Logo

Description automatically generated Logo

Description automatically generated

CSY1018

Web Development – I

Assignment II

Project Report

**Submitted By**

Student Name: Aashish karki

Uni. ID: ­­ 23834873

Group: L4 (C)

Date: 14-MAY-2023

Table of Contents

[1 Introduction 4](#_Toc134954428)

[2 Check List 5](#_Toc134954429)

[4 Testing 7](#_Toc134954430)

[4.1 Movement of Character movement. 7](#_Toc134954431)

[4.2 Movement of missile ejecting from top. 7](#_Toc134954432)

[4.3 Movement of random missile. 7](#_Toc134954433)

[4.4 Movement of exploding bomb on the ground. 8](#_Toc134954434)

[4.5 Movement of players lives. 8](#_Toc134954435)

[4.6 Movement of arrow keys. 8](#_Toc134954436)

[4.7 Movement of start button. 9](#_Toc134954437)

[4.8 Movement of lives of player after die. 9](#_Toc134954438)

[4.9 Movement of collision between arrow and missiles. 9](#_Toc134954439)

[4.10 Movement of showing the die animation after game over. 10](#_Toc134954440)

[5 Evaluation 17](#_Toc134954441)

[6 Conclusion 19](#_Toc134954442)

[7 Bibliography 20](#_Toc134954443)

[8 Appendix 21](#_Toc134954444)

[8.1 Game.js 21](#_Toc134954445)

**Table of Figures**

[Figure 1: First position of character in game. 6](#_Toc134521243)

[Figure 2: The character is moved up by pressing the up-arrow key. 7](#_Toc134521244)

Figure 3: The missiles are generating randomly from top axis…………………………7

Figure 4: The missiles are exploding on the ground that is displayed on green color.8

Figure 5: 3 lives of character on the top right…………………………………………….8

Figure 6: When missile hit the player one life is decreased…………………………….9

Figure 7: Missile are generating in random direction……………………………………9

Figure 8: Hitting arrow key on the forward direction when pressed space button….10

Figure 9: Start button disappear after clicking …………………………………………10

Figure 10: Game over after three lives are gone……………………………………….11

Figure 11: Collision between arrows and missiles……………………………………..11

Figure 12: One life remaining in game ………………………………………………….12

Figure 13: Die animation after dying………………………………………………..……12

**Table of Tables**

[Table 1: Test case for Character Movement. 8](file:///C:\Users\ASUS\Desktop\CSY1018CSY1018-Web-Developement-Assignment-II-23834873%20-Aashish-Karki-Project-Report.docx#_Toc134521252)

[Table 2: Test case for generating missiles. 8](file:///C:\Users\ASUS\Desktop\CSY1018CSY1018-Web-Developement-Assignment-II-23834873%20-Aashish-Karki-Project-Report.docx#_Toc134521252)

[Table 3: Test case for generating missiles randomly . 8](file:///C:\Users\ASUS\Desktop\CSY1018CSY1018-Web-Developement-Assignment-II-23834873%20-Aashish-Karki-Project-Report.docx#_Toc134521252)

[Table 4: Test case for exploded missiles on the ground. 9](file:///C:\Users\ASUS\Desktop\CSY1018CSY1018-Web-Developement-Assignment-II-23834873%20-Aashish-Karki-Project-Report.docx#_Toc134521252)

[Table 5: Test case for lives of the character. 9](file:///C:\Users\ASUS\Desktop\CSY1018CSY1018-Web-Developement-Assignment-II-23834873%20-Aashish-Karki-Project-Report.docx#_Toc134521252)

[Table 6: Test case for arrow key when pressing space button. 9](file:///C:\Users\ASUS\Desktop\CSY1018CSY1018-Web-Developement-Assignment-II-23834873%20-Aashish-Karki-Project-Report.docx#_Toc134521252)

[Table 7: Test case for movement of startbutton. 10](file:///C:\Users\ASUS\Desktop\CSY1018CSY1018-Web-Developement-Assignment-II-23834873%20-Aashish-Karki-Project-Report.docx#_Toc134521252)

[Table 8: Test case for lives of the player after die. 10](file:///C:\Users\ASUS\Desktop\CSY1018CSY1018-Web-Developement-Assignment-II-23834873%20-Aashish-Karki-Project-Report.docx#_Toc134521252)

[Table 9: Test case for collision between arrows and missiles. 10](file:///C:\Users\ASUS\Desktop\CSY1018CSY1018-Web-Developement-Assignment-II-23834873%20-Aashish-Karki-Project-Report.docx#_Toc134521252)

[Table 10: Test case for die animation after game over. 11](file:///C:\Users\ASUS\Desktop\CSY1018CSY1018-Web-Developement-Assignment-II-23834873%20-Aashish-Karki-Project-Report.docx#_Toc134521252)

# Introduction

I am planning to develop a simple game as part of this project. Initially, I am going to begin with my plans for what I will do. First, I have made a schedule for how I am going to start the game, which is going to be my first step. I intend to add a couple of functions to make my project more unique and different as well as create a different type of game. In addition, I plan to increase the number of bombs that will drop randomly in order to enhance my project more interesting. These are my plans to ensure my proposal is successful in a proper way.

As I am making a game, the character will be the simple one. When you press any left or right arrow button, the character will move left and right and up and down. So I am doing my best to do it right by using code and variables to help me. My plan in this project is to create the bomb and this bomb has to be launched from the top and, if the bomb hits the player, the bomb will explode and the player will die, so my goal is to make this game as simple as possible and as efficient as possible. i am intending to use some sort of arrow keys. When we press the space key in the keyboard the bomb has to explode earlier so this is my whole concept and my ambitions about a game which I am going do in this project.

Let me begin by describing what I have done and what I have executed in order to accomplish this project. In order to accomplish this project, I have used a variety of methods, but the main thing is that I applied my plans effectively and efficiently. As a starting point, I made a plan on how I intend to say this in my project, and after that I started to analyze it gradually, and I have also explored a lot of things regarding coding such as Java script, and CSS, which are important for my project. Also, I was able to get assistance from different websites and videos. In addition to that, I started using these resources to accomplish my project and as a result, I received significant results as a result of using them to accomplish my project. Additionally, on top of that, I executed my plans every time I had to, as well as I got help on how to execute my plans from my tutor. In this way I was able to accomplish my project to the best of my ability.

# Check List

|  |  |  |
| --- | --- | --- |
| Feature | Y/N | Notes |
| Bare pass (D- - D+) | | | |
| When the start button is pressed the game should begin and the start button should no  longer be visible | y | Game starts when button is pressed |
| Spaceships get created at a random position at the top of the screen | y | When starts it created top of the screen. |
| Bombs fall down the screen towards the player | y | When game is start missiles start moving towards player. |
| When the bomb hits the ground (green grass area) it explodes | y | When missiles hit the ground they explode on the green ground. |
| If the player is in the radius of the bomb, print “game over” and set the players animation to  dead (css class “character dead”) | y | When the game is over it will print game over and show animation as well. |
| Good pass (C- - B-) | | | |
| The bombs should explode at random points on the grass, not all at the same height | y | When game starts the bombs explode in a variety of direction not same height. |
| The player will have three lives signified at the top right | y | Player has three lives on top right corner of the screen. |
| Each time the player gets hit by a bomb the player loses a life and display the “hit”  animation (css class, e.g. “character hit left”). The life should be removed from the panel in  the top right | y | If the bomb hit the player it loses a life and removed life from top corner. |
| If the player loses all three lives, display the character dead animation and print “game over”  to the screen | y | When the player loses the 3 lives it will print game over. |
| Display a “play again?” message to allow the player to restart the game if they lose | y | When the game is over it will allow the player to restart. |
| Very good pass (B – A-) | | | |
| Implement a scoring system with high scores. Count the number of bombs the player managed to avoid and when they die have them enter their name. When the game ends, ask their name and log the score using Local Storage. When the game ends display the high scores. | n | No scoring system in this game. |
| Make the game more difficult by randomizing the spaceships speed and frequency at which the bombs are dropped. | y | The speed of the bomb and frequency is change by 20 seconds. |
| Make the bombs fall at different angles rather than straight down. | y | Bombs are fell from different angle with different heigh. |
| Excellent pass (A – A+) | | | |
| Add levels of increasing difficulty. For example, 10 slow bombs in level 1, 15 bombs in level 2, etc. Extra marks are available if there are an infinite number of levels that gradually increase in difficulty. | n | Slow and fast bombs are available and game is hard. but the levels are not included. |
| There is an “arrow” class and a “fire” animation for the player. Make it so the player can fire up and destroy the bombs before they hit the ground by pressing the space key. When the key is pressed   1. The player should be given the class “character stand up fire” 2. The player should not be able to move for 0.5s while the arrow is being fired 3. The arrow should fire up from the player’s position and if it hits a bomb, both the arrow and the bomb should be destroyed 4. The player should only be able to fire one arrow every 0.5s | y | When press space button it will display arrow and that is moved upward direction after 0.5s and when arrow meet the bombs both they are destroyed. |

# Testing

## Movement of Character movement.

|  |  |
| --- | --- |
| **Objective** | Testing of movement of character when press arrow keys |
| **Parameter** | The program is run and character will move |
| **Expected Result** | When the up arrow key is pressed the character has to move upward |
| **Actual Result** | The character move upward when pressed Up key. |
| **Conclusion** | Test is successful passed |

Table 1: Test case for player Movement.

## Movement of missile ejecting from top.

|  |  |
| --- | --- |
| **Objective** | Missile are generated from the top of the screen after press start button |
| **Parameter** | The program is run and missiles start generate from top axis |
| **Expected Result** | When pressing the start button the missile have to emerge from the top axis. |
| **Actual Result** | The missiles are emerged from top axis after pressing the start button. |
| **Conclusion** | Test is successful passed |

Table 2: test case for generating missile.

## Movement of random missile.

|  |  |
| --- | --- |
| **Objective** | Testing of random missile |
| **Parameter** | The program is run and missiles are generating randomly |
| **Expected Result** | After pressing start button the missiles are start coming randomly form top axis. |
| **Actual Result** | The missiles are eject randomly from top side |
| **Conclusion** | Test is passed |

Table 3: test case of generating missile randomly from top

## Movement of exploding bomb on the ground.

|  |  |
| --- | --- |
| **Objective** | Testing of missile exploding on the ground . |
| **Parameter** | The program is run and missiles have to explode on bottom of the screen |
| **Expected Result** | When I press the start button the missile have explode on ground. |
| **Actual Result** | The missiles are exploded on the ground after pressing start. |
| **Conclusion** | Test is successfully passed. |

Table 4: test case for exploding the missile on the ground.

## Movement of players lives.

|  |  |
| --- | --- |
| **Objective** | Testing of players lives when missile meet the player |
| **Parameter** | The program is run and hit the player the life has to decreased. |
| **Expected Result** | When arrow touched the payer the one life has to be subtract. |
| **Actual Result** | The arrow touch the player the life will be decreased. |
| **Conclusion** | Test is successful completed. |

Table 5: test case for lives of the charcter.

## Movement of arrow keys.

|  |  |
| --- | --- |
| **Objective** | Testing of movement of arrow keys when I click the space button |
| **Parameter** | The program is run and pressed the space button the arrow have to execute in 0.5 sec. |
| **Expected Result** | When I pressed the space key the arrow has to go forward after taking 0.5 sec pause. |
| **Actual Result** | The arrows are gone to top direction when I pressed the space button. |
| **Conclusion** | Test is successfully passed. |

Table 6:test case for arrow key when pressing space button.

## Movement of start button.

|  |  |
| --- | --- |
| **Objective** | Testing of start button after pressing |
| **Parameter** | The program is run and start button will be disappeared |
| **Expected Result** | When click the start button the button should be removed. |
| **Actual Result** | After pressing the start button the button will be removed |
| **Conclusion** | Test is successfully passed. |

Table 7: test case for movement of start button.

## Movement of lives of player after die.

|  |  |
| --- | --- |
| **Objective** | Testing of movement of lives after game over |
| **Parameter** | The program is run and after game over there will not be any life of player |
| **Expected Result** | When the game is finished the lives have to eject. |
| **Actual Result** | When game is over there is no any lives remaining |
| **Conclusion** | Test is successful. |

Table 8: test case for players lives after die

## Movement of collision between arrow and missiles.

|  |  |
| --- | --- |
| **Objective** | Testing of movement arrows and missile between each other |
| **Parameter** | The program is run and the result will be exploded. |
| **Expected Result** | When the arrow and missile meet together there will be the collision. |
| **Actual Result** | When the arrow meet to the missile there will be the collision and missile and arrow will be disappear. |
| **Conclusion** | Test is successful passed |

Table 9: test case for collision between arrows and missiles.

## Movement of showing the die animation after game over.

|  |  |
| --- | --- |
| **Objective** | Testing of showing die animation after die |
| **Parameter** | The program is run and character is dead after that character will show the animation |
| **Expected Result** | When the game is over and the character will show the dying animation |
| **Actual Result** | The game is over the will be the dying animation |
| **Conclusion** | Test is successfully passed |

Table 10: test case for die animation after game over.

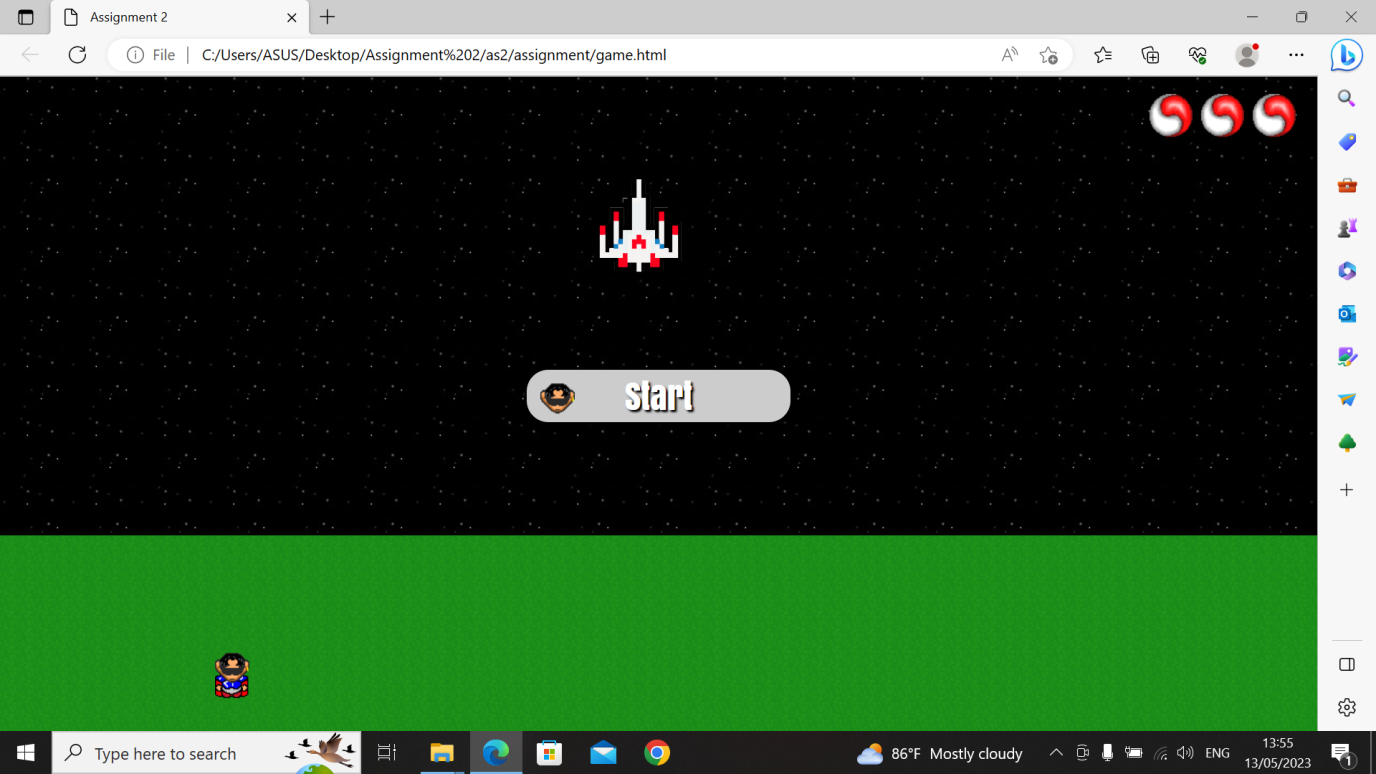


Figure 1: first position of character in game.

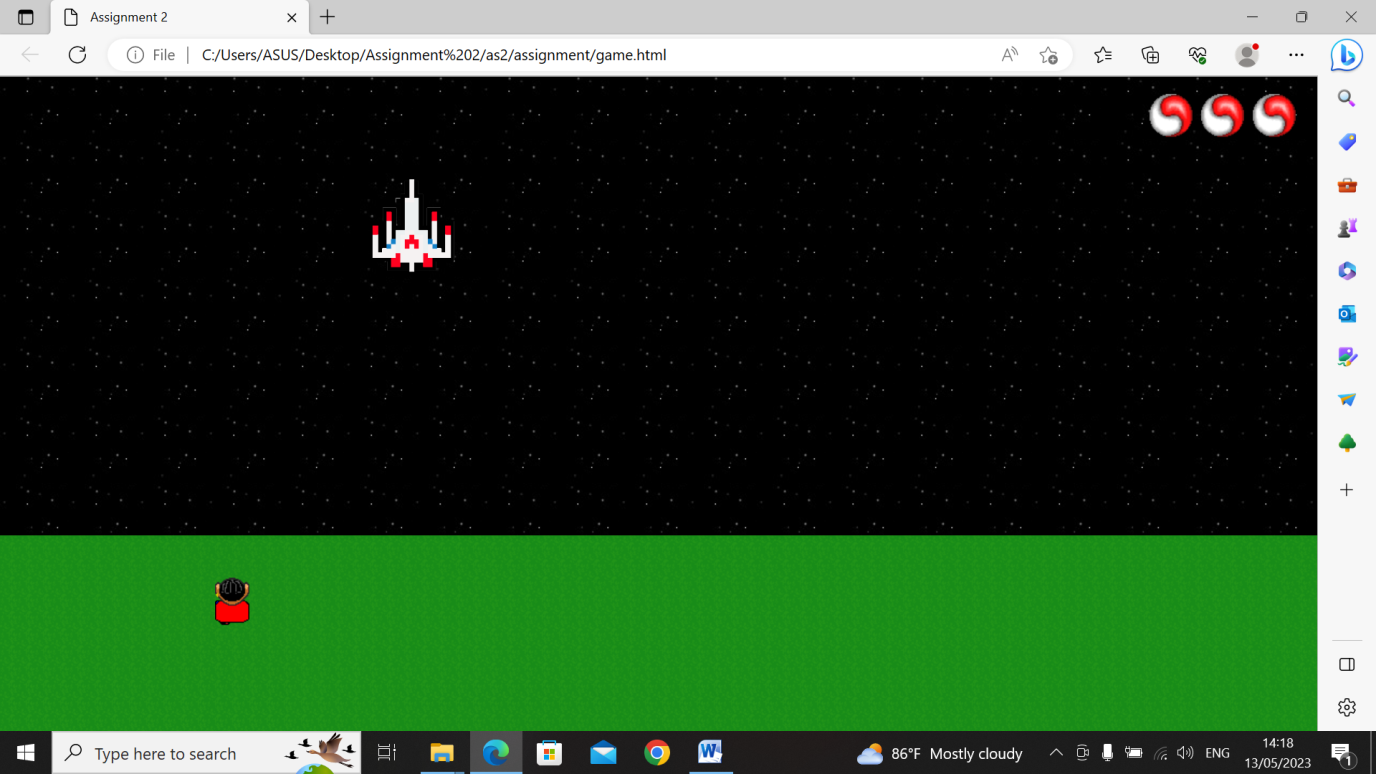


Figure 2: The character is moved up by pressing the up-arrow key.

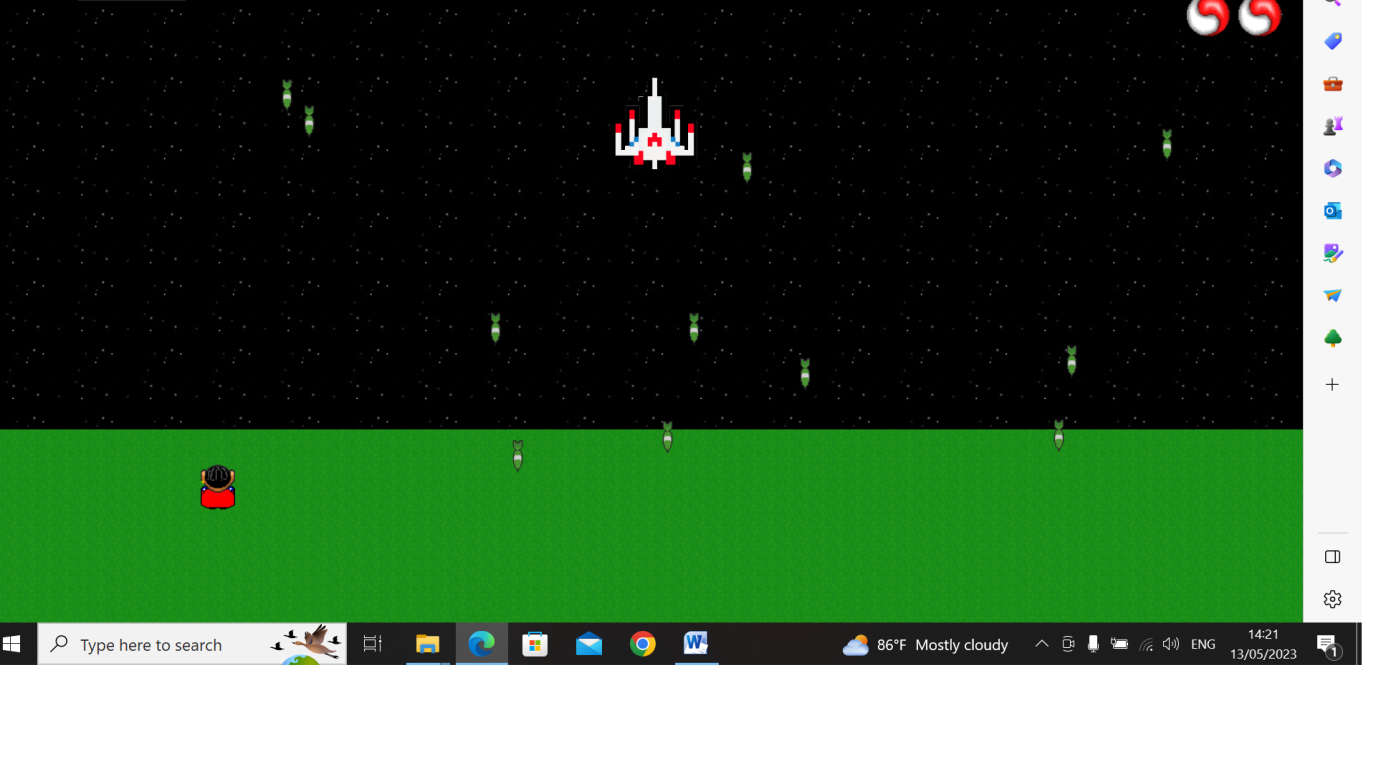


Figure 3: The missiles are generating randomly from top screen.

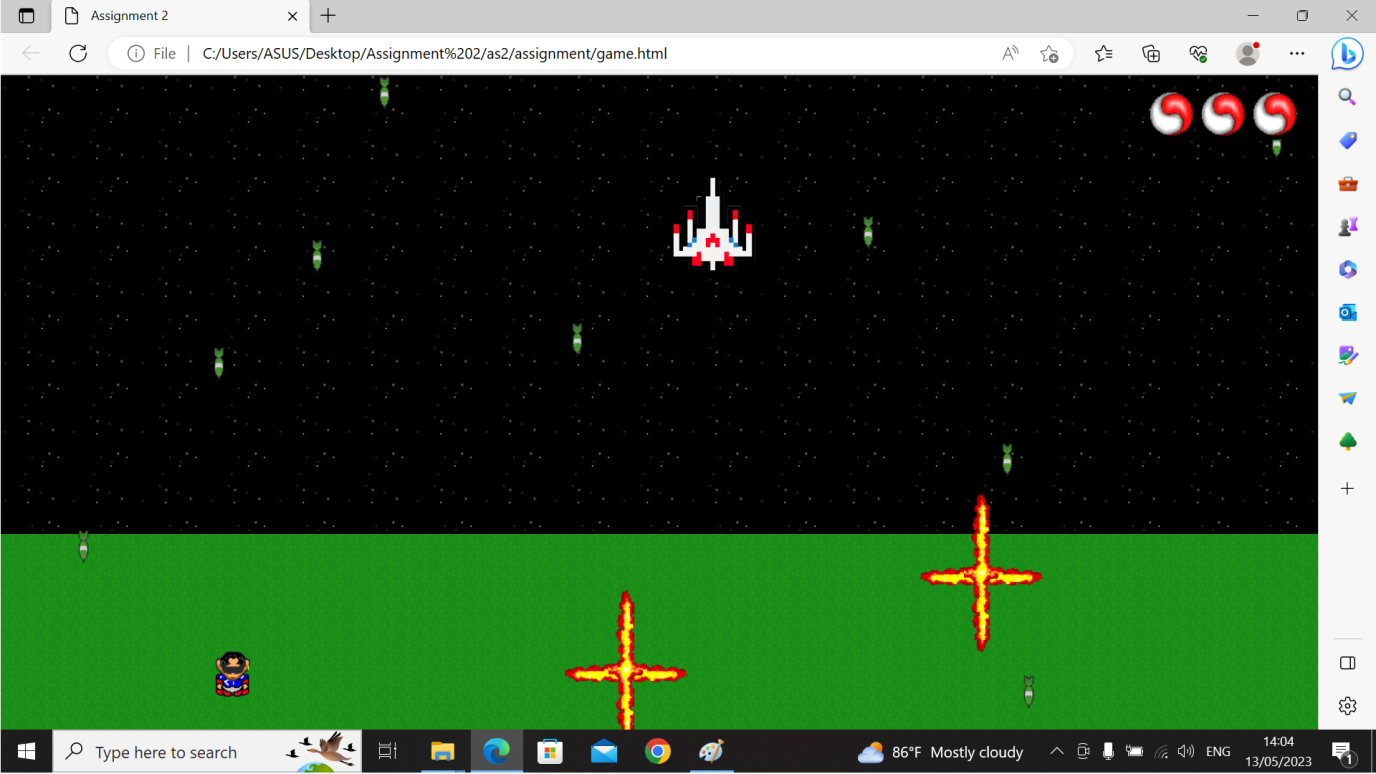


Figure 4: the missiles are exploding on the ground that is displayed on green color.

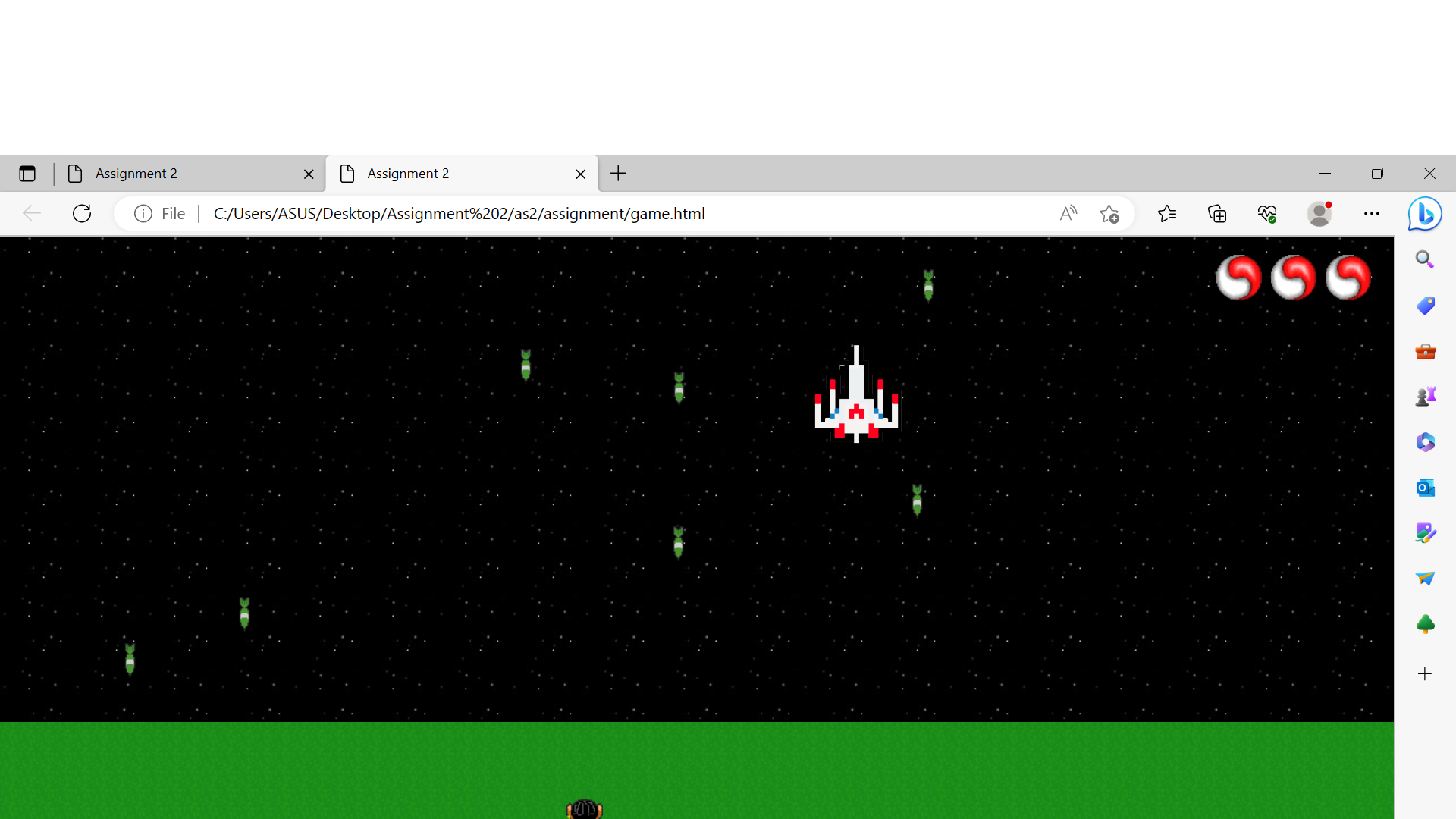


Figure 5: 3 lives of character at first on the top right.

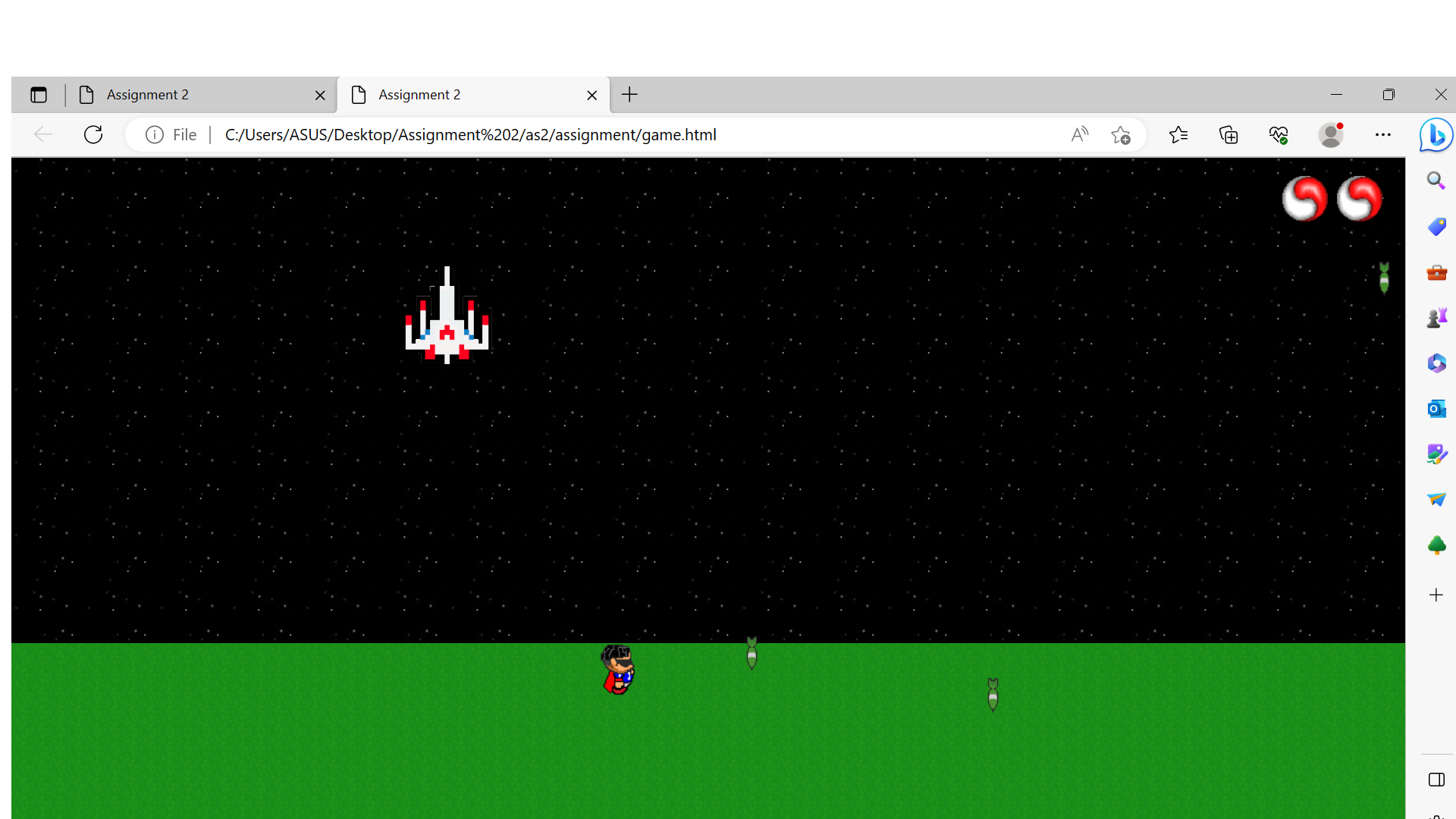


Figure 6: When missile hit the player one life is decreased.

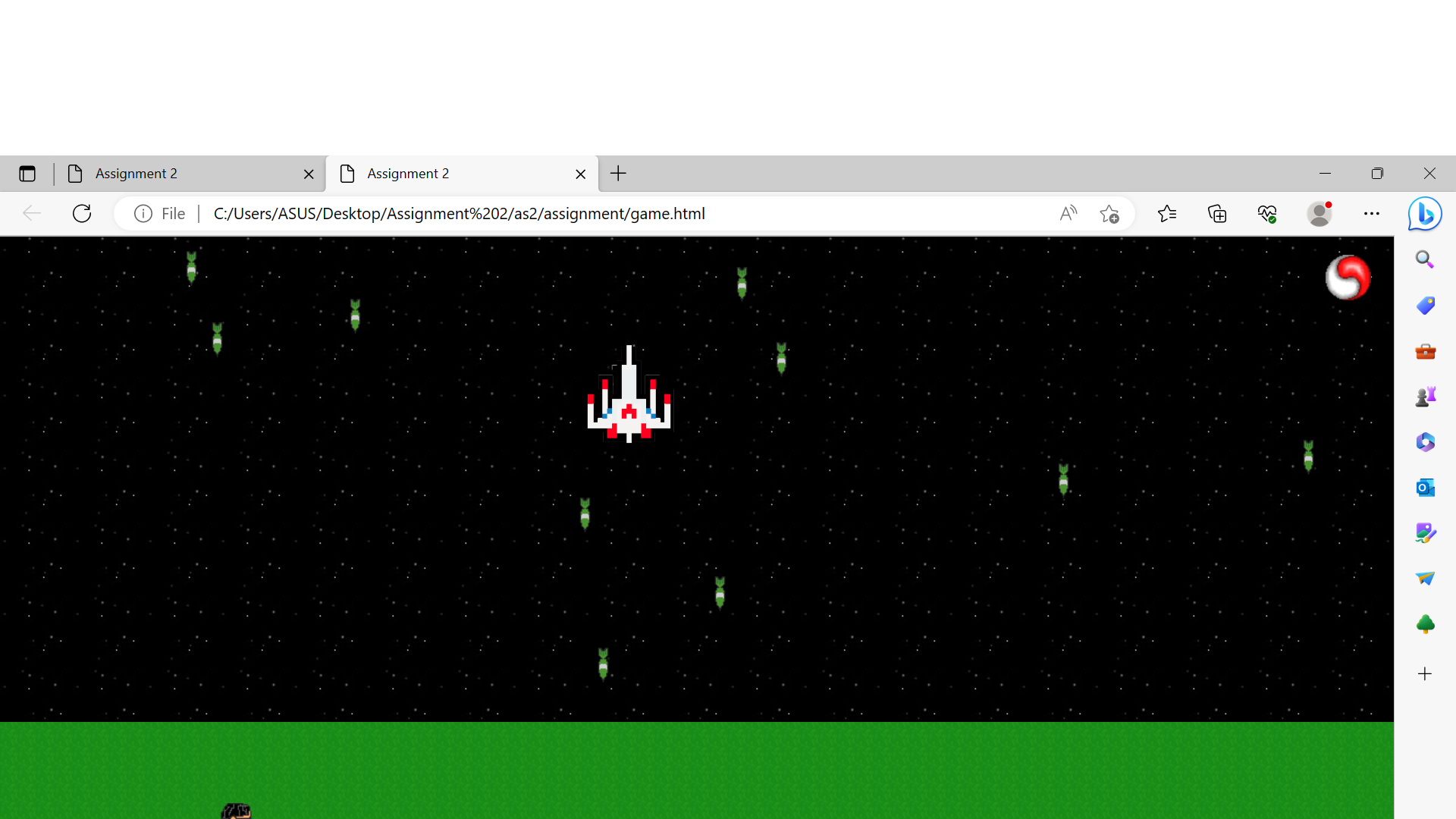


Figure 7: missile are generating from random direction.

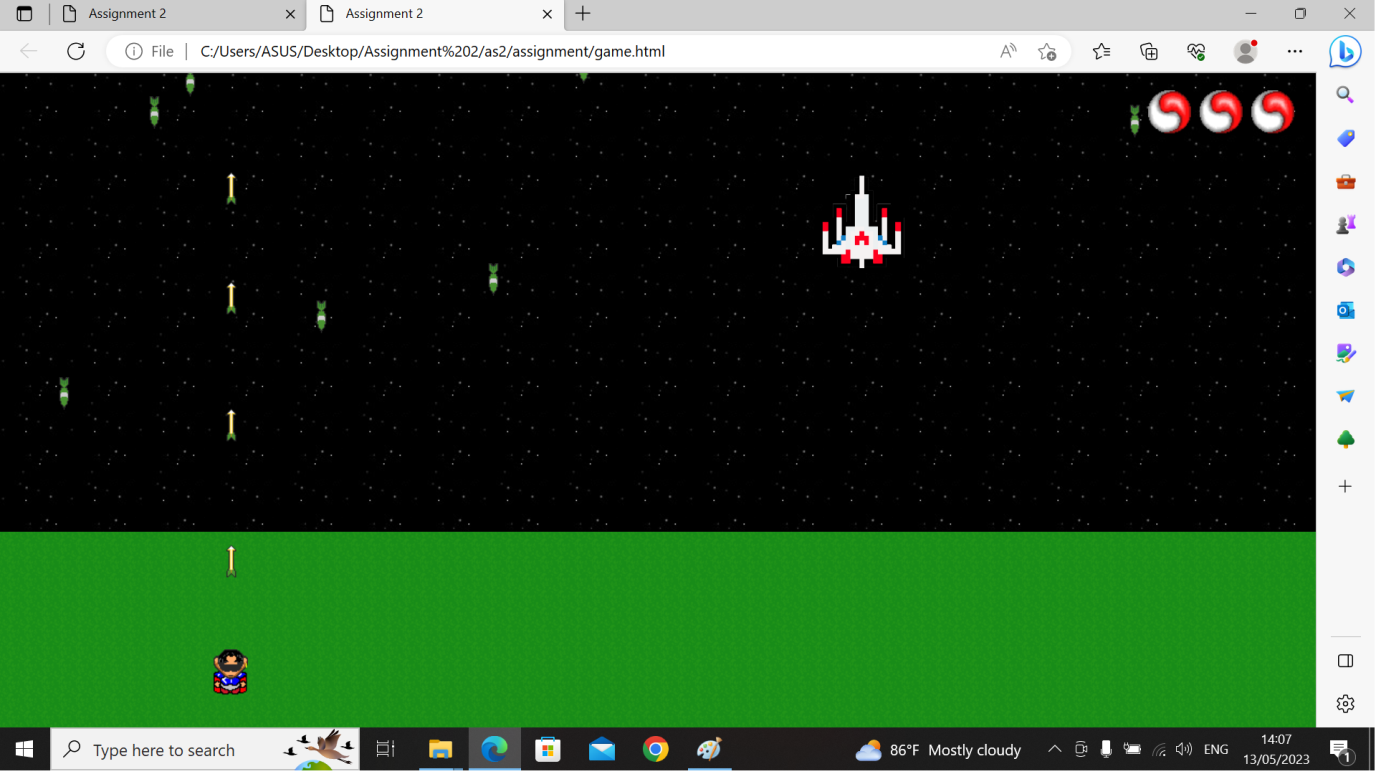


Figure 8:Hitting arrow key on the forward direction when pressed space button.

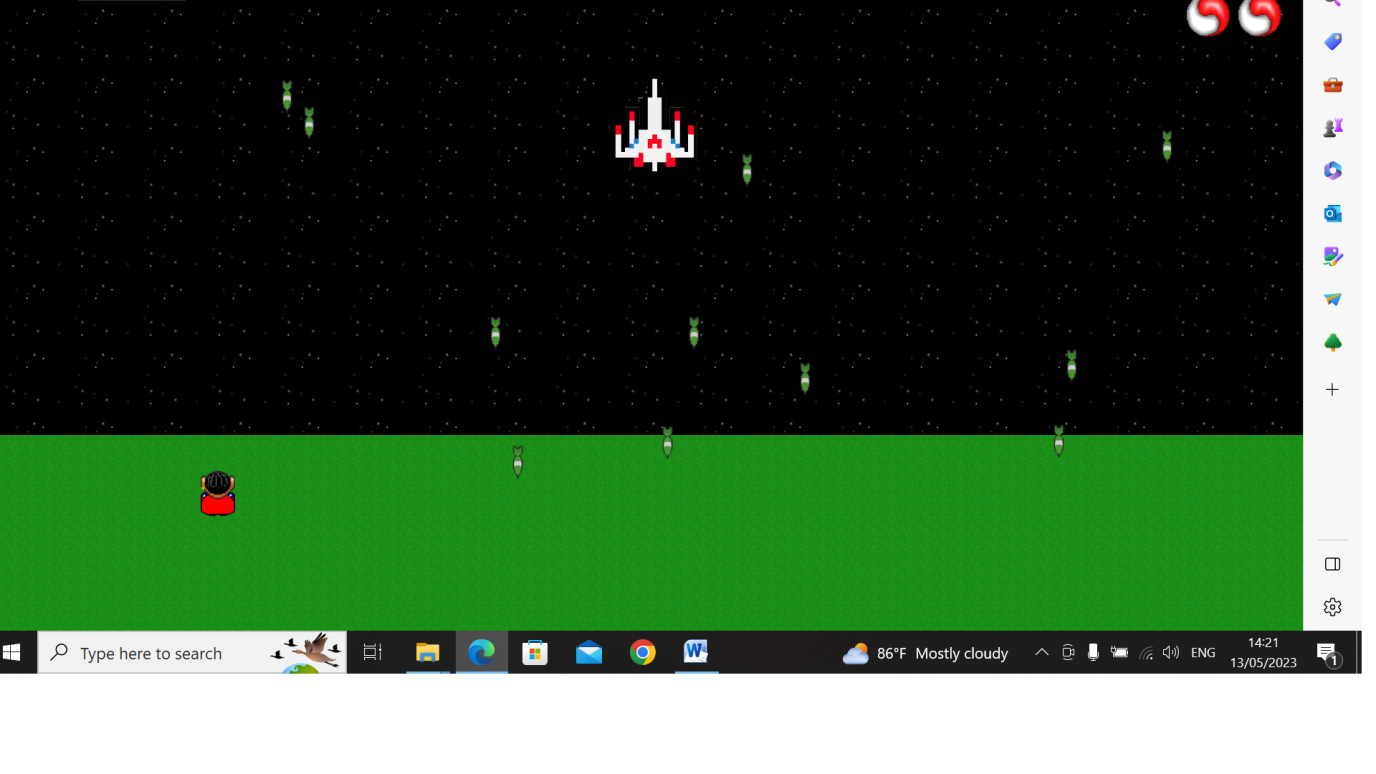


Figure 9: Start button disappears after clicking .

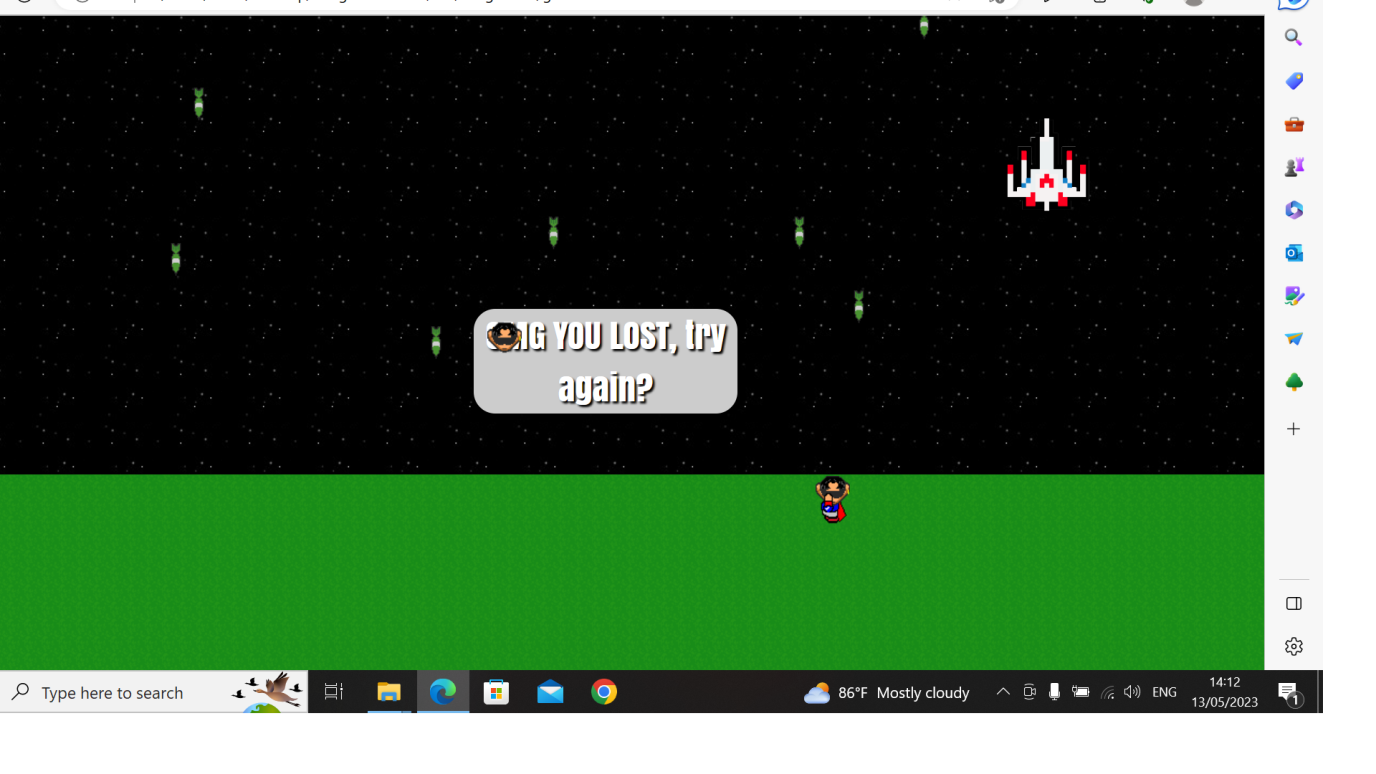


Figure 10: Game over after three lives are gone.

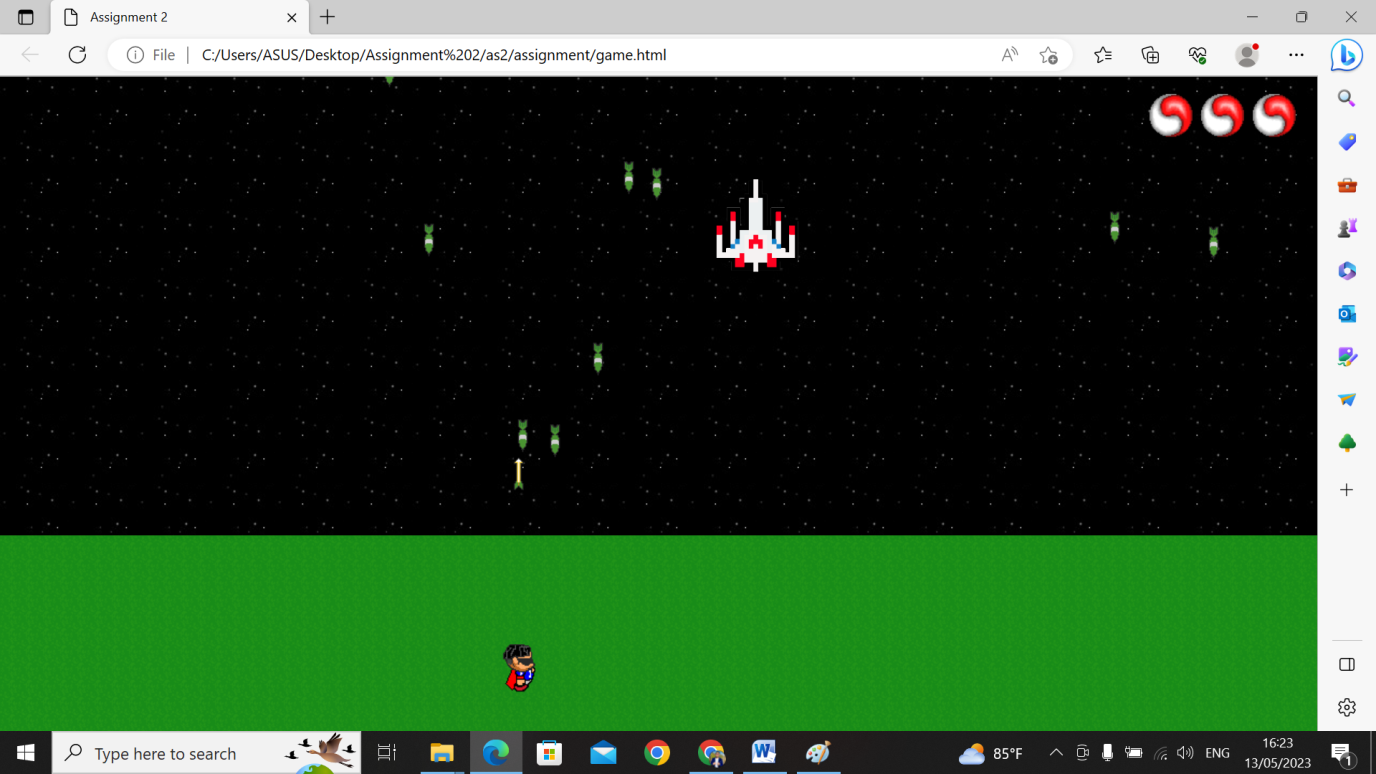


Figure 11: Collision between arrow and missile.

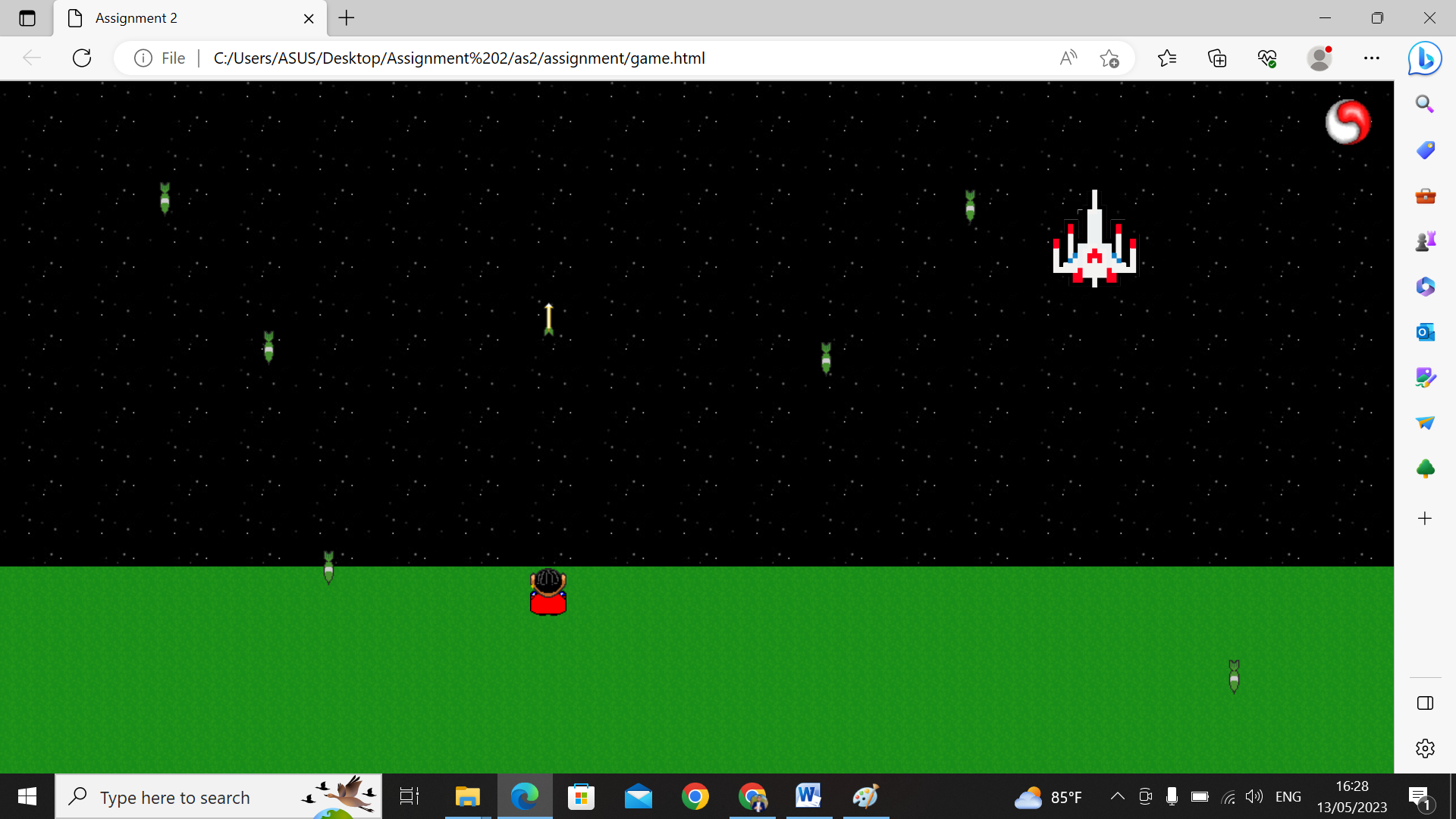


Figure 12: one life remaining in game.

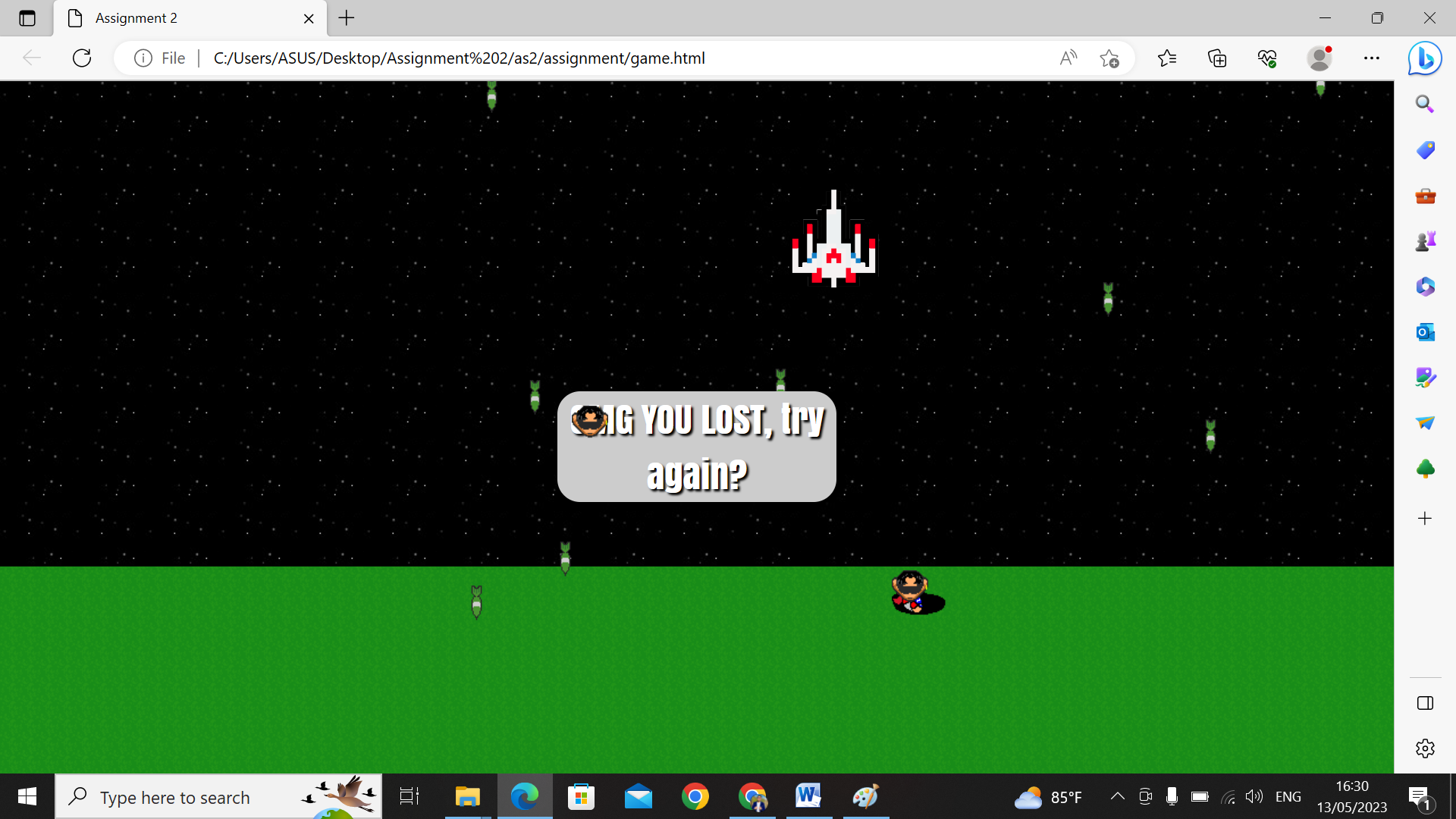


Figure 13: Die Animation after dying.

# Evaluation

There are two weakness or bugs in this game firstly i want to mention the missiles are not coming from the alien the alien is moving continuously left to right and the missiles are come from the top position i want to make my game more different so i did this and i consider it as my weakness. The score system and level have not been defined in this game because the score system has to count per missile, but I did not mention that.  
The two weaknesses are evident in my game. However, apart from that we can enjoy playing the game and I have added a lot of features to make my game more efficient.

As the game worked very well for me in my project, I have added quite a few things to make my project more efficient in order to make it work even more efficiently in the future. It is important that you understand how the game works specifically, that you know the scripting language as well as the query language so that on this project it worked perfectly fine and there are no JS errors.

There are some improvements I could be made are disused below:

A scoring system is essential for any game as it provides players with a sense of progress and accomplishment. I can create a simple scoring system where the player earns points for each enemy they defeat or each level they complete. If missiles are not fired from the correct object, I may need to adjust the code that controls their behavior. Make sure missiles are spawned at the correct location and their trajectory is accurate. You can also add different types of missiles with different behaviors, such as homing missiles or missiles that cause area-of-effect damage. To make your game more engaging and exciting, consider incorporating more features such as power-ups, different enemy types, or obstacles. I can also add different levels that increase in difficulty as the player progresses. Once I've added these improvements and features, thoroughly test my game to ensure its balance and enjoyment. I may need to adjust gameplay or difficulty to make it more engaging for players. I think I could make those kind of improvements to make my project advanced.

As a result of dividing the code into smaller variables and independent modules with specific responsibilities, it would be easy for me to hear it, which in turn would help me to add new features and use a consistent naming convention. It makes my code much more easy to read and understand, especially when I am collaborating on large projects. In addition, my design patterns also play an important role in making my project more efficient. It should then be relatively easy to extend the codebase if you create a well-structured, modular, and documented codebase.

If I had more time what would I do in this game:

If I had more time to accomplish in this game I would include a lot of advanced features such as hiding objects and developing missiles from a variety of aliens I would add the more alien. Moreover, I would enhance the refined mechanics, as well as improve the game visuals which would make my game more complex. To increase player engagement, each level should have unique enemies and objects, so I would also add more levels. The game would also be supported in mobile not just desktop, so I could optimize the controls and layout of the screen to ensure that it is playable on different screen sizes. On the other hand, I would mention sound effects and music. This is because it could greatly enhance the overall experience of the game as well as play a crucial role in any game.

So, if I had more time, I could add those things to make my game better

If I had to develop the same game in the future I would follow a different approach to make my game more quality. I am going o talk about the process. first of all, I would make the plan for the game in advance such as before cresting the gaming process and analyzing the game's maniac project and visual style. it can help ensure that the creation process goes efficiently and smoothly and that the final product meets my expectations. Moreover, I would use a game engine it is used to provide pre-built game mechanics and tools making it easier to develop and test the game till final. Furthermore, I would use version control such as Git can help me keep track of changes to game code and assets and finally, I could use Gather feedback and iterate because throughout the game process gathers feedback from other developers and play testers. it is used to feedback to identify areas for improvement and iterate on the game to make it more engaging and enjoyable, as well as I would add the complex and complicated to make players more attractive.

# Conclusion

In the course of completing this project, I had the opportunity to reflect on the learning module, and I found it to be an extremely useful module to learn. There are many aspects of our daily lives outside of web development that can be applied to this. In the course of the module, I had an excellent experience learning from my tutor in a highly efficient and supportive environment in which to learn. As a student, I learned about HTML, CSS, and JS, which are essential tools and elements of coding. I also discovered a variety of resources and materials to further my learning. As I acquired these languages from scratch, I gradually applied what I learned to my daily life. I found the experience and learning environment highly beneficial.

Speaking of my project, I put a lot of effort into developing it. I included features such as arrow keys, an exploding missile character, and life points. One aspect of the project that was particularly educational was understood how function variables work and be linked together. Additionally, I created a variety of query sections to ensure that my game was suitable for mobile phone use as well as desktop.

My best part of the project was the collision and communication between the missile and arrow keys. I made sure that every function was linked properly with clear and concise comment conventions, making the assignment more accessible for players. To add complexity to the game, I increased the missile speed every 20 seconds. It was generated from a variety of directions in the top section.

After completing the project, I executed many more things related to scripts and programming languages. I explored the connection between two functions and two variables and learned about JavaScript's role in my project. Additionally, I discovered the importance of systematic comments to understand the code. I also discovered how to increase the speed and placement of objects in the game.

Finally, I explored how to add different animation types to make my game even more engaging. Overall, I gained a deeper understanding of how JavaScript works and found it beneficial to learn a programming language or script from scratch. In conclusion, this learning module and project have expanded my knowledge and skills in programming and will continue to be useful in my future endeavors.

# Bibliography

Data, R. (1998, AUG 18). *W3 schools*. Retrieved from W3 schools : https://www.W3schools.com

Robbins, J. N. (2007). *Learning Web Design.*

https://www.learningwebdesign.com

Sims, Z. (2011, AUG 11). Retrieved from Codeacademy: https://www.codeacademy.com

The purpose of this section is to discuss some references that I have found. There are some books and websites that helped me complete this project. Their names and URLs have been provided above. As I was in the middle of doing some assignments, I was in a bind, so I took some help from those sites, and most importantly, I took away an idea on how to solve my problem after reading those websites. I have also found that W3 schools have helped me a great deal by helping me to achieve my goal with in a specific amount of time. To sum up these are the references I used to complete my project.

# Appendix

## Game.js

var upPressed = false;

var downPressed = false;

var leftPressed = false;

var rightPressed = false;

var lastPressed = false;

//------------------------------------------All the varibles are made by user------------------------------------------

var start;

var HP = 3;   //life points

var player;   //player

var Missiles = [];    //all the missile generated

var MovementIncrese = 1; // speed multiplier to increase difficulty

var PlayTime = 0; // game played time

var GameOver = false; // defines game state

var Points = 0; // game Points

var playerHit = false; // player animation when hit

var life; // Life/Health HUD

var PlayerShoot = false; // player animation when fired arrow

var Movements = [];

var high;

var LeftSpeed = [];

var DownSpeed = [];

//-----------------------------------------------------------------------------------------------------------------------------------

W3shools

function startGame() {

    start[0].style.display = 'none';

    move();

    Toop();

    sarAlien();

}

function sarAlien() {

    const alien = document.getElementById("alien");

    const maxLeft = document.body.clientWidth - alien.offsetWidth;

    const currentLeft = parseInt(alien.style.left) || 0;

    const direction = alien.dataset.direction === "right" ? 1 : -1;

    let newLeft = currentLeft + 10 \* direction;

    if (newLeft > maxLeft) {

        newLeft = maxLeft;

        alien.dataset.direction = "left";

    } else if (newLeft < 0) {

        newLeft = 0;

        alien.dataset.direction = "right";

    }

    alien.style.left = newLeft + "px";

}

//---------------------The function is to generate the missile ---------------------------------------------------------------------------------------------------------------

function Misile() {

    var missile = document.createElement('Div');

    missile.className = 'bomb'

    document.body.appendChild(missile);

    span(missile);

    Missiles.push(missile);

    LeftSpeed.push(0);

    DownSpeed.push(1);

}

function span(missile) {

    var ScreenWidth = window.innerWidth;

    var RandoWidth = Math.ceil(Math.random() \* ScreenWidth);

    var RandoHeight = Math.random() \* (1000) + 1;

    missile.style.top = -1 \* RandoHeight + 'px';

    missile.style.left = RandoWidth + 'px';

}

// if the game is over

function Toop() {

    // if the game is over don't need to do anything

    if (GameOver) {

        return;

    }

    // when the game time is 0 change of speed

    if (PlayTime > 100) {

        if (MovementIncrese < 6) {

            for (var i = 0; i < 5; i++) {

                Misile();   //creates more bomb

            }

            MovementIncrese = MovementIncrese + 0.75;

            PlayTime = 0;

        }

    }

    //---------------this code is used to generated the missile from the top direction -------------

    for (var i = 0; i < Missiles.length; i++) {

        var BombTopPos = parseFloat(Missiles[i].offsetTop);

        ExplodeMissile = Math.floor(Math.random() \* (window.innerHeight - sky[0].offsetHeight)) + sky[0].offsetHeight;

        //-------drop bomb on the bottom portion  on the screen--------------------------------

        if (Missiles[i].offsetTop == ExplodeMissile || Missiles[i].offsetTop > window.innerHeight) {

            Pataka(Missiles[i]);

            span(Missiles[i]);

            SpeedLeft(i);

            DownSpeed[i] = ((Math.ceil(Math.random() \* 2))) + 1;

        }

        else {

            if (ElementCollide(player, Missiles[i]) != 0) {

                Pataka(Missiles[i]);    // when missile hits the payer explode the bomb

                span(Missiles[i]);  // rebot the exploded missile

                SpeedLeft(i);

                DownSpeed[i] = ((Math.ceil(Math.random() \* 2))) + 1;

            }

            else {

W3Schools

                var TopPos = BombTopPos + DownSpeed[i] \* MovementIncrese;

                var LeftPos = parseFloat(Missiles[i].offsetLeft);

                Missiles[i].style.top = TopPos + 'px';

                if (LeftPos > window.innerWidth - Missiles[i].offsetWidth || LeftPos < 0) {

                    SpeedLeft(i);

                }

                Missiles[i].style.left = LeftPos + LeftSpeed[i] + 'px';

            }

        }

    }

}

learning web design

//---------------------This function is stop the game and show the score board after die-------------------------------------------------------------------------------------------------------------

function Maro() {

    GameOver = true;

    player.className = 'character dead';

    var btn = document.createElement('div');

    btn.className = 'start';

    var DeadPromt = document.createTextNode('OMG YOU LOST, try again?');

    btn.appendChild(DeadPromt);

    btn.addEventListener('click', () => location.reload());

    document.body.appendChild(btn);

    //----------------------------This code will display the score  --------------------------------------------------------------------------------------------------------

    var ProPlayer = localStorage.key(0);

    var MaxPoints = parseInt(localStorage.getItem(ProPlayer));  //collect total point

    var MaxPoints = document.createElement('div');

    document.body.appendChild(MaxPoints);

    MaxPoints.className = 'Points';

    if (!MaxPoints || Points > MaxPoints) {

        localStorage.clear(); MaxPoints

        var PlayerName = prompt("You've scored the highest, save your score with a name");      //score is new ask the player name

        ProPlayer = PlayerName;     //changes the name of player

        MaxPoints = Points; // changes  the player gained score

    }

    localStorage.setItem(ProPlayer, MaxPoints);     //stores player name and score

    MaxPoints.innerHTML = 'Player Name: ' + ProPlayer + '<br> ' + 'HighScore: ' + MaxPoints;    //shows player name and score at the end after die.

    return;

    //---------------------------------------------------------------------------------------------------------------------------------------------------------

}

//--------------------------------this code is for collision--------------------------------------------------------------------------------------------------------------------

function ElementCollide(ElementA, ElementB) {

    //to detect height,width and top/left of an object for collision

    if (ElementA.offsetTop < ElementB.offsetTop + ElementB.offsetHeight) {

        if (ElementA.offsetTop + ElementA.offsetHeight > ElementB.offsetTop) {

            if (ElementA.offsetLeft < ElementB.offsetLeft + ElementB.offsetWidth) {

                if (ElementA.offsetLeft + ElementA.offsetWidth > ElementB.offsetLeft) {

                    return true;

                }

            }

        }

    }

    return false;

}

**Codeacademy**

// this function is used to fire key

function ArrowAayo(arrows, FirInterval) {

    var ArrowPosTop = arrows.offsetTop;

    arrows.style.top = ArrowPosTop - 1 + 'px';

    for (var i = 0; i < Missiles.length; i++) {

        if (ElementCollide(Missiles[i], arrows)) {

            arrows.remove();        //removes arrow

            PlayTime = PlayTime + 2;        //calculates gameplay timers

            span(Missiles[i]);  //span bomb's position

            SpeedLeft(i);

            DownSpeed[i] = ((Math.ceil(Math.random() \* 2))) + 1;

            Points = Points + 10;           //updates the Points with arrow collides

            high.innerHTML = 'Points:' + Points;

            clearInterval(FirInterval);

        }

        if (ArrowPosTop < 0) // removes arrow if arrow goes outside the screen

        {

            arrows.remove();

            clearInterval(FirInterval);

        }

    }

}

function hola() {

    var arrows = document.createElement('div');

    player.classList.add('fire');

    arrows.className = 'arrow up';

    document.body.appendChild(arrows);

    arrows.style.top = player.offsetTop + 'px';

    arrows.style.left = player.offsetLeft + 'px';

    PlayerShoot = true;

    setTimeout(

        function () {

            player.classList.remove('fire');

            PlayerShoot = false;

        }, 500);

    //go arrow in top direction.

    var FirInterval = setInterval(

        function () {

            ArrowAayo(arrows, FirInterval)

        }

        , 5);

}

//------------------------------------------All the function is to detect bomb location and  hp loss.----------------------------

function Pataka(bomb) {

    var explosion = document.createElement('div');

    explosion.className = 'explosion';

    document.body.appendChild(explosion);

    explosion.style.top = bomb.offsetTop + 'px';

    explosion.style.left = bomb.offsetLeft + 'px';

    console.log("COOL99")

    if (ElementCollide(player, explosion)) {

        console.log("COol")

        plauyerHit = true; //sets player collision to true

        player.classList.add('hit');    //plays human's animation when hit

        HP--; //decrease one health point

        life[0].remove(); //removes life from top left of the screen

        if (HP == 0) //if all 3 lives are gone call die function

        {

            Maro();

        }

        setTimeout(function () {

            playerHit = false; //sets player's state to normal

            player.classList.remove('hit'); //also removes the exploded animation

        }, 1000);

    }

    else {

        high.innerHTML = 'Points:' + Points;

        Points = Points + 5; //if player didnt get hit increment score points

        PlayTime++; //increment play time

    }

    setTimeout(function () {

        explosion.remove(); //explosion is removed after 100ms

    }, 100);

}

//----------------------------------The following function is for giving random speed in different  random direction-----------------------------------------------------------------------------------------------------------------------

function SpeedLeft(Index) {

    var rando = Math.floor(Math.random() \* 2);

    if (rando == 0) {

        LeftSpeed[Index] = ((Math.ceil(Math.random() \* 3)));

    }

    else {

        LeftSpeed[Index] = ((Math.ceil(Math.random() \* 3))) \* -1;

    }

}

function keyup(event) {

    if (GameOver) {

        return;

    }

    if (event.keyCode == 37) {

        leftPressed = false;

        lastPressed = 'left';

    }

    if (event.keyCode == 39) {

        rightPressed = false;

        lastPressed = 'right';

    }

    if (event.keyCode == 38) {

        upPressed = false;

        lastPressed = 'up';

    }

    if (event.keyCode == 40) {

        downPressed = false;

        lastPressed = 'down';

    }

    player.className = 'character stand ' + lastPressed;

}

function move() {

    if (GameOver || PlayerShoot) {

        return;

    }

    var positionLeft = player.offsetLeft;

    var positionTop = player.offsetTop;

    if (downPressed) {

        if (player.offsetTop < window.innerHeight - 30) {

            var TopPos = positionTop + 2;

            player.style.top = TopPos + 'px';

        }

        if (leftPressed == false) {

            if (rightPressed == false && !playerHit) {

                player.className = 'character walk down';

            }

        }

    }

    if (upPressed) {

        var TopPos = positionTop - 2;

        var element = document.elementFromPoint(0, TopPos);

        if (element.classList.contains('sky') == false) {

            player.style.top = TopPos + 'px';

        }

        if (leftPressed == false) {

            if (rightPressed == false && !playerHit) {

                player.className = 'character walk up';

            }

        }

    }

    if (leftPressed) {

        var newLeft = positionLeft - 2;

        if (newLeft > 0) {

            player.style.left = newLeft + 'px';

        }

        if (!playerHit) { player.className = 'character walk left'; }

    }

    if (rightPressed) {

        var newLeft = positionLeft + 2;

        var element = document.elementFromPoint(0, player.offsetTop);

        if (newLeft < window.innerWidth - player.offsetWidth) {

            player.style.left = newLeft + 'px';

        }

        if (!playerHit) {

            player.className = 'character walk right';

        }

    }

}

function keydown(event) {

    if (GameOver || PlayerShoot) {

        return;

    }

    if (event.keyCode == 32) {

        hola();

    }

    if (event.keyCode == 37) {

        leftPressed = true;

    }

    if (event.keyCode == 39) {

        rightPressed = true;

    }

    if (event.keyCode == 38) {

        upPressed = true;

    }

    if (event.keyCode == 40) {

        downPressed = true;

    }

}

// keeps on calling start game function 60times/1sec

function pap() {

    setInterval(startGame, 1000 / 60);

}

// When the page is load player calls 20 bombs

function myLoadFunction() {

    high = document.createElement('div');

    high.className = 'Points';

    document.body.appendChild(high);

    player = document.getElementById('player');

    document.addEventListener('keydown', keydown);

    document.addEventListener('keyup', keyup);

    start = document.getElementsByClassName('start');

    start[0].addEventListener('click', pap);

    sky = document.getElementsByClassName('sky');

    for (var i = 0; i < 14; i++) {

        Misile();

    }

    life = document.getElementsByClassName('health')[0].getElementsByTagName('li');

}

document.addEventListener('DOMContentLoaded', myLoadFunction);

W3 school