Interactive Graphics – Final Project

Alessandro Appolloni - 1757950

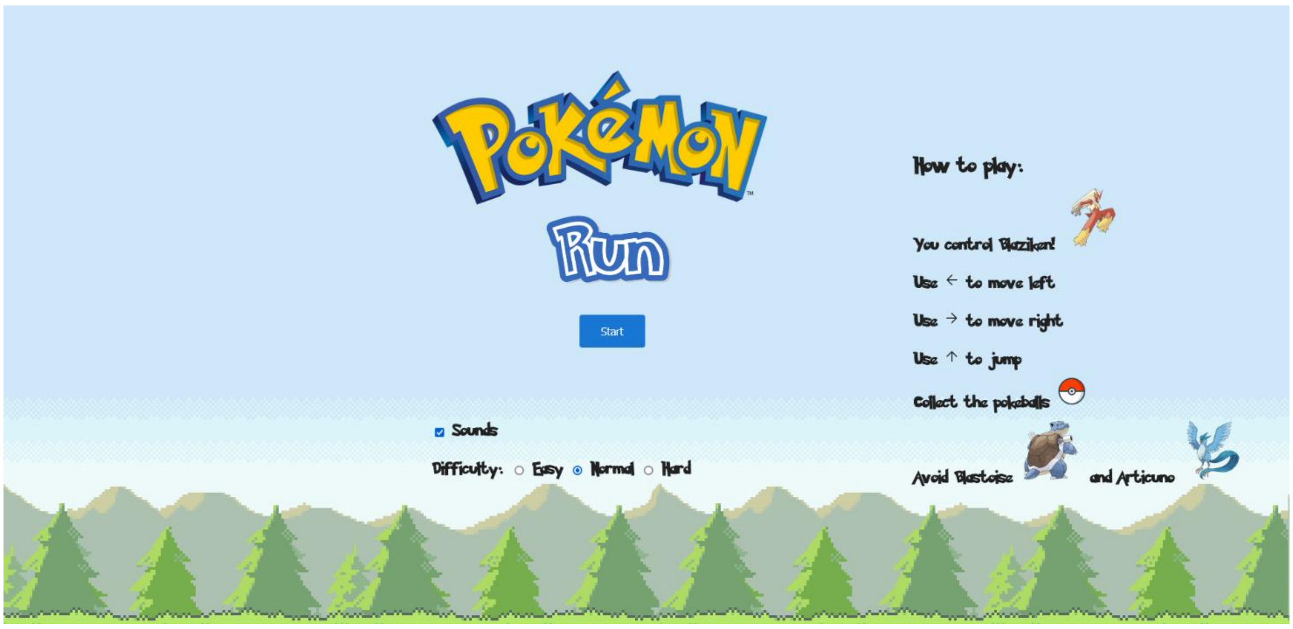
1. Introduction

Pokémon Run is an endless runner game inspired by the universe of Pokèmon. You control Blaziken and his goal is to collect as many Poké Balls as possible avoiding Blastoise and Articuno.

2. The Game – User Interaction

The first screen is the main menu, here the user can:

- enable/disable the sounds,
- choose the speed difficulty of the game,
- read the istructions on how to play the game and which commands can use.
- Start the game



[✓] Sounds

Difficulty: ○ Easy [✓ Normal] ○ Hard

How to play:

You control Blaziken!

Use ← to move left

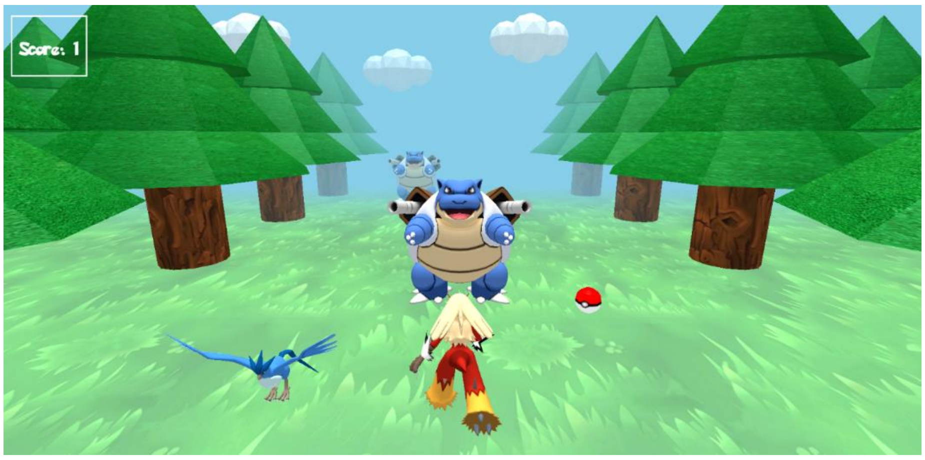
Use → to move right

Use ↑ to jump

Collect the pokeballs

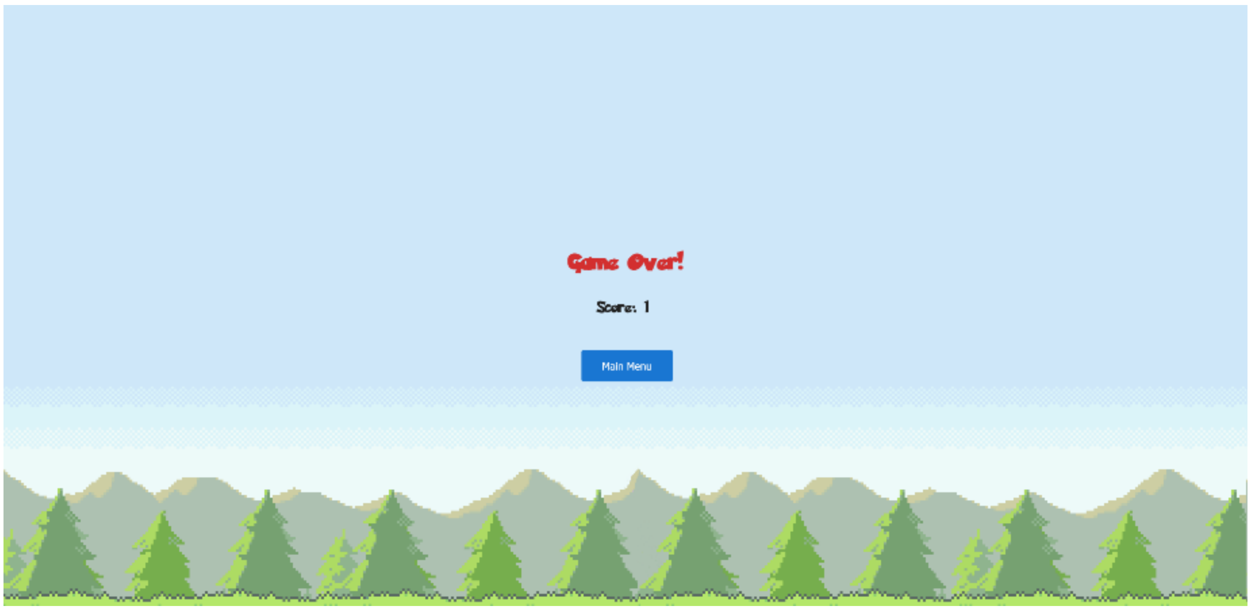
Avoid Blastoise and Articuno

When the user click on “Start” the game starts and the scene shows Blaziken running. The user controls Blaziken through the keyboard arrows, with the left arrow Blaziken moves to the left, with the right arrow moves to the right, with the up arrow jumps.



The Poké Balls and the other two Pokémons (Blastoise and Articuno) can spawn only in three specific lanes (invisible) and Blaziken can move only in these lanes. Outside of the three lanes there are the trees and in the background there are the clouds.

The score increases when Blaziken collects a Poké Ball, but watch out for Blastoise and Articuno, if the user hit them the game is over and a riepilogative screen with the score is showed.



Then the user can go back to the main menu.

3. The code

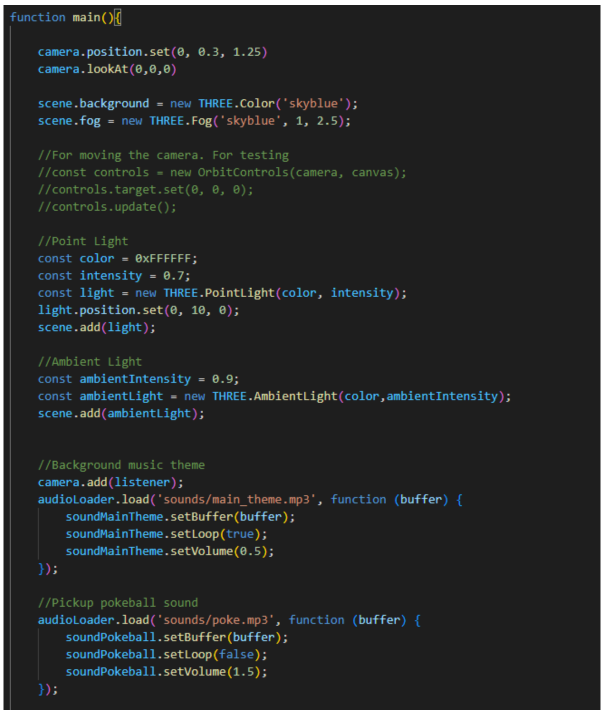
The game is written in javascript with three.js. The code is organized in three files:

- index.html
- main.js
- index.css

The following libraries has been used:

- Three.js
- GLTFLoader: used for importing and loading 3d models
- Tween.js: for smooth animations

In the main.js file the scene is initialized in the main() function. In the first part of the function are setted up the Camera, the color Background, the Fog, the lights and the sounds:

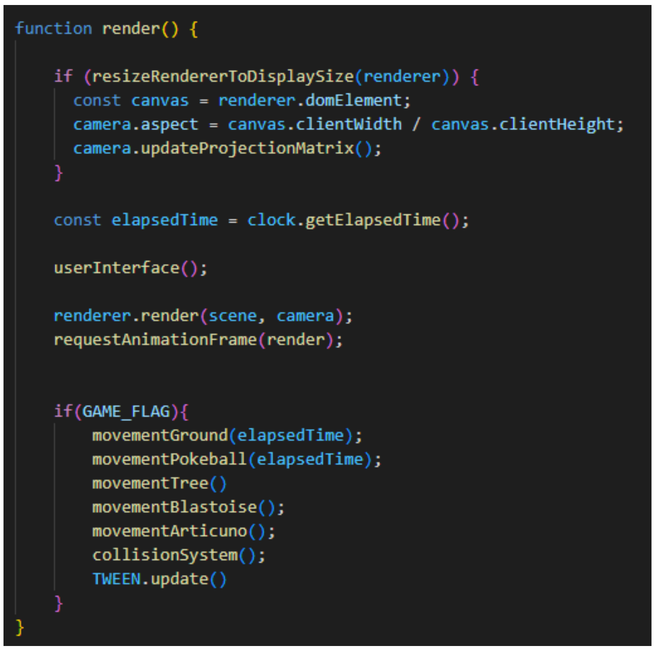


```
function main(){

    camera.position.set(0, 0.3, 1.25)
    camera.lookAt(0,0,0)

    scene.background = new THREE.Color('skyblue');
    scene.fog = new THREE.Fog('skyblue', 1, 2.5);

    //For moving the camera. For testing
    //const controls = new OrbitControls(camera, canvas);
    //controls.target.set(0, 0, 0);
    //controls.update();

    //Point Light
    const color = 0xFFFFFF;
    const intensity = 0.7;
    const light = new THREE.PointLight(color, intensity);
    light.position.set(0, 10, 0);
    scene.add(light);

    //Ambient Light
    const ambientIntensity = 0.9;
    const ambientLight = new THREE.AmbientLight(color,ambientIntensity);
    scene.add(ambientLight);


    //Background music theme
    camera.add(listener);
    audioLoader.load('sounds/main_theme.mp3', function (buffer) {
        soundMainTheme.setBuffer(buffer);
        soundMainTheme.setLoop(true);
        soundMainTheme.setVolume(0.5);
    });

    //Pickup pokeball sound
    audioLoader.load('sounds/poke.mp3', function (buffer) {
        soundPokeball.setBuffer(buffer);
        soundPokeball.setLoop(false);
        soundPokeball.setVolume(1.5);
    });
```

In the second part are setted up all the objects: ground, clouds, trees, pokèballs, Blaziken, Articuno, Blastoise. In the main() is salso called the onKeyDown function, which is responsible to capture the user inputs.

The functions blaziken(), articuno() and blastoise() are also responsible of the idle animations set-up of the respective pokèmons.

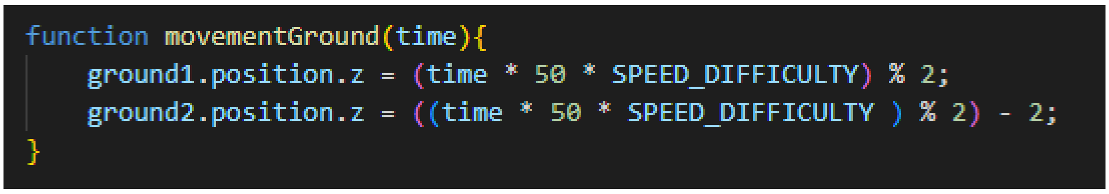
At the end is called the render() functions which is responsible of updating Tween.js (for the animations) and the positions of the objects.



```
function render() {

    if (resizeRendererToDisplaySize(renderer)) {
      const canvas = renderer.domElement;
      camera.aspect = canvas.clientWidth / canvas.clientHeight;
      camera.updateProjectionMatrix();
    }

    const elapsedTime = clock.getElapsedTime();

    userInterface();

    renderer.render(scene, camera);
    requestAnimationFrame(render);


    if(GAME_FLAG){
        movementGround(elapsedTime);
        movementPokeball(elapsedTime);
        movementTree()
        movementBlastoise();
        movementArticuno();
        collisionSystem();
        TWEEN.update()
    }
}
```

The way in which the object are translated in the scene creates the illusion of the endless in game. First of all Blaziken is stopped at the center of the scene and all other objects move towards him.

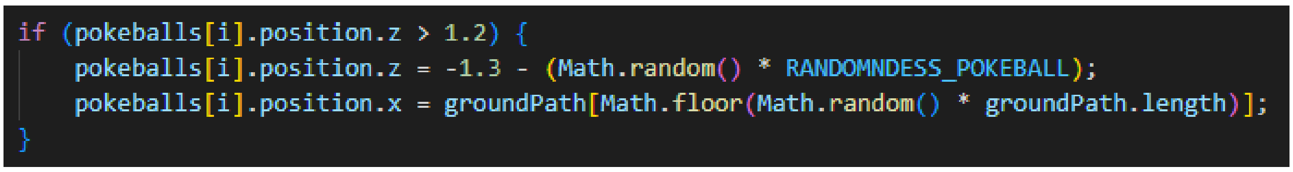
For example, the movement of the ground is managed by the movementGround() function.



```
function movementGround(time){
    ground1.position.z = (time * 50 * SPEED_DIFFICULTY) % 2;
    ground2.position.z = ((time * 50 * SPEED_DIFFICULTY ) % 2) - 2;
}
```

The ground is subdivided in two parts. The first one starts from position z=0, then through the variable *time* that is incremented automatically, moves towards z=2 and when it reaches this point the position is resetted in z=0 and so and so on. The second one works in the same way but it is translated and starts from z=-2

The movement of Trees, Pokéballs, Blastoise and Articuno works in a similar way: they start from a random position on the Z axis, move towards Blaziken and when they reaches the end of the scene the position is resetted to other side of the scene in a random way:



```
if (pokeballs[i].position.z > 1.2) {
    pokeballs[i].position.z = -1.3 - (Math.random() * RANDOMNDESS_POKEBALL);
    pokeballs[i].position.x = groundPath[Math.floor(Math.random() * groundPath.length)];
}
```

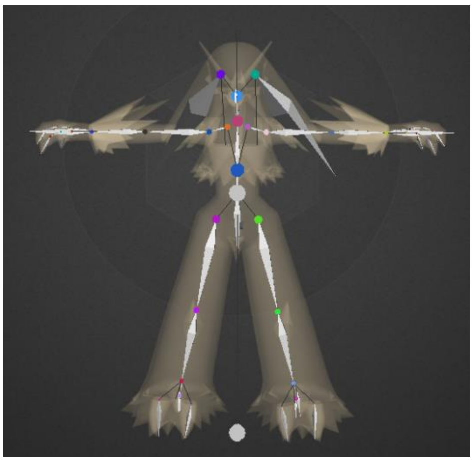
The fog located at the horizon increases the illusion of inifinity and hides the random spawn of objects

In render() there is also collisionSystem() function which is responsible to check if Blaziken collects a Pokéball (if yes, the position of the Poké Ball is moved to the other side of the scene) or hit one of the two other Pokémons.

4. Models

4.1 Blaziken

This is the Pokemon that the user can control. The model has a hierarchical structure. It has been taken from Sketchfab.



4.2 Blastoise

One of the two Pokémon obstacle. This model has a hierarchical structure and it has been taken from Sketchfab.

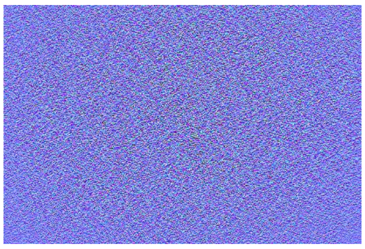
4.3 Articuno

The other Pokémon obtacle. This model has a hierarchical structure and it has been taken from Sketchfab.

4.4 Tree

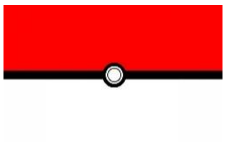
Trees are defined in the Tree class. They are composed of a Trunk (a cylinder geometry) and a Crown, divided in three parts (cone geometries). The trunk has a basic image texture while the cones have a image texture + a normal map texture, generated with NormalMap Online.

Trunk texture: Cone texture:



4.4 Poké Ball

The Poké Ball, defined in its own class, is a sphere geometry and has a texture applied:



4.5 Cloud

Clouds, defined in its own class are composed of three Octahedron geometries. One at the center, one on the left side and one on the rigt side. All of them with a white texture attached.

White texture:

4.6 Ground

The ground, defined in its own class, is a simple Plane geometry with a texture attached that looks like grass.

5. Animations

Animations of the three Pokémons has been made with Tween.js.

5.1 Blaziken

Blaziken perform a running animation, defined in animationBlaziken() function. It moves the Body, Legs, Arms and Tail simulating a person who runs.

The animations performed when it moves on the left/right or jumps are defined in the onKeyDown() function. When it jumps, it moves also the hair.

5.2 Blastoise

Blastoise performs an idle animation moving arms and the two guns attached on the shoulders. Animations are defined in animationBlastoise().

5.3 Articuno

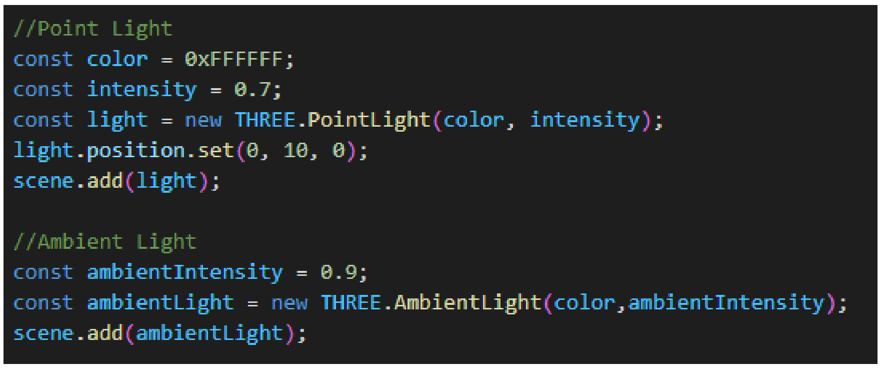
Articuno performs an idle animation moving the body up and down, together with the two wings and the two legs. It moves also the tail. Animations are defined in animationArticuno().

5.4 Poké Ball

The Poké Balls have a simple rotation animation.

6. Lights

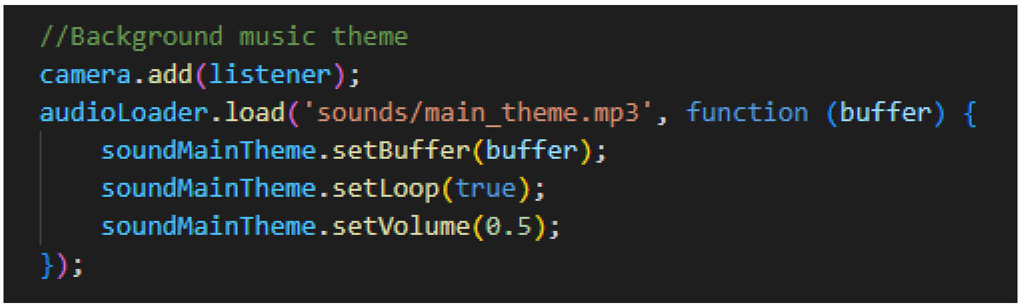
The scene has two different lights, one is an ambient light, the other is a point light. Both of them are needed in order to have a more realistic illumination of the scene.



```
//Point Light
const color = 0xFFFFFF;
const intensity = 0.7;
const light = new THREE.PointLight(color, intensity);
light.position.set(0, 10, 0);
scene.add(light);

//Ambient Light
const ambientIntensity = 0.9;
const ambientLight = new THREE.AmbientLight(color,ambientIntensity);
scene.add(ambientLight);
```

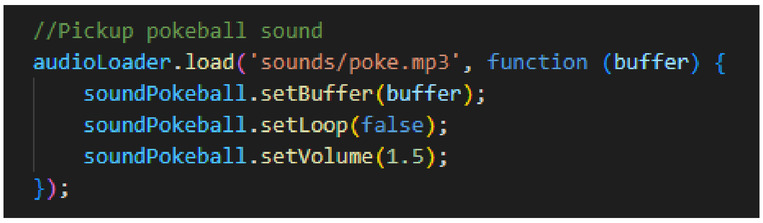
7. Sounds

The music used in the game is the Opening Theme of the official Videogames Pokémon Red and Pokémon Blue.



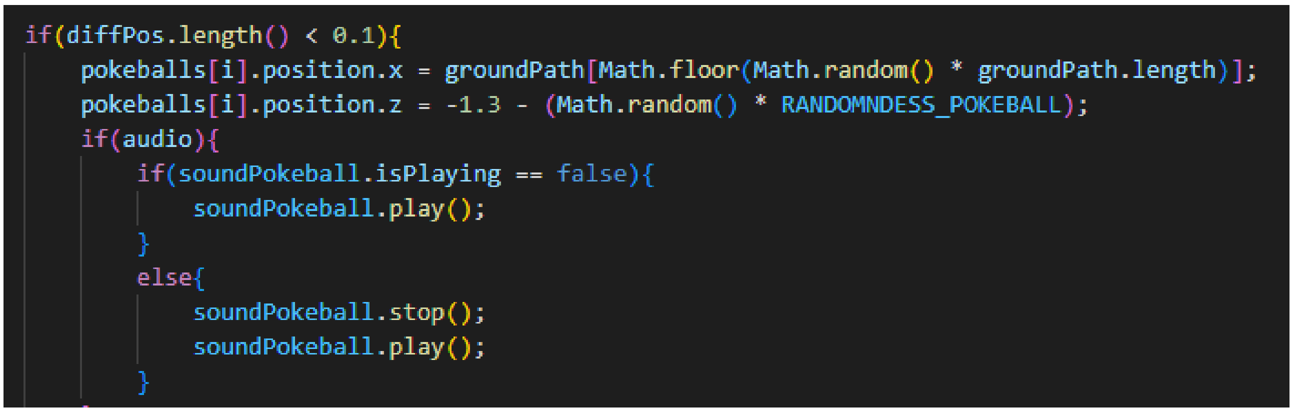
```
//Background music theme
camera.add(listener);
audioLoader.load('sounds/main_theme.mp3', function (buffer) {
    soundMainTheme.setBuffer(buffer);
    soundMainTheme.setLoop(true);
    soundMainTheme.setVolume(0.5);
});
```

It is also used a Sound when Blaziken collects a Poké Ball.



```
//Pickup pokeball sound
audioLoader.load('sounds/poke.mp3', function (buffer) {
    soundPokeball.setBuffer(buffer);
    soundPokeball.setLoop(false);
    soundPokeball.setVolume(1.5);
});
```

When collisionSystem() detects the collision between the character and the ball, the sound is played.



```
if(diffPos.length() < 0.1){
    pokeballs[i].position.x = groundPath[Math.floor(Math.random() * groundPath.length)];
    pokeballs[i].position.z = -1.3 - (Math.random() * RANDOMNDESS_POKEBALL);
    if(audio){
        if(soundPokeball.isPlaying == false){
            soundPokeball.play();
        }
        else{
            soundPokeball.stop();
            soundPokeball.play();
        }
```

8. Browser testing

The game has been tested on:

- Mozilla Firefox 105.0
- Microsoft Edge 105.0.1343.42
- Google Chrome 105.0.5195.127