CS 319

Object-Oriented Software Engineering

Project Analysis Report

Defender

GROUP 2-D

Taha Khurram - 21701824

Mian Usman Naeem Kakakhel - 21701015

Muhammad Saboor - 21701034

Sayed Abdullah Qutb - 21701024

Balaj Saleem - 21701041

1. Introduction

In the Defender[1], the Earth is being invaded by aliens and you, as the player, will commandeer a spaceship in an attempt to shoot down all enemy aliens. The player aims to avoid taking damage from the enemy aliens whilst trying to stop them from abducting humans on the surface of Earth. The aliens are abducting humans so that they can absorb their spiritual energy and mutate resulting in a massive boost in power. The aliens aim to capture and kill all humans on the planet as the total spiritual power absorbed will be enough for their entire species to mutate and unlock their full potential becoming extremely powerful with the intention to now destroy the Earth as there are no more surviving humans. It is solely up to the player to save the world and his species from total annihilation.
The defender is an endless runner game that allows the player to control a spaceship and shoot down invading aliens. The player will have to conserve their ammo and prioritize saving the captured humans to prevent the aliens from becoming more powerful and consequently destroying the player's homeworld.

2. Overview

Defender is an endless scrolling shooter game with a modern Graphical User Interface following the modern norms. The game is designed for both Windows and Linux machines. The game will start in full-screen mode. After starting the game the user will be shown a title screen from where they can decide to start the game or to quit the application and return to the desktop. The player will be able to pause anytime in-game where they will be shown a menu allowing the user to access the shop, resume, or quit option.

2.1 Gameplay

The game follows an endless scroller format. The game starts with the player already inside the spaceship, the player will use the direction keys to control the spaceship to avoid incoming enemy fire while simultaneously trying to shoot at the enemy using the shoot button. The player can choose between three attacks, the first consists of normal bullets, the second will be a heat-seeking missile launched towards the nearest enemy at the time of the shooting, and the third will be a nuclear bomb which will kill all on-screen enemies. Some enemies will be trying to abduct humans on the ground (located at the bottom of the screen) and the player will have to stop those enemies by killing them after which they will drop their human and consequently the player will have to control their spaceship to catch them. If the enemy can take the humans to the top of the screen, they will mutate into Crescent (enemy types are explained later).

The game will calculate the player's score as the time spent surviving, and will keep the highest score achieved by the player on top of the screen. The player may also come into contact with warp zones which will randomly transport the player on the map. If all the humans are captured, it will result in all enemies getting mutating into Crescents. The game will continue until the spaceship is destroyed. The player may choose to pause in-game and will have an option to access the shop or resume/quit the game. The player will earn coins by killing enemies and will be able to spend these coins on upgrades

in the shop menu. These upgrades include upgrading the health tank, the fuel tank or changing the skin color of the spaceship.

2.2 Characters

Character in the game includes the player, humans on the ground and the enemy spaceships.

2.2.1 Player Spaceship(Al Zarrar)

The player will be flying a spaceship named “Al Zarrar”, the following features will be related to the player:

1. **Health:** The player’s spaceship will have a health pool and after their health is depleted their spaceship will be destroyed. Initially, the health will be 100 points. The player will be able to increase the health capacity by buying health tanks of 10, 20, 30, or 40 points for 40, 50, 60, or 70 coins respectively from the shop. If the health of the spaceship drops to 0 points, the spaceship will be destroyed.
2. **Fuel:** This will be the amount of fuel left in the spaceship’s tank and if the spaceship runs out of fuel it will crash resulting in the death of the player. The player will be able to increase the fuel tank capacity by buying additional fuel tanks of 10, 20, 30, or 40 points for 40, 50, 60, or 70 coins respectively from the shop. If the fuel of the spaceship drops to 0 points, the spaceship will be destroyed.
3. **Speed:** The spaceship will have a speed of 30 (in game velocity units).
4. **Color:** The initial skin color of the spaceship will be green. However, the player will be able to change the skin colors in exchange for 35 coins. The possible skin colors will be red, blue, yellow, and orange.
5. **Score:** The current score of the player will be shown and will be incremented by 1 each second. The initial score will be 0 whenever a new game is started.
6. **Coins:** The current coin count of the player will be shown and will be incremented by the bounty of each enemy killed. The initial coin count will be 0 whenever a new game is started.
7. **Bullet:** The spaceship will fire blue bullets which will damage the enemies. The speed of the blue bullet is 40. A successful hit of the enemy by the bullet awards 50 points to the player.
8. **Missiles:** The player will be able to see whether the missile is available for use or not. If it is not available, the number of seconds will be shown after which it will be available for use. After firing the missile, the cooldown will be 5 seconds. The speed of the blue bullet is 80. The damage of the missile is 100 points.
9. **Nuclear Bomb:** This weapon will kill all enemies on screen for the player. After using the bomb, the cooldown will be 15 seconds.

2.2.2 Human

The humans do not perform any action themselves and they will only change states between being idle on the ground, being abducted by the “BugEye” enemy, falling, or caught by the spaceship. After killing the abducting enemy if the player fails to catch the human they will fall to the ground and be killed. If the enemy manages to reach the top, the human will die and the enemy will mutate into Crescent.

2.2.3 Enemies

The enemy will work in different ways to destroy the spaceship, ultimately to capture the Earth. We have set the number of enemies at the start of the game according to the screen resolution. If the number of enemies in the game at any point falls below this number, we randomly generate the enemy on the map.

The game has several different types of enemies who will have varying functions:

1. **BugEye:** These enemies are responsible for picking up humans. It will not fire any weapon towards the spaceship. Its health will be 100 points. Its speed will be 20. It will find the nearest human and go towards it to abduct it. After abducting, BugEye will start moving upwards and when it reaches the top, it will mutate itself into Crescent.
2. **Crescent:** This enemy results after the mutation of weaker enemies and will only attack the player. Its health will be 100 points. Its speed will be 30. It will be equipped with OrangeGlow with a damage of 30 points. This enemy will move towards the spaceship and start orbiting around it while shooting the spaceship.
3. **Saucer:** This is a low-level enemy which shoots RedBolt bullets with a damage of 20 points at the player. Its health will be 50 points. Its speed will be 20. The direction of this enemy is randomly decided at generation time and then it continues to move in that direction until it bounces off of the edges of the map and continues in its new bounced direction.
4. **Saucer Blades:** These are stronger versions of Saucers in terms of speed. The speed of Saucer Blades will be 25. Its health and damage to its weapon (RedBolt) will be the same as Saucer. The direction of this enemy is randomly decided at generation time and then it continues to move in that direction until it bounces off of the edges of the map and continues in its new bounced direction.
5. **Slicer:** This is a low-level enemy that only attacks the player. Its speed is 20. Its health will be 50 points. It will be equipped with RedBolt bullets with a damage of 25 points. The direction of this enemy is randomly decided at generation time and then it continues to move in that direction until it bounces off of the edges of the map and continues in its new bounced direction.
6. **Spikey:** These enemies will try to attack the player by getting close to the spaceship and exploding dealing damage of 15 points.

2.3 Map and Zones

2.3.1 The Main Map

The game will have one fixed map in which the player will navigate their spaceship, there will be a minimap overhead in which a portion of the map is shown to show the player their current location on the overall map.

The map will have humans on the bottom of the screen i.e. on the ground, and the enemies will be shown in their respective positions.

2.3.2 Warp Zone

There will be random warp zones generated on the map that will transport the spaceship(player) from one location to another. The warp zone will be generated randomly on the current screen. When the

spaceship collides with the warp zone, it will be teleported to a random location on the map which is not on the current screen.

2.4 Game Frame

The game frame will be used as a hud to show several important information tabs in the game:

1. **Health Points:** The spaceship's health points will be shown in the game frame.
2. **Fuel Points:** The current fuel points will be shown and will depict the status of the player's fuel tank.
3. **Coins:** The number of coins will be shown.
4. **Score:** The player's current score will be shown in the game frame and will be updated over time.
5. **HighScore:** The player will also be able to see their high score on top of the screen. HighScore will always be taken from the save file if available. Otherwise, it will be set to 0.
6. **Missile Cooldown:** The time after which the missile will be available for use to the player will be shown.
7. **Nuclear Bomb Cooldown:** The time after which the nuclear bomb will be available for use to the player will be shown.
8. **Map View:** A minimap will be shown, this minimap will show a boundary around a small section of the map in which the player is located.

2.5 Additional Features:

Additional features of the game which we proposed to do are as follows:

2.5.1 Asteroids

We will implement an asteroid shower event in which several asteroids will fall down the map and the player has to avoid them to assure his/her safety. If an asteroid hits the spaceship, it will deal 50% damage to the spaceship.

2.5.2 Fuel

The spaceship will have a fuel attribute which will be depleted by 1 point every second. The fuel will start from a full tank which will initially have 100 points. The fuel tank can be upgraded from the shop. If the fuel hits 0 points the spaceship will be destroyed.

2.5.3 Weapons

We are providing new weapons to the spaceship:

1. **Missile:**
   When the missile is not available for use, the number of seconds will be shown after which it will be available for use. After firing the missile, the cooldown will be 5 seconds. The speed of the blue bullet is 80. The damage of the missile is 100 points. The missile will target the nearest enemy to the spaceship. If the enemy dies before the missile collides with it, the missile will retain its previously set direction.
2. **Nuclear Bomb:**

This weapon will kill all enemies on screen for the player. After using the bomb, the cooldown will be 15 seconds.

2.5.4 Pickups

These pickups will provide the user with essential status recovering items: health and fuel. These items will drop down the screen randomly.

1. **Health:**
   The player will be able to utilize a health pickup which will restore the space ship's health by 10 points and if the player's health is already full it will not affect the spaceship's health.
2. **Fuel:**
   The player will be able to utilize a fuel pickup which will restore the space ship's fuel by 70 points and if the player's fuel is already full it will not affect the spaceship's fuel.

2.5.5 Shop

The game will include a shop feature. It will be accessible from the pause menu. It will have 3 categories of items:

1. **Health Upgrades:**
   The player will be able to increase the health capacity by buying health tanks of 10, 20, 30, or 40 points for 40, 50, 60, or 70 coins respectively from the shop.
2. **Fuel Upgrades:**
   The player will be able to increase the fuel capacity by buying fuel tanks of 10, 20, 30, or 40 points for 40, 50, 60, or 70 coins respectively from the shop.
3. **Skin Mutation:**
   The player will be able to change the skin colors in exchange for 35 coins. The possible skin colors will be: red, blue, yellow, green, and orange.

3. Requirements

3.1 Functional Requirements

The functional requirements of the game are explained below. After the game is opened and the main menu is displayed, the user/player can do any of the below:

1. **Start a New Game:**
   Player can start a new game in which the player's health, missile count and bomb count will be set to full as explained in the overview. The score will be initialized to 0 and increases as the game progresses. Humans will be spawned on the earth in different positions randomly. Enemies will also be randomly spawned on the map, and the enemies will keep spawning as the already spawned enemies die. The number of spawned enemies and humans depends on the screen resolution.

The following functional requirements occur after 'Start a New Game' is pressed and are as follows:

2. **Move the Spaceship:**

This option gives the player the ability to move the spaceship around using the arrow keys on the keyboard. The spaceship will move in any direction the user wants to until the fuel runs out or the spaceship is damaged enough so that the health becomes empty. (Details of the spaceship in section 2.2.1, and details of fuel in section 2.5.2).

3. **Fire Bullets and Missiles:**
   As the player moves the spaceship around, enemies start appearing into the screen, then the user can fire blue bullets at those enemies and prevent the enemies from hitting back at the spaceship. (Refer to section 2.2.3 for Enemy details). The player can also fire missiles if the spaceship has any available missiles. (Refer to section 2.2.1 and 2.5.3 for more info on bullet and missile damage)
4. **Use Nuclear Bomb:**
   There might come times when the screen is full of enemies, so then the user can use the Nuclear Bomb ability of the spaceship and clear the screen of all enemies. This also depends on the availability of the nuclear bomb. (Refer to section 2.5.3 for more info on the nuclear bomb)
5. **Use Pickup Items:**
   There are extra items, called Pickup Items, which the user can get and boost the spaceship further. These pickup items are health and fuel, which are spawned at random times and in random places throughout the map. (Details on Pickup Items is in Section 2.5.4)
6. **Warp to a location:**
   Warp zones are randomly picked areas on the map which the player can enter from one location and exit from another location. (Details of the Warp zone is explained in section 2.3.2)
7. **Pause Menu:**
   The player may enter this menu mid-game after pressing the pause button(ESC key). The Pause Menu will provide the player with 4 different options, including ‘Continue Game’, ‘Main Menu’, ‘Shop Menu’, and ‘Quit Game’.
8. **Shop Menu:**
   The shop menu is located in the pause menu and the player will be allowed to spend their coins earned here on different upgrades for the spaceship. (Refer to section 2.5.5 for more info on Shop)
9. **Quit:**
   This option will exit the game. If the player presses on the Quit Game button in the Main Menu, the game will quit itself. Also if the player is mid-game and presses the Pause button (ESC key), the player can close the game by pressing the Quit button.

3.2 Non-Functional Requirements

3.2.1 Performance

To ensure an enjoyable game experience the performance of the game is to be kept optimal,i.e. At all times, the frame rate will not drop below 30 frames per second, on an intel core 2 duo or higher processor machine. Rendering will be done according to the camera frame, enabling low resource usage and better overall performance, in other words, items out of camera display will not be rendered. Furthermore, the upper limit of objects visible on the screen will be set to 50, including bullets, enemies, humans, to achieve higher performance.

3.2.2 Usability

Menus, title screens (possibly splash screens) and other such scenes are to be designed in a way to ensure accessibility and user-friendliness, by using fewer than 5 buttons on any available menu. Basic start and pause menus are provided to the user to allow continue gameplay at his/her discretion. Throughout the game no more than 8 keys will be used, to allow for easy understanding of controls.

3.2.3 Player Engagement

Features, such as currency, are added to ensure the player engages with the game for longer periods due to increased variation and strategies. Furthermore, to create a form of competition the player will have to manage their health and fuel points to achieve the highest score possible which will be shown at the top of the screen, the player will aim to achieve a higher score in future sessions. On average it is expected that the user will play the game for more than 1 game session.

3.2.4 Extendibility

The codebase is to be written such that new features to the game are easily includable. These extended features may in the future include the following 5 items:

Improved mechanics (better shooting and spaceship control), improved graphics (better sprites and animations), variety and functionality of enemies, spaceship selection, planet selection. Hence the base classes are to ensure that that these features are added with a minimum amount of rework.

4. System Models

4.1 Use Case model

Use Case diagram has one primary external actor controlling the spaceship.

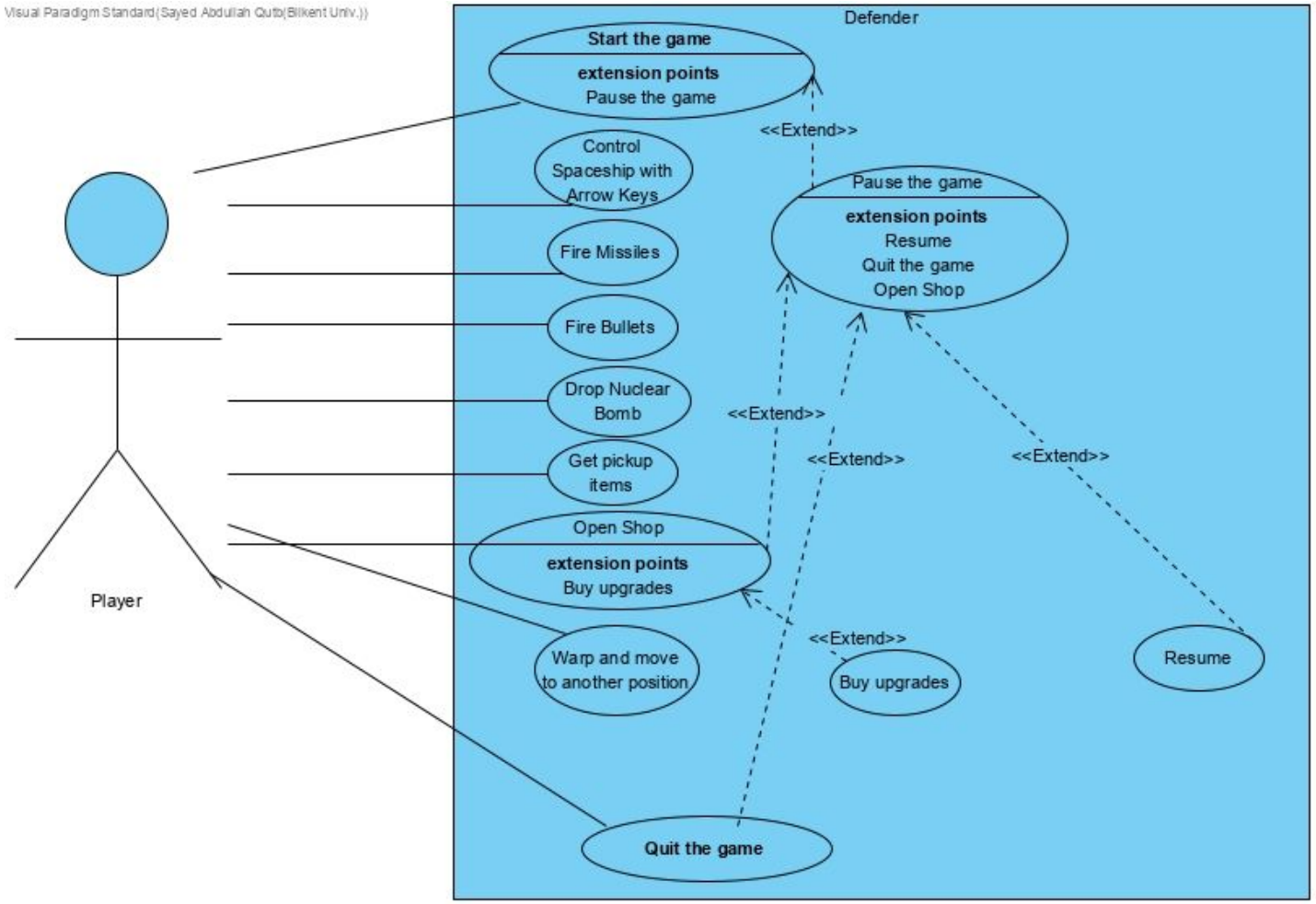


Figure 1 (Use Case Diagram)

4.1.1 Use Case Descriptions

4.1.1.1 Start the game

Name: Start the game

Participating Actor(s): Player

Entry Condition(s):

- The player opens the game

Exit Condition(s):

- The player starts playing and controlling the spaceship in the game

Flow of events:

a. The player opens the game
b. The game window shows up
c. Player clicks on “Start”
d. Player gets to see the spaceship, the enemies, and the humans

Special Requirement(s):

- The player can pause the game by pressing the ESC button on the keyboard.

4.1.1.2 Control Spaceship with arrow keys

Name: Control Spaceship with arrow keys

Participating Actor(s): Player

Entry Condition(s):

- The player starts the game

Exit Condition(s):

- The player ends up controlling the spaceship and might fire missiles
- Spaceship moves from one spot to another

Flow of events:

a. The player opens the game
b. The game window shows up
c. Player clicks on “Start”
d. Player gets to see the spaceship, the enemies, and the humans
e. The player presses any of the arrow keys on the keyboard and the spaceship starts moving

Special Requirement(s): None

4.1.1.3 Fire Missiles

Name: Fire Missiles

Participating Actor(s): Player

Entry Condition(s):

- Player starts the game
- Player presses “Space” key

Exit Condition(s):

- Player fires a missile which travels from the spaceship and hits either an enemy or the missile is wasted
- Missile comes out of the spaceship
- If the missile hits the enemy, player gets coins
- Missile is more powerful than the bullet and causes more damage to the enemy

Flow of events:

a. Player opens the game
b. Player clicks on “Start”
c. Player may adjust the spaceship according to the enemies’ position
d. Player presses a key to fire the missile and the missile starts traveling towards the player
e. Missile impacts the enemy if it hits them
f. Player gets coins for successful hits

Special Requirement(s): None

4.1.1.4 Fire Bullets

Name: Fire Bullets

Participating Actor(s): Player

Entry Condition(s):

- Player starts the game
- Player presses “Space”

Exit Condition(s):

- A bullet ends up going from the spaceship towards the enemies’ side
- Bullet either hits the enemy or is wasted
- Successful hits result in coins being awarded to the player

Flow of events:

a. Player opens the game
b. Player clicks on “Start”
c. Player controls the spaceship and moves it according to the enemies’ position
d. Player fires a bullet by pressing the space key on the keyboard
e. A bullet is being fired from the spaceship and goes towards the enemies
f. Bullet either hits the enemy or hits nowhere
g. Successful hits give player coins

Special Requirement(s): None

4.1.1.5 Drop Nuclear Bomb

Name: Drop Nuclear Bomb

Participating Actor(s): Player

Entry Condition(s):

- Player starts the game

Exit Condition(s):

- The game screen ends up being free of enemies
- All enemies are gone for a small amount of time

Flow of events:

a. Player opens the game
b. Player clicks on “Start”
c. Player drops a “Nuclear Bomb” missile and the screen is cleared and empty of enemies.

Special Requirement(s): None

4.1.1.6 Get pickup items

Name: Get pickup items

Participating Actor(s): Player

Entry Condition(s):

- Player starts the game

- Pickup item is dropped from above

Exit Condition(s):

- The player picks up the falling item.
- The spaceship is refueled if the pickup item is fuel.
- The spaceship regains full health if the pickup item is health.
- The player might also get upgraded missile type.

Flow of events:

a. Player opens the game
b. Player clicks on “Start”
c. Player plays until a pickup item is dropped from the top and the player catches it

Special Requirement(s): Pickup items are dropped at most one at a time, either health or fuel item.

4.1.1.7 Open shop

Name: Open shop

Participating Actor(s): Player

Entry Condition(s):

- Player starts the game
- Player has enough coins to buy an item

Exit Condition(s):

- The player gets to see shop items ready for sale
- The player might buy a shop item if he has enough coins.
- He might buy an upgraded missile or fuel for the spaceship if he has enough coins.

Flow of events:

a. Player opens the game
b. Player clicks on “Start”
c. Player presses “Esc” key and the game menu appears
d. Player clicks on Shop and the shop open

Special Requirement(s): Player needs coins if he wants to buy items from the shop.

4.1.1.8 Warp and move to another position

Name: Warp and move to another position

Participating Actor(s): Player

Entry Condition(s):

- Player starts the game
- Player moves to the warp zone

Exit Condition(s):

- Player comes out of the warp zone in a different position than before.

Flow of events:

a. Player opens the game
b. Player clicks on “Start”
c. Player moves the spaceship towards the warp zone
d. Player enters the warp zone and is transported to a different position on the game screen.

Special Requirement(s): None

4.1.1.9 Quit the game

Name: Quit the game

Participating Actor(s): Player

Entry Condition(s):

- Player opens the game

Exit Condition(s):

- Player exits from the game

Flow of events:

a. Player opens the game
b. Player clicks on “Quit”

Special Requirement(s): None

4.1.1.10 Pause the game

Name: Pause the game

Participating Actor(s): Player

Entry Condition(s):

- Player starts the game

Exit Condition(s):

- The game is paused
- Game menu is shown to the player

Flow of events:

a. Player starts the game
b. Player presses “ESC” key to pause the game

Special Requirement(s): None

4.1.1.11 Buy Upgrades

Name: Buy Upgrades

Participating Actor(s): Player

Entry Condition(s):

- Player starts the game
- Player pauses the game by pressing “ESC” key

Exit Condition(s):

- Player is equipped with upgrades

Flow of events:

a. Player starts the game
b. Player presses “ESC” key to pause the game
c. Player opens Shop
d. Player clicks on a shop item and buys if he has enough coins

Special Requirement(s): Player needs coins to buy upgrades

4.1.1.12 Resume

Name: Resume

Participating Actor(s): Player

Entry Condition(s):

- Player starts the game
- Player pauses the game by pressing “ESC” key

Exit Condition(s):

- Game on!

Flow of events:

a. Player starts the game
b. Player presses “ESC” key to pause the game
c. Player clicks on “Resume”
d. Game is continued again

Special Requirement(s): None

4.2 Dynamic models

4.2.1 Sequence Diagrams

The game’s models consist of 2 sequence diagrams, one is for the scenario in which the spaceship is hit by orange glow weapon from Crescent enemy and then the spaceship picks up a health pickup to restore its health. The other one is for the scenario in which the human is abducted and the spaceship tries to save it from its death.

4.2.1.1 Spaceship is hit by OrangeGlow and picks up a health pickup

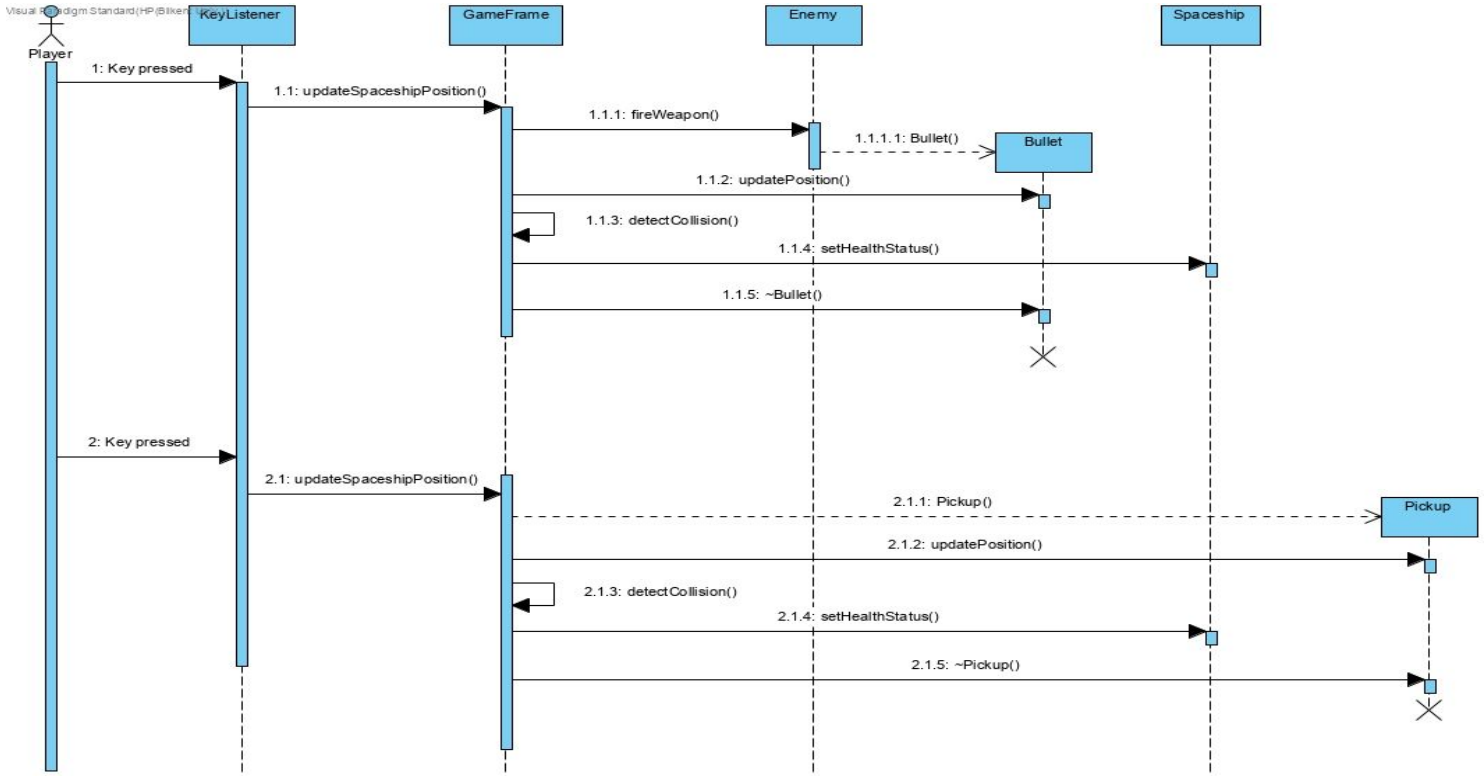


Figure 2 (The spaceship is hit by enemy fire and later gets a pickup)

Scenario:
The player is moving the spaceship when they get shot by the enemy (Crescent), the bullet shot by the enemy (orange glow) manages to hit the spaceship so the spaceship takes damage and its health is reduced.
After this, the player moves the spaceship around and the game frame creates a health pickup object whose position is changing. When the spaceship collides with the health pickup the health of the spaceship is recovered and the health pickup is removed from the game.

Explanation:

1. When the player presses a key (right arrow key) the key listener detects it and the game frame calls the updateSpaceShipPosition() class to move the spaceship. Next, the enemy (crescent) calls the fireWeapon() method and shoots a bullet (orange glow) at the player. The position of the bullet is updated using the updatePosition() method and when the bullet collides with the spaceship the gameframe calls the detectCollision() method following which it calls the setHealthStatus() method to change the health of the spaceship as it just got hit. After this, the game frame destroys the bullet object as it has already collided with the spaceship by calling the ~Bullet() method.

2. When the player presses a key (left arrow key) the key listener detects it and the game frame calls the updateSpaceShipPosition() method to move the spaceship. Next, the game frame

creates a pickup object in the game. The position of the pickup object is updated using the updatePosition() method in the game frame and if the spaceship collides with the pickup the detectCollision() method is called by the game frame after which the game frame calls the setHealthStatus() method on the spaceship to increase its health. As the pickup had been used the game frame at the end calls the ~Pickup() method and removes the pickup object from the game.

4.2.1.2 Sequence Diagram For Human Abduction, and Death.

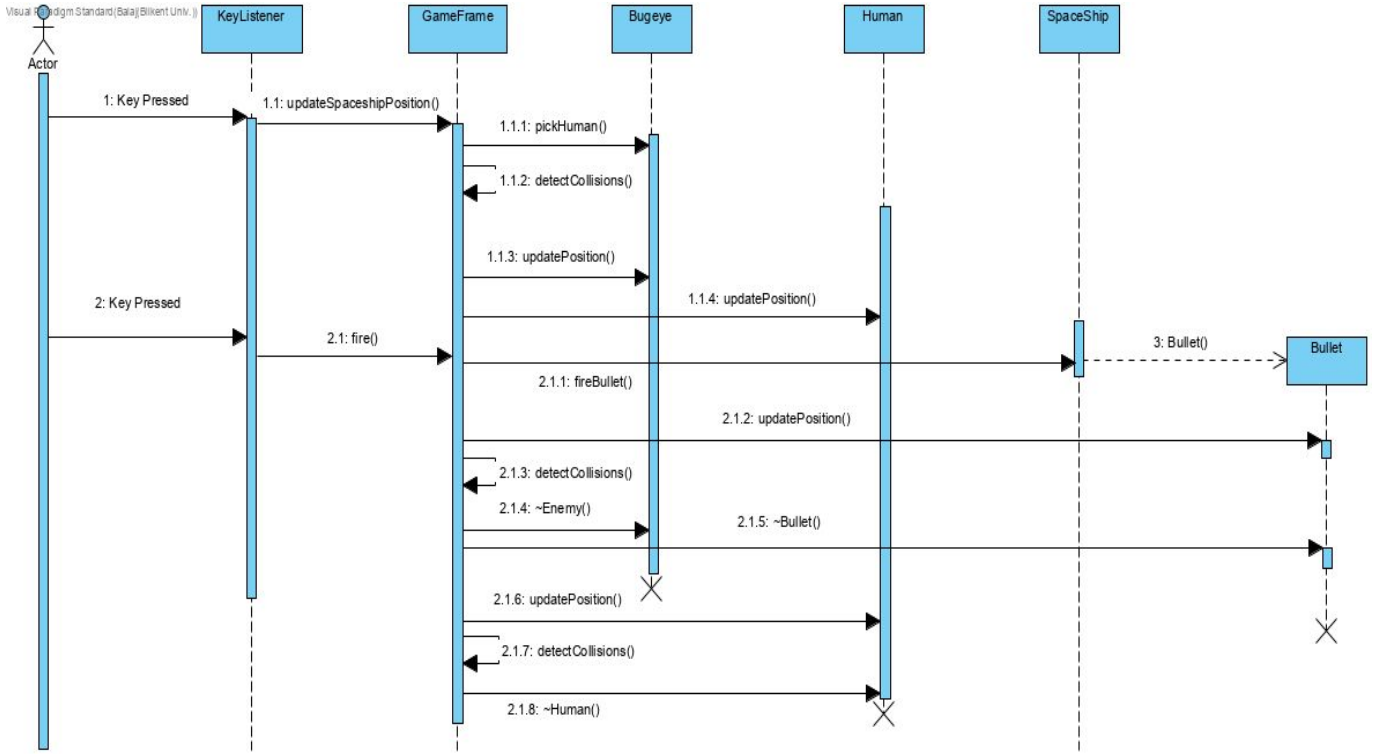


Figure 3 (A human is abducted, spaceship failed to rescue, humans died)

Scenario:
The player is moving the spaceship, when BugEye (a subclass of Enemy) picks up a human, upon collision with it. Bugeye's position is updated as it starts to move up with the human, the human's position updates along with this. The player (spaceship) thereafter fires to hit the bugeye this destroys both the bullet and the Bugeye; the human is released and moves towards the ground. The human collides with the ground and dies.

Explanation:

1. The player presses an arrow key (up), the keyListener receives this and calls the updateSpaceshipPosition method of the game frame. The pickHuman() method of bug eye is called after this, the game frame calls detectCollisions() on itself to determine when the bug eye collides with the human to pick it up. After this updatePosition() of both bug eye and human is called by the game frame as they start to move up.

2. The player presses Z after this to fire a bullet, the keyListener detects this and calls the fire() method of the game frame which in turn calls the fireBullet() of the SpaceShip, this creates the bullet class through its constructor. The updatePosition method of bullet() is also called by the game frame as the bullet travels. Thereafter the game frame detectsCollisions() between the bullet and the bugeye, and calls destructors of both classes (~Enemy() and ~Bullet()). The human starts to fall after bug eye is destroyed and the updatePosition() of Human is called by the GameFrame. Finally, a collision between the ground and the human is detected by detectCollisions() of game frame, and the destructor of Human class ~Human() is called by GameFrame.

4.2.2 Activity Diagrams

4.2.2.1 Pickup Generator

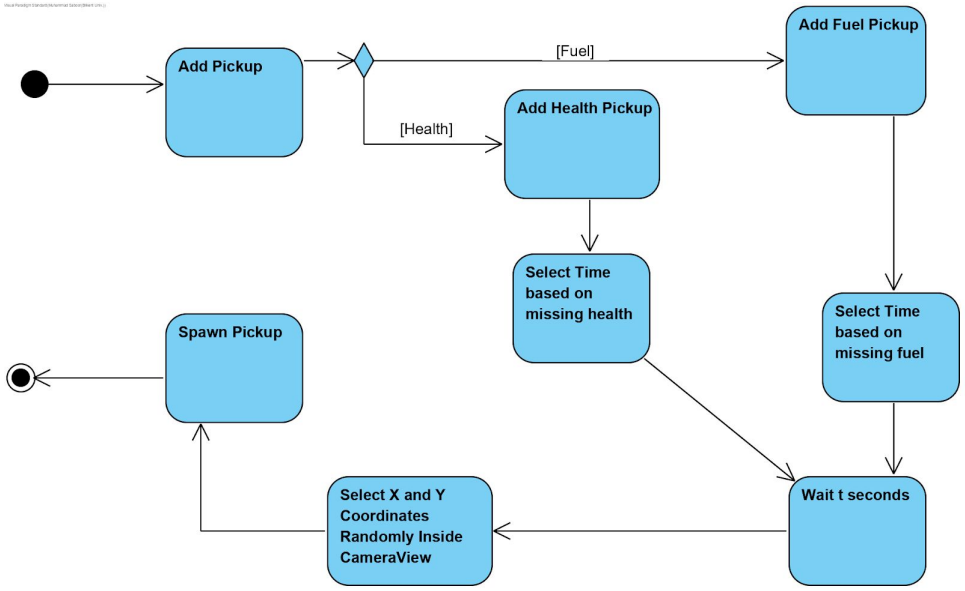


Figure 4 (Pickup Generation Activity Diagram)

Add pickup is an activity concerning the random generation of pickup items on our map. Simply put these items will be dispensed at random locations and the frequency of these spawns will depend on the situation that the player is in. The activity starts when the addPickUp() method is called. Since we have 2 kinds of pickups that will be made available to the player during the game, the diagram is branched into 2 parts: health and fuel. If we are dispensing a Fuel item, we will first see the fuel condition of the spaceship and then will apply an algorithm that will calculate the time which we have to wait before dispensing a fuel item. If the fuel is very low then this time will be very low as well and if the fuel is not at a dangerous level, then the time will be a little higher than before.In other words, this determines how critically the player(spaceship) needs fuel, and dispenses fuel accordingly. Thus creating a proportional relation with the current fuel level. The same algorithm is applied for dispensing a health pickup. After this random x and y coordinates are selected inside the camera view boundary and the pickup is dispensed.

4.2.2.2 Spaceship Collision

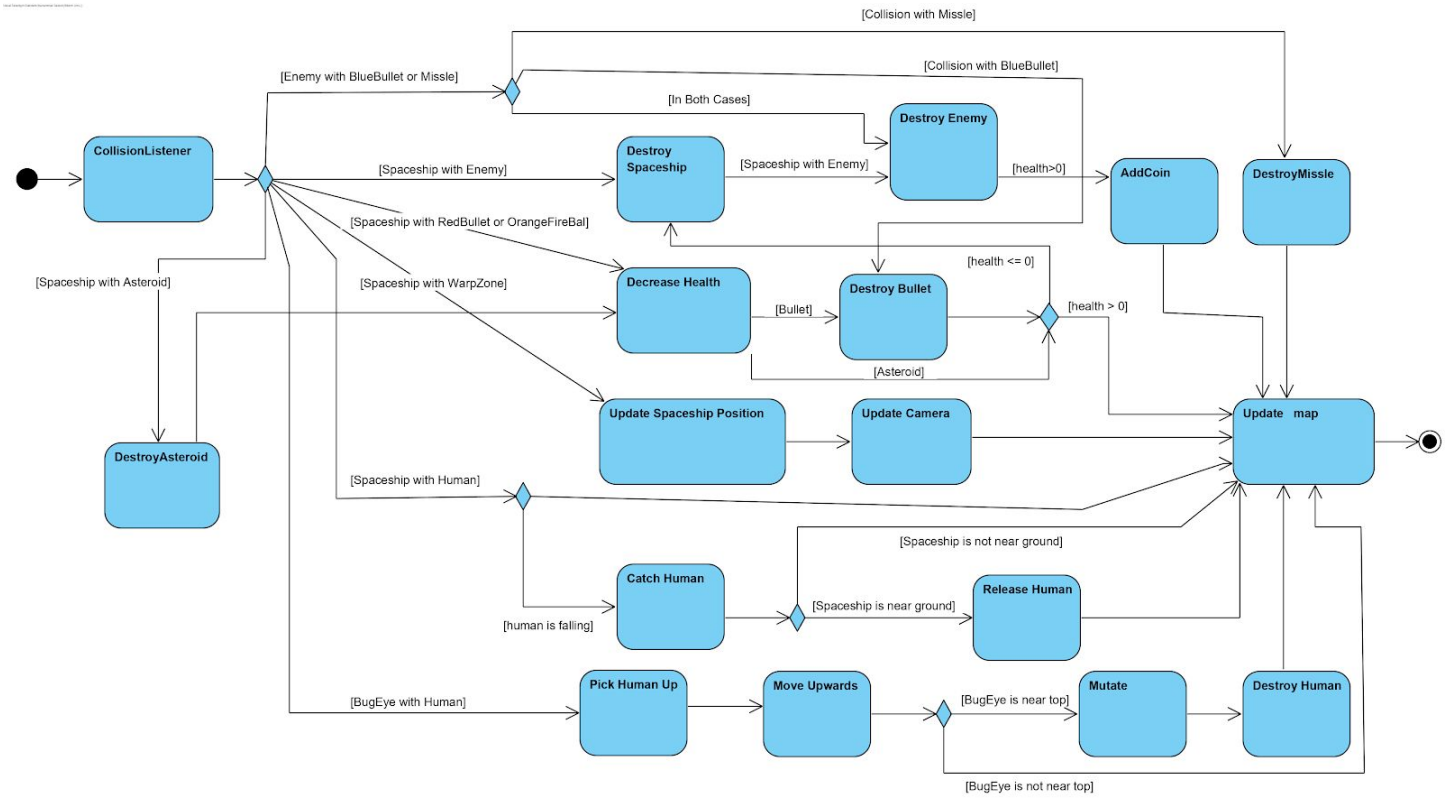


Figure 5 (Collision Detection Activity Diagram)

The description of the activity diagram is as follows:

1. The first action is a collision between spaceship and player, in this case, the system destroys both the spaceship and the enemy.
2. The second interaction is a collision between the spaceship and (enemy) bullet, in this case, the system must decrement spaceship health and destroy the collided bullet (furthermore if the bullet decrements the spaceship's health to less than or equal to zero, then the system must destroy the spaceship).
3. The third activity is a collision between spaceship and warp zone, the system handles this by updating the spaceship position and updating the camera position to follow the player accordingly.
4. The spaceship may also fire a bullet, in this case, if a collision is detected between the enemy and the bullet the system must destroy the bullet and decrement health of the enemy.
5. The spaceship may fire missiles as well and the missile after colliding with the enemy will be destroyed along with the enemy.
6. If a falling human collides with a spaceship the system must allow the spaceship to catch it and when it reaches the ground the spaceship automatically releases the human.
7. The bugeye (enemy class) may also collide and with a human, pick it and then move upwards, in that case, if it reaches the top of the camera view the human bugeye mutates (into the stronger enemy) and the human is destroyed.
8. If the bugeye is destroyed whilst carrying a human, that human will be dropped and put in the falling state.

9. We will also have asteroids on the map and if a collision occurs the spaceship will take heavy damage while the asteroid will be destroyed.
10. The game includes health and fuel pickups if the player collides with any of these items the player will have their stats restored and the pickup will disappear.
11. Upon death enemies will drop coins, however, the player does not need to collide with these coins and the coins will disappear into the player's inventory shortly after being dropped by enemies.

    For each of the aforementioned events, the map and camera are updated.

4.2.3 State Diagrams

4.2.3.1 Human State Diagram

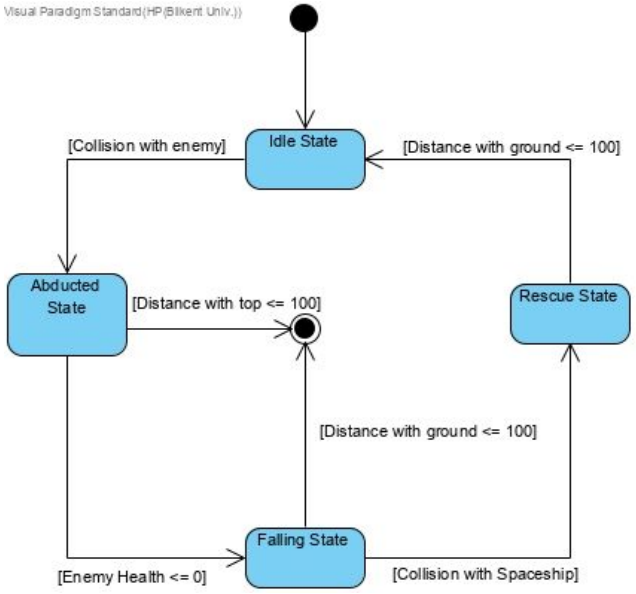


Figure 6 (Human state diagram)

When a human is created inside the game, he will be placed on the ground and will be in the idle state. If the bugeye enemy collides with the human it will pick up the human and start moving up towards the top of the screen, during this the human is in the abducted state. If the enemy bugeye is able to carry the human till the top of the screen the human will be killed however, if the spaceship is able to kill the enemy bugeye by reducing its health to 0 then the enemy will drop the human, following this the human will be in the falling state in which the human is falling down the screen. If the human reaches the ground in this state it will be killed but if the spaceship collides with the human mid air then the spaceship will start to carry this human and the human will be in the rescue state. When the spaceship carrying the human reaches the ground the human will be dropped off and will now be located on the ground which will result in the human going back to the idle state.

4.2.3.2 Spaceship Health State Diagram

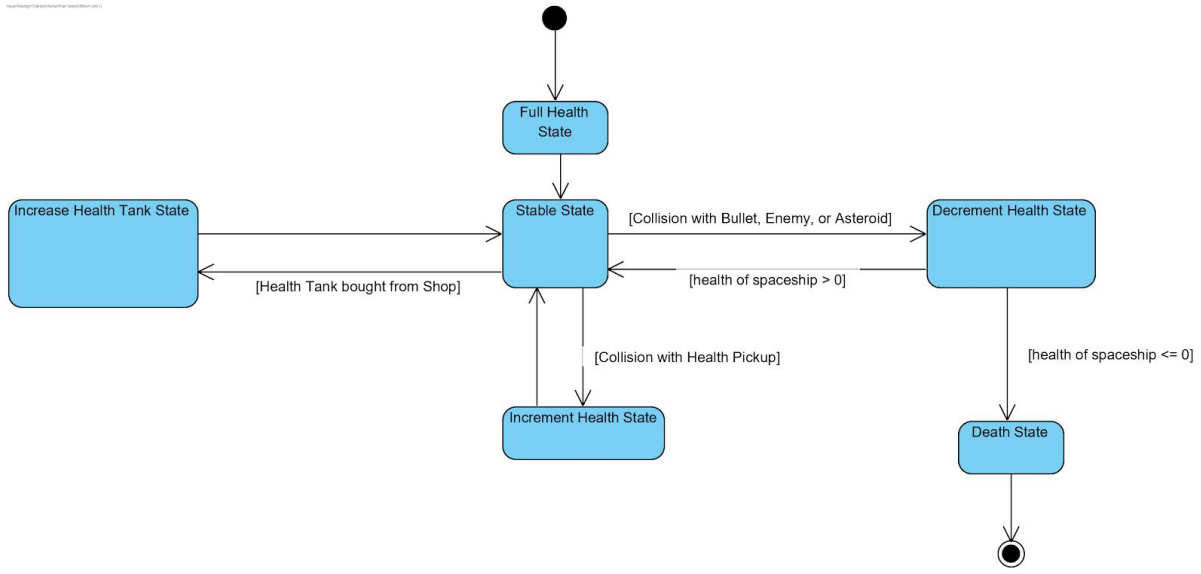


Figure 7 (Spaceship's Health State Diagram)

When the game starts, the initial state of the health of the spaceship will be such that it will have full health (100 points). The state will move to a stable state unconditionally. The stable state is such that It will neither increase nor decrease the health of the spaceship. When the spaceship collides with either enemy, bullet or asteroid, it will move into a decrease health state where the health of the spaceship will decrease by the amount of the damage specified by the type of collision. From that state, if the health of the spaceship is less than or equal to zero go to death state and then end-state diagram. From the decrease health state, if the health is greater than 0 then go to a stable state. If the spaceship is in a stable state, and it collides with a health pickup, it goes to increase the health state which increases the health of the spaceship by the amount specified by the pickup. Then it goes back to a stable state. If a health tank is bought from the shop, it goes to increase the health tank state and increase the size of the health tank according to the size bought from the shop. Then it returns to a stable state.

4.2.3.3 Coin State Diagram

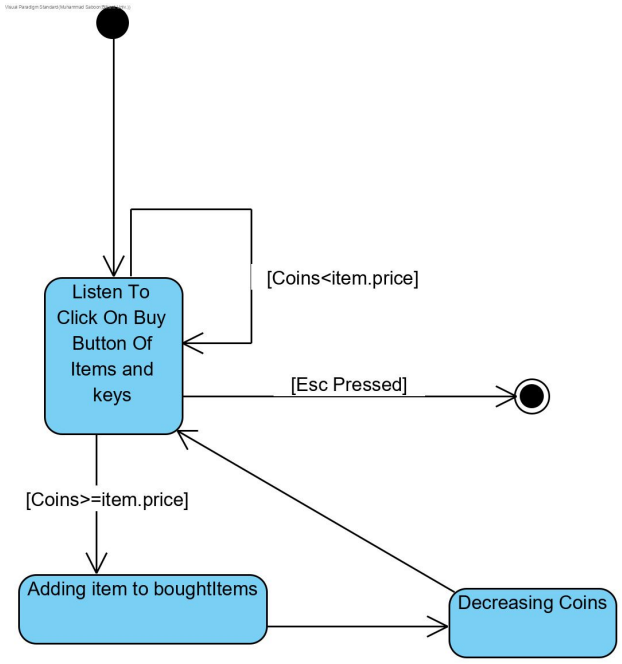


Figure 8 (Coin State Diagram)

In the beginning, the game will listen to the buttons pressed by the user, if the user pressed the button to buy an item and the number of coins is less than the price the game will keep listening and the purchase will not be made. If the user has coins more than or equal to the price of the item, the item will be bought and the number of coins will be decreased.

4.3 Object and Class model

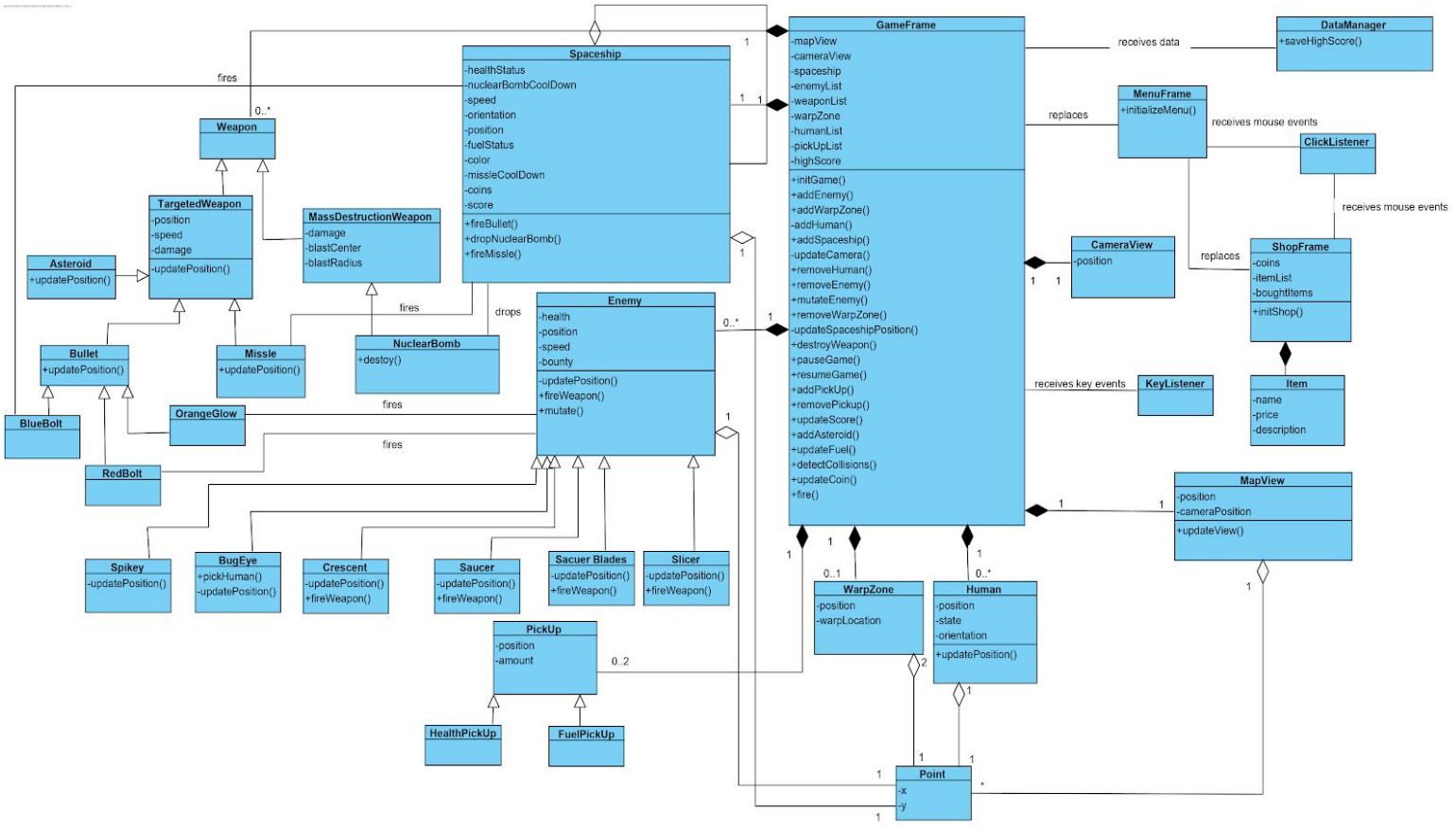


Figure 9 (Object and Class Diagram)

This diagram assumes that constructors, getters, setters, and destructors are present for every class.

This diagram explains the whole class structure of any problem domain components participating in the game. It can be seen, the main class that controls everything in the game is the GameFrame. This diagram assumes that constructors, getters, setters, and destructors are present for every class.

GameFrame is the main class that will be instantiated when a new game is started. This class has two jobs, one is display everything on the screen and the other is to control the flow of the game and bridge the gap between all the other components of the game. GameFrame has many methods that help in initializing the many components we want in the game such as the spaceship, enemies, humans, asteroids, pickups, etc. This class also handles the generation of new enemies, pickups, and asteroids. It also handles the event from the key listener and does appropriate tasks depending on the user input. It also provides detectCollision() method which performs actions based on different collision in the game (e.g. between human and spaceship).

Spaceship is a class that is initialized at the start of the game and it can be controlled by the player. This class has some attributes such as health, fuel, orientation, position, score, coins, etc. These help the placement of the spaceship on the map and tracking the progress of the player and whether they crashed the plane or not. This class also has some methods such as fireBullet(), dropNuclearBomb(),

etc. These methods help in accomplishing the different tasks of the spaceship such as firing weaponry. This class can not exist without the game frame and has to be instantiated in it.

Enemy is a class from which all of the enemy types are inherited. It has a lot of methods and attributes which are common among all of the enemy types such as health, speed, position, and bounty of the enemy the same as the spaceship. It also has fireWeapon() and mutate() to complete the duties it will have during the game. This class is inherited by 6 enemy types. These 6 enemies have a fireWeapon() for each of the types as the type of weapon they fire is different, for example, Spikey and BugEye will not fire any weapon whereas Saucer Blades will fire RedBolt Bullets. The BugEye also has a pickHuman() method as it is the only one that can pick up humans.
Their updatePosition() method also differs from each other, therefore, it should be overridden in child classes. These classes are also not independent of the GameFrame and have to be instantiated in it.

Weapon is a class that is inherited by 2 different kinds of weapons in the game. These types are MassDestructionWeapon and TargetedWeapon. The MassDestructionWeapon can only be fired by the spaceship. This class contains the starting point of the attack and the radius of the attack and aims to destroy any enemy in that radius. Its method destroy() does exactly that. On the other hand, TargetedWeapon can be fired by the enemy as well. The TargetedWeapon is inherited by Asteroid, Bullet and Missile and the Bullet are further inherited by BlueBolt, RedBolt, and OrangeGlow. The Blue Bolt is fired by the player, the OrangeGlow and RedBolt are fired by the enemy. We decided to inherit BlueBolt, RedBolt and OrangeGlow from Bullet class, in case any changes (for example, in updatePosition() method) are to be made in the system in the future. The Missile is an enemySeekerMissile and can only be fired by the spaceship. The asteroids are independent of any enemy or player as it will be spawned by the system to target the spaceship.

The human class is rather simple as the humans will only be instantiated on top of the frame and their position will be recorded to see where they are placed and their collisions will be recorded as we have to allow the BugEye or Spaceship to pick them up on contact.

The WarpZone is the class instantiated on the GameFrame and their collisions will be recorded to see whether the spaceship has collided with it to transport it to the final warpLocation. The WarpZone also has a position attribute which will show the coordinates on which this WarpZone's initial and final positions are located.

The Pickup class shows which pickups are available in the game. It is inherited by HealthPickup and FuelPickup and both of them have the position on which they both are at that moment and the amount of change they will bring to the amount of fuel or health the spaceship when they collide with the spaceship.

The MapView is the whole map of the game. It shows the place where spaceship, enemies, and humans on the map are and also shows where the camera is currently using its cameraPosition attribute.

The CameraView is the class that monitors which part of the Mapview should be shown on the screen and it updates these coordinates when the spaceship moves.

The ShopFrame is the class which is instantiated independently from the GameFrame and where the player can buy different upgrades for the Spaceship. It contains coins, itemList, boughtIems attributes. These help in comparing whether the player can afford the item that the player wants to buy.

We have a DataManager class that is required to save the current score of the player if it is greater than the previous highScore to save the new HighScore.

The ClickListener and the KeyListener are classes that listen to any activity on the class they have been attached to and are used to interact with the user of the game.

4.4 User Interface - Navigation Paths and Screen Mockups

Figure 10 (Start Menu)

This is the menu initially presented to the user as the game starts in which the user can start a new game or quit the game to the windows.

Figure 11 (Pause Menu)

This is the menu presented to the user when the game is in the paused state. Here the player can choose to start a new game, resume the current game, go to the shop to buy some items, or quit the game.

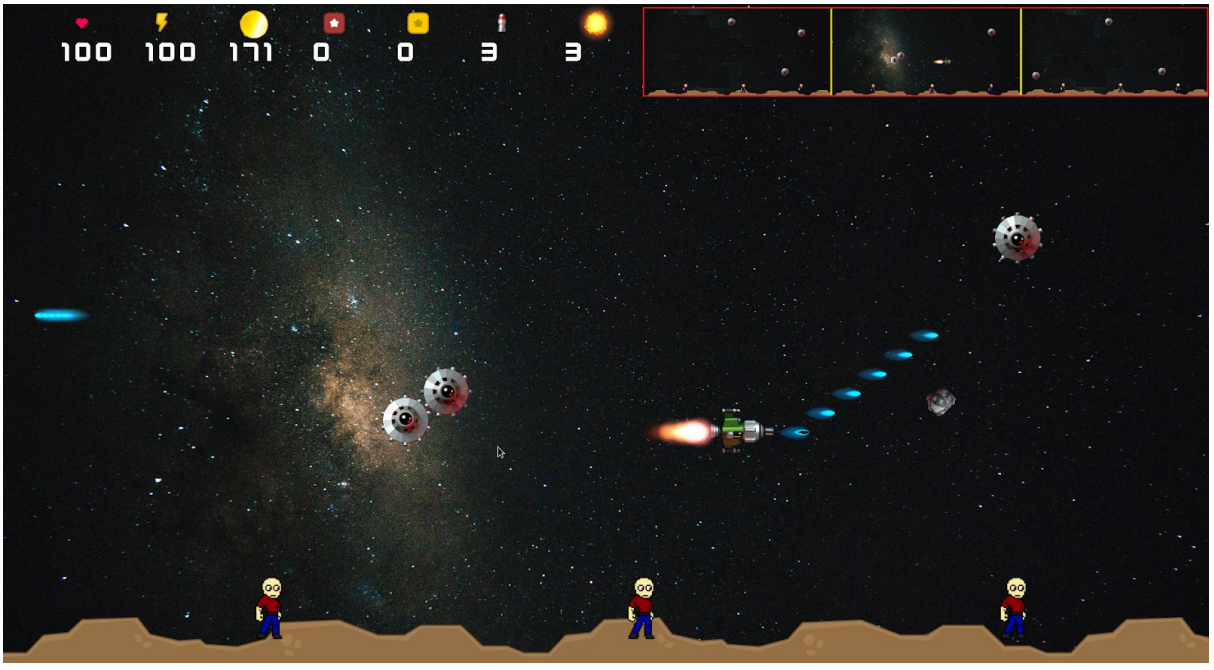


Figure 12 (Gameplay)

This is an example of what the Game Frame will look when the game is played. On the top left of the screen, the user is shown a head-up display (HUD) which shows spaceship's health, fuel, coins, and score. It also shows a high score and missile and nuclear bomb cools down. On the top right of the screen, the user is shown a view of the map showing current positions of the enemies, spaceship, and humans. It also highlights the current camera view on the map. The other part of the screen shows the game screen in which the player can move shooting enemies and saving humans from falling. The enemies will be doing their job (shooting, exploding) as explained in Section 2.2.3. Asteroids will be falling from which the player has to save himself. The pickup and warp zones will be generated on this screen and the player can utilize them by colliding with them.

Figure 13 (Shop Frame)

The above picture shows Shop Frame which is accessible to the user from Pause Menu. The user can buy different types of upgrades namely: Health, Fuel and Skin colors. The user will be shown the description and the price of each item that can be bought. The user will also show the number of available coins on the top right corner of the screen. The user can go back to the main menu using the back button on the top left corner of the screen.

Player(Spaceship)

Bugeye(Enemy)

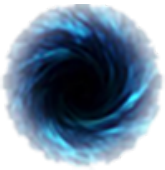
Slicer(Enemy)

Spikey(Enemy)

Saucer(Enemy)

Saucer Blades(Enemy)

Saucer(Enemy)



Warp Zone

Meteor

Human

Blue Bolt

Red Bolt

Orange Glow

Missile

Health(Pickup)

Fuel(Pickup)

Coin

Score

5. Improvement Summary

On top of the standard version of the defenders game we have included new features such as a shop menu where the player can spend the coins they earn in-game on items such as health upgrades, fuel upgrades, and skin color, these upgrades will be available for the lifetime of that game and the player will have to make these purchases again if they start a new game. We have also added an asteroid event in which an asteroid is generated on top of the screen and the player has to avoid it otherwise his spaceship will receive damage. The spaceship has both health and fuel values, the fuel is a new feature added so that the player would feel more involved in the game where he has to manage his resources. In terms of weapons, we have included two new weapons the missile and the nuclear bomb each having their unique mechanism and damage values. The final new feature of our game is the pickup system, the pickups include health and fuel pickups if the player manages to collide with either pickup it will restore their stats resulting in prolonged and competitive gameplay.

References

[1] “Classic Arcade Games For Sale,” Classic Arcade Game Machines for Sale. [Online]. Available: https://www.arcadeclassics.net/80s-game-videos/defender. [Accessed: 30-Nov-2019].