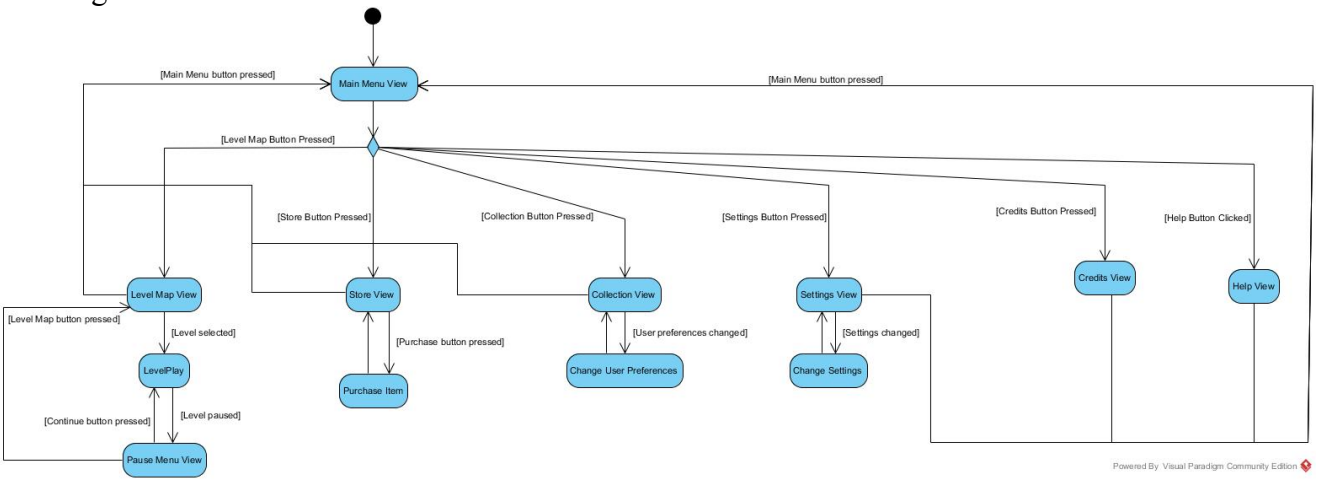


***Figure 48 Activity Diagram of Game Play***

## *3.2.2.Sequence Diagrams*

### *3.2.2.1. Start Game*

**Scenario:** Player Ali requests to view Level Map by pressing relative button in the Main Menu. Main Menu loads Level Map View. Ali chooses a level, Level 1, from the Level Map by pressing the relative level icon. System requests GameManager to initialize the game. GameManager first gets the full screen window and graphics from Screen Manager to manipulate the screen to start the level. Secondly, the System requests current selection of UserPlane and Pilot from the CollectionManager. CollectionManager returns the selected UserPlane and Pilot. GameManager creates the corresponding UserPlane and Pilot objects. Then CollectionManager specifies the purchased Packages as well. Then System requests level details, including the number of game objects, their types, routes and appear times, from the FileManager. FileManager returns the corresponding file to GameManager. The system loads the level view and requests from LevelView to create corresponding objects. While determining BonusPackages, the system also considers information returned form the CollectionManager. LevelManager creates a UserPlane and a Pilot and places the object to the specified place. The GameObjects which are supposed to appear at the beginning of the level are created, a BonusPackage, a TargetPlane and an Obstacle. System loads created objects to the Level. Then GameManager starts the level.

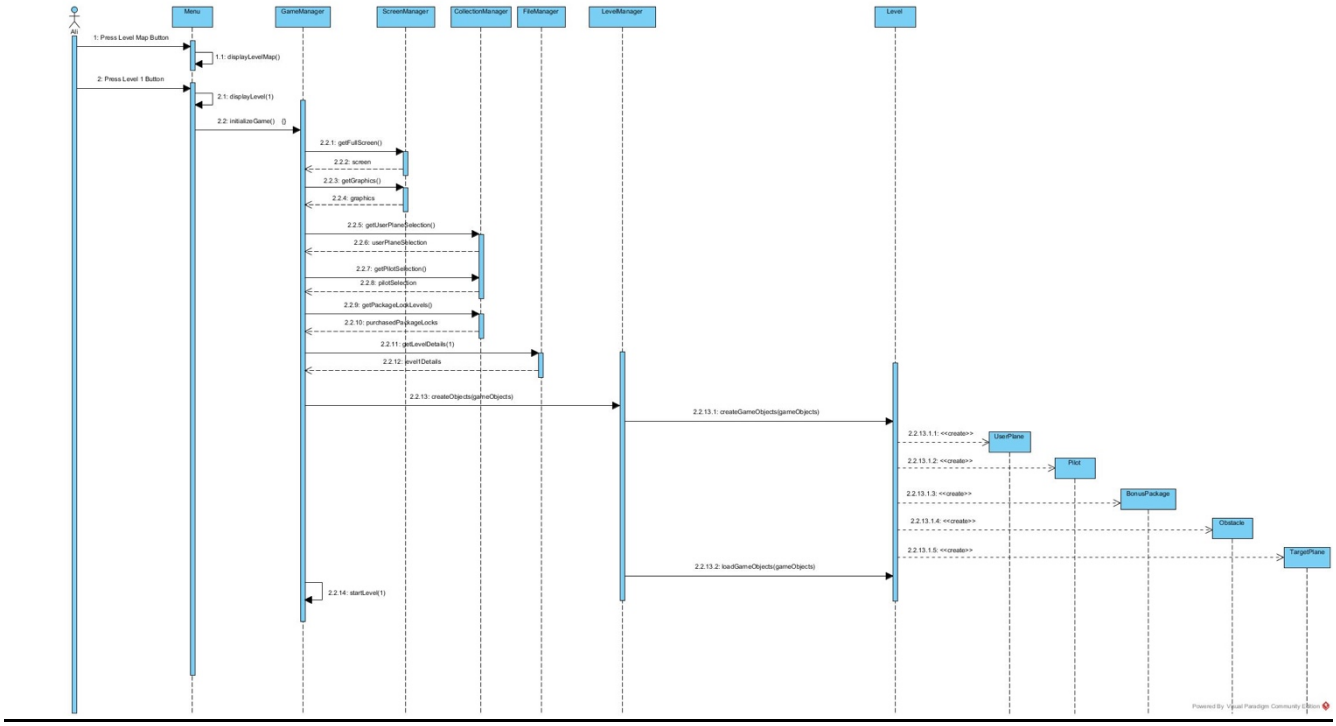


*Figure 49 Start Game Sequence Diagram*

### *3.2.2.2. Creation of GameObjects During Level*

**Scenario:** (Continued after Start Game) The system starts the time allocated for the selected level, Level 1. LevelManager checks time left. Then the System searches for GameObjects specified in the Level File. If the appearance time of a GameObject matches the current time, LevelManager creates the object, a TargetPlane and a BonusPackage in this scenario. Then the system adds the created GameObjects to Level. This process of creating and loading GameObjects continue until the game is over, time can be over or Player can die.

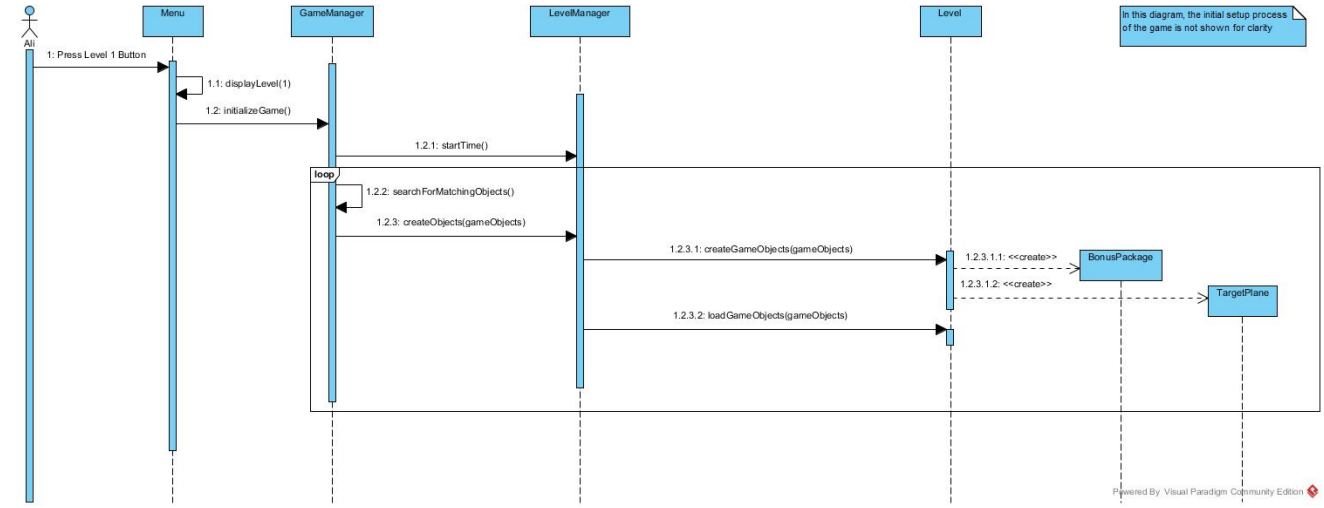


**Figure 50 Create Game Objects During Level Sequence Diagram**

### *3.2.2.3. Level Play*

**Scenario:** (In this scenario the loop of Creation of GameObjects During Level is not represented for clarity) Player Ali requests to start Level 1 by pressing on the corresponding button. The system starts Level 1 by starting the time, 2 minutes. The system checks the left time and since it is larger than 0, the level continues. Player Ali presses 'down' key to move the UserPlane down. The InputManager takes the keyboard input and asks LevelManager to handle operation. LevelManager requests UserPlane to move and the UserPlane moves down. The system checks for collisions. CollisionManager specifies that there is no collision. Player Ali presses 'up' key. The system responds by moving the plane upwards. Again, LevelManager asks CollisionManager if collision occurred. CollisionManager specifies that collision occurred and indicates that the collision has occurred between UserPlane and Weapon object with id w23. LevelManager handles collision. Then the system checks whether the health of the UserPlane is depleted. Since this is not the situation the game continues. Player Ali presses space button. The system first checks whether the User has the weapon. LevelManager responds by returning false. Then Player Ali presses 'C' key to change the weapon type. The system updates the selected weapon. Then Player Ali presses space key. The system again checks if the User has the weapon. LevelManager returns true this time and creates a Weapon object. The system adds the Weapon object to Level. Then the LevelManager requests the weapon to move in the specified direction. Then the system checks whether collision has occurred. CollisionManager indicates that collision has occurred and it is between newly created weapon and Carriage object with id c8. The system handles the collision by updating health of the Carriage and the points of User. Then the system checks whether the life of Carriage has depleted. Carriage indicates that its health has depleted. Then the system requests the Carriage and Weapon to disappear with animation. Then this game loop continues until the system detects that the time is over. When the time is over the system checks the total points earned, whether it is larger than the threshold for the level. Then the system turns the points into coins. GameManager indicates that the level is passed and Player Ali is directed back to the Level Map Page.

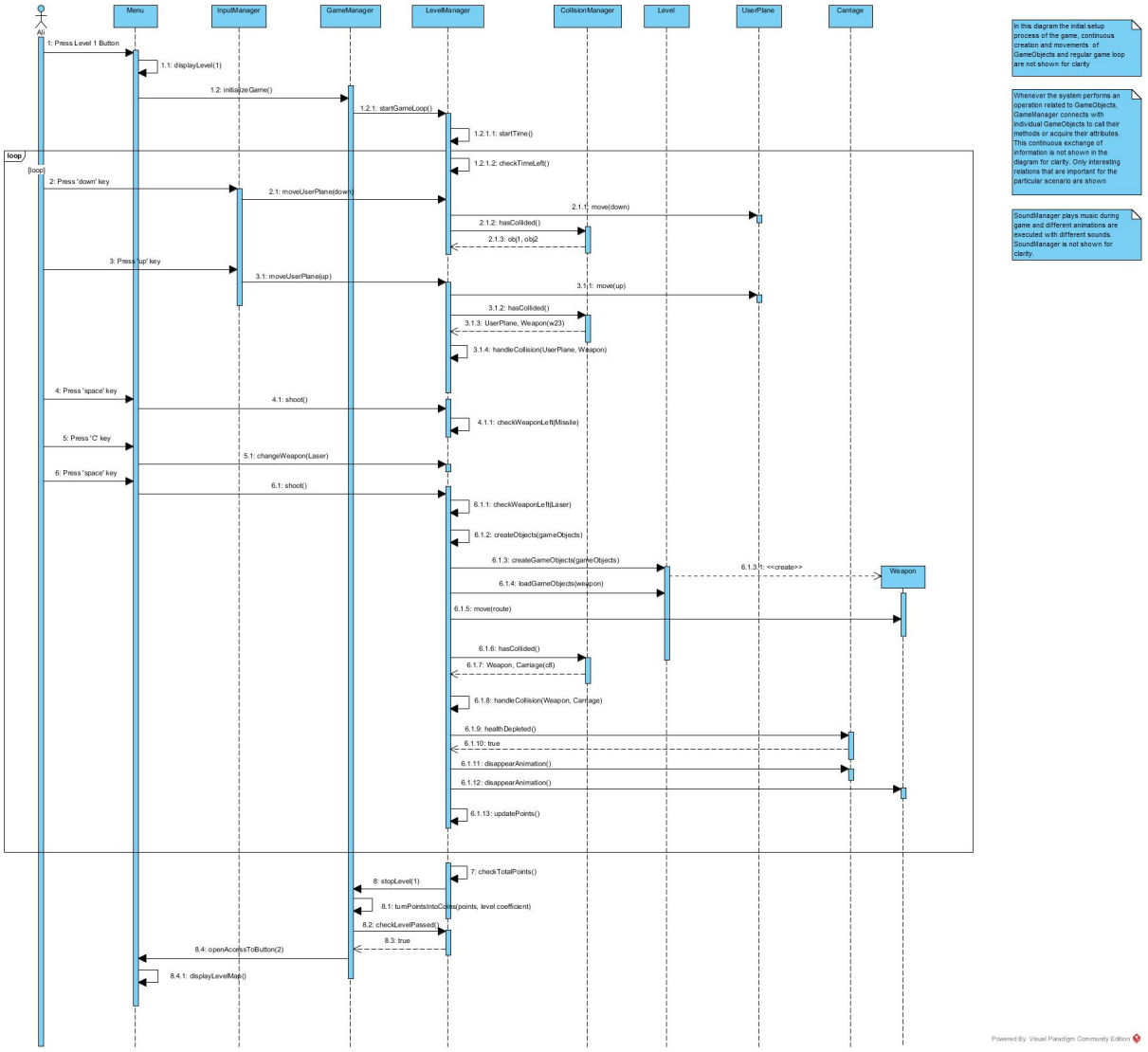


**Figure 51 Level Play Sequence Diagram**

### *3.2.2.4. Collecting Present Bonus Package*

**Scenario:** (In this scenario the loop of Creation of GameObjects During Level and the actual game loop details are not shown for clarity) While playing Level 2, Player Ali presses 'up' key. LevelManager requests UserPlane to move and the UserPlane moves down. The system checks for collisions. CollisionManager specifies that collision occurred and indicates that the collision has occurred between UserPlane and HeathBonusPackage object with id hb1. In order to handle the collision, LevelManager requests the health from the HealthBonusPackage. HealthBonusPackage returns the amount. LevelManager adds the specified amount to the health of the UserPlane. Then the game loop continues as usual.

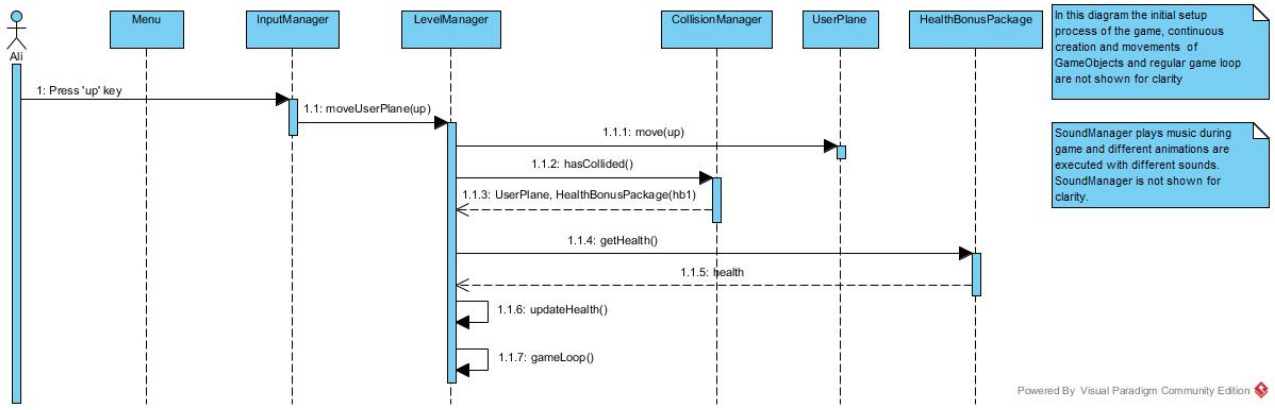


**Figure 52 Collecting Present Bonus Package Sequence Diagram**

### *3.2.2.5. Collecting Trap Package*

**Scenario:** (In this scenario the loop of Creation of GameObjects During Level and the actual game loop details are not shown for clarity) While playing Level 2, Player Ali presses 'up' key. LevelManager requests UserPlane to move and the UserPlane moves up. The system checks for collisions. CollisionManager specifies that collision occurred and indicates that the collision has occurred between UserPlane and PlaneEnlargeTrapPackage object with id pet1. In order to handle the collision, LevelManager requests from UserPlane to enlarge. UserPlane enlarges itself. Then the LevelManager waits for the TrapPackage period to end and at the end of the period requests UserPlane to come to its default size. Afterwards, the game loop continues as usual.

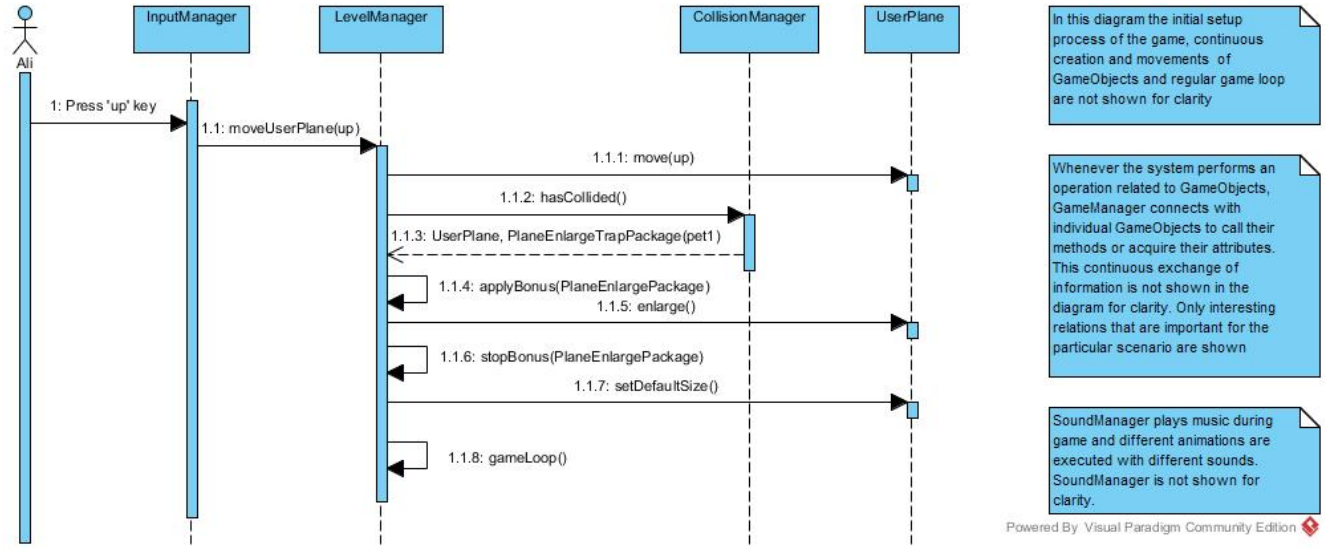


**Figure 53 Collecting Trap Bonus Package Sequence Diagram**

### *3.2.2.6. Obstacle Hit*

**Scenario:** (In this scenario the loop of Creation of GameObjects During Level and the actual game loop details are not shown for clarity) While playing Level 2, Player Ali presses 'up' key. LevelManager requests UserPlane to move and the UserPlane moves up. The system checks for collisions. CollisionManager specifies that collision occurred and indicates that the collision has occurred between UserPlane and Obstacle object with id o18. LevelManager stops the level and notifies Ali that the level has not been passed via Menu View. Menu View directs Ali to Level Map.

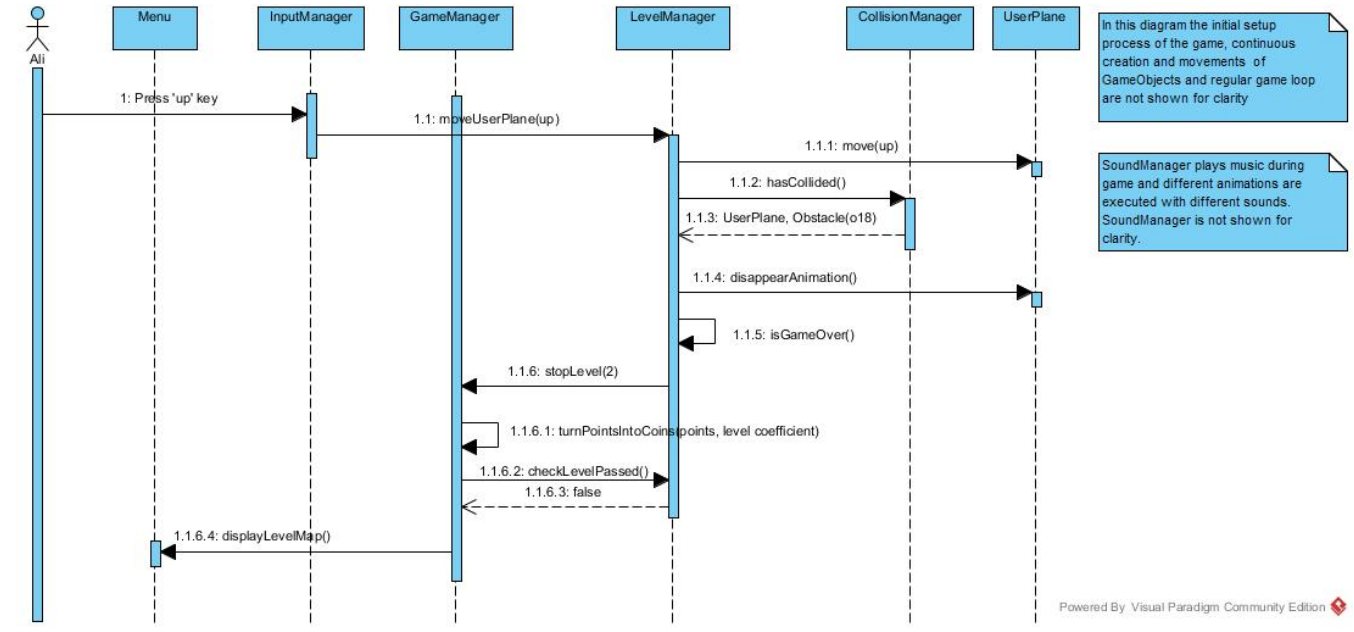


**Figure 54 Obstacle Hit Sequence Diagram**

### *3.2.2.7. Using an Explosive as Weapon*

**Scenario:** (In this scenario the loop of Creation of GameObjects During Level is not represented for clarity) While playing level 2, Player Ali presses 'C' key iteratively until he reaches to Bomb which is an Explosive Weapon. Player Ali presses space button. The system first checks whether the User has the weapon. LevelManager returns true and creates an Explosive object. The system adds the Explosive object to Level. Then the GameManager gets the object and requests it to move in the specified direction. Then the system checks whether collision has occurred. CollisionManager indicates that collision has occurred and it is between newly created Explosive and TargetPlane object with id tp31. Then the system searches for any GameObject within the damage area of the Bomb. LevelManager updates the health of objects and points of the User. The System also requests Bomb to disappear with an animation. Since the UserPlane is amongst these objects, its health is also decreased by the system. Then the system checks whether the health of the UserPlane is depleted. Since this is not the situation the game continues.

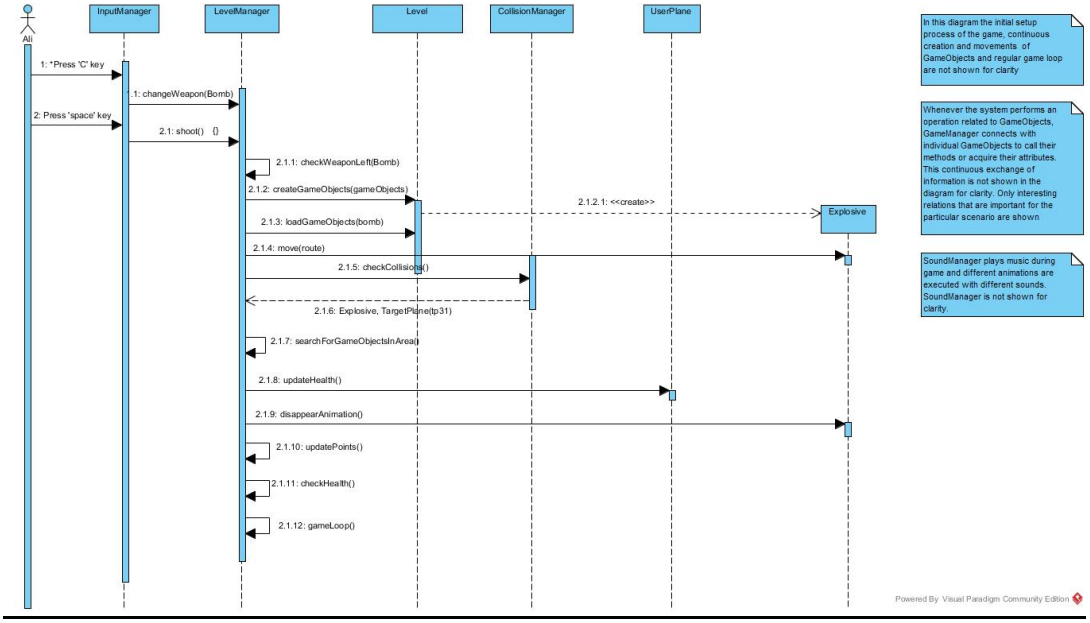


**Figure 55 Using an Explosive as Weapon Sequence Diagram**

### *3.2.2.8. Shooting A Rocket*

**Scenario:** (In this scenario the loop of Creation of GameObjects During Level is not represented for clarity) While playing level 2, Player Ali presses space button. The system first checks whether the User has the weapon. LevelManager returns true and creates a Weapon object. The system adds the Weapon object to Level. Then the LevelManager requests the weapon to move in the specified direction. Then the system checks whether collision has occurred. CollisionManager indicates that collision has occurred and it is between newly created Weapon and Rocket object with id r3. The system first checks whether the health of the Rocket has depleted. Rocket indicates that this is the situation. The LevelManager requests Rocket to disappear. Then the system searches for any GameObject within the damage area of the Rocket. For each GameObject within the area, LevelManager updates health of objects and points of the User. Since the UserPlane is amongst these objects, its health is also decreased by the system. Then the system checks whether the health of the UserPlane is depleted. Since this is not the situation the game continues.

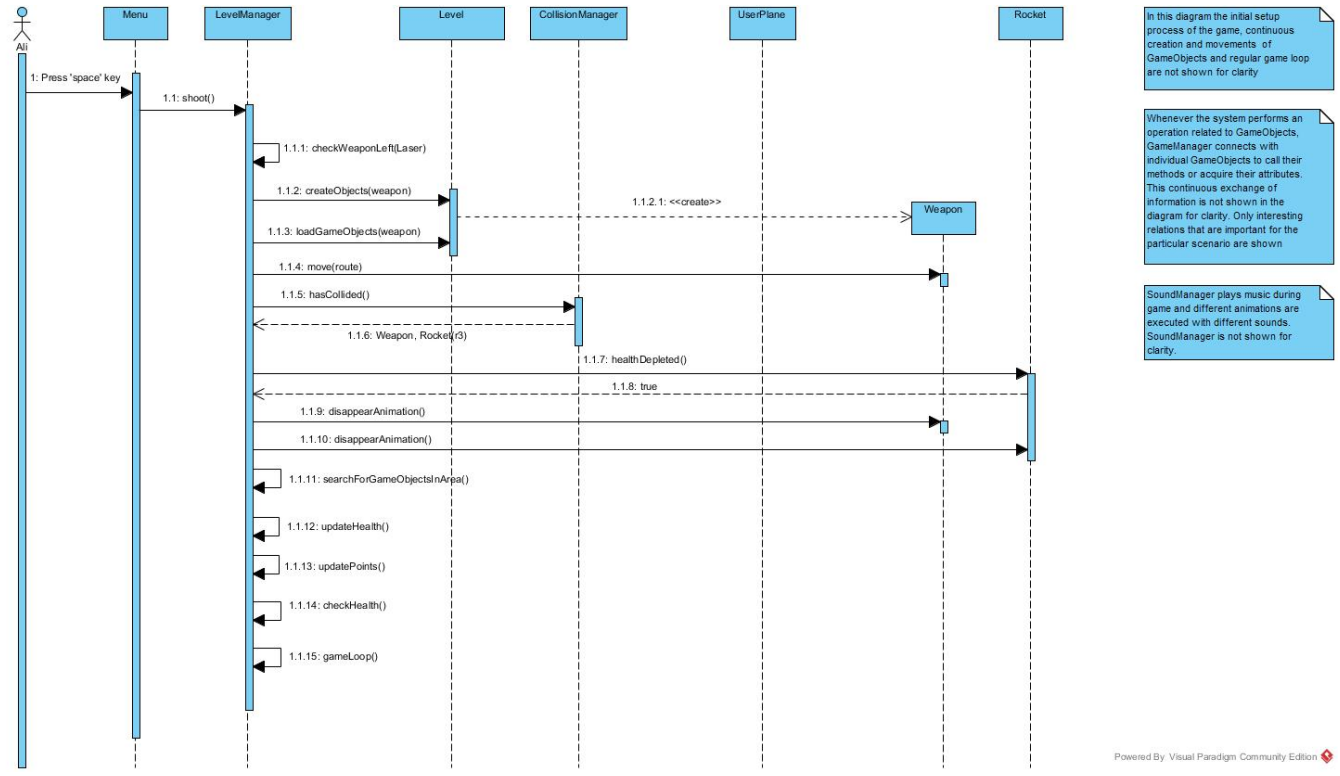


**Figure 56 Shooting a Rocket Sequence Diagram**

### *3.2.2.9. Playing Bonus Mission*

**Scenario:** (It is assumed that the Player has shot the BonusMissionTarget in the level and Bonus Mission option is visible in the Level Map.) Player Ali requests to view Level Map by pressing relative button in the Main Menu. Main Menu loads Level Map View. Ali chooses the Bonus Mission, Ship Shooting in this scenario by pressing the proper icon.

System requests GameManager to initialize the game. GameManager first gets the full screen window and graphics to manipulate the screen to start the Bonus Mission. Then the System requests Bonus Mission details, including the number of Ship objects and their appearance times from the FileManager. FileManager returns the corresponding file to GameManager. The system loads the Bonus Mission view and requests LevelManager to create corresponding objects. LevelManager creates a UserPlane and a Pilot and places the objects to the specified places. System checks the appearance time of Ships from the file. The Ships which are supposed to appear at the beginning of the level are created. System loads created objects to the Level. Then LevelManager starts the game loop. The system starts the time allocated for the BonusMission. Then the

system checks the time and searches for Ship objects specified in the Level File. If the appearance time of a Ship object matches the current time, LevelManager creates the object, another Ship object in this scenario. Then the system adds the created Ship object to Level. This process of creating and loading GameObjects continue until the Bonus Mission is over.

Player Ali presses ‘left’ key to move the UserPlane left. LevelManager requests UserPlane to move and the UserPlane moves left. Then Player Ali presses space button. LevelManager creates a Weapon object, Torpedo specifically. The system adds the Weapon object to Level. Then the LevelManager gets the object and requests it to move in the specified direction. Then the system checks whether collision has occurred. CollisionManager indicates that collision has occurred and it is between newly created Torpedo and Ship object with id s10. The system handles the collision by updating the points of User. Then the system requests the Ship to disappear with animation. Then this game loop continues until the system detects that the time is over. When the time is over the system turns the points into coins. LevelManager indicates that the Bonus Mission is completed and Player Ali is directed back to the Level Map Page. Bonus Mission button is removed from Level Map until Player Ali shoots another BonusMission Target.

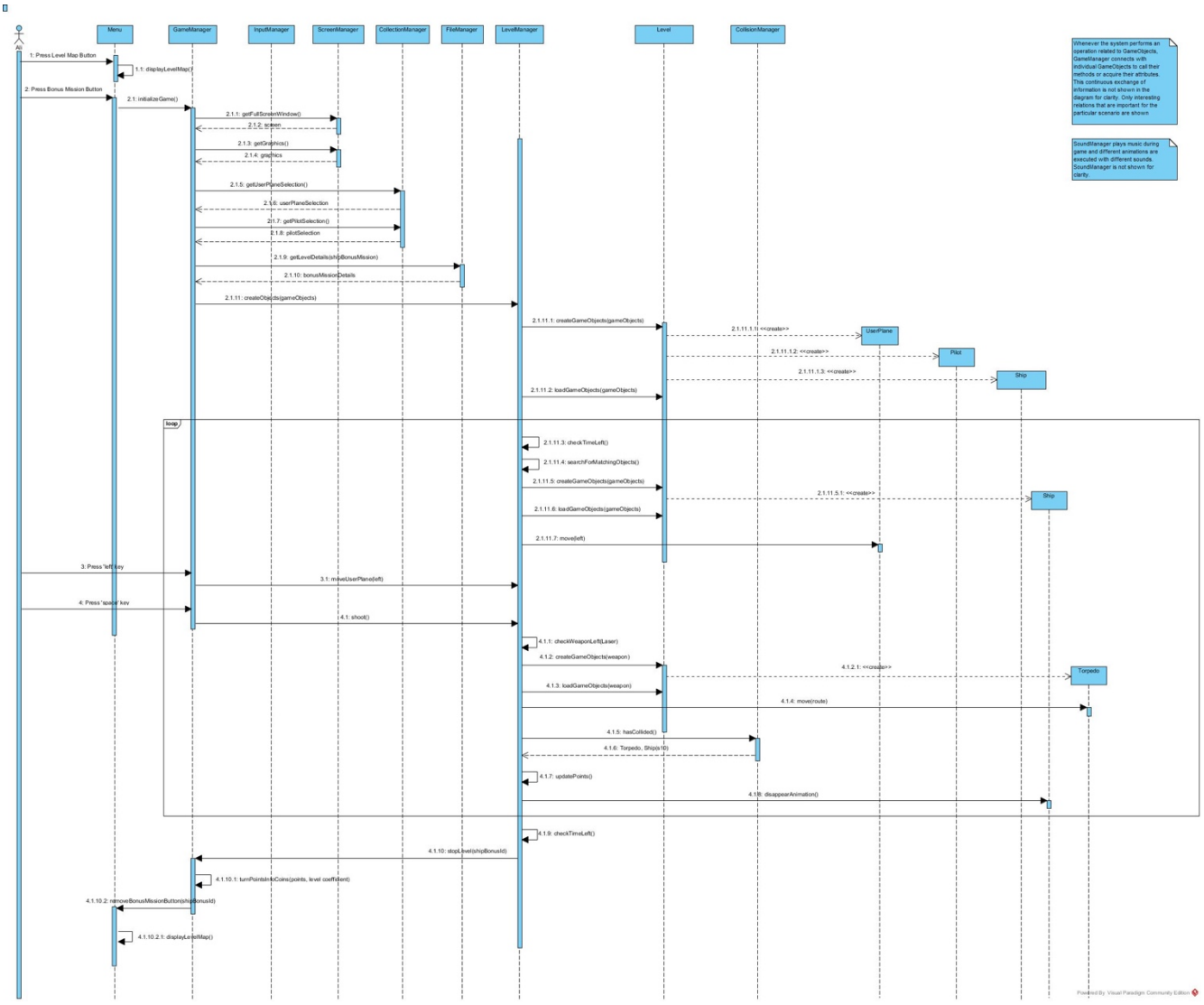


**Figure 57 Playing Bonus MissionSequence Diagram**

### *3.2.2.10. Purchasing UserPlane from Store*

**Scenario:** Player Ali requests to view Store by pressing relative button in the Main Menu. Main Menu loads the Store View. The system requests amount of coins from the CollectionManager. Then the System

locks GameObjects which User cannot afford. Player Ali selects a Weapon by pressing the item view, Flame Gun in this scenario. Then Player Ali specifies the number of weapons he wants to buy, 10 in this scenario. The system checks whether the total price of Weapons exceeds the number of coins in the User account. The CollectionManager indicates that there are necessary coins in the account. Then the system completes the purchase. Then Player Ali presses Main Menu button and the System displays the main menu.

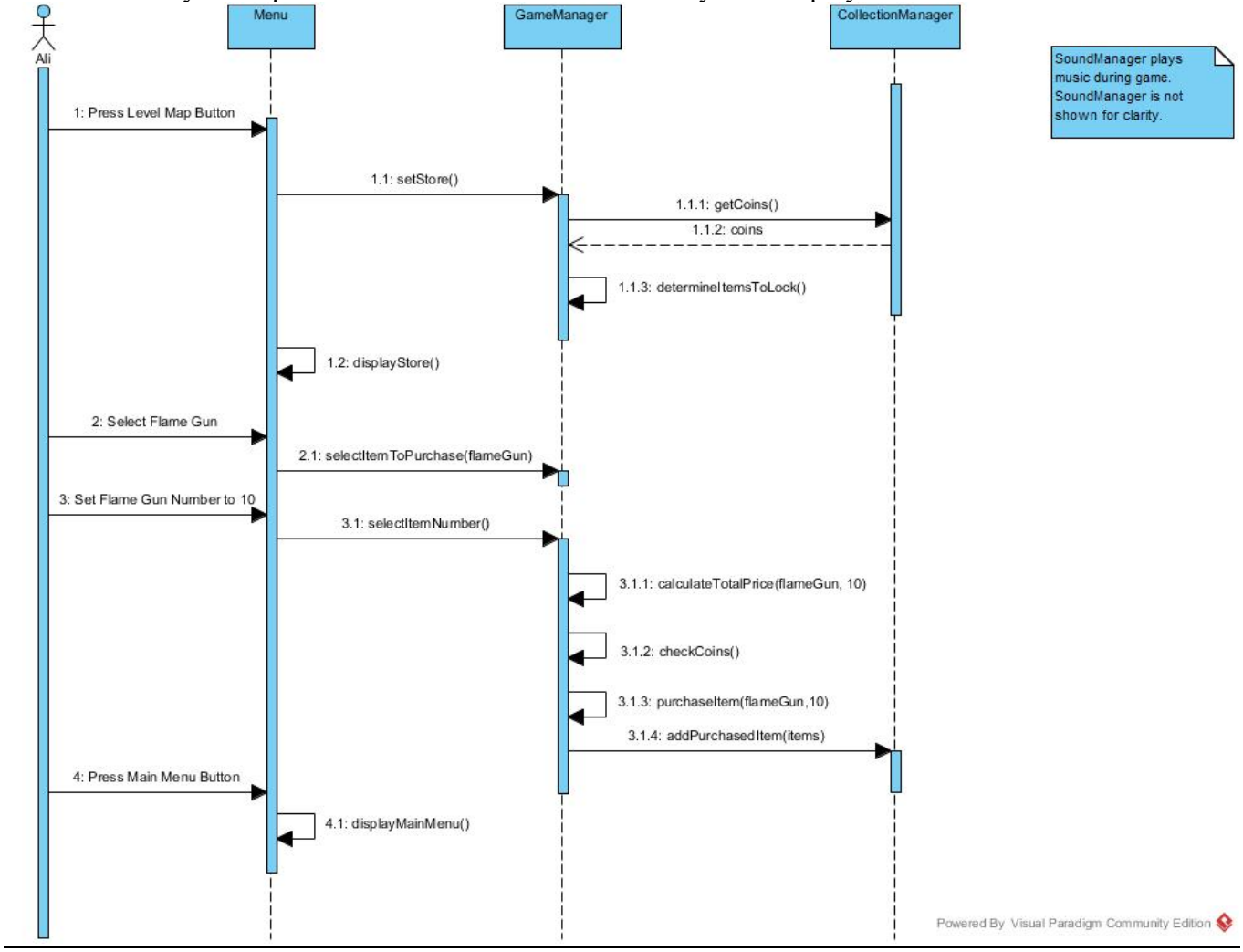


**Figure 58 Purchasing a UserPlane from Store Sequence Diagram**

### *3.2.2.11. Changing UserPlane Preference from Collection*

**Scenario:** Player Ali requests to view Collection by pressing relative button in the Main Menu. Main Menu loads the Collection View. The System requests all purchased items from the CollectionManager. The system updates the Collection view accordingly. Then Player Ali changes Weapon choice by selecting F-22 plane. The System requests CollectionManager to update the UserPlane choice. Then Player Ali presses Main Menu button hence the system directs him to Main Menu.

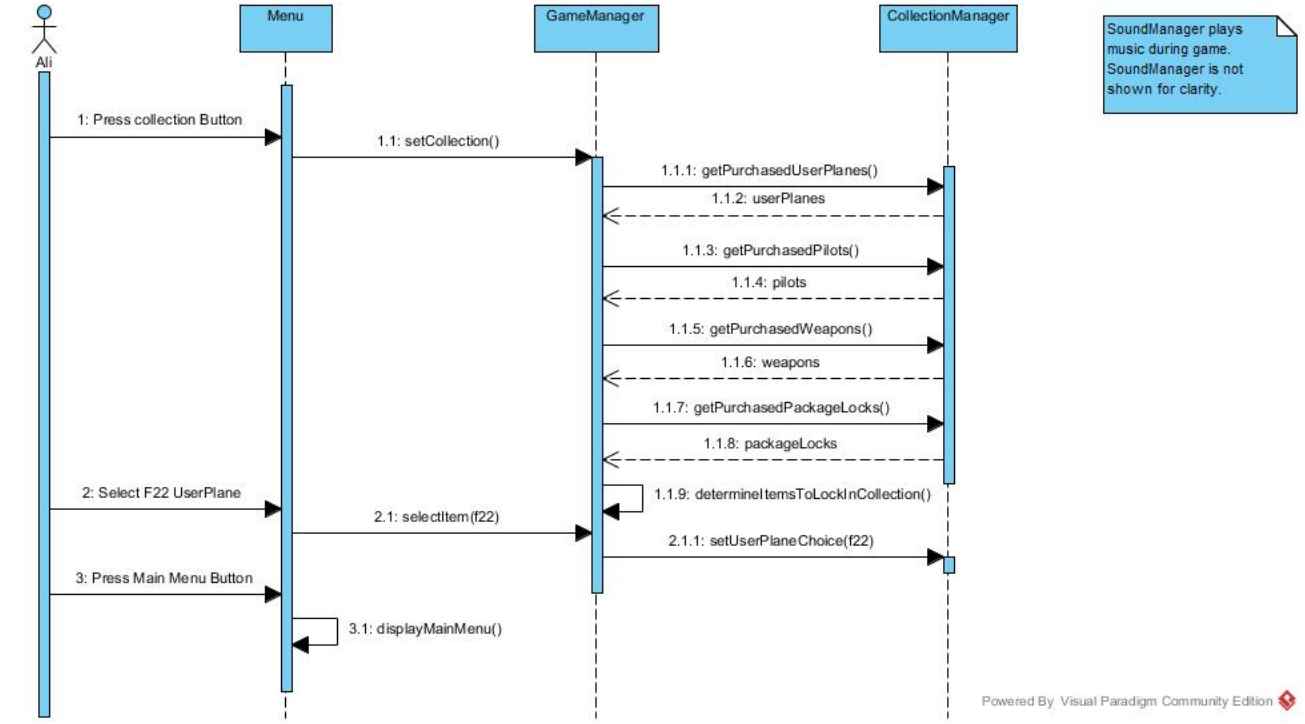


**Figure 59 Changing UserPlane Preference from Collection Sequence Diagram**

### *3.2.2.12. Changing Settings*

**Scenario:** Player Ali requests to view Settings by pressing relative button in the Main Menu. Main Menu loads the Settings View. Player Ali changes Volume level. The System updates the settings. Player Ali presses Main Menu button and System directs him to Main Menu.

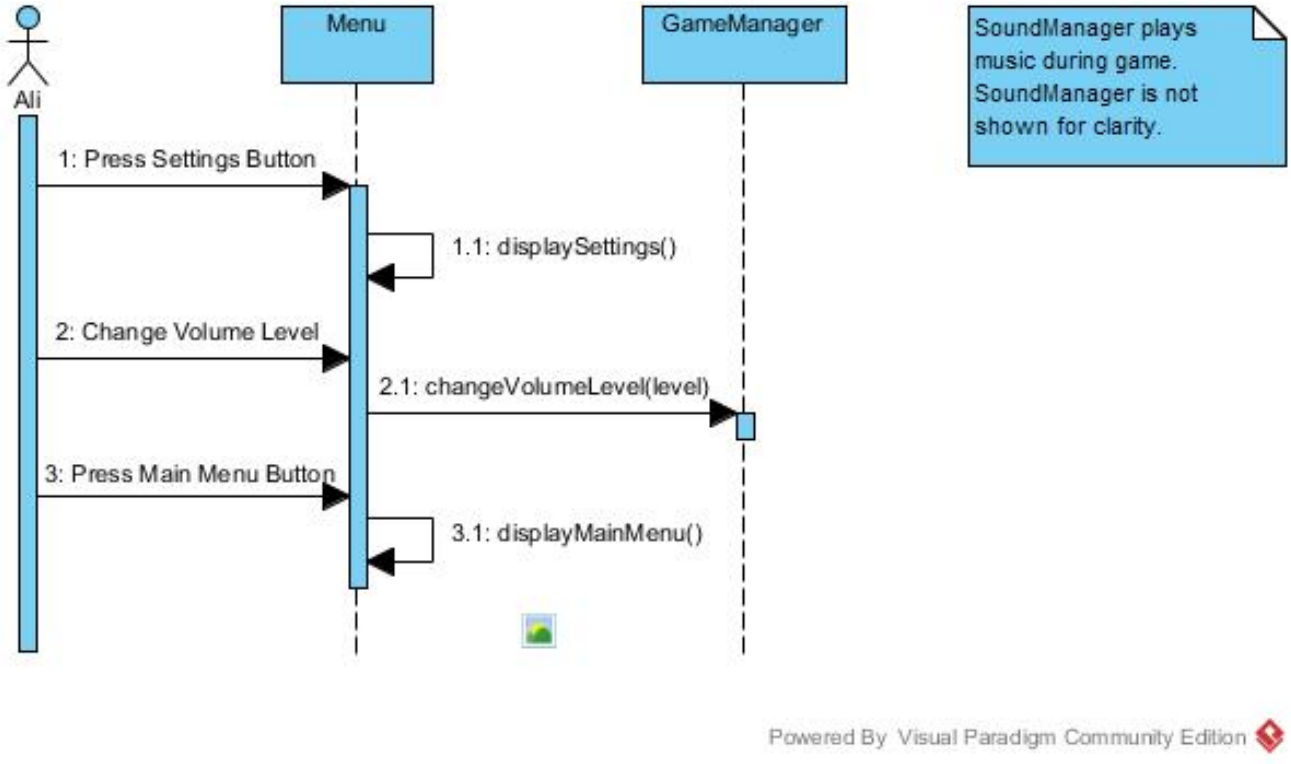


**Figure 60 Changing Settings Sequence Diagram**

# 4. *Design*
## *4.1.Design Goals*

### Reliability

Reliability is fundamentally the ability of the system to perform required operations. We aim our system to be reliable, that is the system should be able to perform regular game operations until the user stops playing. Hence, the system should be resistant to external and internal failures and prevent crashes. In order to achieve this goal, boundary conditions should be determined carefully. During implementation, necessary precautions should be taken to prevent system from crashing. Moreover, if a power-loss or any other problem occurs, the system should not result in data loss. The levels the player has completed, his selections and shopping should be restored if the system fails.

**Traceability of Requirements**

Traceability is important for design of "Sky Wars" in terms of refining requirements into lower-level design components, in understanding the consequences of a possible change and in justification of the existing requirements. In order to achieve this goal, linking each system requirement to real-world dimensions is necessary.

### Usability

**User-Friendliness**

In our software, the user friendliness will be achieved by considering efficiency, effectiveness and satisfaction. The program will be developed in a user-centred way and it will be tested to see the user experience so that it can be evaluated and improved. A pleasant and an explanatory graphical user interface will help us to achieve this design goal.

**Ease of Use**

The game will be easy to use because it will be easy to learn. The interface of the game will be easy to navigate and there will be an efficient error handling. Also the menu and the instructions at the beginning of the game will help the user to play game without trouble.

**Understandability**

Sky Wars, its design model, architecture, implementation and documentation should be understandable. We aim to design the system in a way that users, developers and reviewers can understand the system boundaries, functions and structure. In order to achieve this goal, we need to focus on consistency and completeness during development process.

### Performance

**Rapid-Development**

Since the software will be developed by aiming to make it an object-oriented program, the components will be reused. The models and the UML diagrams will greatly reduce the implementation process. On the other hand, our programming language will be Java. It provides us visual aid tools which also maintains a rapid-development.

**High Performance**

In non-functional requirements of the system, it is declared that the system should respond to the Player input immediately. This is crucial for "Sky Wars" in order to keep player entertained and interested in the game. Besides, the system must also move the game objects smoothly and handle animations without any pause. In order to reach a good game performance and smooth animations, images can be loaded to memory before the game starts.

**Efficiency**

In order to make Sky Wars as efficient as possible, performance goals must be reconsidered in terms of complexity. Thus, possible bottlenecks of the system design can be found and reorganized before the

implementation phase. Writing maintainable and readable code would also be helpful through the implementation in terms of efficiency. So that, the code segments would be eligible for reconsideration.

## Supportability

**Flexibility**

A flexible software should be able to adapt itself for future changes. In order to achieve that, we will keep the coupling of the components at minimum so that a change will not the affect the whole system. We will limit the locations of the necessary modifications. Also, the boundaries between the objects will be kept simple.

**Modifiability**

We aim Sky Wars to be Modifiable, changes should be easy to implement. When the functionality of the system needs to be changed, this change should not alter the whole structure. In order to achieve this goal, the coupling between subsystems should be minimized. Components of Sky Wars should be independent enough that a change in a component does not alter other components much.

**Maintainability**

One of the design goals is Maintainability, the ability to change the system to fix bugs and introduce new technologies. This goal can be achieved by minimizing the complexity of the system. Hence the system should be decomposed to subsystems in a way that complexity is reduced.

**Adaptability**

Since an adaptable design allows developers to easily customize the system, through adapting the existing design, and also gives the privilege of quickly developing new and upgraded models, it is important for the system. Since Java is one of the few programming languages, which provides cross-platform portability, it will be helpful through the development process for the system adaptability. Operating system constraints will not be a problem. Another key point for the adaptability of Sky Wars would be writing the code as readable and maintainable as possible.

## Trade-offs

**Functionality vs. Ease of Use**

In Sky Wars, to allow user to easily adapt to game, we use only arrow keys, space button and C button to play the game. We thought that having too many buttons can create distraction while user is playing the game. Thus ease of use is more important than functionality in our game.

**Rapid Development vs. Functionality**

Since we have limited time, we have to analyze and design our game faster than a normal procedure. Because of that we cannot add many things. Thus we reduce functionality. However, we will develop Sky Wars with a sufficient functionality which user can demand.

**High Performance vs. Reliability**

Sky Wars is an arcade game because of that user has to see all moving objects without any skipping. Thus we believe that performance of Sky Wars is really important. We also want our game to be reliable, consistent even in failure. We also aim to prevent data-loss in a system crash. Hence, we aim higher performance as long as the system is reliable.

**High Performance vs. Memory**

In order to maximize performance memory is effectively used in Sky Wars. For instance, game images and sounds are loaded to memory before the game is started. However, when memory usage is tried to restricted, the performance decreases.

**High Performance vs. Adaptability**

Sky Wars could be written in another programming language such as C/C++ for performance advantages. However, easily customizing the system through adapting the existing design is one of the important design goals of Sky Wars. Thus, a trade-off between performance and adaptability must be done.

# *4.2.Subsystem Decomposition*

Sky Wars system is decomposed to three parts: User Interface, Game Control and Game Model. These layers are ordered hierarchically and they have run-time dependency. User Interface subsystem contains Boundary classes and View components. This subsystem is responsible from interacting with user and handling view operations. It is the top layer and calls Game Control subsystem to handle game-related operations. Game Control subsystem is the middle layer and it contains classes responsible from the fundamental functions and operations of the game. The Game Control subsystem provides services related to all operations of the game including level operations. This subsystem calls Game Model subsystem for retrieving necessary data for game operations. Game Model is the lowest layer. It provides data-related services such as data retrievals and updates. It consists of all classes containing game data.

## Closed Architecture

Sky Wars architecture is a closed architecture in which layers can access to services of only layer below them. Hence, User Interface subsystem can only call services from Game Control subsystem and not from Game Model subsystem. Even though this architecture decreases Efficiency, it gains Flexibility to system since layers become more independent of each other. Moreover, a closed architecture is also more Maintainable since the run-time dependencies decrease.

## Coupling and Coherence of Subsystems

In our subsystem decomposition, we aimed and achieved high coherency. Classes within subsystems share a common concept and they perform similar and related operations. They interact with each other frequently for implementing related functionalities. For instance, Game Control subsystem consists of Controller classes such as GameManager, LevelManager, SoundManager. They are all responsible from handling general game functions related to store, collection, settings and level-specific operations. They interact with each other for all game operations. For instance, GameManager calls LevelManager for LevelControl, LevelControl calls GameManager for obtaining related information. SoundManager is called by GameManager for sound operations. This strong relation and unity among subsystem classes indicate that we have high coherence within our subsystems.

We also aimed and achieved low coupling. Hence, subsystems mostly do not depend on each other. There are weak relations between subsystems therefore, they can act more independently. For instance, only connection between User Interface subsystem and Game Control subsystem is between GameManager and Menu and ScreenManager. Game Control subsystem is connected to Game Model only via relation between LevelManager and Level classes. Hence classes within subsystems mostly do not know about interfaces of other layers. Hence changes in a subsystem generally do not affect other subsystems. Our subsystem architecture demonstrates that we have low coupling.

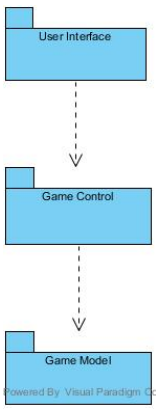


**Figure 61 Basic Layers of Sky Wars**

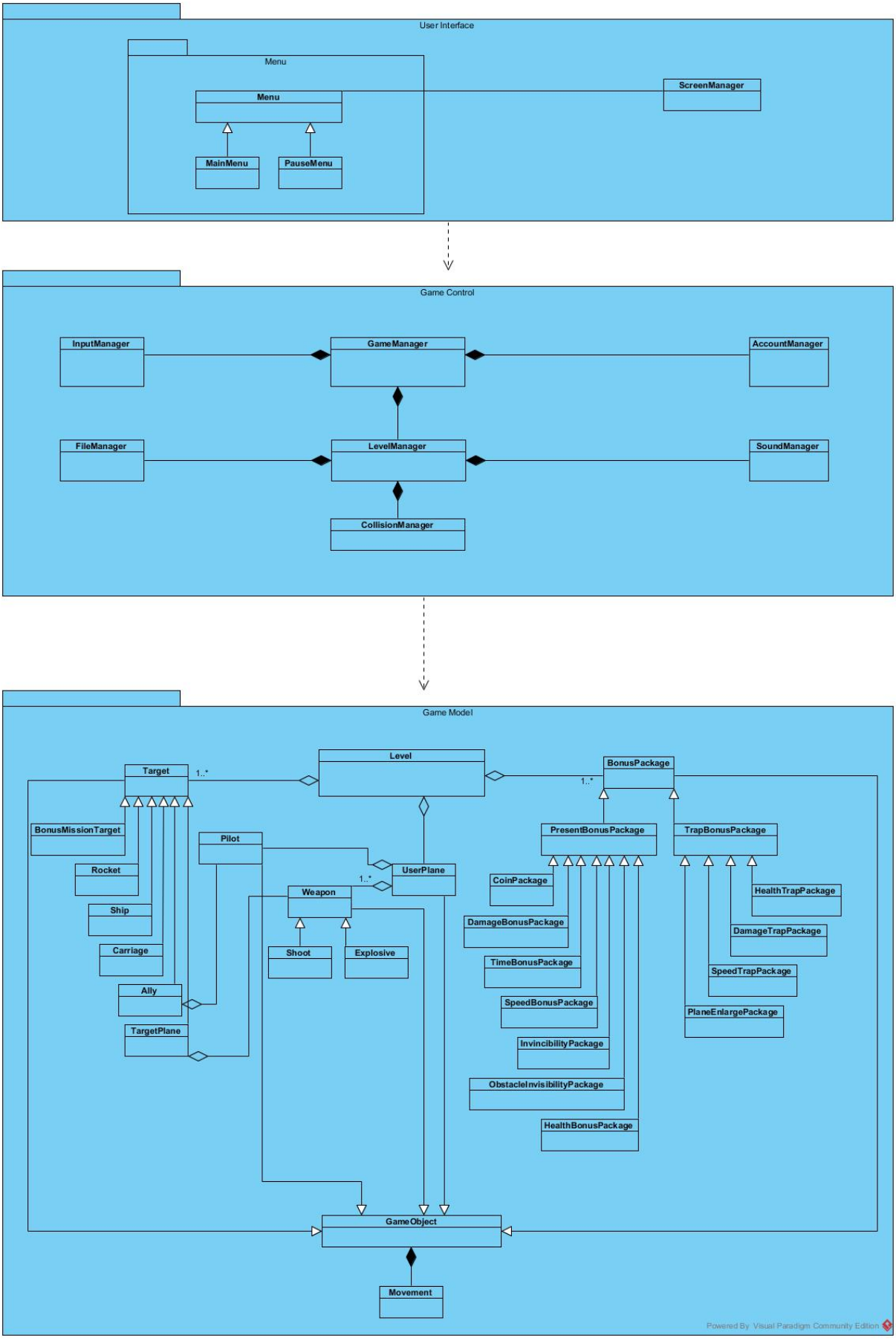


**Figure 62 Sky Wars Subsystem Decomposition with Subsystem Details**

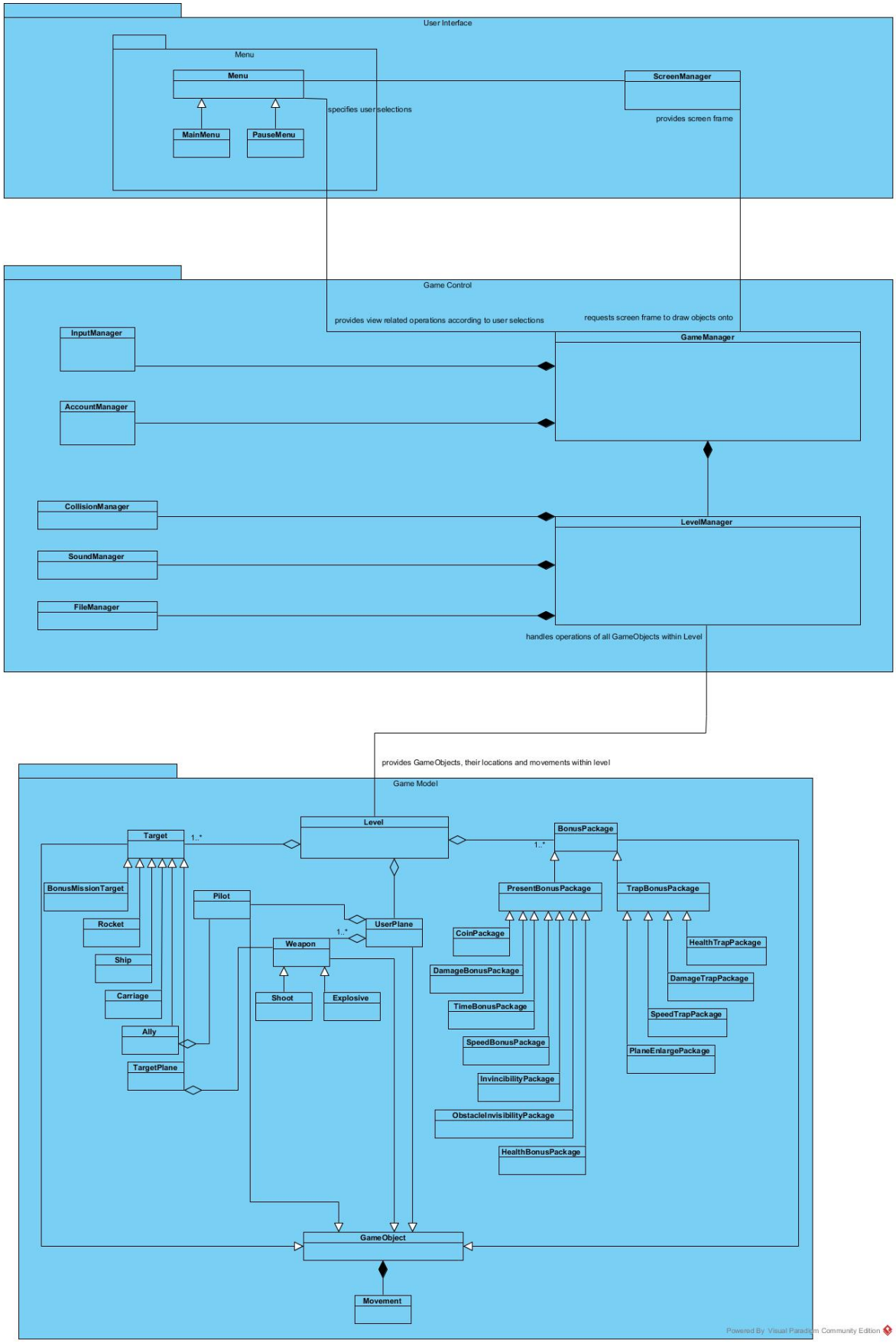


**Figure 63 Subsystem Decomposition with Connection Details**

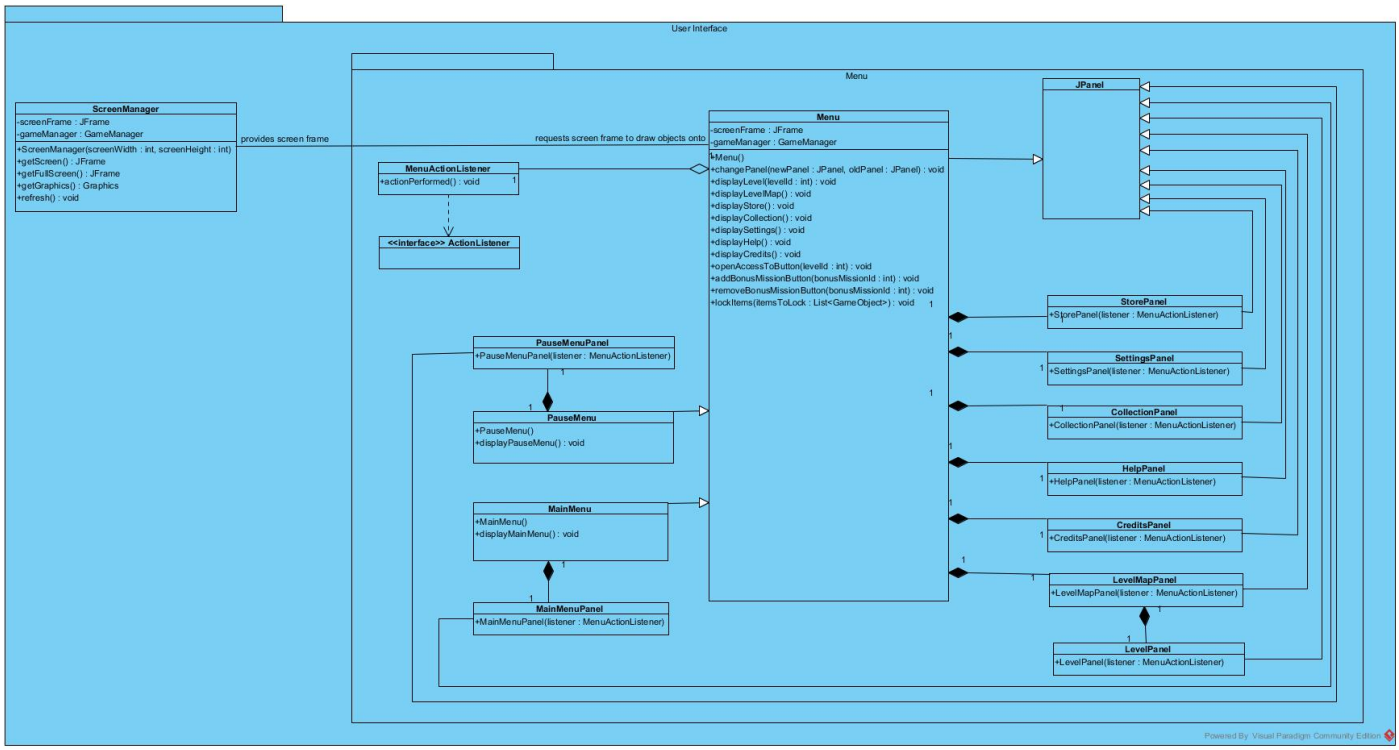


**Figure 64 User Interface Subsystem Details**

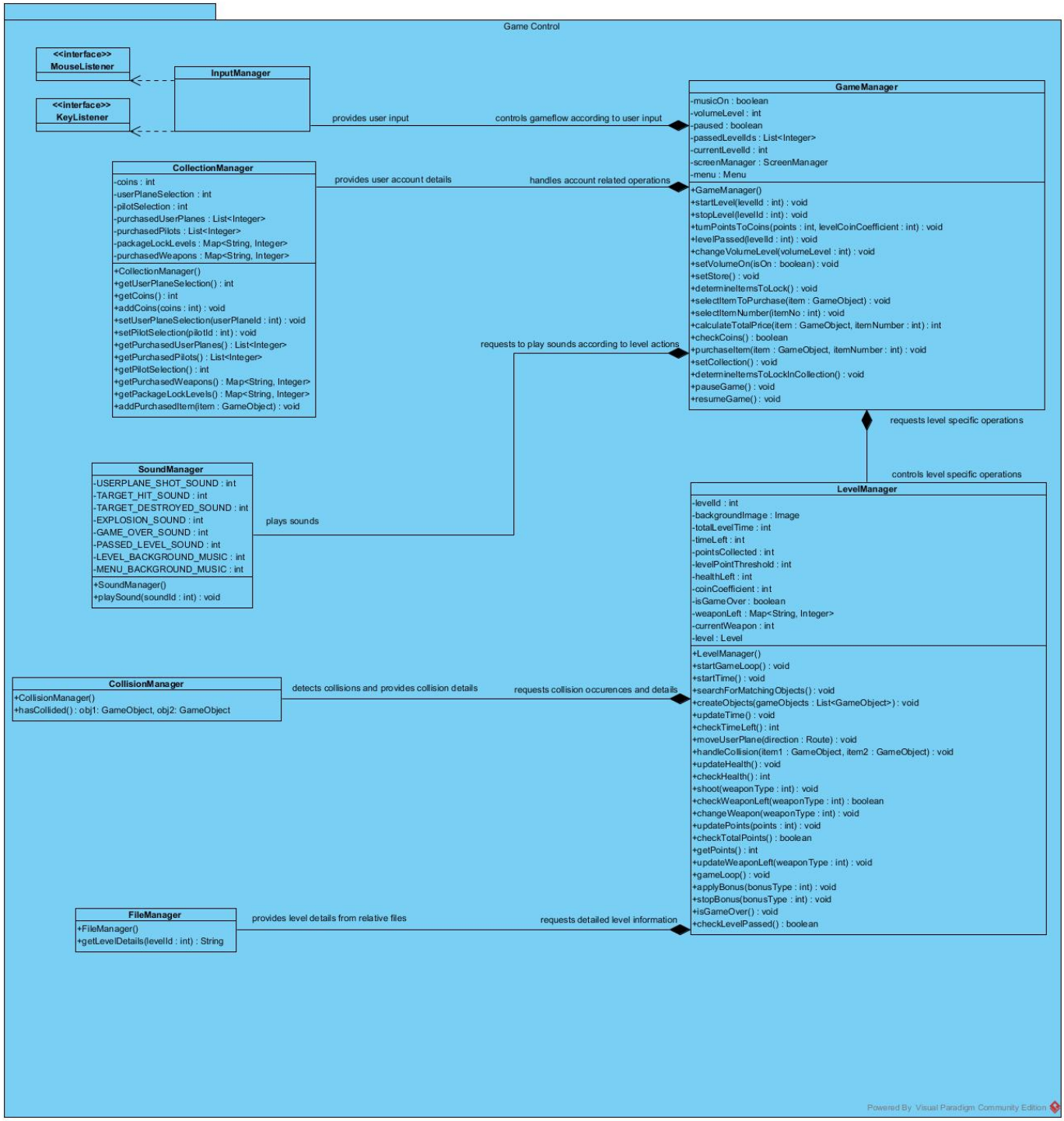


**Figure 65 Game Control Subsystem Details**

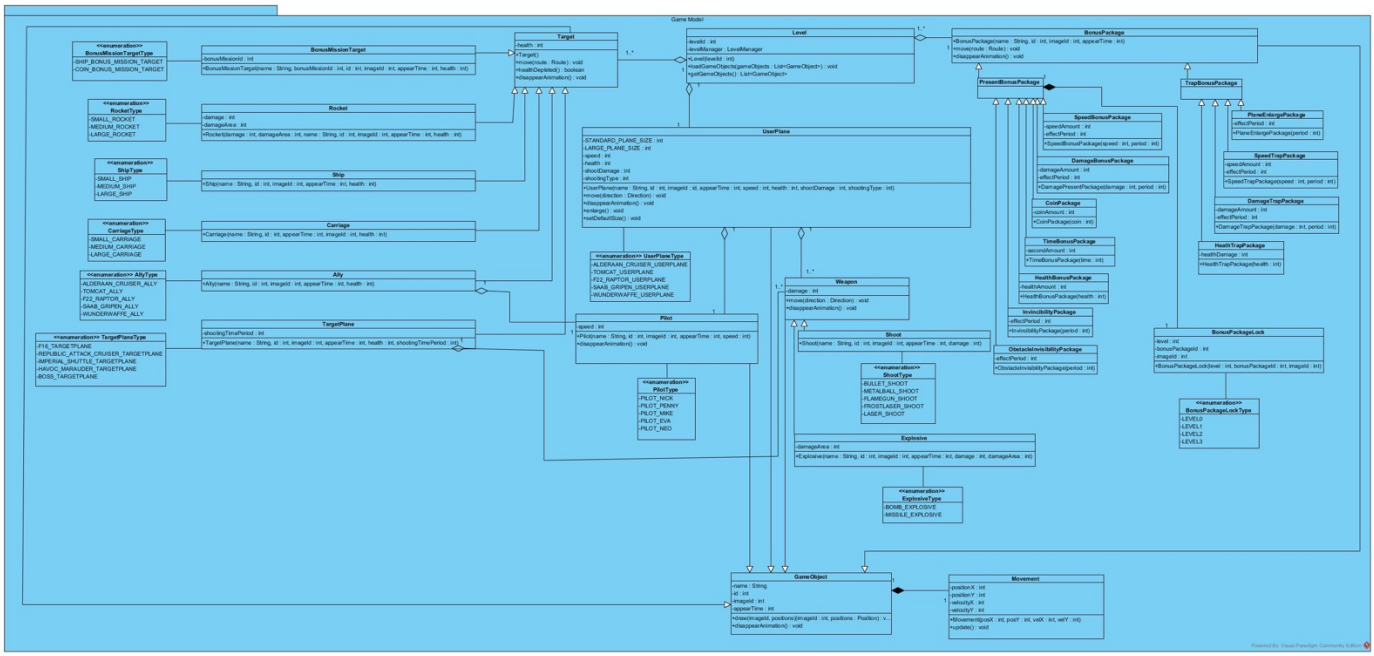


**Figure 66 Game Model Subsystem Details**

# *4.3.Architectural Patterns*

## 3-Tier Architectural Style

We applied 3-Tier Architectural Style for Sky Wars system. Hence the game consists of three hierarchically ordered virtual machines. The first layer is the Presentation Layer. This layer is the layer that communicates with the user. In our application User Interface subsystem is the Presentation Layer. It is the top layer which represents application boundary. The Application layer is the second layer which is the logic part of the system. Game Control subsystem represents the Application level since it is the part which controls the game. The last layer in a 3-layer architecture is Data Layer which provides persistent data of the game. In Sky Wars, Architecture Game Model subsystem is the Data Layer since it holds the game data.
3-tier architecture is appropriate for Sky Wars since the game can be divided to client, logic and data parts. This architectural style makes the game more Manageable and Maintainable.

## Model View Controller Architectural Style

Even though Sky Wars system is decomposed as layers and 3-Tier architecture is the fundamental pattern, we also have a structure which is similar to Model View Controller pattern. The User Interface subsystem contains all View objects of the system. Hence User Interface subsystem acts as the View subsystem which is responsible from displaying game entities to user. The Game Control subsystem corresponds to Controller subsystem naturally, since all Controller classes are in Game Control subsystem. Hence, it acts like Controller and allows communication between Model and View by handling user input, updating models and views. Model subsystem is the Game Model subsystem because it contains all entity objects. It holds the application domain data.

Since Sky Wars can be decomposed to subsystems in a way that entity, boundary and control objects are grouped together, MVC pattern can be followed in system design. Using MVC architecture increases Flexibility, hence it becomes possible to make changes in a subsystem without affecting others.

# *4.4.Hardware/ Software Mapping*

Sky Wars does not have a complex operation system. During level, objects are created, moved, updated and removed. Hence, the computation rate is not demanding for a single processor. One processor will be necessary to maintain the game without flaws. Similarly, there is no need for additional memory space. The game needs keyboard and mouse as external I/O devices for the game control.

Sky Wars is a single user game. Hence, it does not require connection to a server or internet or any other device. Therefore, the only node in the system is UserComputer. The UserComputer should have java compiler installed. Keyboard and Mouse I/O devices are also represented as external nodes.

The components of Sky Wars are associated with subsystems; User Interface, Game Control and Game Model. Game Control component provides game operations interface and User Interface uses these services. Game Model component provides data operations and Game Control uses this interface. This architecture is represented with below diagrams.

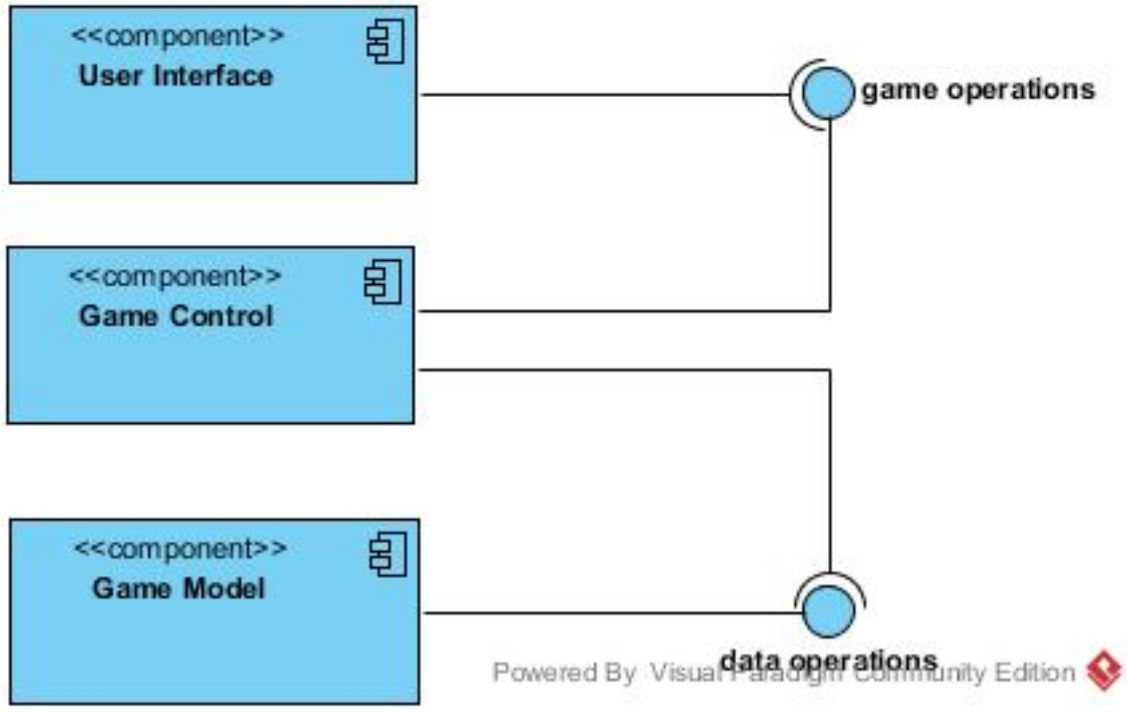


Figure 67 Component Diagram of Sky Wars

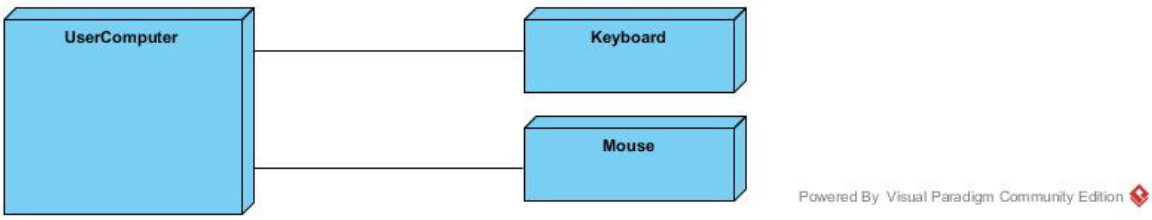


Figure 68 Deployment Diagram of Sky Wars

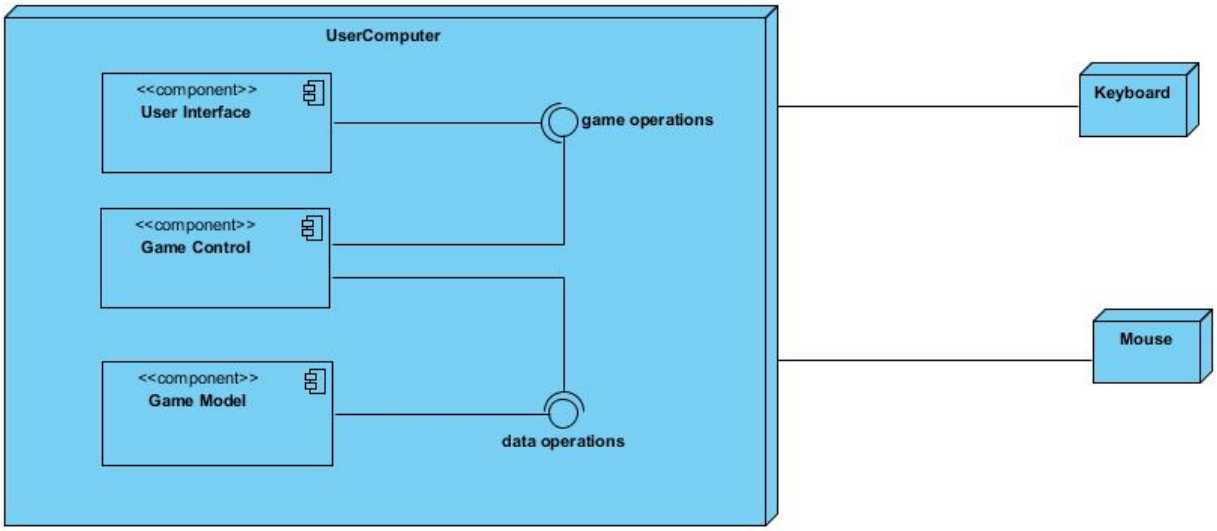


**Figure 69 Detailed Deployment Diagram of Sky Wars**

# *4.5. Addressing Key Concerns*

## *4.5.1. Persistent Data Management*

In Sky Wars, most of the objects are not persistent. Only entities that remain after the game is exited are current level, collection details, bonus level access and settings. The GameObjects and their status within level play will not be saved. Hence for saving, we decided to use file system where objects are saved as text. This data management system will increase efficiency and it will be easy to use since few objects will be recorded. Moreover, we have single writer of the file. Since we do not have multiple users accessing to data storage, it is appropriate to use file system instead of a database. Using a database will increase cost, make development and maintainability harder and it is unnecessary since we do not have a large set of persistent data. Moreover, image and sound files will be saved to drive.

## *4.5.2. Access Control and Security*

As stated earlier, Sky Wars is a single player game. It does not need authentication so anyone with the executable file can start the application and play the game. Since there is only one player and there is no authentication, we did not create an access matrix. All use cases that is all operations defined externally by the system are open to access of any user. Since there is no access control, there cannot be a security issue about the system.

## *4.5.3. Global Software Control*

In Sky Wars we have a fundamental controller class GameManager. Even though there are other controllers they all work under the control of Game Manager. Hence we adopted a Centralized Design for software control. We decided to have an event-driven control for Sky Wars. Since the game is managed according to external events, user control, the software control is based on these events. Therefore, the software system will have a simpler structure and centralized design will be handled within main loop of the game.

## *4.5.4. Boundary Conditions*

**Initialization**

Sky Wars is provided as an executable .jar file. Hence, it does not require installation. It can be ran directly on any computer which has a Java compiler.

**Termination**

Player will be able to quit the game at any point of the game. Close button of the JFrame can be used for system termination. However, if the game is played in full screen mode, Quit buttons on game on Main Menu and Pause Menu screens can be used to exit game. During level, the player should pause the level first and click on Quit from Pause Menu. When the user quits the game, the levels he has passed, the bonus mission level access, the items he has purchased so far and his preferences in Settings and Collection will be saved to file document used for persistent data management. If the user quits in the middle of a level, his progress in level will not be saved and he will be required to start the level over.

**Failure**

Sky Wars contains plenty of image and sound files within. Hence, if initialization of the process is affected from this files and a problem occurs, the game will be started without sounds.

If a system error occurs during level play, the level will be restarted. The progress of the user in level will be lost if such an error occurs, however the game will not need restart.

If a certain screen does not respond to user, the panel will be closed by the system and main screen will be displayed.

If the system crushes, it will be restarted. The game will be automatically saved in certain time intervals to prevent data-loss in a system crash.

# 5. *Conclusion*

In this report the application domain of Sky Wars was examined and its requirements were extracted. These requirements were analysed in order to plan the solution domain. Functionalities of the system were examined and modelled in the report. Moreover, the system design is examined in detail, subsystem decompositions, architectural styles and design decisions are explained.

First of all, general themes and functions of the game were identified. The aim, context and control of the game were specified. Game concepts and items were explained in detail. Hence, a basic understanding of Sky Wars, what is it, how it is played, what are the properties and tasks of the game, was built.

Considering the description of the game, requirements were extracted. What Sky Wars should be able to do for certain functionalities were identified. Furthermore, non-functional requirements and system constraints were specified. In the guidance of requirements, sample scenarios were created for various functionalities. Then these scenarios were generalized as use cases. For a better understanding of the game, interface of Sky Wars was included in the report. Screen designs and pictures of game objects were provided.

Analysis part followed the Requirement Elicitation section. Class diagrams were created, system objects were identified and their operations and attributes were determined. The functional flow is represented with Activity Diagrams. For detailed explanation of interactions among objects, Sequence Diagrams were provided. These UML diagrams created a bridge between application and solution domains.

Last part of the report was System Design. First of all, design goals and trade-offs were explained. The system is decomposed into subsystems and architectural patterns are identified. Deployment and Component diagrams are used for elaborating Hardware/Software mapping decisions. Other system decisions such as data management and software control are explained in detail.