TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Proposal

On

Space Shooter

Submitted By

Amit Raj Pant - THA076BCT005

Arahanta Pokhrel - THA076BCT009

Pilot Khadka - THA076BCT025

Sudeep Kaucha - THA076BCT044

Submitted To

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

July 2021

DECLARATION

We hereby declare that the report of the project entitled **"Space Shooter"** which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Computer Engineering**, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Amit Raj Pant **(**THA076BCT005) _____

Arahanta Pokhrel (THA076BCT009) _____

Pilot Khadka (THA076BCT025) _____

Sudeep Kaucha (THA076BCT044) _____

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a project work entitled "**Space Shooter**" submitted by **Amit Pant**, **Arahanta Pokhrel**, **Pilot Khadka** and **Sudeep Kaucha** in partial fulfillment for the award of Bachelor's Degree in Computer Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's degree of Computer Engineering.

Project Supervisor

Er. Saroj Shakya

Department of Electronics and Computer Engineering, Thapathali Campus

Examiner

Raj Kumar Dhakal

COPYRIGHT

ACKNOWLEDGEMENT

We would like to thank the Department of Computer and Electronics Engineering for the opportunity to learn and implement our knowledge by granting us a platform to do project work.

We would like to express our sincere gratitude towards our lecturer Er. Saroj Shakya for providing resources and support . We are thankful to him for providing guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project. Moreover, we would like to give special thanks to our classmates who encouraged, discussed and exchanged ideas with each other. Special gratitude to our senior brothers for helping us to create ideas.

Finally, we would like to thank all the people who are directly or indirectly related during our study and preparation of this project.

Amit Raj Pant - THA076BCT005
Arahanta Pokhrel - THA076BCT009
Pilot Khadka - THA076BCT025
Sudeep Kaucha - THA076BCT044

ABSTRACT

Space shooter is a game that allows the player to go over multiple levels that progress in difficulty and speed. Within each level, there are three types of asteroids: easy, medium and hard. Each asteroid level contains its own artwork, health and point embedded to it. Also, some asteroids drop power-ups for the player to increase ship power and mobility throughout the levels. It's a simple 2D video game based on C++ programming language and created using the Simple Fast Multimedia Library (SFML) Framework.

Keywords: SFML, Video Games, Space shooter

Table of Contents

List of Figures

List of Abbreviations

2D	Two Dimensional
COVID 19	Coronavirus Disease 2019
DIY	Do It Yourself
et al.	And Others
GUI	Graphical User Interface
H.G.Wells	Herbert George Wells
HP	Health points
IDE	Integrated Development Environment
IDLE	Integrated Development Environment
OOP	Object Oriented Programming
PC	Personal Computer
SFML	Simple Fast Multimedia Library
VS Code	Visual Studio Code

1. INTRODUCTION

Game industry is one of the fastest growing industries, allowing people from all around the world to play games of many different genres. It provides many jobs to people with different talents and produces a big number of games each year, thus resulting in big revenues for the companies that make them. These games are played worldwide by millions of people on a variety of different platforms, including Microsoft Xbox One, Sony PlayStation 4, Nintendo Switch, different mobile platforms and, of course, the biggest and best PCs. There is a game for every person: from first person shooters to racing games, from fighting games to sports games, from strategy games to puzzle games, etc.

Due to its continued growth, software developing companies started making more applications for game development and, at the present time, there are many different tools that allow a single person or a very small team of people to successfully develop and sell video-games, which accelerates the market growth of this industry even more. Games are played by people of all ages and genders. This fact provides an easy way to connect parents and children, or different people around the globe. Taking into account this information, it is easy to see why game development is a good project to get into.

Shoot'Em Up, also known as shmup, is a video-game genre in which the player-controlled character engages in a battle against hordes of enemies while at the same time having to dodge all incoming attacks which can consist of different types of projectiles, map objects and enemies themselves.

Space Shooter is a Shoot'Em up game that involves piloting spacecraft in an outer space setting. It is a single level, single player strategy game on the Windows platform. The player will progress through levels which require precise manipulation of the environment, through the game. The game Encourages creativity and daring via branching pathways.

The goal is to eliminate all of the aliens by shooting them. While the player has three lives, the game ends immediately if the enemies reach the bottom of the screen. The aliens attempt to destroy the player's cannon by firing projectiles. As aliens are defeated,

their movement and the game's music both speed up. Defeating all the aliens brings another wave which starts lower, a loop which can continue endlessly.

1.1 Background Introduction

Space Invaders (1978) is most frequently cited as the “first” in the genre. A seminal game created by Tomohiro Nishikado at Japan’s Taito Corporation, it led to shooter games becoming prolific. Space Invaders pitted the player against multiple enemies descending from the top of the screen at a constantly increasing speed. Nishikado came up with the game’s concept by combining elements of Breakout (1976) with those of earlier target shooting games, along with alien creatures inspired by The War of the Worlds (by H.G. Wells) because the hardware was unable to render the movement of aircraft, with the game set in space as the available technology only permitted a black background. It had a more interactive style of gameplay than earlier target shooting games, with multiple enemies who respond to the player-controlled cannon’s movement and fire back at the player, leading to a game over when the player is killed by the enemies. While earlier shooting games allowed the player to shoot at targets, Space Invaders was the idea of giving the player multiple lives, and popularized the concept of achieving a high score, as it saved the player’s score.

With these elements, Space Invaders set the general template for the shoot 'em up genre. It became one of the most widely cloned shooting games, spawning more than a hundred Space Invaders clones. It has influenced most shooting games released since then, which have continued to rely on "the multiple life, progressively difficult level paradigm" of Space Invaders.

Following the success of Space Invaders, Space Shooters were the dominant sub-genre during the late 1970s to early 1980s. These games can overlap with other subgenres as well as space combat games.

1.2 Motivation

Despite the economic instability and crisis deeply affecting the world due to COVID-19, The Analysts published that the game industry has grown at a rate of 57% surprisingly. [7]As people are stuck at home, gaming has become a new getaway.

Millions of people are playing games in front of their computers. The gaming industry has been able to appeal to a wide variety of users of all ages with different tastes. Games like Minecraft, Among us have risen to popularity with game streaming at all times high.

Being stuck in the same situation, we decided game creation would be a worthwhile as well as an entertaining project for us. We were always fascinated by the idea and engineering behind game logic. We have faith that our project of developing games would help to implement the coding knowledge we possess. It would grow more interest towards programming and enhance the programming skill. We hope designing the game in a team will be a joyful and entertaining endeavor for us.

1.3 Problem Definition

1.3.1 Game Engine

How do we structure the code in order for four people to efficiently? Implement features iteratively, seamlessly, and concurrently? How can we solve common game-programming problems, such as collision, and level generation?

1.3.2 Graphics

How can we implement certain specific graphical effects in a visually pleasing, yet efficient, manner? Which of these effects are easy to implement, and? Which are difficult?

1.3.3 Collaboration

How do we collaborate as a group of four members on the same software? How do we select which features to implement, and which to skip, with regards to the time constraint set on the project?

1.4 Objectives

The main objectives of our project are listed below :

- To entertain and enthuse players by implementing a more interactive style of gameplay

- To learn the basics of game engine and game physics and work as a team.

2. LITERATURE REVIEW

The earliest concept for a space shooter game dates back to 1978 named Space Invaders developed by Japanese designer Tomohiro Nishikado, who spent a year designing the game and developing the necessary hardware to produce it. The game's inspiration is reported to have come from varying sources, including an adaptation of the electro-mechanical arcade game Space Monsters released by Taito in 1972, and a dream about Japanese school children who are waiting for Santa Claus when they are attacked by invading aliens. Nishikado himself has cited Atari's arcade game Breakout (1976) as his original inspiration behind the game's concept, wanting to adapt the same sense of achievement and tension from destroying targets one at a time, combining it with elements of target shooting games [1] [3] [4].

2.1 Methods, Techniques and Algorithms

In Tomohiro Nishikado's original Space Invader (1978), typical level consisted of a player piloting a laser canon to battle columns of descending aliens while using shields to block alien fire. The speed of the alien approach increased as the game progressed, adding to the tension. [5] A bonus alien spaceship appeared from time to time, which offered the player an opportunity to score additional points by blowing it up. The game used a similar layout to that of Breakout (1976) but with different game mechanics; rather than bounce a ball to attack static objects, players were given ability to fire projectiles at moving enemies. [6] Players were given multiple lives, had to repel hordes of enemies, could take cover from enemy fire, and use destructible barriers.

2.2 Importance and Application

As one of the earliest shooting games, Space Invaders set precedents and helped pave the way for future titles and for the shooting genre. Space Invaders popularized a more interactive style of gameplay, with enemies responding to the player-controlled cannon's movement, and was the first video game to popularize the concept of achieving a high score, being the first to save the player's score. [2] [7]While earlier shooting games allowed the player to shoot at the targets, Space Invaders was the first in which multiple enemies could fire back at the player, and in contrast to earlier arcade

games which often had a timer, Space Invaders introduced the “concept of going round after round.” [5]

Drawbacks and Limitations

Nishikado was unable to program the game as he wanted-the Control Program board was not powerful enough to display the graphics in color or move the enemies faster-and he ended up considering the development of the game’s hardware the most difficult part of the whole process. [9]While programming the game, Nishikado discovered that the processor was able to render each frame of the alien’s animation graphics faster when there were fewer aliens on the screen. Since the alien’s positions updated after each frame, this caused the aliens to move across the screen at an increasing speed as more and more were destroyed. Early enemy designs for the game included tanks, combat planes, and battleships. [4]

3. METHODOLOGY

Here we explain the algorithms implemented and how different entities of our game like player, enemies, bullets, asteroids, etc. are governed by. These all aspects of our game are explained below.

3.1 Design of Main menu

Our main menu provides a gateway to access many functions of the program. It contains a very interactive user interface, two options for the user to either enter the game or to exit the game when s/he doesn't like to play or so. A large block text Space Shooter in the main menu is the title of our project and below it there are two options titled Initiate Millenium Falcon which is a tribute to iconic Millenium Falcon from Star wars that enters to the game itself and then another option below this option is Exit which quits the application itself.

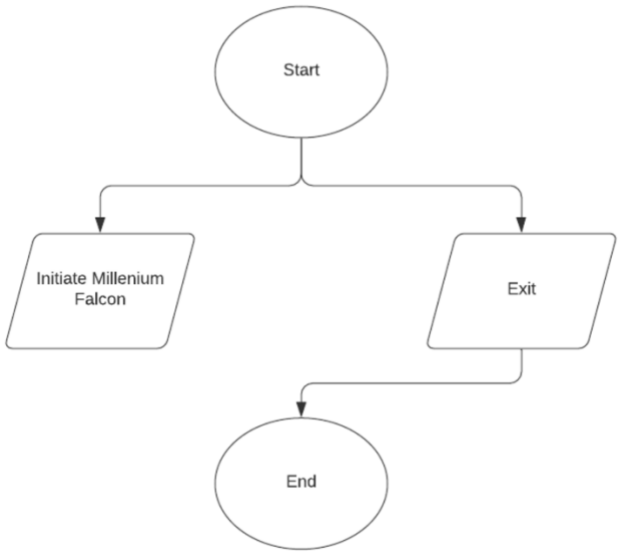

Figure 1:Block diagram of main menu

3.2 Initiate Millenium Falcon

This section of the game contains the main function of the game itself that is to enable players to play the game. Various entities of the game itself are drawn here like the spaceship of player, random spawn of enemies, constant flow of asteroids and the

enemies themself shooting bullets targeting towards the position of the player. The inner function wheels of various entities in this section are explained below.

3.2.1 Player's spaceship

Players can shoot bullets to any direction inside the game window and control their spaceship to any direction and to any extent they like, this is possible through the mouse movement and keyboard inputs provided by the players. Rotation input through the mouse gives the direction to which spaceship should rotate and keyboard inputs "A,S,D,W" gives the direction up, down, left, right where the spaceship should move. Right click mouse input directs the player spaceship to shoot bullets to the direction where the player spaceship is pointing.

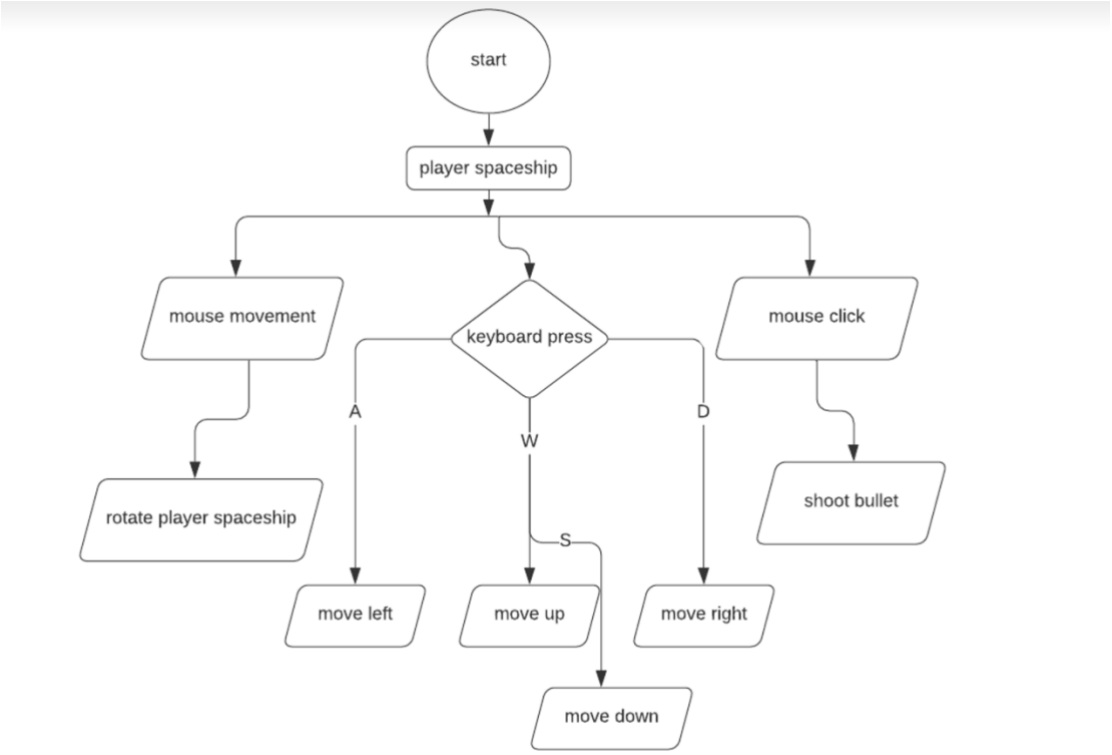

Figure 2: Block diagram for for Player spaceship inputs

3.2.2 Enemy and Player Bullets

Bullets themselves are separate entities in game and they can be used both by enemies' spaceships and players' spaceships and have different functions on the basis of who

shoots them. Players can shoot bullets in the direction they are pointing and enemies on the other hand calculate the relative position and direction of the player to them and shoot the bullet continuously to that direction after some instant of their spawning. When a bullet shot by a player encounters another enemy spaceship or even asteroid then it decreases the health point of the encountered entity and the same is for the bullet shot by the enemy spaceship. If the bullet touches the game windows borders then it itself destructs.

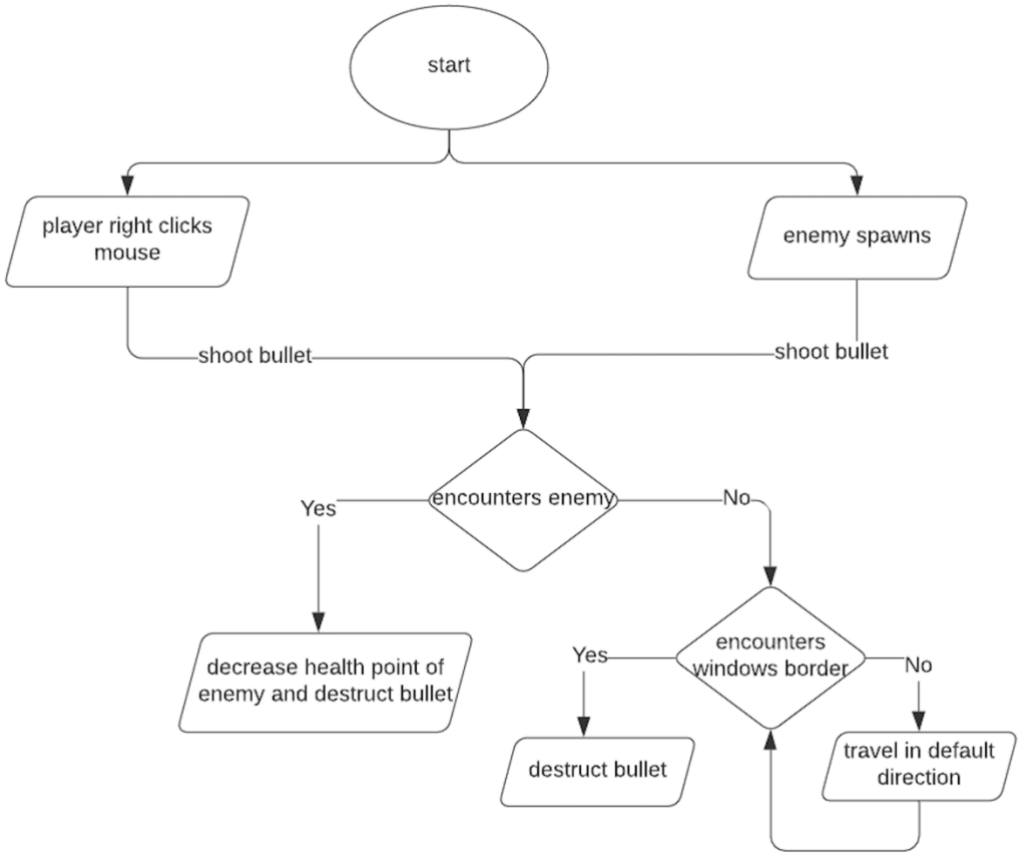

Figure 3: Block diagram for enemy and player bullet

3.2.3 Score and health system

The player spaceship, enemy spaceship and asteroids are initially given their own health which decreases when one entity touches another entity or the bullet shot by one entity encounters the other. When the bullet shot by the player encounters the enemy then the health point of the enemy decreases, the bullet self-destructs and the player gains points based on the entity scored. On the other hand if the bullet shot by the enemy spaceship encounters the player spaceship then the health point of the player decreases and the

bullet self destructs. When the health of any enemy entity decreases to zero then the entity self-destructs and after some instant an entire fleet of enemy spaceships spawn at some random position. The player spaceship has its own score and health point system which decreases when an enemy bullet touches or when some other enemy entity comes crashing towards it. When the health point of our player spaceship decreases down to zero then a gameover window prompts.

3.2.4 Enemy spaceships

An entire fleet of enemy spaceships spawn at some random position inside the game window. Each spaceship contains position arguments and the placeholder for the relative position of the player spaceship which it uses to shoot the bullet towards the player spaceship position. It can rotate but not move from its originating position to shoot and point towards the player spaceship. When it is hit by the player bullet or even when a player spaceship crashes to it then its health point decreases and it flashes bright red colour to signify this state. When its health point decreases to zero then the entity self-destructs and waits for the original fleet of spaceships to destruct and then after a certain instant of time the entire fleet spawns at a completely random position.

3.2.5 Asteroids

They are self rotating and constantly moving entities that move towards the left part of the window. They come in various shapes and sizes randomly spawning from the right side of the game window. The speed of the asteroid is determined on the basis of the shape of the asteroid but these entities don’t shoot bullets if you haven’t figured it out yet. Their sole power comes from crashing into player spaceships which in turn decreases their own health point and the health point of the player. When these entities encounter the left side of the windows then they self-destruct or when their health point decreases to zero.

4. SYSTEM DESCRIPTION

The game has different functions and classes to manage different parts of the game. We have implemented various SFML and C++ libraries. The system description of each module is discussed below.

4.1 Main menu

The main menu is made up of two sections : Initiate Millennium Falcon and Exit. A space themed background has been applied that goes well with the theme of the game. The player can choose between the two options with the help of a keyboard. Choosing 'Initiate Millennium Falcon' starts the game and 'Exit' closes the game. SFML library was used to create and record keystrokes from the user.

.GetPressedItem() function of the SFML library records the keystroke of the user. Keyboard words are recognized as keywords in the SFML library so when a key is pressed, .Moveup(), .MoveDown(),.. functions can be used to move the control.

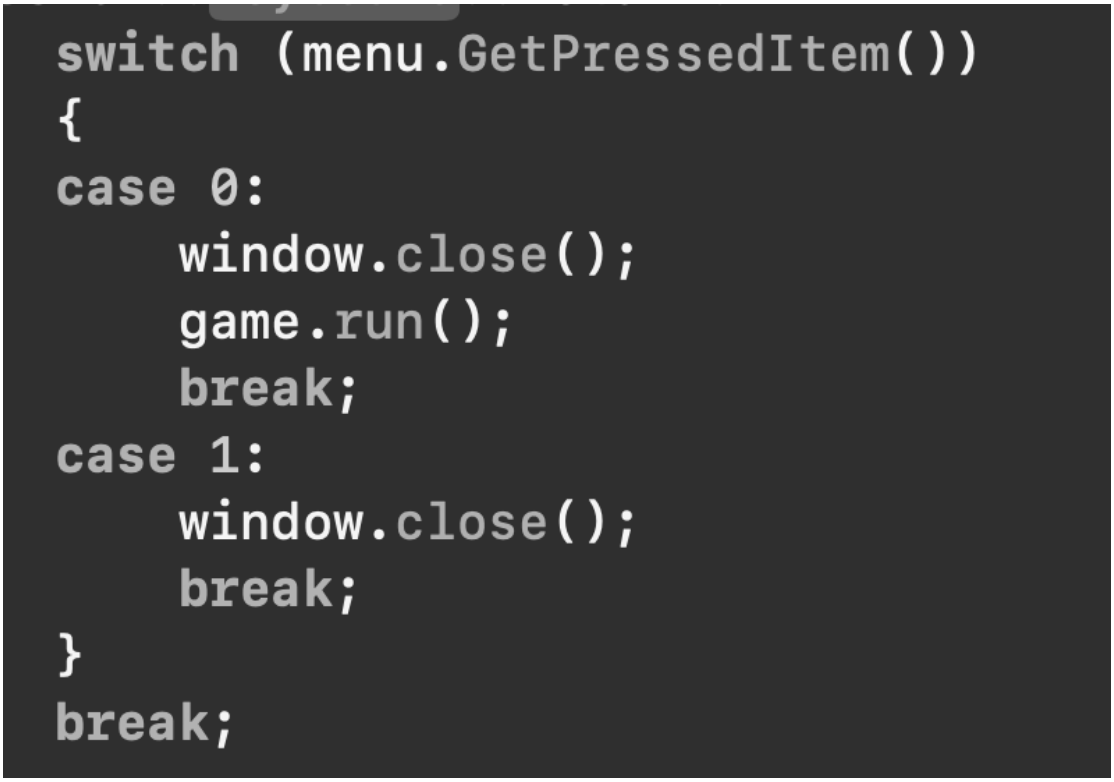

Figure 4 : Main menu switch case

4.2 Game

Game contains the main logic of the game. In the game header file, all the functions and classes required to run the game have been declared. The Game class manages game components: Player, Bullet, Asteroid and Enemy. It is the core component of the game and it runs the game loop.

The game class calls the initialize functions of other classes :the window, background textures required by the game. It records the player movement ,input and updates the game. At the end, all the game related objects are destroyed.

4.3 Player

Player class handles the spaceship. .initVariables() of the player class initializes the health, speed and attack cooldown of the spaceship. initTexture() and initSprite() of the player class sets the texture of the spaceship.

The funcion setPlayerPosition sets the location of the ship when the game is launched. Then the player is able to move freely with the help of the movePlayer function. Keyboard functions like “A”, “S”, “W”, “D” keyboard to move the player and Mouse to point and shoot the enemies

```
const Vector2f& Player::getPosition() const
{
    return sprite.getPosition();
}

const FloatRect Player::getbounds() const
{
    return this->sprite.getGlobalBounds();
}

const int& Player::getHp() const
{
    return this->health;
}
```

Figure 5: Player position and HP

The above functions return the current player position and health points of the player. The game loop uses this information to point the enemy ship to the player location,

make sure the player does not go out of the screen and display the current HP of the player on the screen. When the getHP() returns 0, the game displays Game Over screen.

4.4 Asteroid

Asteroid header file initializes the asteroid class.The asteroid class keeps track of all the asteroids in the screen and updates them every frame of the game.

Constructor has been used to initialize the sprite and texture, health of the asteroid. getposition() returns the current position of the asteroid , getPoint() returns the point awarded to the player once the asteroid is destroyed.

```
void Asteroid::initVariables()
{
    this->healthMax=1;
    this->health = this->healthMax;
    this->damage=1;
}
```

Figure 6: Asteroid initialization

initVaribles() of the asteroid class initializes the health and damage of the asteroid. healthMax sets the asteroid health to 1 and damage is set to 1. When the player ship collides with the asteroid, 1 damage is done to the player.

update() moves the asteroid from the spawn point to the left side of the screen, towards the player.

4.5 Enemy

Enemy header file contains enemy related variables and functions. Constructor is used to initialize the sprite and texture of the enemy ship. initVariables() is used to initialize the enemy class. Function updateAttackcooldown() determines when the enemy spaceship can attack after spawning.

initVariables() if the enemy class initializes the HP, damage and the point received of the enemy shit. healthMax of the enemy has been set to 2 so two bullet hits are required to destroy the ship.Once the ship is destroyed, 500 points is awarded to the player.

```
this->healthMax = 2;
this->health = this->healthMax;
this->damage = 1;
this->points = 500;
```

Figure 7: Enemy initialization

4.6 Bullet

Bullet class manages the shape, size and direction of the bullet fired. pos_x and pos_y contain the location of the player ship in the game. .setPosition() sets the position of the bullet to the player ship. .setRotation function calculates the mouse position and the bullet points to the direction of the mouse pointer.

When a bullet is fired, bulletSpeed moves the bullet by set speed every frame in the direction set by the shapeBullet.setRotation.

```
this->shapeBullet.setTexture(*texture);
this->shapeBullet.setPosition(pos_x, pos_y);
this->direction.x = dir_x;
this->direction.y = dir_y;
this->shapeBullet.setRotation(rotation + 180);
this->bulletSpeed = bulletSpeed;
```

Figure 8: Bullet initialization

5. RESULTS AND ANALYSIS

Project planning and scheduling is a part of project management. The project planning stage requires several inputs, including conceptual proposals, project schedules. The development of this project is not successfully done without proper planning and scheduling. Project planning and scheduling is a very important stage.

5.1 Main menu

The main menu is the start page of the game. The user can choose to continue play the game or exit the game.

Figure 9: Game main menu

5.2 Game

When “Initiate Millenium Falcon " is pressed, the game is started. The player can then interact with the game environment. The scoreboard is shown at the top left corner of the screen which represents the health status and score of the player.

The user ship can fire bullets while moving through space. Mouse is used to point to the direction the bullet is to be fired. The bullet does damage to the asteroid and the enemy ship.

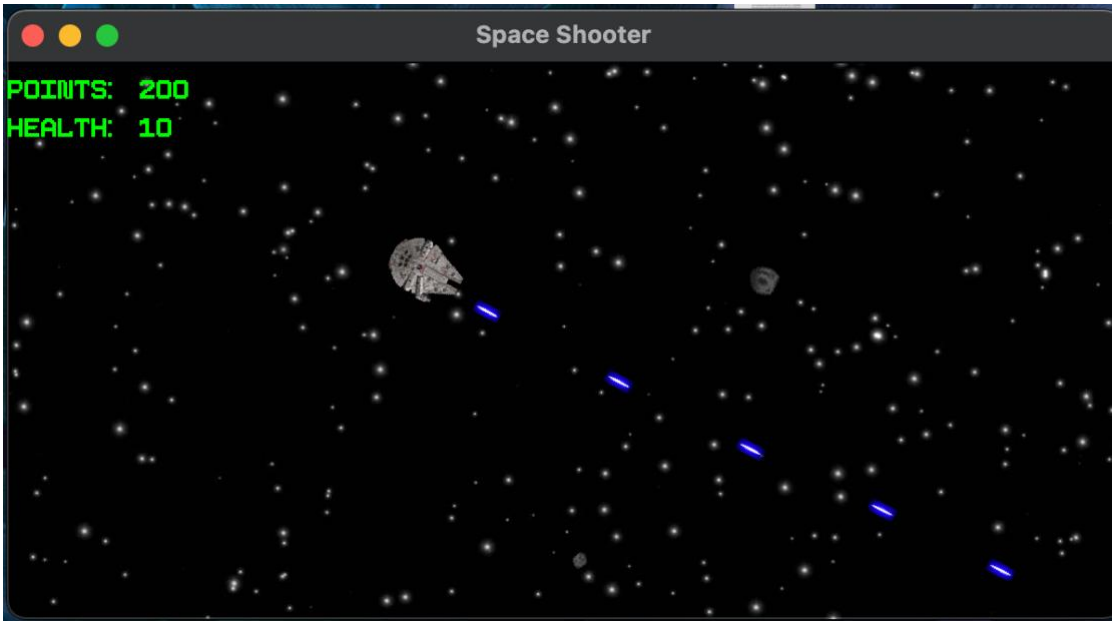

Figure 10: Game screen

5.3 Game Over Screen

When the player is hit with an enemy bullet or an asteroid, one health point is lost. The player has a total of 10 health points. When the player's health score drops to zero, the game reaches its end point and the user is shown the game over screen.

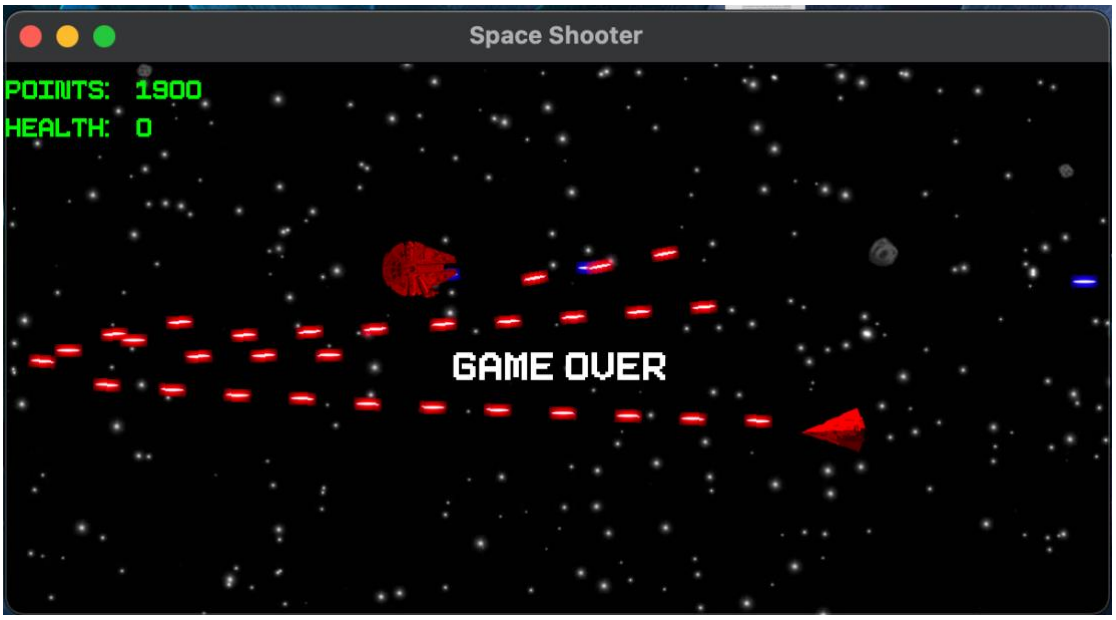

Figure 11: Game Over screen

5.4 Analysis

Every game is made to entertain the user, and our project fulfills that. The goal of the game is simple. To destroy as many asteroids and enemy ships without dying. The player is awarded points for every asteroid and ship destroyed. Although the game is simple in nature, one needs countless hours to beat the high score. The game is easy to play and learn, but it can be difficult to master. The game tests the patience and skill of the player by dodging and shooting at the same time.

The graphics and background of this game is somewhat different from the other Space Shooter games available in the market. The spaceship design and sound effects have been inspired from the original ‘Stars Wars’ as the game takes place in space. The player ship is the coveted Millennium Falcon and the enemy ship design is Star Destroyer. The ship shoots using a “Blaster”, a special type of weapon from the series.

6. CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

An implementation of the various OOP concepts learnt is our project Space Shooter. It helped us to work as a team and helped to implement various steps of problem solving and create this beautiful creation we call Space shooter.

6.2 Future enhancement

Space shooter was built for the fulfillment of the project work. It was built in a very limited time. The game has huge potential. Different features can be added to the game to make it more engaging. Features that could be added to different parts of the game for further enhancement are mentioned below.

6.2.1 Enemies

We have only one type of enemy in the game. Enemies create difficulties for the player to move through the space. There are a lot of things that could be added to the enemy class in future to diversify them. We.can add More types of enemies to the game. Different types of enemies with different kinds of damage effects and sound effects.

6.2.2 Player

At present, the game contains only one playable space ship. The game can be expanded by adding other types of ship with different bullet configurations like double barrelled, homing missiles, etc. Different sizes of ships can also be introduced. The player could also be given an option of customising their ship with different colors and designs.

Powerup systems like rapid-fire, invincibility, helper ship,etc could also be added that can increase playability of the game.

6.2.3 Game Environment

Currently the game background and surrounding environment doesn't have different graphics and designs. We can work on enhancing them to make the game look more

attractive. Different healing and killing effects can be added to the game. Also, adding different kinds of sound effects will ensure more engagement.

6.3 Limitations

The limitations of the project that we built are

- High score feature is not available due to time limitation,
- Player cannot choose between various spaceship designs,
- Level up features and various in-game power ups are not implemented.

7. APPENDICES

7.1 Appendix A: System Requirements

Items	Specifications
RAM	More than 500MB
OS	Microsoft windows 7/8/10
Processor	Intel i3-all above versions

References

[1] A. Insight, "analyticsinsight," 31 October 2020. [Online]. Available: https://www.analyticsinsight.net/gaming-boom-in-covid-19-times-analysis-insights/.

[2] K. Bowen, "worldvideogamehalloffame," [Online]. Available: https://www.worldvideogamehalloffame.org/games/space-invaders.

[3] P. J. Whitehead, "Game Genres: Shmups".

[4] A. Seabrook, Interviewee, *Replay: the Evolution of Video Game Music.* [Interview]. 2008.

[5] A. Williams, "History of Digital Games: Developments in Art, Design and Interaction," 2017.

[6] C. Glenday, Top 100 Arcade Games: Top 5, Guinness World Records. Guinness, 2008.

[7] M. Bielby, in *The Complete YS Guide to Shoot 'Em Ups*, 1990, p. 30.

[8] K. Bowen, "worldvideogamehalloffame," [Online]. Available: https://www.worldvideogamehalloffame.org/games/space-invaders.

[9] D. H. Ryan Gedde, "ign," 23 7 2008. [Online]. Available: http://games.ign.com/articles/840/840621p1.html.