Name: Mustafa Baran Ercan
ID: 2881055520 **Section:** 01

Name: Neslihan Pelin Metin
ID: 71047171244 **Section:** 02

CMPE 360 PROJECT 9 REPORT

PART 1

- What each person in your group did

First, we determined the concept of the art gallery we would create in our project. Based on the movies we both like, we chose the leading characters that we could use in our project. After a long research and elimination, we decided on Batman, Jack Sparrow, Darth Vader, and John Wick. To start our project, we arranged the distribution of tasks and planning. We both installed node.js and npm on our computers, but since we could not install three.js on Pelin's computer, we started working on Baran's computer. We organized regular meetings and together we made the skybox, the text, the walls, the floor and the animated cube and ring in the middle. Then we divided the movie characters we chose among ourselves and added the characters and posters. Baran made the characters and posters of the Batman and Pirates of the Caribbean movies, and Pelin made the Star Wars and John Wick movies.

- Add screenshots of objects (8 objects)

Image 1: Cube, Ring, Text, Batman, Batman Poster, Jack Sparrow, Pirates of Caribbean Poster.

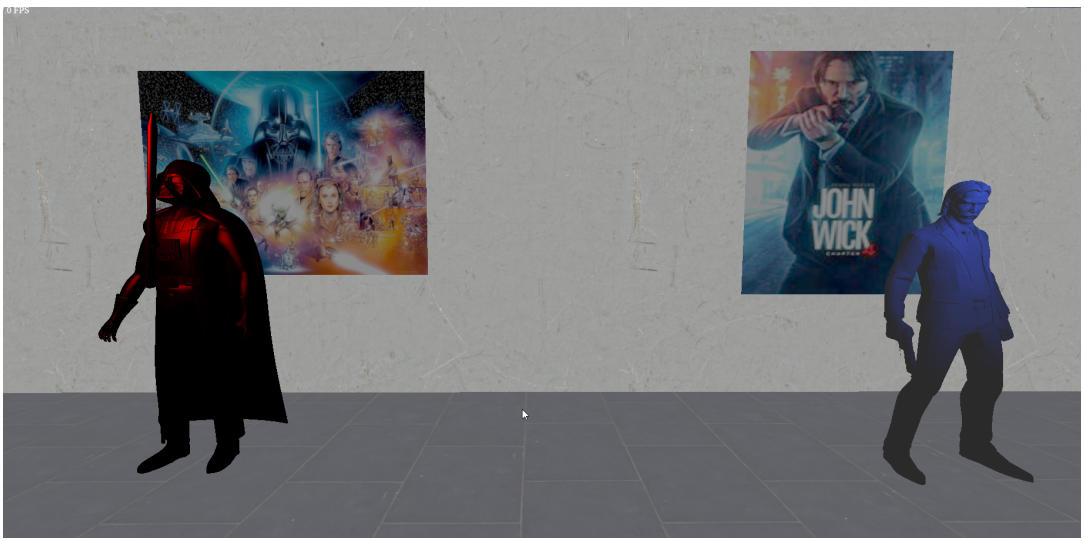

Image 2: Darth Vader, Star Wars Poster, John Wick, and his poster.

Name: Mustafa Baran Ercan
ID: 2881055520 **Section:** 01

Name: Neslihan Pelin Metin
ID: 71047171244 **Section:** 02

- Write geometry types (at least 3) geometry you use.

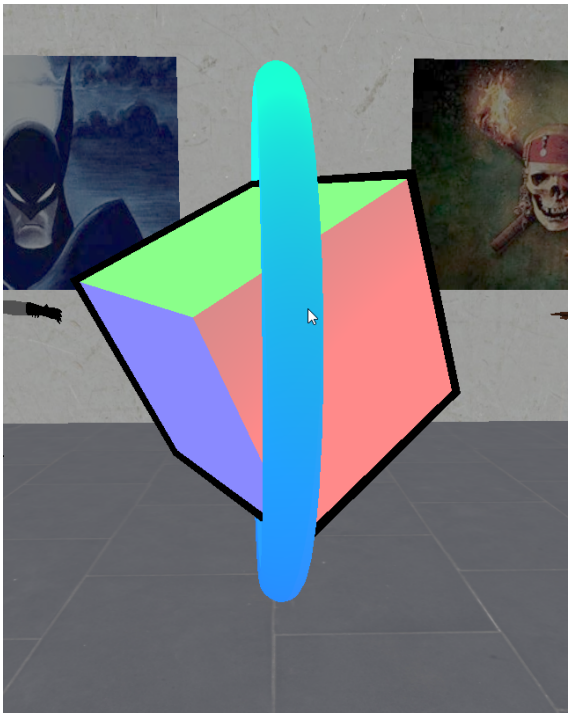

Image 3: Cube and Ring

Image 4: Custom OBJ models

Image 5: Plane (Walls, Floor and Posters)

Name: Mustafa Baran Ercan
ID: 2881055520 **Section:** 01

Name: Neslihan Pelin Metin
ID: 71047171244 **Section:** 02

PART 2

- Add multiple lights, skybox and texture mapping and explain your process.

Lights

```
    //Ambient Light
    const ambientLight = new THREE.AmbientLight(0xffffff, 0.8);
    scene.add(ambientLight);

    // These 3 lights are used to light up the cube with different colors.
    //Point Light - 1
    const pointLight1 = new THREE.PointLight(0x0000ff, 50, 7);    // Blue
    pointLight1.position.set(-2, 0, 0);
    scene.add(pointLight1);

    //Point Light - 2
    const pointLight2 = new THREE.PointLight(0xff0000, 50, 7);    // Red
    pointLight2.position.set(2, 0, 0);
    scene.add(pointLight2);

    //Point Light - 3
    const pointLight3 = new THREE.PointLight(0x00ff00, 50, 7);    // Green
    pointLight3.position.set(0, 4, 0);
    scene.add(pointLight3);
```

Our approach begins by establishing an ambient light source, providing a gentle, overall illumination throughout the scene with a soft white hue. Calibrated with an intensity of 0.8, this light illuminates the entire environment evenly. Additionally, we have integrated three distinct point lights, each emitting a distinct color - blue, red, and green. These lights, carefully positioned using position.set(), contribute focused and colored illumination from specific locations in the scene. Set with intensities at 50 and a reach of 7 units, these point lights enhance the visual ambiance by selectively casting vibrant, targeted lighting within the space, allowing us to highlight specific elements or areas within our scene.

```
        // Red Special Light just for Darth Vader.
        const vaderLight = new THREE.PointLight(0xff0000, 1000, 5);
        vaderLight.position.x = 6.5; vaderLight.position.y = 5; vaderLight.position.z = 14;
        scene.add(vaderLight);
```

```
        // light for john wick.
        const johnWickLight = new THREE.PointLight(0x122368, 1000, 5);
        johnWickLight.position.x = -7; johnWickLight.position.y = 5; johnWickLight.position.z
= 14;
        scene.add(johnWickLight);
```

We also wanted to add special point lights with different colors for Darth Vader and John Wick.

Name: Mustafa Baran Ercan
ID: 2881055520 **Section:** 01

Name: Neslihan Pelin Metin
ID: 71047171244 **Section:** 02

Skybox

```
    //Skybox
    const CubeTextureLoader = new THREE.CubeTextureLoader();
    const skyboxMaterials = CubeTextureLoader.load([
        'assets/skybox/1.jpg',
        'assets/skybox/6.jpg',
        'assets/skybox/5.jpg',
        'assets/skybox/2.jpg',
        'assets/skybox/3.jpg',
        'assets/skybox/4.jpg',
    ]);
    scene.background = skyboxMaterials;
```

We integrated a skybox using the CubeTextureLoader property. Our approach involved taking advantage of this specialized loader designed to manage textures on six sides of a cube. Initially, we used the CubeTextureLoader's load() method and provided a set of file paths leading to individual images representing the front, back, top, bottom, left and right perspectives of the virtual cube. These images collectively formed the panoramic view that encompassed the entire scene. Following this, we assigned the uploaded textures to scene.background, effectively wrapping the entire environment with a harmonious backdrop.

Texture Mapping

We applied texture mapping to the skybox, the floor, the walls, the characters, and the posters. Since we explained how we made the skybox above, let's explain how we apply it to the rest of the objects.

1. Floor

```
    // Floor
    const floorGeometry = new THREE.PlaneGeometry(100, 100, 1, 1);
    const floorTexture = new THREE.TextureLoader().load('assets/floor/rubber_tiles.jpg');
    const floor = new THREE.Mesh(floorGeometry, new THREE.MeshBasicMaterial({ map:
floorTexture }));
    const repeat = 13;
    floorTexture.wrapS = THREE.RepeatWrapping;
    floorTexture.wrapT = THREE.RepeatWrapping;
    floorTexture.repeat.set(repeat, repeat);

    floor.rotation.x = -Math.PI / 2;
    floor.position.y = -0.6;
    scene.add(floor);
```

We created a floor using PlaneGeometry and applied it to create a textured surface by including the texture rubber_tiles.jpg. Laying the texture along the floor increased its visual appeal and we carefully positioned it parallel to the floor.

2. Walls

```
    // Walls
        // Front Wall
        const wallGeometry = new THREE.PlaneGeometry(30, 17, 1, 1);
        const wallTexture = new THREE.TextureLoader().load('assets/walls/concrete.jpg');
        const wall = new THREE.Mesh(wallGeometry, new THREE.MeshBasicMaterial({ map:
wallTexture, color: 0xffffff, side: THREE.DoubleSide, transparent: true}));

        const repeatWallInsideWidth = 2, repeatWallInsideHeight = 1;
        wallTexture.wrapS = THREE.RepeatWrapping;
        wallTexture.wrapT = THREE.RepeatWrapping;
        wallTexture.repeat.set(repeatWallInsideWidth, repeatWallInsideHeight);
```

We also created walls using PlaneGeometry and customized them with textures like concrete.jpg using MeshBasicMaterial. Our deliberate adjustment of the repetition of textures along these walls changed their appearance based on certain tiling parameters.

3. Characters

```
         // Batman texture
        const batmanTexture = new
THREE.TextureLoader().load('assets/characters/batman/Batman_texture.png');
        const batmanMaterial = new THREE.MeshPhongMaterial({ map: batmanTexture });

        // Batman
        loaderOBJ.load('assets/characters/batman/Batman.obj', (batman) => {
            batman.traverse(function(child) {
                if (child instanceof THREE.Mesh) {
                    child.material = batmanMaterial;
                }
            });
            batman.position.x = -5;
            batman.position.y = 2.65;
            batman.position.z = -6;
            batman.scale.multiplyScalar(0.3);

            scene.add(batman);
        });
```

We included Batman's 3D model (Batman.obj) in the scene and obtained Batman's texture (Batman_texture.png) via TextureLoader(). We applied this texture to different parts of Batman's model using MeshPhongMaterial, effectively changing their visual representation in our scene. Batman was then precisely positioned and scaled appropriately for an accurate fit.

4. Posters

```
        // Batman Poster
        const darkKnightTexture = new
THREE.TextureLoader().load('assets/posters/batman/batman.jpg');
        const darkKnightMaterial = new THREE.MeshPhongMaterial({ map: darkKnightTexture});
        var darkKnightPoster = new THREE.Mesh(new THREE.PlaneGeometry(7, 5),
darkKnightMaterial);
        darkKnightPoster.position.x = -5.6;
        darkKnightPoster.position.y = 5;
        darkKnightPoster.position.z = -9.9;
        scene.add(darkKnightPoster);
```

We also created creating plane-like objects with textures such as batman.jpg, which we loaded as textures for these planes. These images acted as textures applied through MeshPhongMaterial and then we placed them in our scene as posters of the movies.

- Which features are animated, explain your process.

1. Cube

```
    cube.rotation.x -= 0.003; cube.rotation.y += 0.003;
```

2. Ring

```
    ring.rotation.y += 0.006; ring.rotation.x -= 0.006;
```

3. Text

```
    textMaterial.uniforms.time.value += 0.005;
```

We applied continuous rotation to the cube by changing the cube.rotation.x and cube.rotation.y properties within the animation loop (init() function). Simultaneously, we animated the ring object by setting the ring.rotation.y and ring.rotation.x properties, allowing it to rotate around its axis. Additionally, we added an animated effect for text displayed on a plane. For this we used a customized textMaterial with a shader that dynamically changes the appearance of the text over time. Within the shader, we continuously increased the time uniformity (textMaterial.uniforms.time.value += 0.005;) within the animation loop (init() function), resulting in a transformation of the visual representation of the text.

Name: Mustafa Baran Ercan
ID: 2881055520 **Section:** 01

Name: Neslihan Pelin Metin
ID: 71047171244 **Section:** 02

- Write shader effects you used explain your process.

```
// Define the color for the color pass
const textMaterial = new THREE.ShaderMaterial({
    uniforms: {
        map: { value: texture },
        time: { value: 0.0 },
    },
    vertexShader: `
        varying vec2 vUv;
        void main() {
            vUv = uv;
            gl_Position = projectionMatrix * modelViewMatrix * vec4(position, 1.0);
        }
    `,
    fragmentShader: `
        uniform sampler2D map;
        uniform float time; // To animate the text.
        varying vec2 vUv;

        void main() {
            vec4 texColor = texture2D(map, vUv);

            vec3 colors = vec3(
                0.5 + 0.5 * sin(time * 2.0 + vUv.x * 10.0),
                0.5 + 0.5 * sin(time * 2.0 + vUv.x * 10.0 + 2.0),
                0.5 + 0.5 * sin(time * 2.0 + vUv.x * 10.0 + 4.0)
            );

            gl_FragColor = vec4(texColor.rgb * colors, texColor.a);
        }
    `,
    transparent: true,
});
```

We have integrated shader effects into the textMaterial definition. This textMaterial is specifically designated as a ShaderMaterial and surrounds both vertexShader and fragmentShader features. While the vertexShader remains a basic transition shader, we have taken responsibility for the important fragmentShader. Our implementation dynamically changes the appearance of the displayed text by modulating its color based on sinusoidal functions of time. By customizing the fragment shader, we extract the original text color from the texture and blend it intricately with dynamically generated color variations, all controlled by the time uniform.

Name: Mustafa Baran Ercan
ID: 2881055520 **Section:** 01

Name: Neslihan Pelin Metin
ID: 71047171244 **Section:** 02

- Add two screenshots with different camera views of your scene when you finish your work.

Image 6 Camera View 1

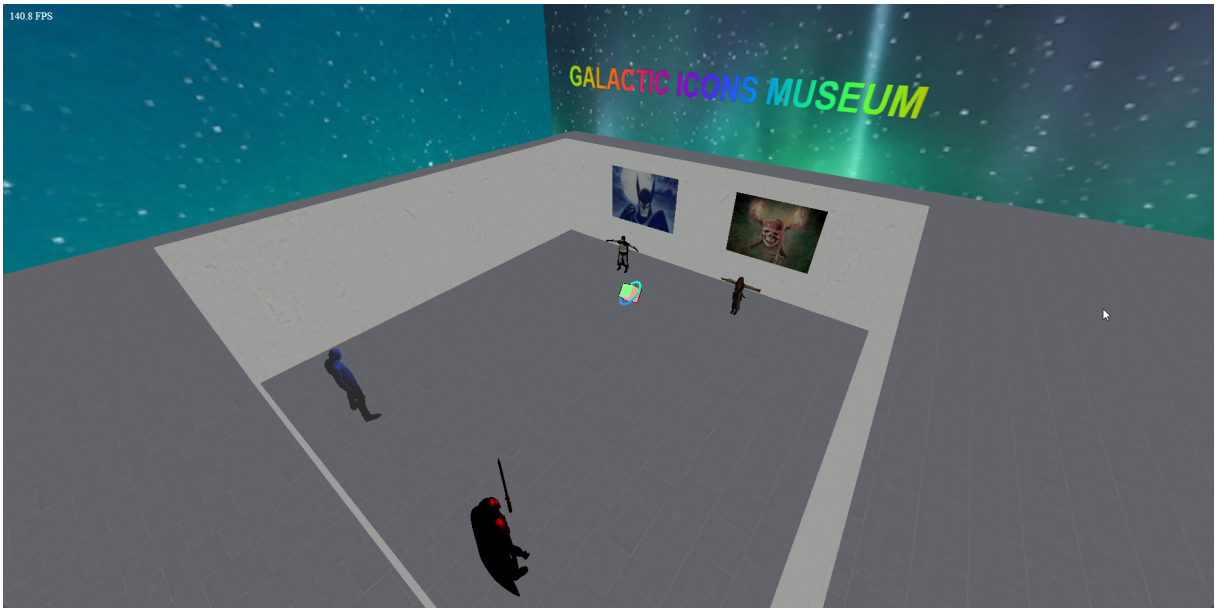

Image 7 Camera View 2

- Please record a video the all options and copy the drive link to your report

https://drive.google.com/file/d/1ksTjnqMxKocQMhBInPq72kxTylaaL2xP/view?usp=sharing