Christopher Quach

Your Student ID: 923836573

rexchris2

CSC317-Section # 1

Assignment 3 - JavaScript Game

Description:
In this assignment we are required to make a game in javascript. Some games are banned. We have to make a javascript game that works and runs decently. We learned and used the canvas modules for our game. The main goal was to learn how to use the canvas tag and use javascript to draw it. We could use images and sound if we wanted. For instance we can just use a box we drew or we can add an image to replace it also with the other objects that are present in the game we could use images. Sound as well can help improve the game playing field making it more immersive. Also no frameworks are allowed and the alert().

Approach / What I Did:
The approach I did was I first played around with the code that was provided on W3 school. Then after playing around with W3 school code for their game. It gave me more ideas on what game I should make. I wanted to make a shooter game like Space invaders. But it was different. The enemies spawned one at a time randomly and you only have 1 life. But first I started with the controls on moving the ship. I made it so the player can move it up down left right with the arrow keys and they can also shoot bullets. After that I used some code from W3 schools for movement and collision. So when the enemy crashes into the player it will end the game. Then after that I added images for the player and enemies and then the background. After that I added sound for the bullets being shot and when the bullets hit an enemy and also background music and death music. After adding all that I made a start screen and then a death screen so that the player can press the restart button to restart the game. I got the sounds and music from soundfxcenter. Note that the bullet function and the border around the canvas was code that was generated from ChatGPT. It's not mine!!

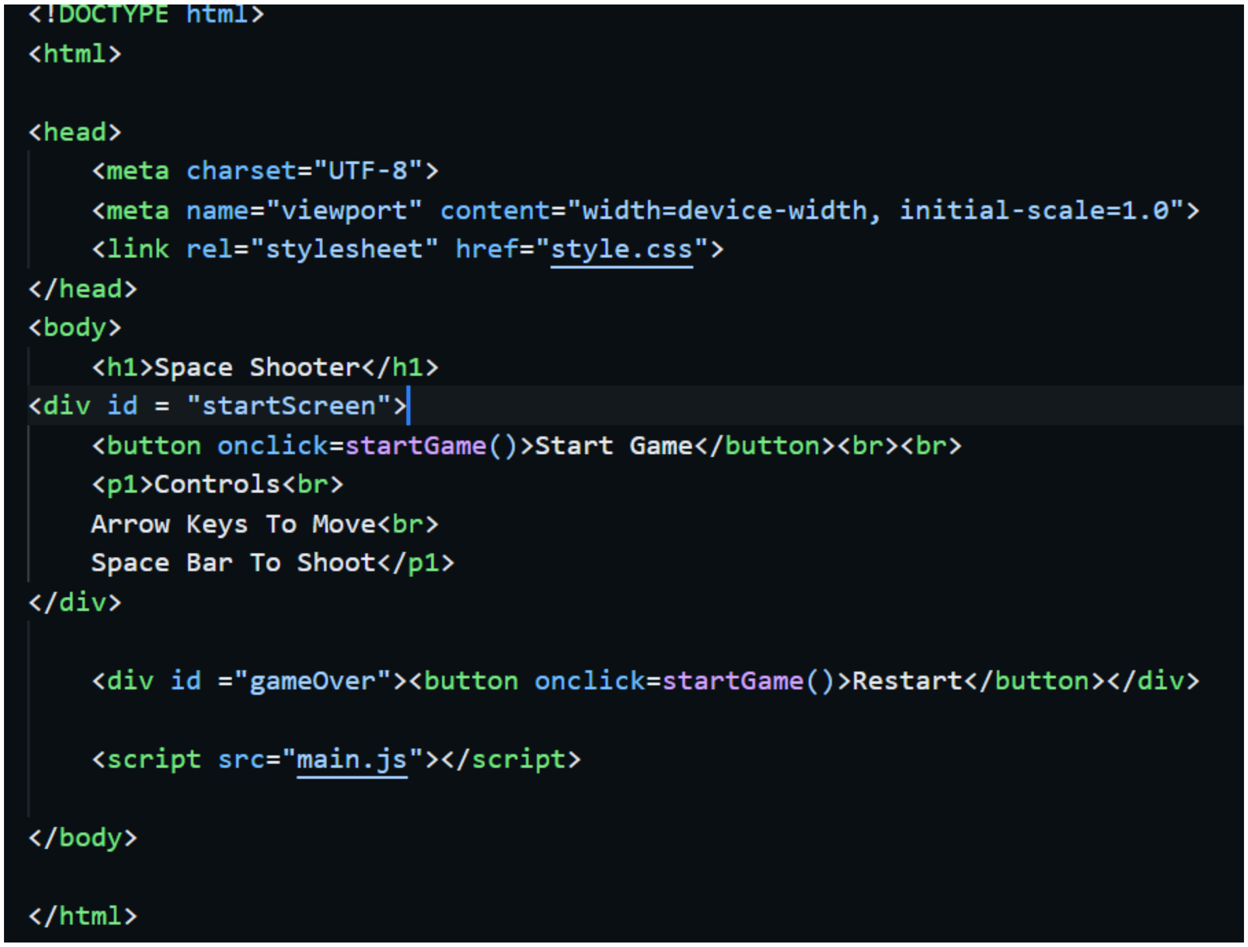
Issues and Resolutions:
Write about issues you encountered while working this assignment and how you resolved them.

Issues I encountered were errors with collision, death restart, bullet shooting fire rate,adding moving background pictures. First I really couldn't figure out how I would make the bullet. I made it work a little bit but it just shot an entire line instead of a singular bullet so I had chat gpt help me out with that one. Then after I was struggling with the first rate of the bullet. The player could hold down the shoot button and it would shoot an entire line of bullets with no delay so I had chat gpt help me with that. Eventually it worked and the bullet function was good. Another was the death restart button. When you died and pressed restart it wouldn’t really restart but it would just continue the game and everything was moving much faster and faster and you wouldn’t die at all. So I figured that I had to reset the variable in the startgame function like resetting the enemy speed back to 0 and the enemies killed, etc.. After I did that it worked out fine the player was reseted and all the enemies variable was reseted to the beginning.

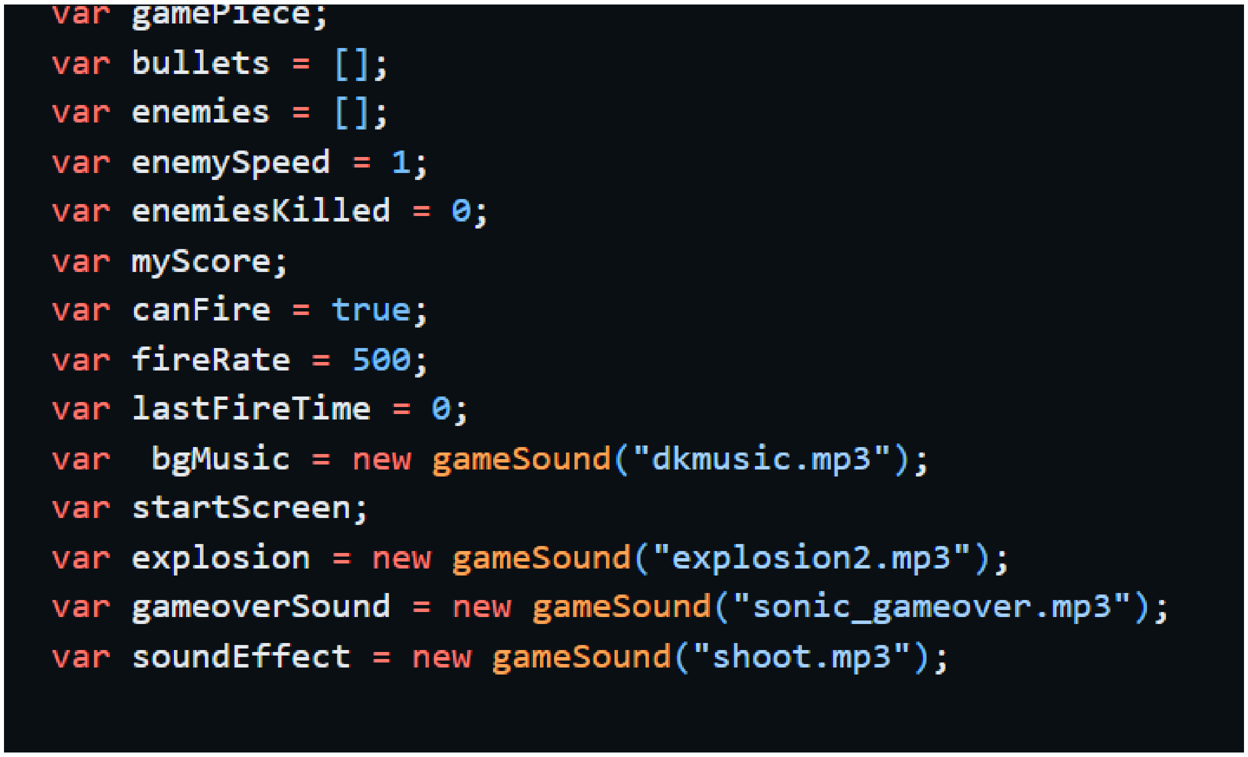
Analysis:
Reflect on what you have learned and how decisions made could impact users.)

What I learned from making these games is a lot. I learned how to make a canvas and add objects into it. I learned how to make it move with control using the keyboard. I learned how to implement the bullet which can help me in future projects that I would need to shoot. Same with the fire rates of the bullets. I learned how to put together a game and how everything runs together. I learned how collisions work with other objects and I learned that x,y plays a big part in making the game. I already knew how sound and background music works from another project I did. This game was different because I did a project with Java in making a game but it had its own framework so It was straightforward to use as they already had functions set up for you. But with this javascript with no function was a little more challenging but once you get the functions down it was easy to puzzle the game together.

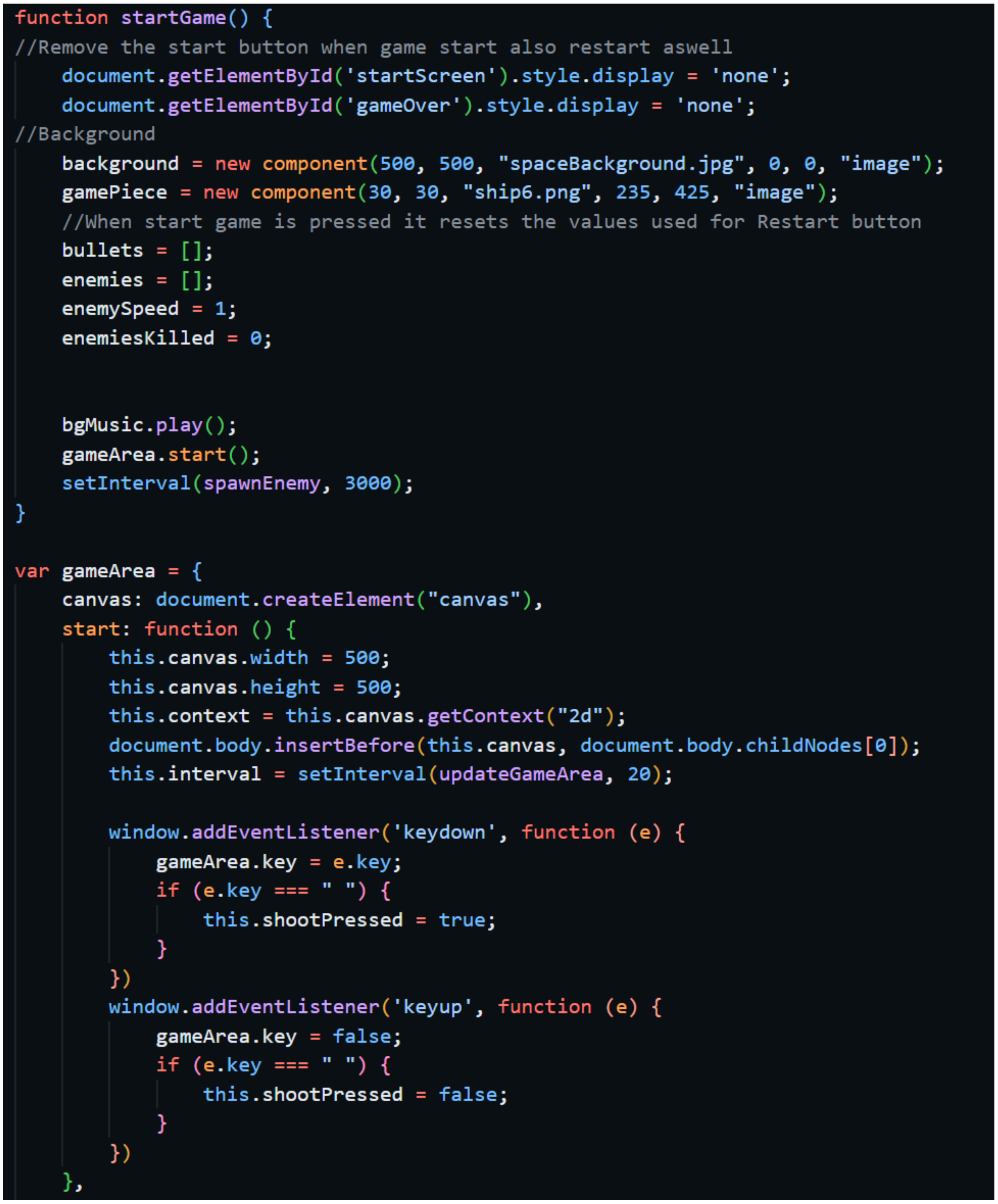
Screen shots:
Provide several screenshots of steps you have taken while working on this assignment.



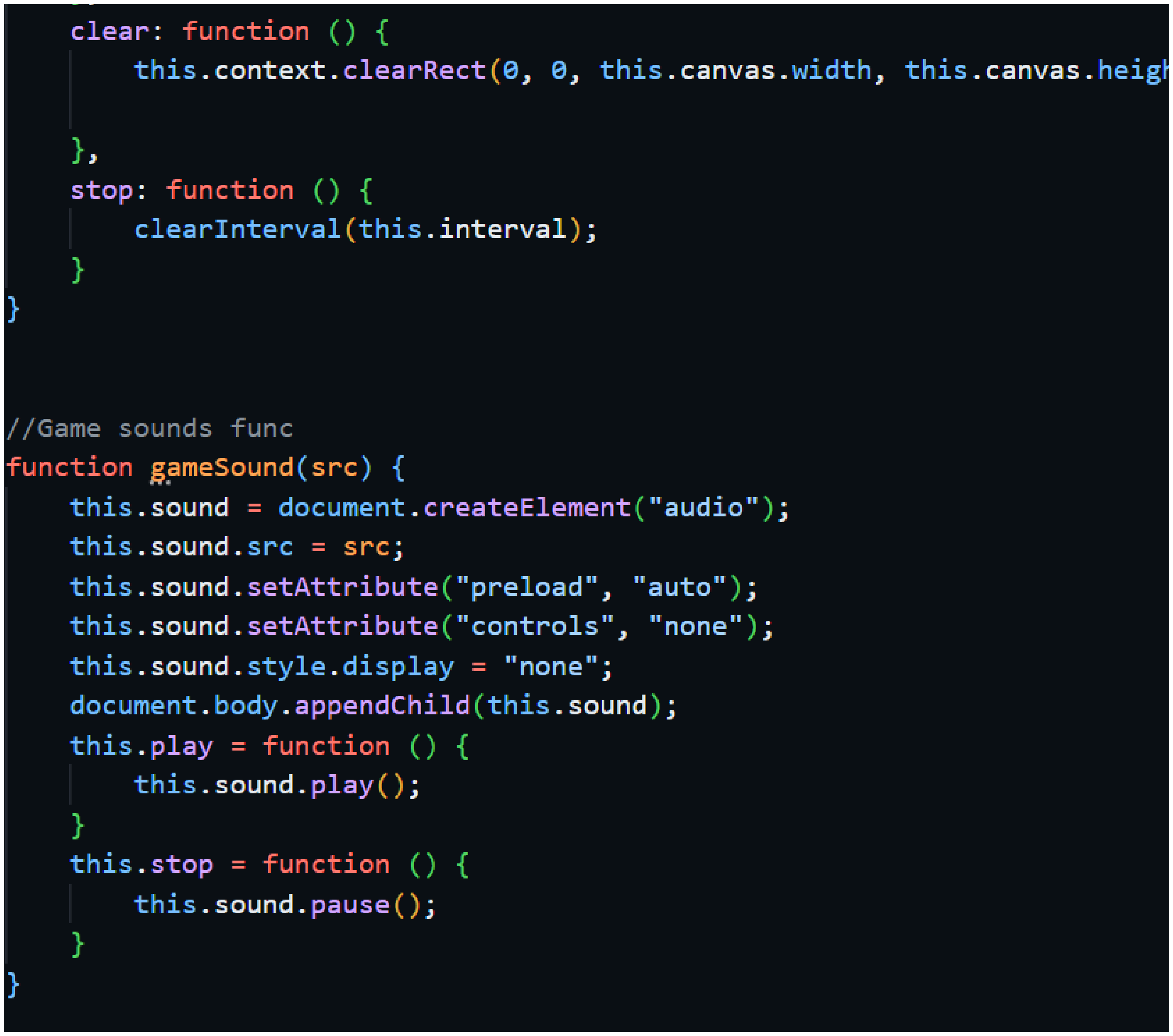
```html
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1>Space Shooter</h1>
<div id = "startScreen">
    <button onclick=startGame()>Start Game</button><br><br>
    <p1>Controls<br>
    Arrow Keys To Move<br>
    Space Bar To Shoot</p1>
</div>

    <div id ="gameOver"><button onclick=startGame()>Restart</button></div>

    <script src="main.js"></script>

</body>

</html>
```



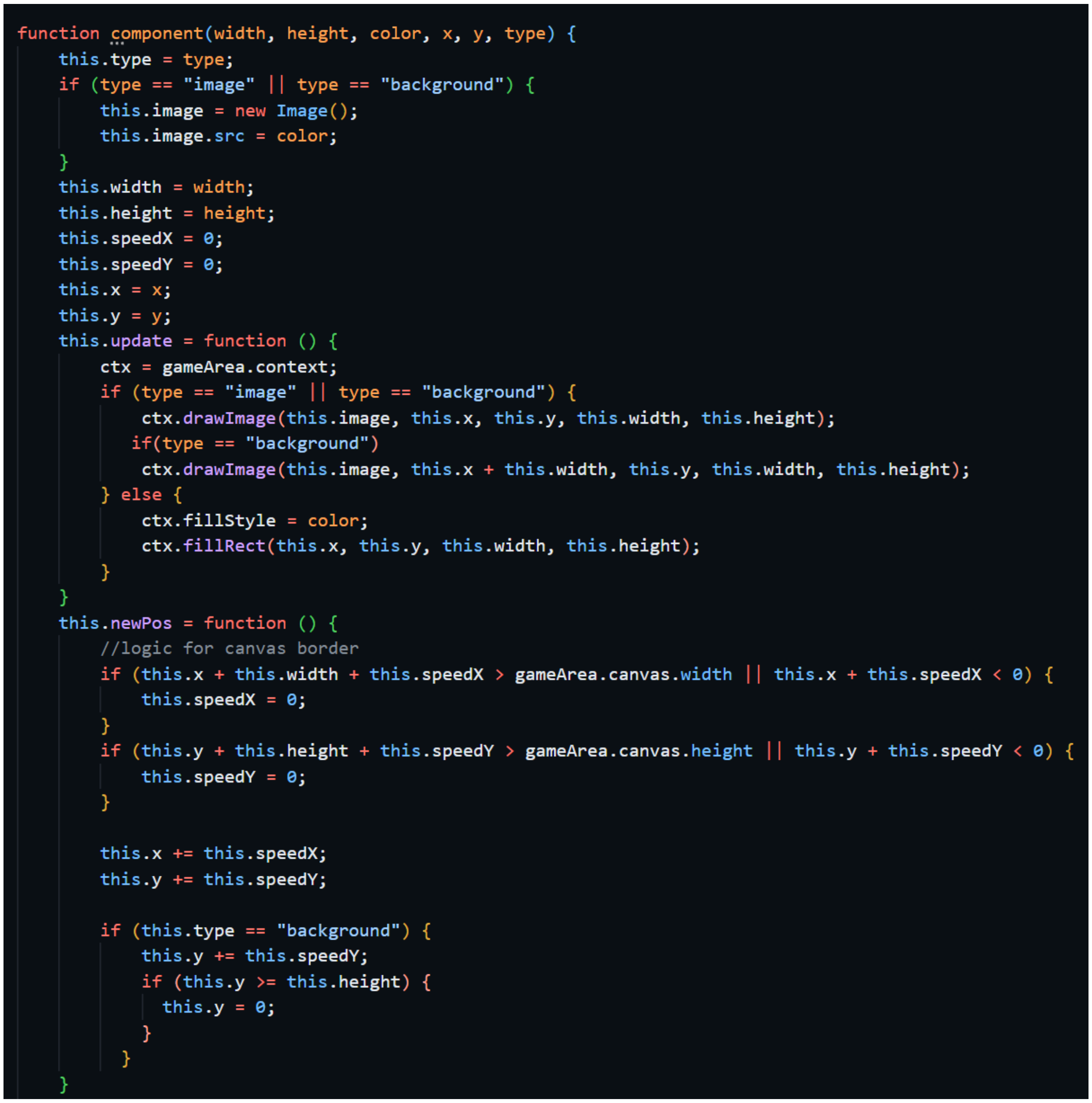
```javascript
var gamePiece;
var bullets = [];
var enemies = [];
var enemySpeed = 1;
var enemiesKilled = 0;
var myScore;
var canFire = true;
var fireRate = 500;
var lastFireTime = 0;
var  bgMusic = new gameSound("dkmusic.mp3");
var startScreen;
var explosion = new gameSound("explosion2.mp3");
var gameoverSound = new gameSound("sonic_gameover.mp3");
var soundEffect = new gameSound("shoot.mp3");
```



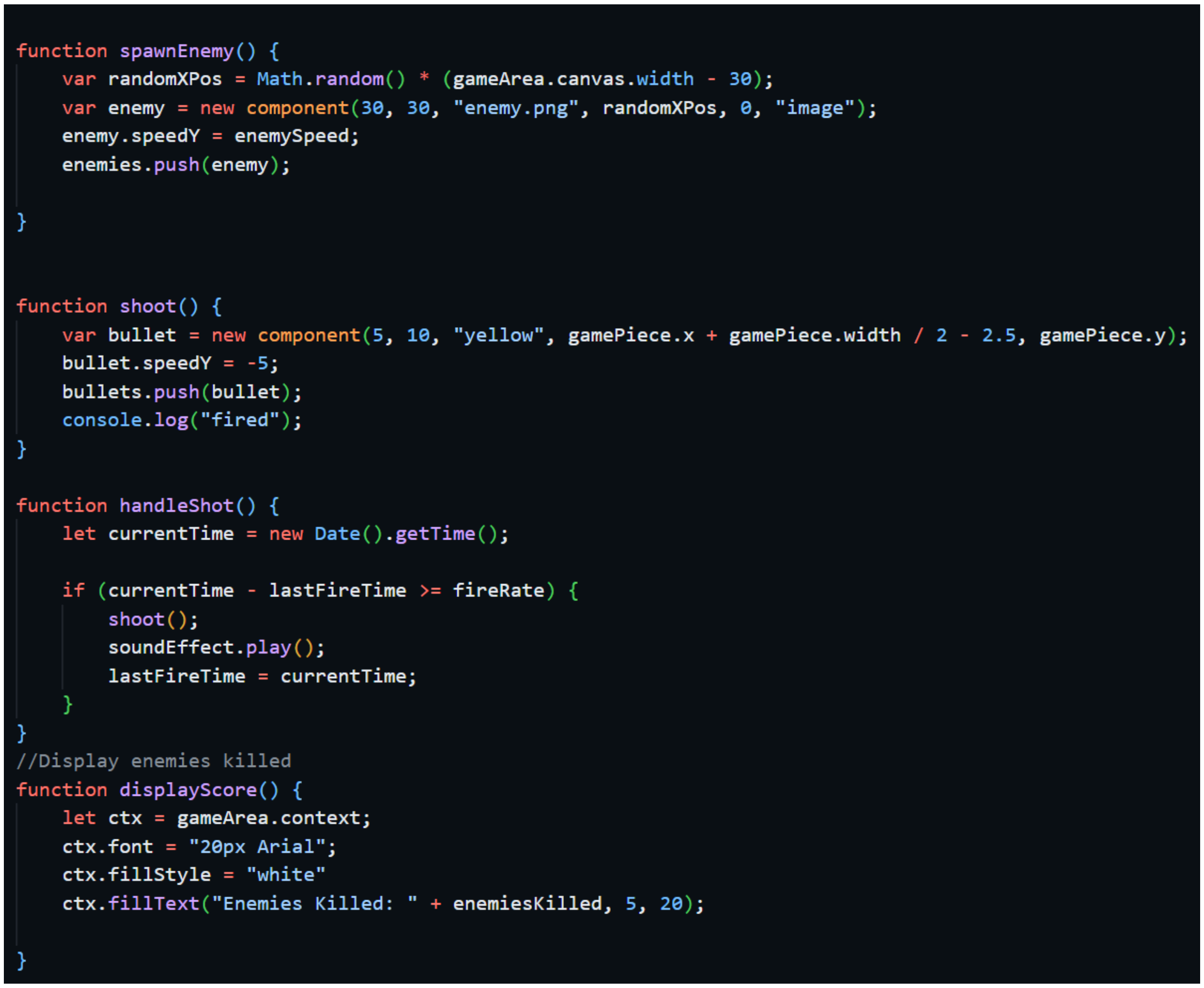
```
function startGame() {
//Remove the start button when game start also restart aswell
    document.getElementById('startScreen').style.display = 'none';
    document.getElementById('gameOver').style.display = 'none';
//Background
    background = new component(500, 500, "spaceBackground.jpg", 0, 0, "image");
    gamePiece = new component(30, 30, "ship6.png", 235, 425, "image");
    //When start game is pressed it resets the values used for Restart button
    bullets = [];
    enemies = [];
    enemySpeed = 1;
    enemiesKilled = 0;


    bgMusic.play();
    gameArea.start();
    setInterval(spawnEnemy, 3000);
}

var gameArea = {
    canvas: document.createElement("canvas"),
    start: function () {
        this.canvas.width = 500;
        this.canvas.height = 500;
        this.context = this.canvas.getContext("2d");
        document.body.insertBefore(this.canvas, document.body.childNodes[0]);
        this.interval = setInterval(updateGameArea, 20);

        window.addEventListener('keydown', function (e) {
            gameArea.key = e.key;
            if (e.key === " ") {
                this.shootPressed = true;
            }
        })
        window.addEventListener('keyup', function (e) {
            gameArea.key = false;
            if (e.key === " ") {
                this.shootPressed = false;
            }
        })
    },
```



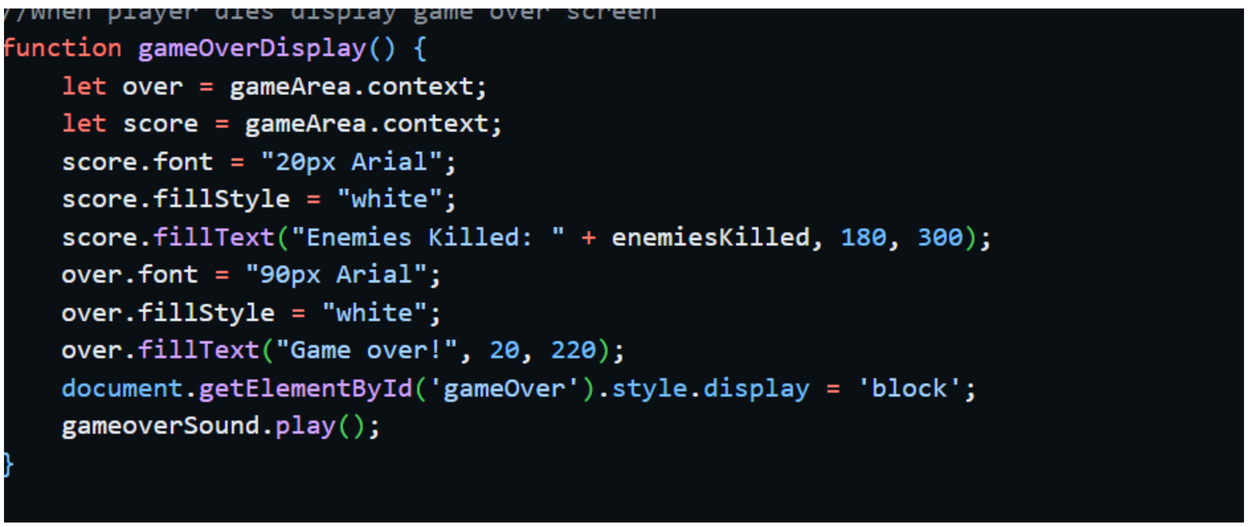
```
    clear: function () {
        this.context.clearRect(0, 0, this.canvas.width, this.canvas.heig

    },
    stop: function () {
        clearInterval(this.interval);
    }
}


//Game sounds func
function gameSound(src) {
    this.sound = document.createElement("audio");
    this.sound.src = src;
    this.sound.setAttribute("preload", "auto");
    this.sound.setAttribute("controls", "none");
    this.sound.style.display = "none";
    document.body.appendChild(this.sound);
    this.play = function () {
        this.sound.play();
    }
    this.stop = function () {
        this.sound.pause();
    }
}
```



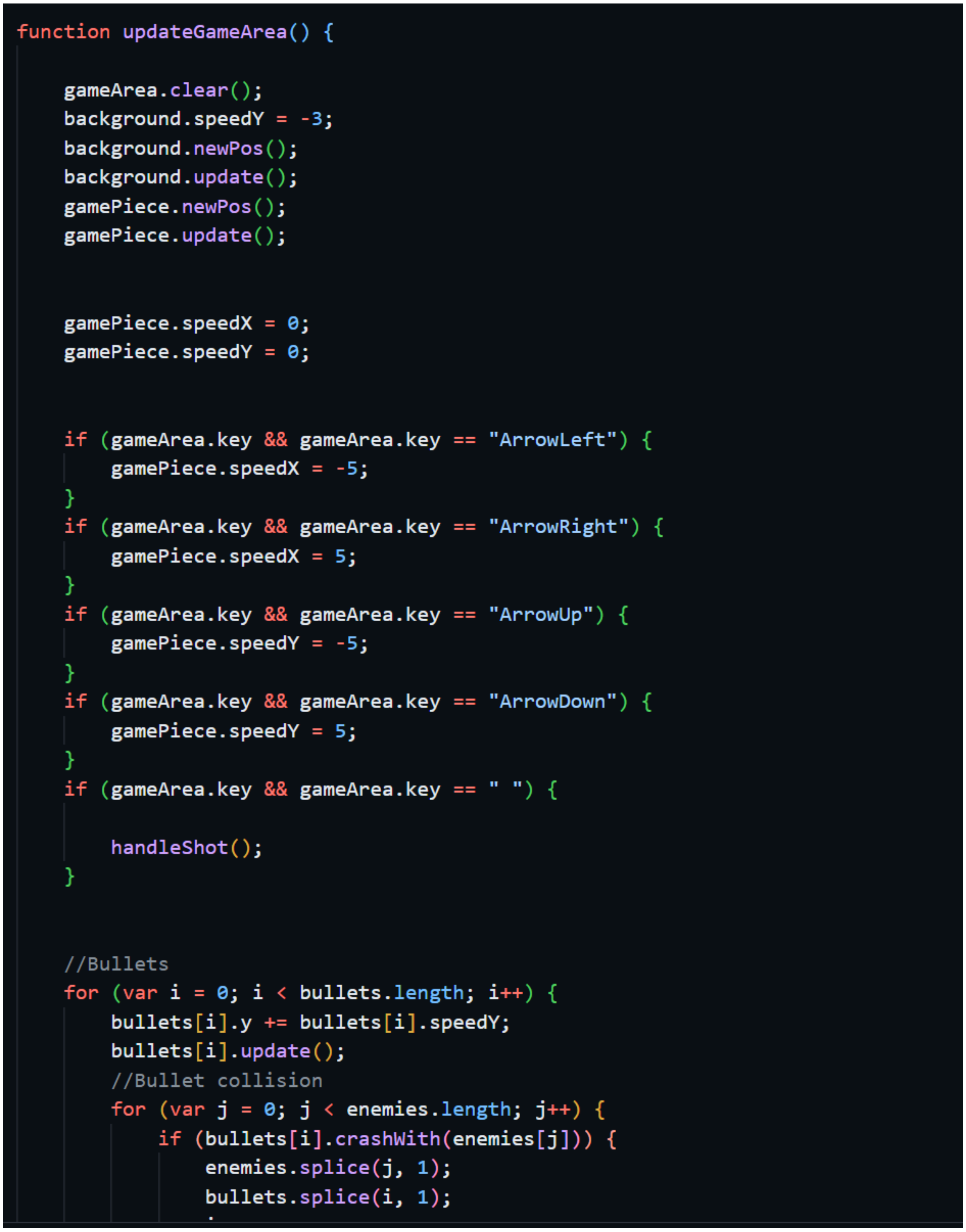
```
function component(width, height, color, x, y, type) {
    this.type = type;
    if (type == "image" || type == "background") {
        this.image = new Image();
        this.image.src = color;
    }
    this.width = width;
    this.height = height;
    this.speedX = 0;
    this.speedY = 0;
    this.x = x;
    this.y = y;
    this.update = function () {
        ctx = gameArea.context;
        if (type == "image" || type == "background") {
            ctx.drawImage(this.image, this.x, this.y, this.width, this.height);
           if(type == "background")
            ctx.drawImage(this.image, this.x + this.width, this.y, this.width, this.height);
        } else {
            ctx.fillStyle = color;
            ctx.fillRect(this.x, this.y, this.width, this.height);
        }
    }
    this.newPos = function () {
        //logic for canvas border
        if (this.x + this.width + this.speedX > gameArea.canvas.width || this.x + this.speedX < 0) {
            this.speedX = 0;
        }
        if (this.y + this.height + this.speedY > gameArea.canvas.height || this.y + this.speedY < 0) {
            this.speedY = 0;
        }

        this.x += this.speedX;
        this.y += this.speedY;

        if (this.type == "background") {
            this.y += this.speedY;
            if (this.y >= this.height) {
              this.y = 0;
            }
          }
    }
```



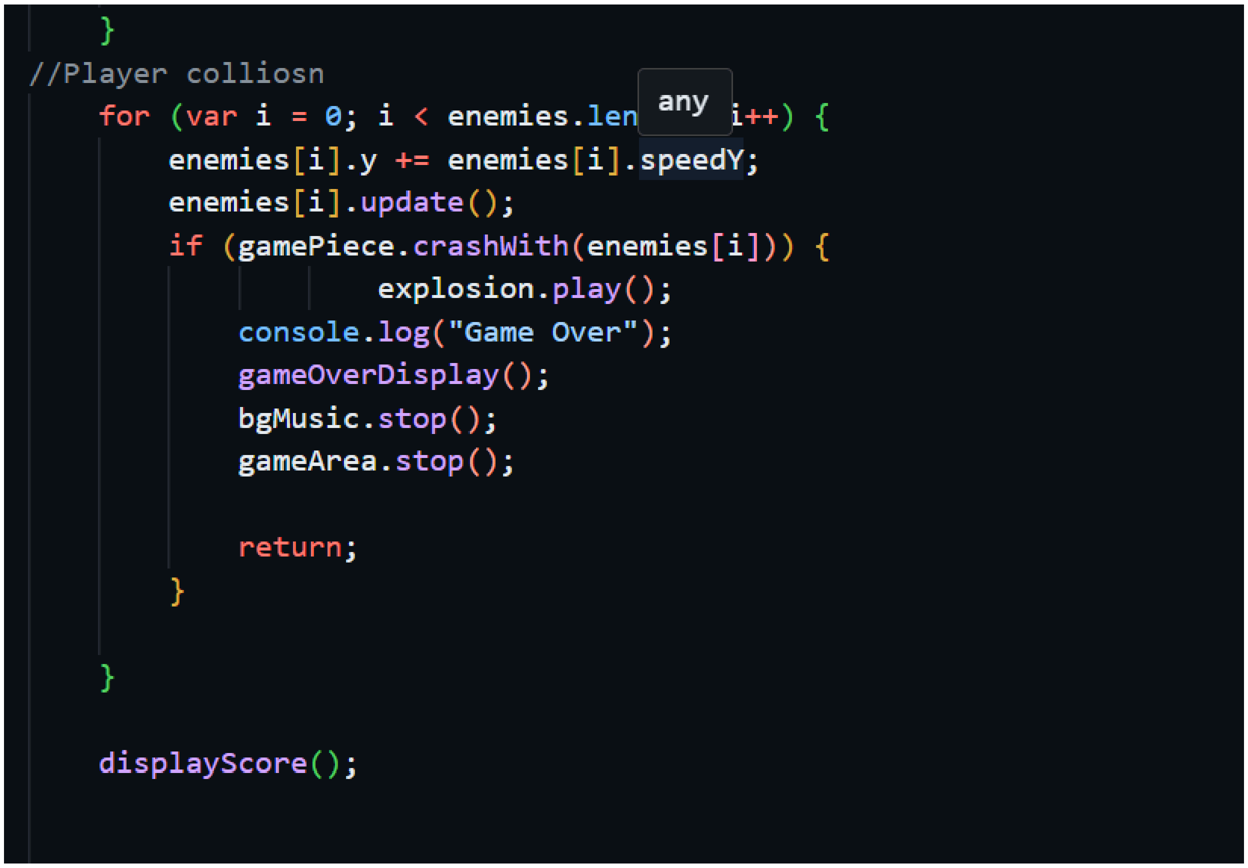
```
function spawnEnemy() {
    var randomXPos = Math.random() * (gameArea.canvas.width - 30);
    var enemy = new component(30, 30, "enemy.png", randomXPos, 0, "image");
    enemy.speedY = enemySpeed;
    enemies.push(enemy);

}


function shoot() {
    var bullet = new component(5, 10, "yellow", gamePiece.x + gamePiece.width / 2 - 2.5, gamePiece.y);
    bullet.speedY = -5;
    bullets.push(bullet);
    console.log("fired");
}

function handleShot() {
    let currentTime = new Date().getTime();

    if (currentTime - lastFireTime >= fireRate) {
        shoot();
        soundEffect.play();
        lastFireTime = currentTime;
    }
}
//Display enemies killed
function displayScore() {
    let ctx = gameArea.context;
    ctx.font = "20px Arial";
    ctx.fillStyle = "white"
    ctx.fillText("Enemies Killed: " + enemiesKilled, 5, 20);

}
```



```
//when player dies display game over screen
function gameOverDisplay() {
    let over = gameArea.context;
    let score = gameArea.context;
    score.font = "20px Arial";
    score.fillStyle = "white";
    score.fillText("Enemies Killed: " + enemiesKilled, 180, 300);
    over.font = "90px Arial";
    over.fillStyle = "white";
    over.fillText("Game over!", 20, 220);
    document.getElementById('gameOver').style.display = 'block';
    gameoverSound.play();
}
```



```
function updateGameArea() {

    gameArea.clear();
    background.speedY = -3;
    background.newPos();
    background.update();
    gamePiece.newPos();
    gamePiece.update();


    gamePiece.speedX = 0;
    gamePiece.speedY = 0;


    if (gameArea.key && gameArea.key == "ArrowLeft") {
        gamePiece.speedX = -5;
    }
    if (gameArea.key && gameArea.key == "ArrowRight") {
        gamePiece.speedX = 5;
    }
    if (gameArea.key && gameArea.key == "ArrowUp") {
        gamePiece.speedY = -5;
    }
    if (gameArea.key && gameArea.key == "ArrowDown") {
        gamePiece.speedY = 5;
    }
    if (gameArea.key && gameArea.key == " ") {

        handleShot();
    }


    //Bullets
    for (var i = 0; i < bullets.length; i++) {
        bullets[i].y += bullets[i].speedY;
        bullets[i].update();
        //Bullet collision
        for (var j = 0; j < enemies.length; j++) {
            if (bullets[i].crashWith(enemies[j])) {
                enemies.splice(j, 1);
                bullets.splice(i, 1);
```



```
    }
//Player colliosn
    for (var i = 0; i < enemies.len any i++) {
        enemies[i].y += enemies[i].speedY;
        enemies[i].update();
        if (gamePiece.crashWith(enemies[i])) {
                    explosion.play();
            console.log("Game Over");
            gameOverDisplay();
            bgMusic.stop();
            gameArea.stop();

            return;
        }

    }

    displayScore();
```