VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI

Mini Project Report on

"INTERACTIVE HOVERBOARD GAME"

Submitted in the partial fulfillment for the requirements of Computer Graphics & Visualization Laboratory of 6th semester CSE requirement in the form of the Mini Project work

Submitted By

POORVI ROHIDEKAR	USN: 1BY17CS113
PRANOVE A B	USN:1BY17CS118

Under the guidance of

Mr. SHANKAR R
Assistant Professor, CSE, BMSIT&M

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
YELAHANKA, BENGALURU - 560064.

2019-2020

BMS INSTITUTE OF TECHNOLOGY &MANAGEMENT

YELAHANKA, BENGALURU – 560064

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project work entitled **"INTERACTIVE HOVERBOARD GAME"** is a bonafide work carried out by **Poorvi Rohidekar (1BY17CS113) and Pranove A B (1BY17CS118)** in partial fulfillment for *Mini Project* during the year 2019-2020. It is hereby certified that this project covers the concepts of *Computer Graphics & Visualization*. It is also certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report.

Signature of the Guide with date
Mr. SHANKAR R
Assistant Professor
CSE, BMSIT&M

Signature of HOD with date
Dr. Anil G N
Prof & Head
CSE, BMSIT&M

INSTITUTE VISION

To emerge as one of the finest technical institutions of higher learning, to develop engineering professionals who are technically competent, ethical and environment friendly for betterment of the society.

INSTITUTE MISSION

Accomplish stimulating learning environment through high quality academic instruction, innovation and industry-institute interface.

DEPARTMENT VISION

To develop technical professionals acquainted with recent trends and technologies of computer science to serve as valuable resource for the nation/society.

DEPARTMENT MISSION

Facilitating and exposing the students to various learning opportunities through dedicated academic teaching, guidance and monitoring.

PROGRAM EDUCATIONAL OBJECTIVES

1. Lead a successful career by designing, analysing and solving various problems in the field of Computer Science & Engineering.
2. Pursue higher studies for enduring edification.
3. Exhibit professional and team building attitude along with effective communication.
4. Identify and provide solutions for sustainable environmental development.

ACKNOWLEDGEMENT

We are happy to present this project after completing it successfully. This project would not have been possible without the guidance, assistance and suggestions of many individuals. We would like to express our deep sense of gratitude and indebtedness to each and every one who has helped us make this project a success.

We heartily thank our Principal, Dr. MOHAN BABU G N, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We heartily thank our Professor and Head of the Department, Dr. ANIL G N, Department of Computer Science and Engineering, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We gracefully thank our Project Guide, Mr. Shankar R, Assistant Professor, Department of Computer Science and Engineering for his intangible support and for being constant backbone for our project.

Special thanks to all the staff members of Computer Science Department for their help and kind co-operation.

Lastly, we thank our parents and friends for the support and encouragement given throughout in completing this precious work successfully.

POORVI ROHIDEKAR (1BY17CS113)

PRANOVE A.B (1BY17CS118)

ABSTRACT

Hoverboard Games Simulator. A self-balancing scooter, Segway or self-balancing two-wheeled board, commonly referred to as a "H**overboard**", is a type of portable, rechargeable battery-powered scooter.

This version of the experiment works by having the main webpage open on your computer and another webpage open on your mobile phone to get the accelerometer data.

Using the DeviceOrientation Web API, the orientation of the phone is detected and sent to the browser on your laptop via WebSockets.

By placing your phone on a skateboard, it can become a controller. You can use the orientation data to apply it to the 3D environment and interact with the game as if you were on a hoverboard.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 BRIEF INTRODUCTION

Our goal in this project is to make an interactive hoverboard game using webGL and JavaScript so as to provide the users with real life experience. Mainly focusing on getting the accelerometer data through our phone and webpage by keeping a website open on each of the respective devices. With the help of device orientation web API the orientation of the mobile can be configured. The skateboard can controlled via mobile. Orientation data that we get can be applied to a 3D environment. The first step is to set up the node js server and websockets. Next we have to get and stream the orientation data following which we have to listen to the orientation data in the game. Loading all the required libraries for making a 3D environment we develop a 3D scene. Next we generate the terrain and load the 3D models. we have our scene set up, we can use the orientation data to update the position of the hoverboard model in our scene.

1.2 MOTIVATION

The main motivation towards building this project is the existing model of a daydream controller. The ultimate motive is to make it accessible to people. The simple fact that one can build their own controller is an exciting opportunity into building this project. It is the primary idea of undertaking a project which is interesting and the opportunity to build and learn such varied and diverse projects in the field of graphics. Since using a web Bluetooth to connect the daydream controller directly to the browser didn't allow us to connect our phones to our laptop using the web Bluetooth API. Both phones and laptops can only be central devices and not peripheral so we have developed this using web sockets for the communication to take place between the phone and the laptop.

1.3 SCOPE

The main scope of this interactive hoverboard game is to give a real life experience to people of all ages by implementing it virtually. It is to make sure that with one click people shall be able to play this game anywhere anytime using their smartphones and laptops.

1.4 PROBLEM STATEMENT

Our problem statement in this project is to make an interactive hoverboard game using webGL and JavaScript so as to provide the users with real life experience. Primarily focusing on getting the accelerometer data through our phone and webpage by keeping a website open on each of the respective devices.

1.5 PROPOSED SYSTEM

The proposed methodology in this project is to overcome the barriers of the system that already exists by developing an interactive hoverboard game. The main purpose is make sure people get to experience the real life hoverboard virtually. This reduces the risk of explosion, usability issues, and neither is it expensive because everything is being computerized

1.6 LIMITATIONS

- Usability restrictions
- Some hoverboards are very expensive
- Injuries and risk of explosion

2. LITERATURE SURVEY

The number of mobile users engaged in interactive games is certainly going to grow in the near future. surveying points to open research issues and captures the state of the art in the field of interactive games over networks i.e phone and laptop being connected to the same network. The mobile revolution has brought us the possibility to enjoy our favorite applications anywhere and anytime. In this context, interactive games over networks embody a fascinating case study both for their commercial success and for their technical challenges, thus, sparking interest and development. From a research point of view, network support to interactive games must overcome several challenges, especially when considering many mobile players simultaneously sharing the same virtual arena. Interactivity (or responsiveness) refers to the delay between the generation of a game event in a node and the time at which other nodes become aware of that event. To ensure an enjoyable game to the final user, the time elapsed from the game event generation at a certain node and its processing time at every other node participating in the same game session must be kept under a certain interactivity threshold. Indeed, interactive games are gaining attention as their users are increasing in number. Mobile users are obviously influenced by this success, and it is becoming more and more common.

3. SYSTEM REQUIREMENTS

3.1 FUNCTIONAL REQUIREMENTS

Material needed:

- A modern mobile phone (any phone that has an accelerometer built-in).
- A computer.
- A projector *(optional)*.

Tech stack:

- HTML
- CSS
- JavaScript (Vanilla JS, no framework knowledge needed!)
- DeviceOrientation Web API
- Node.js with Socket.io

4. SYSTEM ANALYSIS

41 Setting up the Node.js server and web sockets:

To get started, we need to set up a server to serve our files, expose routes and set up socket.io. We require some modules, start an Express server and indicate the path and routes to serve our files. We also set up a basic socket.io connection that will wait for the communication between the server and front-end to be established.

4.2 Getting and streaming orientation data:

Using the DeviceOrientation Web API, we're able to get orientation data from the mobile phone. We instantiate socket.io and when the page loads, we're sending a first message to the server to indicate that the mobile page has been visited.

4.3 Listen to the orientation data in the game:

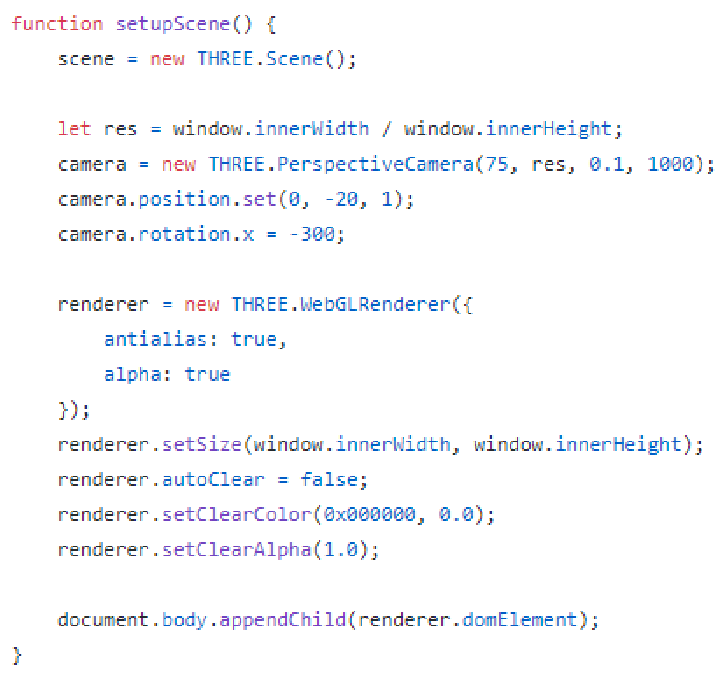
To be able to use the orientation data from the phone on the laptop, we need to be listening to the message sent by the server. Again, we instantiate socket.io, and when we receive the message mobile orientation from the server, we're able to use the tilt data in our game.

4.4 Building the game environment using Three.js and 3D models:

The 3D environment built in this project is using Three.js. To start using it as well as a few add-ons, we include them in our HTML. The main resources needed here are the Three.js library, the noiseSimplex script to generate the plane and the OBJ and MTL loaders to load the 3D assets used for the hoverboard and obstacles.
We initialised and set the position of the camera, the renderer, and we set up ambient light and spotlight in our environment.

5. SYSTEM INTERPRETATION

Implementing Scene :



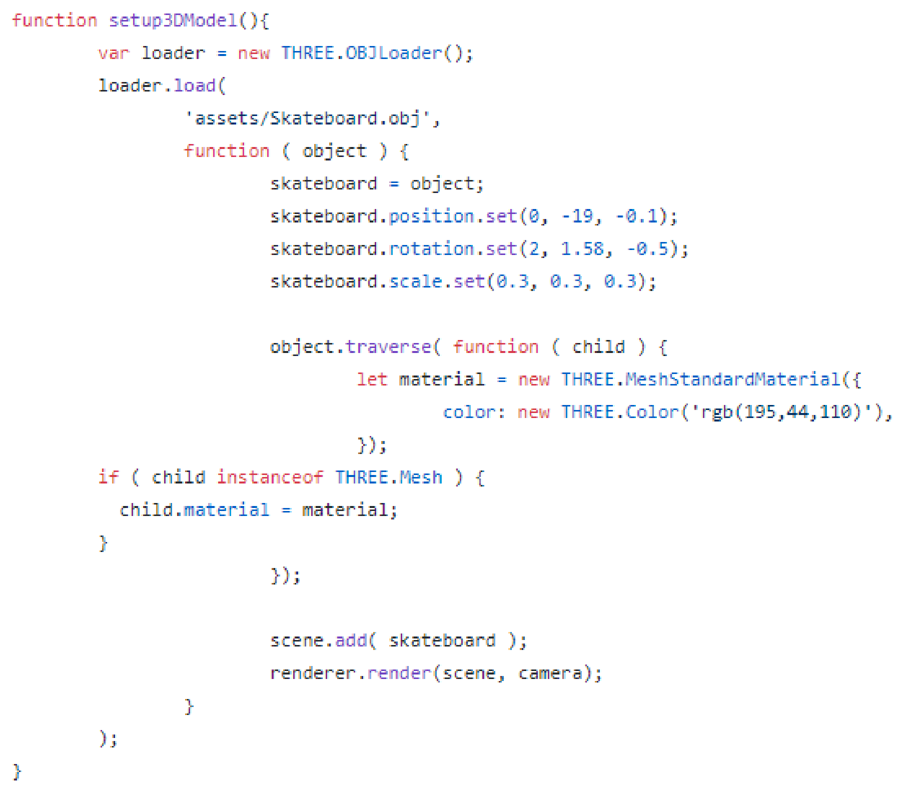
```
function setupScene() {
    scene = new THREE.Scene();

    let res = window.innerWidth / window.innerHeight;
    camera = new THREE.PerspectiveCamera(75, res, 0.1, 1000);
    camera.position.set(0, -20, 1);
    camera.rotation.x = -300;

    renderer = new THREE.WebGLRenderer({
        antialias: true,
        alpha: true
    });
    renderer.setSize(window.innerWidth, window.innerHeight);
    renderer.autoClear = false;
    renderer.setClearColor(0x000000, 0.0);
    renderer.setClearAlpha(1.0);

    document.body.appendChild(renderer.domElement);
}
```

Implementing Hoverboard :

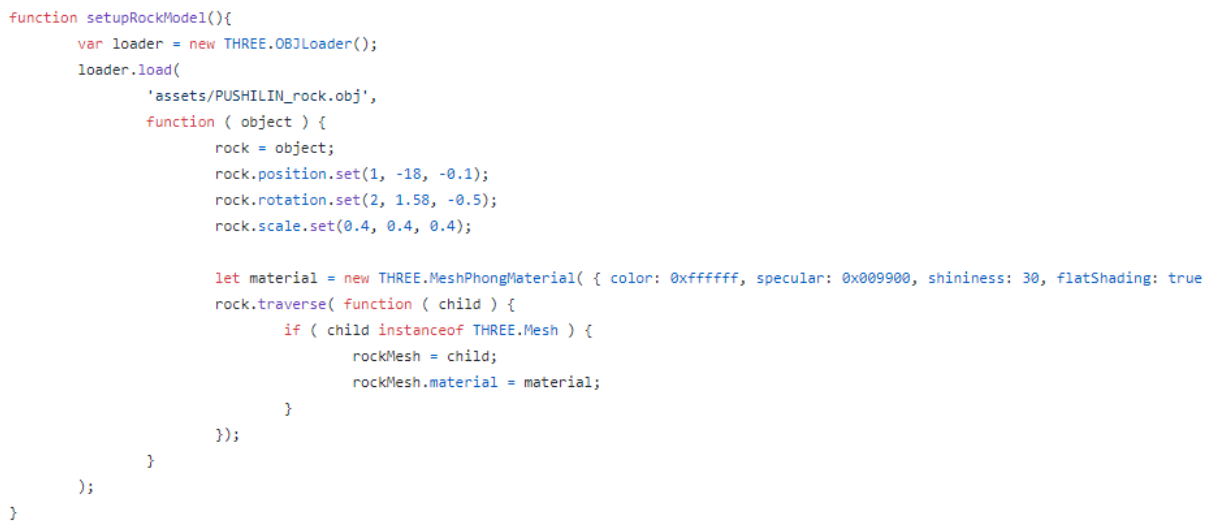


```
function setup3DModel(){
        var loader = new THREE.OBJLoader();
        loader.load(
                'assets/Skateboard.obj',
                function ( object ) {
                        skateboard = object;
                        skateboard.position.set(0, -19, -0.1);
                        skateboard.rotation.set(2, 1.58, -0.5);
                        skateboard.scale.set(0.3, 0.3, 0.3);

                        object.traverse( function ( child ) {
                                let material = new THREE.MeshStandardMaterial({
                                        color: new THREE.Color('rgb(195,44,110)'),
                                });
        if ( child instanceof THREE.Mesh ) {
          child.material = material;
        }
                        });

                        scene.add( skateboard );
                        renderer.render(scene, camera);
                }
        );
}
```

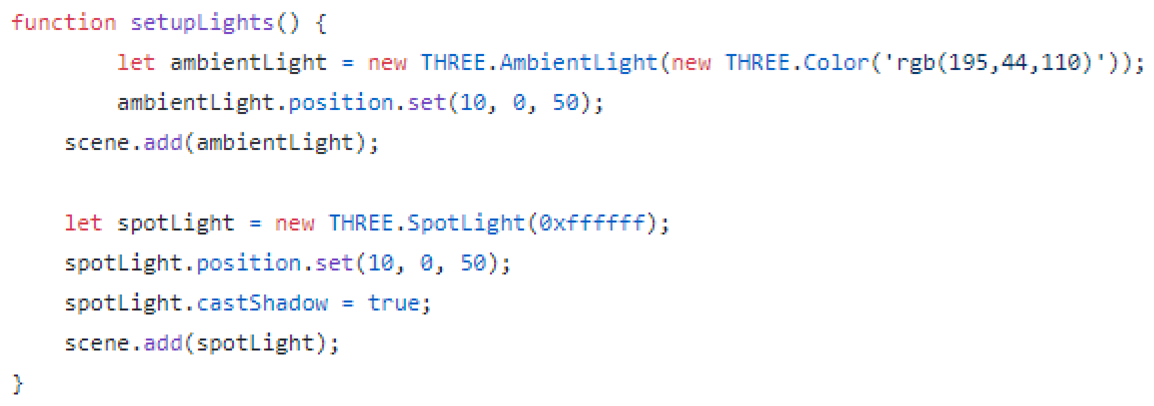
Implementing Rockmodel :



```
function setupRockModel(){
        var loader = new THREE.OBJLoader();
        loader.load(
                'assets/PUSHILIN_rock.obj',
                function ( object ) {
                        rock = object;
                        rock.position.set(1, -18, -0.1);
                        rock.rotation.set(2, 1.58, -0.5);
                        rock.scale.set(0.4, 0.4, 0.4);

                        let material = new THREE.MeshPhongMaterial( { color: 0xffffff, specular: 0x009900, shininess: 30, flatShading: true
                        rock.traverse( function ( child ) {
                                if ( child instanceof THREE.Mesh ) {
                                        rockMesh = child;
                                        rockMesh.material = material;
                                }
                        });
                }
        );
}
```

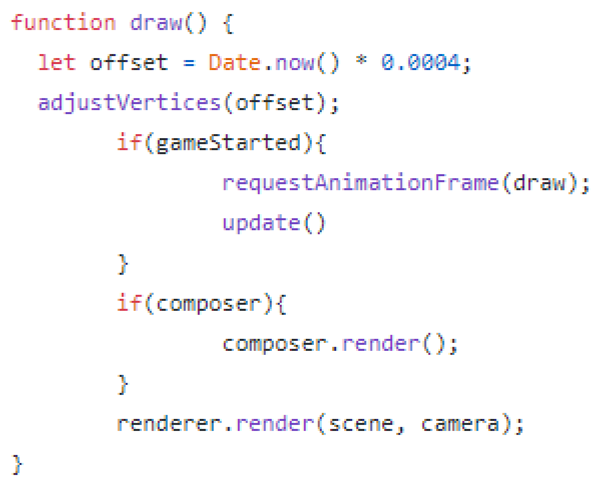
Implementing Lighting :



```
function setupLights() {
        let ambientLight = new THREE.AmbientLight(new THREE.Color('rgb(195,44,110)'));
        ambientLight.position.set(10, 0, 50);
    scene.add(ambientLight);

    let spotLight = new THREE.SpotLight(0xffffff);
    spotLight.position.set(10, 0, 50);
    spotLight.castShadow = true;
    scene.add(spotLight);
}
```

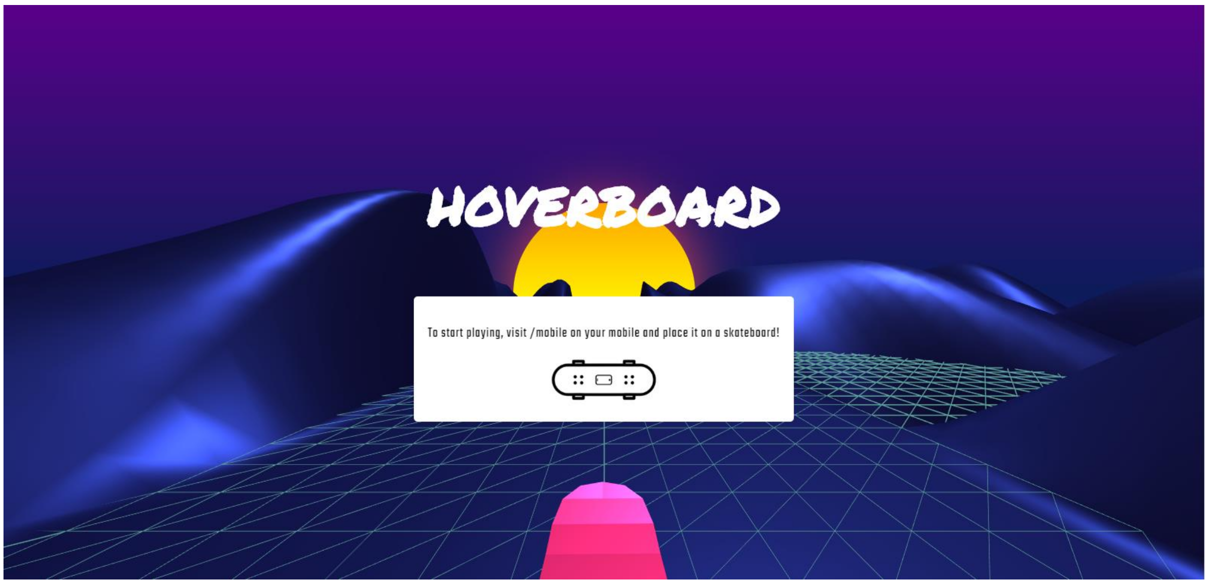
Final Rendering :



```
function draw() {
  let offset = Date.now() * 0.0004;
  adjustVertices(offset);
        if(gameStarted){
                requestAnimationFrame(draw);
                update()
        }
        if(composer){
                composer.render();
        }
        renderer.render(scene, camera);
}
```

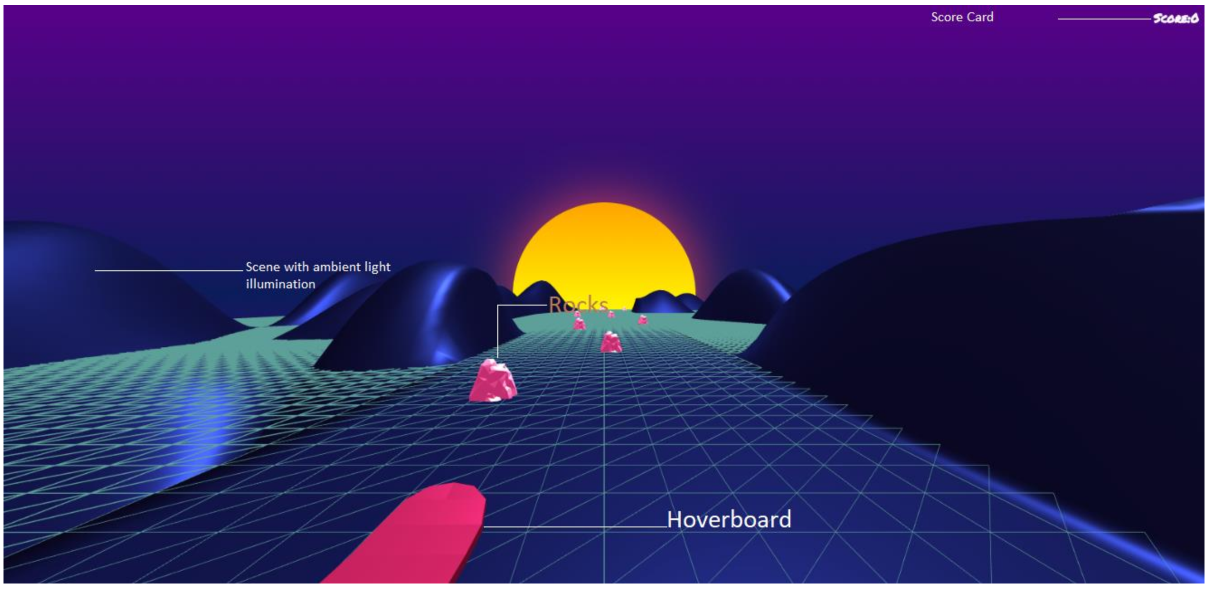
6. INTERPRETATION OF RESULTS

Desktop Intro Page :

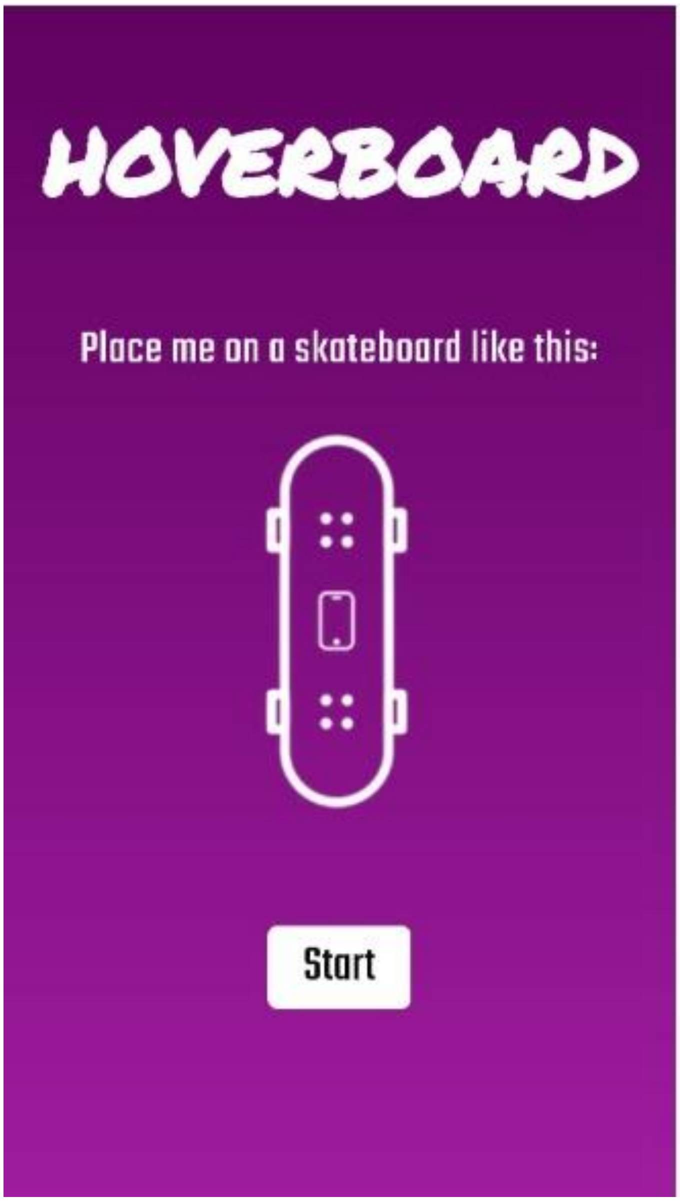


To start playing, visit /mobile on your mobile and place it on a skateboard!

Game in Progress Page :



Mobile Device Page:



CONCLUSION

We have successfully demonstrated a 3D environment to simulate a hoverboard game, using three.js to create a scene and ObjectLoader to implement a hoverboard, and the obstacle rocks. We have set up the camera position, spotlight and ambient light to the objects of the screen, and have made it interactive with the player's device, by streaming the Device orientation in real time to the scene.

BIBILIOGRAPHY

1. https://www.w3schools.com/js/
2. https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/Tutorial:Device Orientation web API
3. https://en.wikipedia.org/wiki/Hoverboard
4. three.js - JavaScript 3D Library : https://threejs.org/
5. Socket.IO - JavaScript library for realtime web applications : https://socket.io/
6. NodeJS : https://nodejs.org
7. https://threejs.org/docs/#examples/en/loaders/MTLLoader