**Contents**

[NOTES: 2](#_Toc129743938)

[7 Feb 2023 Immersification Stream for CSD3120 3](#_Toc129743939)

[Setup Skybox 3](#_Toc129743940)

[Configure Skybox Texture (Part 1) 7](#_Toc129743941)

[Download Skybox [OPTIONAL] 9](#_Toc129743942)

[Configure Skybox Texture (Part 2) 11](#_Toc129743943)

[Setup Camera 14](#_Toc129743944)

[Setup Lights (HemisphericLight) 17](#_Toc129743945)

[Setup Lights (PointLight + HeimisphericLight) 20](#_Toc129743946)

[Setup VideoSkyDome (alternative to Skybox) 23](#_Toc129743947)

[Where to get free 360 videos (to test) 27](#_Toc129743948)

[Adding Inspector Keyboard Shortcut 28](#_Toc129743949)

[Handling window resize 32](#_Toc129743950)

[Setup Models 33](#_Toc129743951)

[Setup Animation 39](#_Toc129743952)

[Setup Particles 44](#_Toc129743953)

[Setup Sounds 50](#_Toc129743954)

[Create Button (with clicking sound) 55](#_Toc129743955)

# NOTES:

- If you haven’t done Part 1 yet, here’s the link to [Part 1](IPAsetup_part1.docx)

- If you haven’t done Part 2 yet, here’s the link to [Part 2](IPAsetup_part2.docx)

**TOPICS COVERED:**

**INSTALLING DEPENDENCIES:**

**FINAL CHANGES:**

webpack.config.js

tsconfig.json

# Developing Immersive Applications: Recorded live coding stream on "Implementing Interactions"

<https://www.youtube.com/watch?v=8zXOif9Ys8c>

## Installing plugins

Enter the following commands in WSL:

npm i copy-webpack-plugin -D

You should be able to see "copy-webpack-plugin" added to your "devDependencies"

A screenshot of a computer

Description automatically generated with medium confidence

**webpack.json**

Add the following codes:

const CopyPlugin = require('copy-webpack-plugin')

Here:

Text

Description automatically generated

Add the following codes:

new CopyPlugin({

            patterns: [

                { from: path.resolve(\_\_dirname, 'public')}

            ]

        }),

Here:

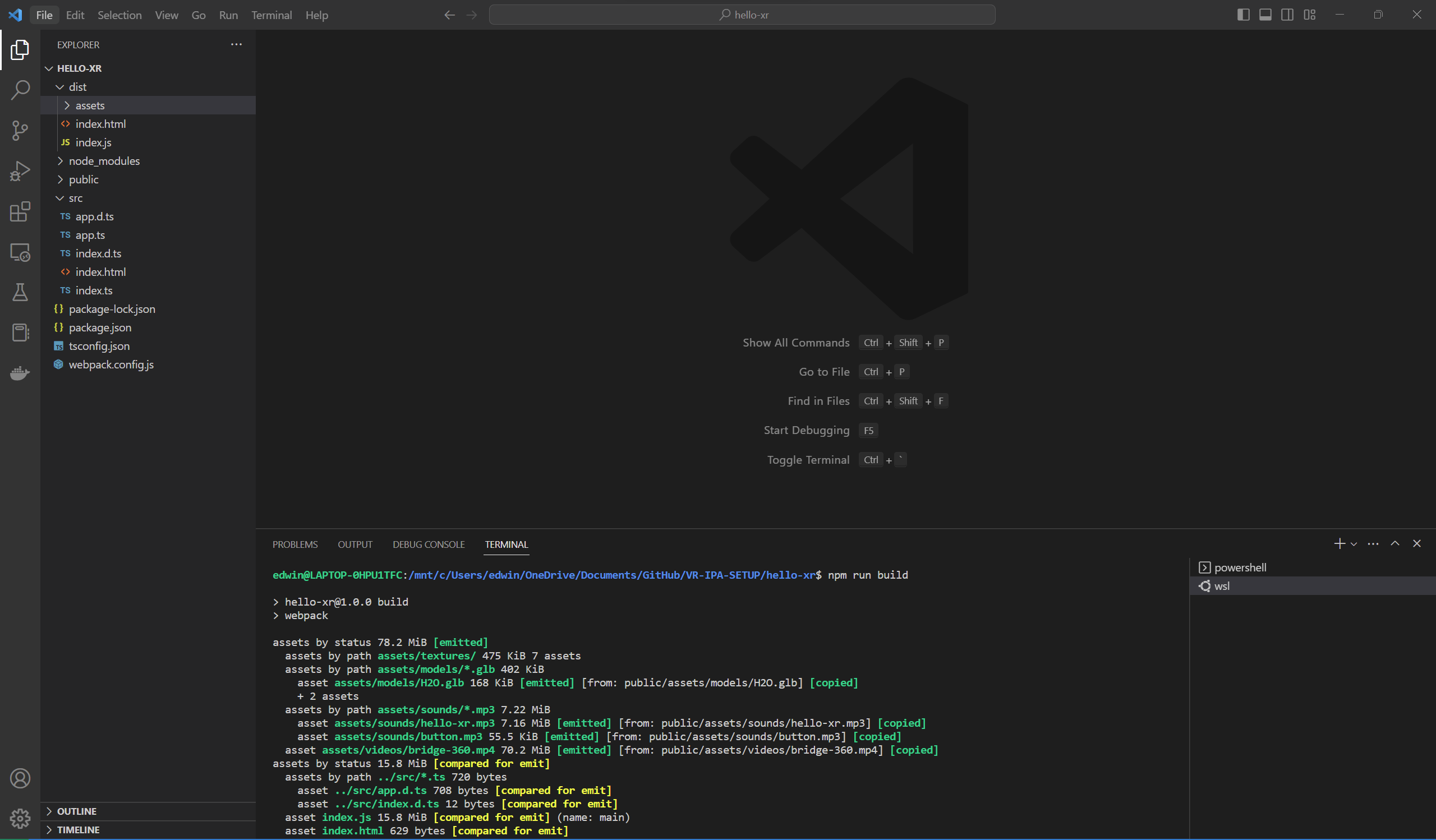
Text

Description automatically generated

Enter the following commands in WSL (to Rebuild the project):

npm run build

You should be able to see your assets folder from your public folder copied to your dist folder.

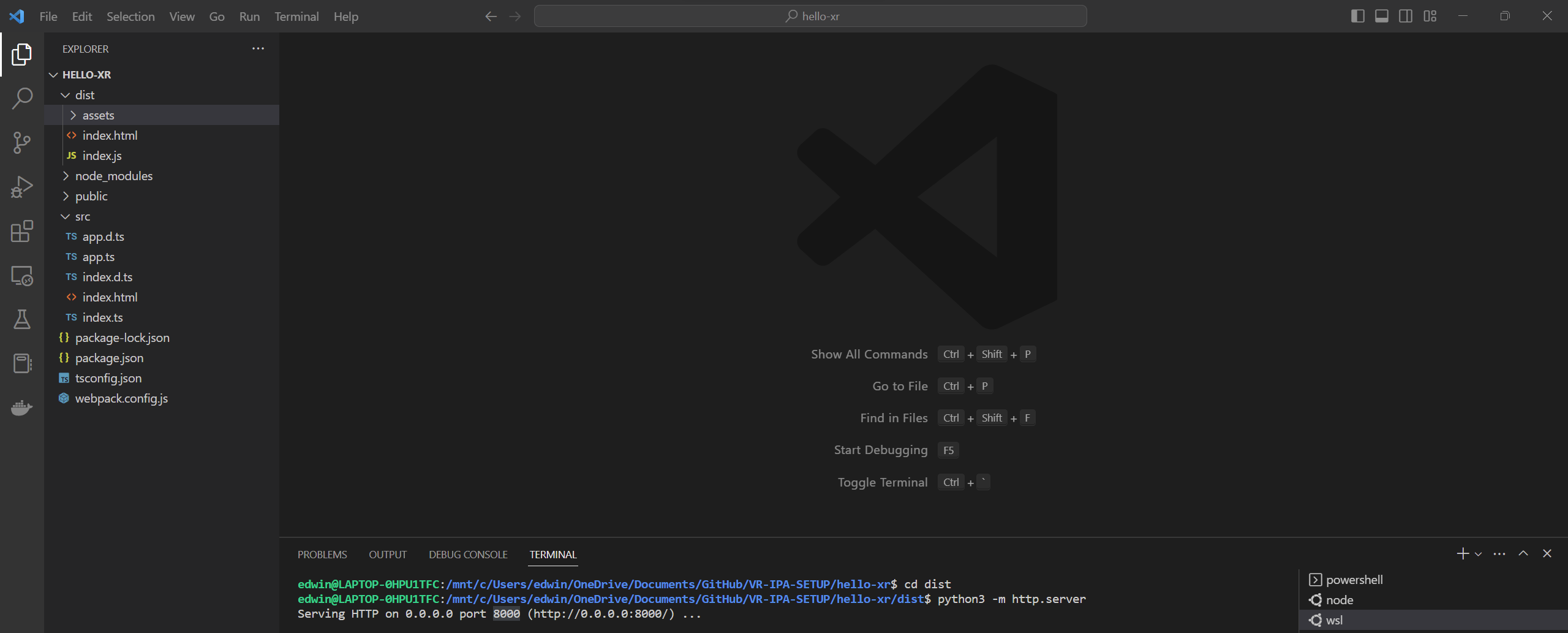


## Deploying to Production Server (instead of Dev Server)

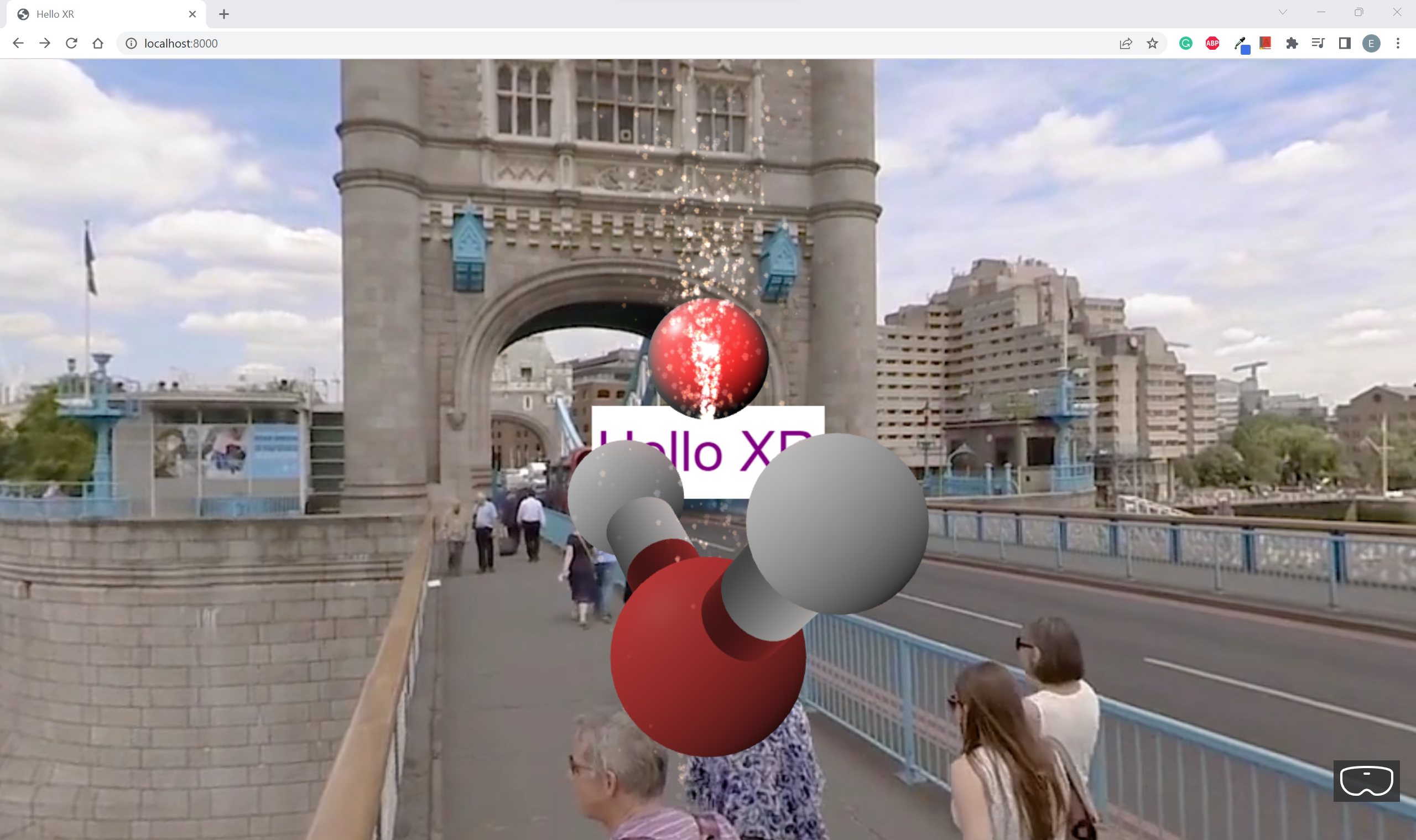
Enter the following commands in WSL:

cd dist

python3 -m http.server



Go to localhost:8000 in your chrome browser.



You should be able to see the Web App working fine on the production server.

## Refactoring codes

**tsconfig.json**

Comment out the following codes:

//"declaration": true,

Here:

Text

Description automatically generated

**app.ts**

Under createScene():

Comment out the following codes:

//this.createVideoSkyDome(scene);

//this.addSounds(scene)

Change this code to:

sphere.position.y = -0.5

Change these codes to:

scene.createDefaultCameraOrLight()

//this.createCamera(scene)

//this.createLights(scene)

*Component based architecture*

### Text Component

Enter the following commands in WSL:

cd src

mkdir -p components/meshes

cd components/meshes

touch index.ts

touch text-plane.ts

You should be able to see "components\meshes" folders added to your "src" folder.

**A screenshot of a computer

Description automatically generated with medium confidence**

Go to **components\meshes\index.ts** and add the following codes:

export \* from './text-plane'

Go to **components\meshes\text-plane.ts** and add the following codes:

import { Mesh, MeshBuilder, Scene } from "babylonjs";

import { AdvancedDynamicTexture, TextBlock } from "babylonjs-gui";

export class TextPlane {

  public mesh: Mesh

  public textBlock: TextBlock;

  constructor(

    name: string,

    width: number,

    height: number,

    x: number,

    y: number,

    z: number,

    text: string,

    backgroundColor: string,

    textColor: string,

    fontSize: number,

    scene: Scene

  ) {

    const textPlane = MeshBuilder.CreatePlane(name + " text plane", {

      width: width,

      height: height,

    });

    textPlane.position.set(x, y, z);

    const planeTexture = AdvancedDynamicTexture.CreateForMesh(

      textPlane,

      width \* 100,

      height \* 100,

      false

    );

    planeTexture.background = backgroundColor;

    const planeText = new TextBlock(name + " plane text");

    planeText.text = text;

    planeText.color = textColor;

    planeText.fontSize = fontSize;

    planeTexture.addControl(planeText);

    this.mesh = textPlane

    this.textBlock = planeText

  }

}

**app.ts**

Add the following codes:

import { TextPlane } from "./components/meshes"

Here:

Text

Description automatically generated

Replace CreateText(scene:Scene) with the following codes:

createText(scene:Scene)

    {

        const helloPlane = new TextPlane(

            "hello plane",

            3,

            1,

            0,

            2,

            5,

            "Hello XR",

            "white",

            "purple",

            60,

            scene

        )

        helloPlane.textBlock.onPointerUpObservable.add(evtData => {

            alert('Hello Text at: \n x: ' + evtData.x + ' y:' + evtData.y)

        })

        helloPlane.textBlock.onPointerDownObservable.add(() => {

            this.sound.play()

        })

    }

Here:

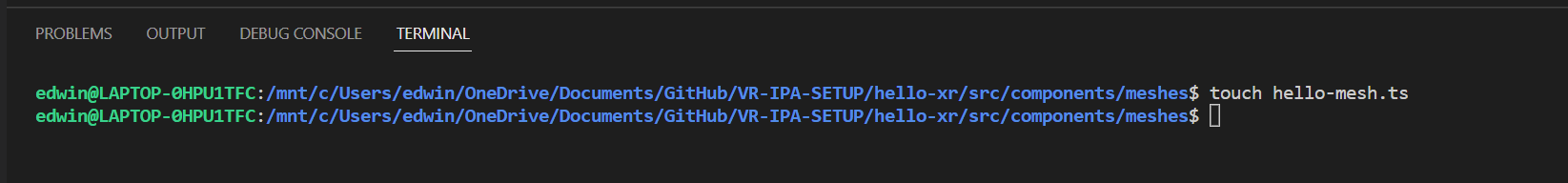
Text

Description automatically generated

### Mesh Component

Enter the following commands in WSL:

touch hello-mesh.ts

****

You should be able to see "hello-mesh.ts" file added to your "components\meshes" folder.

**Text

Description automatically generated**

Change these codes under createScene() to:

//simple sphere

        const sphere = MeshBuilder.CreateSphere('sphere', { diameter: 1.3 }, scene)

        sphere.position.y = -0.5

        sphere.position.z = 5

        //hello sphere

        const helloSphere = new HelloSphere("hello sphere", {diameter: 1}, scene)

        helloSphere.position.set(0, 1, 5)

Go to **components\meshes\index.ts** and add the following codes:

export \* from './hello-mesh'

Go to **components\meshes\hello-mesh.ts** and add the following codes:

import {

  AbstractMesh,

  Mesh,

  MeshBuilder,

  Scene,

  StandardMaterial,

} from "babylonjs";

import { TextPlane } from "./text-plane";

export interface HelloMesh {

  scene: Scene;

  mesh: Mesh;

  label: TextPlane;

}

export class HelloSphere extends AbstractMesh implements HelloMesh {

  scene: Scene;

  mesh: Mesh;

  label: TextPlane;

  constructor(name: string, options: { diameter: number }, scene: Scene) {

    super(name, scene);

    this.scene = scene;

    this.mesh = MeshBuilder.CreateSphere("hello sphere mesh", options, scene);

    this.mesh.material = new StandardMaterial("hello sphere material", scene);

    this.addChild(this.mesh);

    this.label = new TextPlane(

      "hello sphere label",

      1.5,

      1,

      0,

      options.diameter / 2 + 0.2,

      0,

      "hello sphere",

      "purple",

      "white",

      25,

      scene

    );

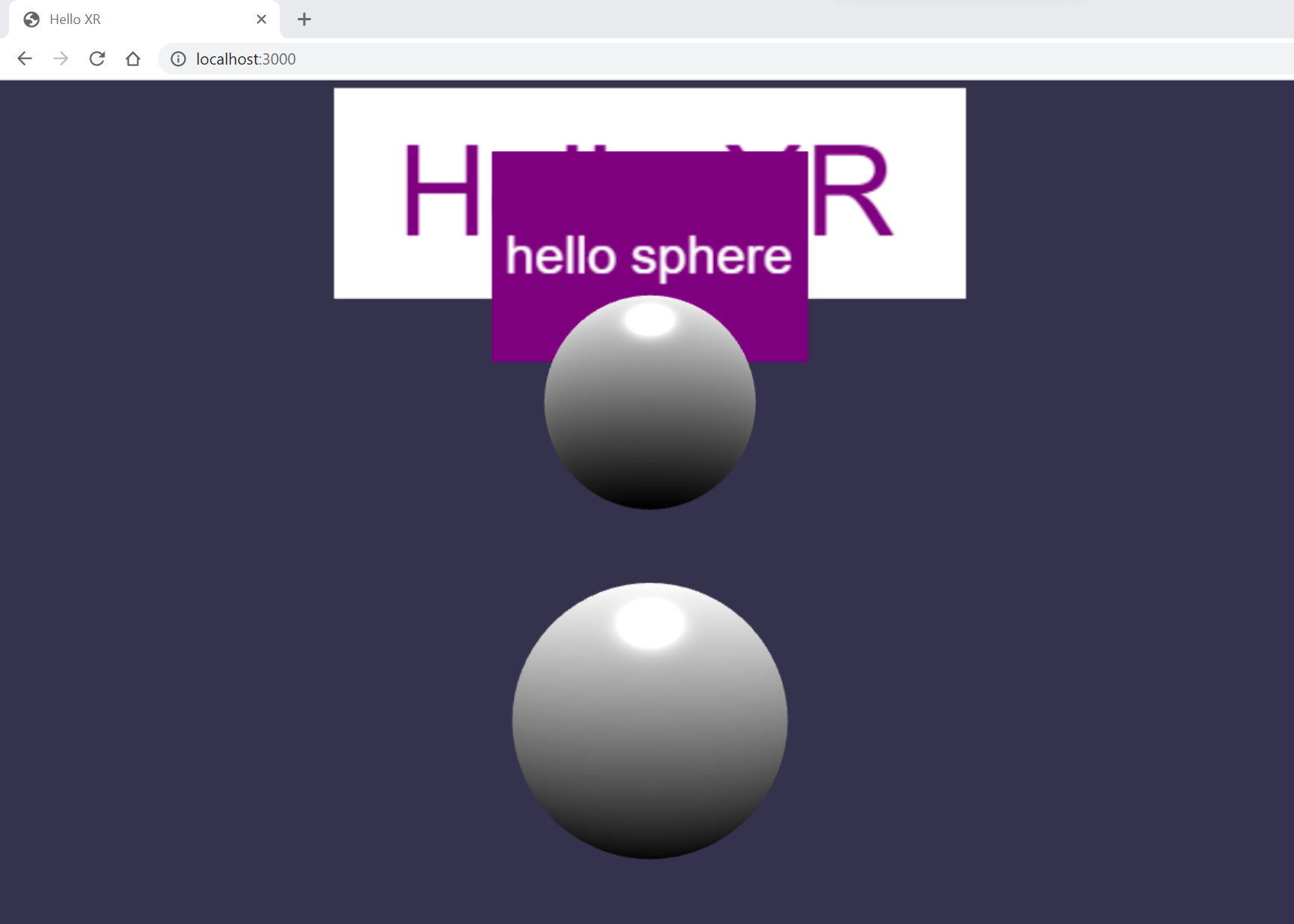
    this.addChild(this.label.mesh);

  }

}

**Check**

You should see 2 Spheres rendered in your web app.



## Interactions