RAMAIAH
Institute of Technology

DEPARTMENT OF

ISE

Project Report

on

HYBRID WEB DEVELOPMENT

CFC SHOOTER GAME

STUDENT NAME : SARANG R

USN : 1MS22CY063

Ramaiah Institute of Technology

(Autonomous Institute, Affiliated to VTU)

MSR Nagar, MSRIT Post, Bangalore-560054

October – November, 2023

INDEX

Problem Statement

Create an immersive and educational space shooter game with the goal of increasing awareness about the harmful effects of chlorofluorocarbons (CFCs) on the ozone layer. Players will navigate a spacecraft armed with anti-CFC weapons in a dynamic interstellar setting. The main mission is to intercept and eliminate CFC particles before they penetrate the ozone layer, providing players with a simulated experience of the genuine environmental threat posed by these compounds to Earth's protective atmosphere.

Introduction

Welcome to the CFC Shooter, an innovative and educational space shooter game designed to shed light on the detrimental impact of chlorofluorocarbons (CFCs) on the ozone layer. In this engaging gaming experience, players assume control of a spacecraft armed with anti-CFC weaponry, embarking on a mission to safeguard the Earth's atmospheric shield.

PURPOSE:

The primary purpose of the CFC Shooter is to raise awareness about the environmental consequences of CFCs, synthetic compounds known for their role in ozone layer depletion. Through interactive gameplay, users are not only entertained but also educated about the urgency of addressing this global environmental challenge.

WORKING OF THE PROJECT:

Players navigate through an immersive interstellar environment, facing the task of intercepting and destroying CFC particles before they breach the ozone layer. The game employs realistic physics and mechanics to simulate the actual threat posed by these compounds, providing players with an authentic experience of the consequences of CFC emissions.

USAGE:

The CFC Shooter is designed for players of all ages who seek an entertaining yet educational gaming experience. Whether you're a casual gamer or an environmental enthusiast, the game offers an accessible platform to learn about CFCs and their impact on the ozone layer while enjoying an engaging space shooter adventure.

By blending entertainment with education, the CFC Shooter not only aims to captivate players but also instill a sense of environmental responsibility. As you embark on this thrilling journey through space, you'll contribute to a greater understanding of the importance of preserving our planet's protective atmospheric layer. Get ready to shoot down CFCs and become a champion in the fight for a sustainable and healthier Earth!

Implementation

ozone.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" type="png" href="o3.jpg">
    <title>CFC Shooter</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1 style="text-align:center;color: cyan"><u>CFC Shooter Game</u></h1>
    <img src="bgim.jpg" id="bgim">
    <canvas id="canvas1"></canvas>
    <div class="hidden">
        <img src="1101534968.png" id="planet" >
        <img src="image.png" id="ozone">
        <img src="player.png" id="player">
        <img src="asteroid.png" id="asteroid">
    </div>
    <script src="game.js"></script>
</body>
</html>

```

style.css

```css
html {

    background-repeat: no-repeat;
    background-position: center center;
}

body{
    background: url('bgim.jpg');
    background-size: cover;
}

#bgim {
    margin: 0;
    padding: 0;
    display: flex;
    position: relative;
}



#canvas1{
    background-image: url("Hubble_ultra_deep_field.jpg");
    position: absolute;
    top: 10%;
    left: 27.5%;
}
.hidden{
    display: none;
}
```

game.js

```
class Planet{
    constructor(game){
        this.game=game;
        this.x=350;
        this.y=-200;
        this.radius=300;
        this.image=document.getElementById('planet')
    }
    draw(context){
        context.drawImage(this.image,this.x-500,this.y-328);
        context.beginPath();
        //drawing the circle
        context.arc(this.x,this.y,this.radius,0,Math.PI);
        context.stroke();


    }
}
class Ozone{
    constructor(game){
        this.game=game;
        this.x=350;
        this.y=600;
        this.radius=91;
        this.image=document.getElementById('ozone')
    }
    draw(context){
        context.drawImage(this.image,this.x-180,this.y-105);
        context.beginPath();
        context.arc(this.x,this.y,this.radius,0,Math.PI,2*Math.PI);
        context.stroke();


    }
}
class Player{
    constructor(game){
        this.game=game;
        this.x=259;
        this.y=510;
        this.radius=40;
        this.angle=Math.PI*0.5;
        this.image=document.getElementById('player');
        this.aim;
    }


    draw(context) {
        context.save();
        context.translate(this.x, this.y);
        context.rotate(this.angle);
        context.drawImage(this.image, -this.radius, -this.radius, this.radius * 2, this.radius * 2);
        if(this.game.debug){
            context.beginPath();
            context.arc(0, 0, this.radius, 0, Math.PI * 2);
            context.stroke();
        }

        context.restore();
    }
    update(){
        this.aim=this.game.calcAim(this.game.mouse,this.game.ozone);
        const centerX = this.game.ozone.x;
        const centerY = this.game.ozone.y;
        const orbitRadius = 135;
        const cursorAngle = Math.atan2(this.game.mouse.y - centerY, this.game.mouse.x - centerX);
        this.x = centerX + orbitRadius * Math.cos(cursorAngle);
        this.y = centerY + orbitRadius * Math.sin(cursorAngle);
        this.x = Math.max(this.radius, Math.min(this.x, this.game.width - this.radius));
        this.y = Math.max(this.radius, Math.min(this.y, this.game.height - this.radius));
    }
    shoot(){
        const projectile=this.game.getProjectile();
        if(projectile)projectile.start(this.x+this.radius*this.aim[0],this.y+this.radius*this.aim[1],this.aim[0],this.aim[1]);
    }

}
```

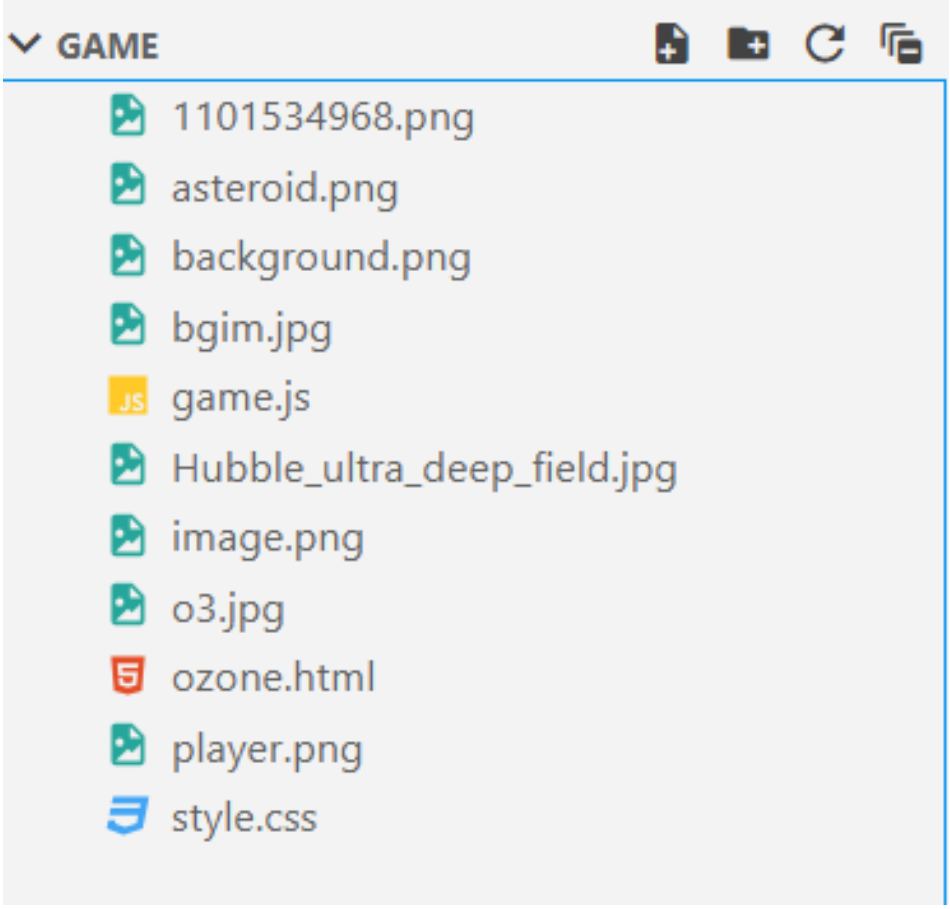
```
class Projectile {
    constructor(game) {
        this.game = game;
        this.x;
        this.y;
        this.radius = 5;
        this.speedX = 1;
        this.speedY = 1;
        this.speedModifier=5;
        this.free = true;
    }

    start(x, y,aimX,aimY) {
        this.free = false;
        this.x = x;
        this.y = y;
        this.speedX=aimX*this.speedModifier;
        this.speedY=aimY*this.speedModifier;
    }

    reset() {
        this.free = true;
    }

    draw(context) {
        context.save();
        context.beginPath();
        context.arc(this.x, this.y, this.radius, 0, Math.PI * 2);
        context.fillStyle = 'gold';
        context.fill();
        context.restore();
    }

    update(context) {
        if (!this.free) {
            this.x += this.speedX;
            this.y += this.speedY;
            this.draw(context);
        }
        if(this.x<0 | this.x>this.game.width || this.y<0 || this.y>this.game.height){
                this.reset();
        }
    }
}

class Enemy {
    constructor(game) {
        this.game = game;
        this.x = 0;
        this.y = 0;
        this.radius = 40;
        this.width = this.radius * 2;
        this.height = this.radius * 2;
        this.speedX = 0;
        this.speedY = 0;
        this.collided=false;
        this.free = true;
    }

```

```
start() {
        this.free = false;
        this.collided=false;
        this.frameX=0;
        this.lives=this.maxLives;
        this.frameY=Math.floor(Math.random()*4);
        if(Math.random()<0.5){
            this.x = Math.random() * this.game.width;
            this.y=0;
        }
        else{
            this.x=Math.random()<0.5?-this.radius:this.game.width+this.radius;
            this.y = Math.random() * this.game.height;
        }


        const aim = this.game.calcAim(this, this.game.ozone);
        this.speedX = aim[1];
        this.speedY = aim[0];
    }

    reset() {
        this.free = true;
    }
    hit(damage){
        this.lives-=damage;
    }

    draw(context) {
        if (!this.free) {
            context.drawImage(this.image,this.frameX*this.width,this.frameY*this.height,this.width,this.height,this.x-this.radius,this.y-this.radius,this.width,this.height);

            if(this.game.debug){
                context.beginPath();
                context.arc(this.x, this.y, this.radius, 0, Math.PI * 2);
                context.stroke();

            }
        }
    }

    update() {
        if (!this.free) {
            this.x += this.speedX;
            this.y += this.speedY;
            if (this.game.checkCollision(this, this.game.ozone)&& this.lives>=1) {
                this.lives=0;
                this.speedX=0;
                this.speedY=0;
                this.collided=false;
                this.gamee.lives--;
            }
            if (this.game.checkCollision(this, this.game.player)&& this.lives>=1) {
                this.lives=0;
                this.collided=true;
                this.game.lives--;
            }
            this.game.projectilePool.forEach(projectile=>{
                if(!projectile.free && this.game.checkCollision(this,projectile)){
                    projectile.reset();
                    this.hit(1);
                }

            });
            if(this.lives<1) this.frameX++;
            if(this.frameX>this.maxFrame){
                this.reset();
                if(!this.collided)this.game.score+=this.maxLives;
            }
        }
    }
}
```

```
class Asteroid extends Enemy{
    constructor(game){
        super(game);
        this.image=document.getElementById('asteroid');
        this.frameX=0;
        this.frameY=Math.floor(Math.random()*4);
        this.maxFrame=7;
        this.lives=3;
        this.maxLives=this.lives;
    }
}


//game class is sued to control everything in the game
class Game{
    constructor(canvas){
        this.canvas=canvas;
        //width and height of main game object should be same as the canvas
        this.width=this.canvas.width;
        this.height=this.canvas.height;
        this.planet=new Planet(this);
        this.ozone=new Ozone(this);
        this.player = new Player(this);
        this.debug=false;
        this.projectilePool=[];
        this.numberOfProjectiles=30;
        this.createProjectilePool();
        this.enemyPool=[];
        this.numberOfEnemies=200;
        this.createEnemyPool();
        this.enemyPool[0].start();
        this.enemyTimer=0;
        this.enemyInterval=900;
        this.score=0;
        this.winningScore=25;
        this.lives=2;

        this.mouse={
            x:0,
            y:0
        }
        window.addEventListener('mousemove',e=>{
            console.log(e);
            this.mouse.x=e.offsetX;
            this.mouse.y=e.offsetY;
        });
        window.addEventListener('mousedown',e=>{
            this.mouse.x=e.offsetX;
            this.mouse.y=e.offsetY;
            this.player.shoot();
        })
        window.addEventListener('keyup',e=>{
            if(e.key==='d') this.debug=!this.debug;
            else if(e.key==='w')this.player.shoot();

        });


    }
    render(context,deltaTime){

        this.planet.draw(context);
        this.drawStatusText(context);
        this.player.draw(context);
        this.ozone.draw(context);
        this.player.update();
        this.projectilePool.forEach(projectile=>{
            projectile.update(context);
        });
        this.enemyPool.forEach(enemy=>{
            enemy.update();
            if (!enemy.free) {
                enemy.draw(context);
            }

        });
        if(!this.gameOver){
            if(this.enemyTimer<this.enemyInterval){
                this.enemyTimer+=deltaTime;
            }else{
                this.enemyTimer=0;
                const enemy=this.getEnemy();
                if(enemy) enemy.start();
            }
            if(this.score>=this.winningScore || this.lives<1){
                this.gameOver=true;
            }
        }

    }
    drawStatusText(context){
        context.save();
        context.textAlign='left';
        context.font='30px Impact';
        context.fillStyle = 'white';
        context.fillText('Score : '+this.score,10,510);
        for(let i=0;i<this.lives;i++){
            context.fillRect(20+15*i,550,10,30);
        }
        if(this.gameOver){
            context.textAlign='center';
            let message1;
            let message2;
            if(this.score>=this.winningScore){
                message1='You have succesfully saved the Ozone Layer';
                message2='Your score is '+this.score+'!'
            }else{
                message1='Ozone layer has depleted completely';
                message2='Be cautious! Try again'
            }
            context.font='37px Impact';
            context.fillStyle='cyan';
            context.fillText(message1,this.width*0.5,200);
            context.fillStyle='cyan';
            context.fillText(message2,this.width*0.5,300);
        }
    }
```

```
calcAim(a,b){
        const dx=a.x-b.x;
        const dy=a.y-b.y;
        const distance= Math.hypot(dx,dy);
        const aimX=dx/distance;
        const aimY=dy/distance;
        return [aimX,aimY,dx,dy];
    }
    checkCollision(a,b){
        const dx=a.x-b.x;
        const dy=a.y-b.y;
        const distance=Math.hypot(dx,dy);
        const sumOfRadii=a.radius+b.radius;
        return distance<sumOfRadii;
    }
    createProjectilePool(){
        for(let i=0;i<this.numberOfProjectiles;i++){
            this.projectilePool.push(new Projectile(this));

        }
    }
    getProjectile(){
        for(let i=0;i<this.projectilePool.length;i++){
            if(this.projectilePool[i].free) return this.projectilePool[i];
        }
    }
    createEnemyPool(){
        for(let i=0;i<this.numberOfEnemies;i++){
            this.enemyPool.push(new Asteroid(this));
        }
    }
    getEnemy(){
        for(let i=0;i<this.enemyPool.length;i++){
            if(this.enemyPool[i].free)return this.enemyPool[i];
        }
    }
}



window.addEventListener('load',function(){
    const canvas=this.document.getElementById("canvas1");
    const ctx=canvas.getContext('2d');
    canvas.width=700;
    canvas.height=600;
    ctx.strokeStyle='white';
    ctx.lineWidth=2;
    const game=new Game(canvas);
    let lastTime=0;
    function animate(timeStamp){
        const deltaTime=timeStamp-lastTime;
        lastTime=timeStamp;
        ctx.clearRect(0,0,canvas.width,canvas.height);
        game.render(ctx,deltaTime);
        requestAnimationFrame(animate);

    }
    requestAnimationFrame(animate);


});
```

Files


GAME
1101534968.png
asteroid.png
background.png
bgim.jpg
game.js
Hubble_ultra_deep_field.jpg
image.png
o3.jpg
ozone.html
player.png
style.css

Output

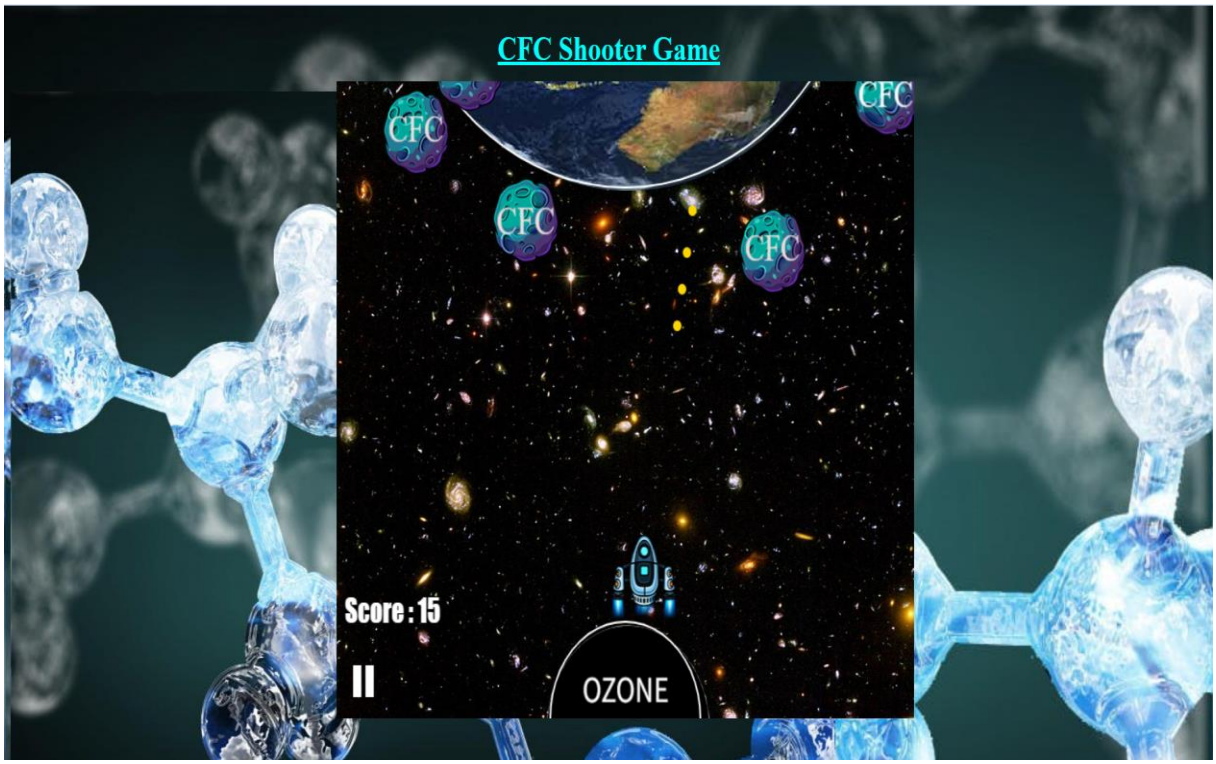


Figure 1: CFC shooter in action

Figure 2: bullets in action, destruction of a cfc carries 3 points and adds 3 points to the score. The two rectangular bars at the bottom depict the lives available.

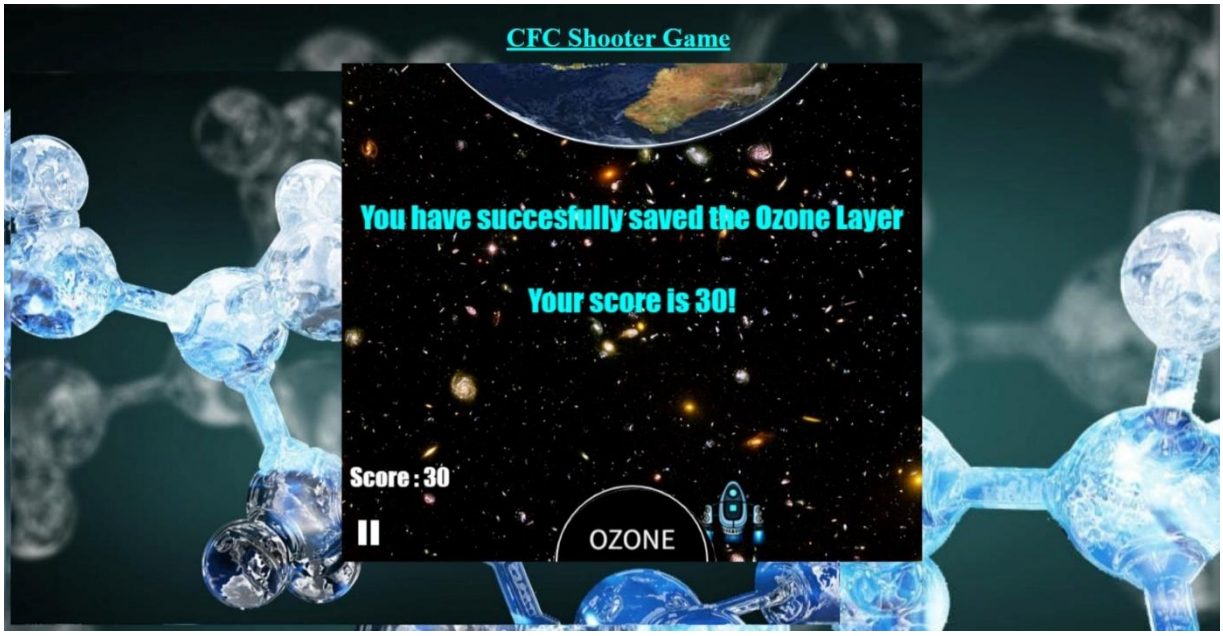


Figure 3: When the score reaches more than 25 then the player has successfully won the game and saved the ozone layer

Figure 4: When the cfc's come in contact with the then the player loses his life. Player has 2 lives and if both the lives are exhausted then the player loses the game. The two rectangular bars at the bottom depict the lives available.