

# Projet Musée Virtuel



Brest 2024



**El Mehdi Haddadi  
Ayoub Elfinou**



# SOMMAIRE

## **I. Contexte du projet**

## **II. Développement**

1. Structure du musée

2. Crédit des tableaux

3. Crédit des portes

4. Crédit des status

5. Points de déplacements

6. Bonus

## **III. Conclusion**



# Table de figures



Figure 1 : Hall

Figure 2 : Premier étage

Figure 3 : Illustration d'un tableau interactif avec des lampes et des éléments de texte interactifs.

Figure 4 : Figure illustrant l'activation du son.

Figure 5 : Code pour initialiser les détecteurs de lampes et de son, ainsi que pour déclencher l'affichage des images et du texte

Figure 6 : Figure représentant l'ouverture et la fermeture de la porte automatiquement.

Figure 7 : Code pour ouvrir et fermer la porte après détection.

Figure 8 : Figure représentant un état avec rotations en cours.

Figure 9 : Code qui importe les objets et fait tourner les états.

Figure 10 : Figure représentant des sphères de téléportation activé.

Figure 11 : Figure représentant un ascenseur.

Figure 12 : Figure représentant un ascenseur après ouverture automatique, détection et affichage de messages.

# I. Contexte du projet

Pour le projet du module « Réalité Virtuelle », notre mission était de concevoir un environnement 3D immersif. Celui-ci devait permettre à l'utilisateur de se déplacer librement et d'interagir avec les objets présents dans la scène. Nous avons opté pour le moteur 3D en temps réel **Babylon.js**.



Ce choix était fait en raison qu'il facilite la création d'éléments de la scène, comme les caméras, les objets et les animations, grâce à ses nombreuses méthodes intégrées.

Pour donner vie à notre musée virtuel, nous avons suivi des directives spécifiques concernant la disposition de l'espace et la structure du bâtiment, notamment la présence de deux étages et une superficie de 30 x 30 mètres. Ces consignes sont détaillées dans le document *enonce.pdf*.

La première phase de notre travail a consisté à concevoir le design du musée et à choisir une thématique permettant d'incorporer des tableaux et des statues en adéquation avec notre sujet.

Notre musée a pour objectif de présenter **la « belle époque » période de 1880 à 1914.**

## **II. Développement**

### **1. Structure du musée :**

Lorsque vous entrez dans le hall de notre musée, vous avez deux options :

- Explorer **les trois salles adjacentes** où divers **tableaux** sont exposés.
- Monter au **premier étage** en empruntant **les escaliers**. À l'étage, vous découvrirez d'autres **tableaux** ainsi que **des objets géométriques** articulés et animés.

Nous avons ajouté **un ascenseur** à gauche du hall, dont les portes s'ouvrent lorsque l'utilisateur s'approche. Cependant, il est actuellement hors service en raison d'un problème de mise en œuvre de la translation de l'ascenseur. Pour maintenir le réalisme, les visiteurs verront **une notification** indiquant que l'ascenseur est temporairement hors service, accompagnée d'excuses pour le désagrément occasionné.

Les Figures 1 et 2 illustrent un premier exemple de design, montrant les différentes salles et espaces répartis sur les deux étages. Voici une présentation succincte de notre idée initiale

#### **Au Hall :**

- Un accueil avec des statues.
- Des escaliers et un ascenseur.
- Trois salles d'exposition avec des peintures.
- Des sphères de téléportation.

#### **Au premier étage :**

- Une salle avec une statue dynamique.
- Des lampes pour l'éclairage.
- Des sphères de téléportation.
- De grands tableaux d'art.



Figure 1 : Hall



Activer Windows

Accédez aux paramètres pour activer Windows.

Figure 2 : Premier étage



## 2. Cration des tableaux :

Dans chaque salle d'exposition de notre **muse virtuel**, nous avons plac **huit tableaux de la Belle poque**. Chaque tableau est accompagn d'une **lampe** qui s'allume lorsque l'utilisateur s'en approche, ainsi que d'une **musique** qui se dclenche. De plus, lorsque l'utilisateur regarde un tableau, **le nom** de celui-ci s'affiche, et s'il reste  le regarder ou effectue **un double clic, une description dtaille** du tableau apparat.



Figure 3 : Illustration d'un tableau interactif avec des lampes et des lments de texte interactifs.

!

Nous avons ajouté de la musique. Donc n'oubliez pas **d'activer le son** pour entendre de la musique lors de la visualisation des tableaux, afin d'immerger pleinement le visiteur dans une ambiance propice à la réflexion.

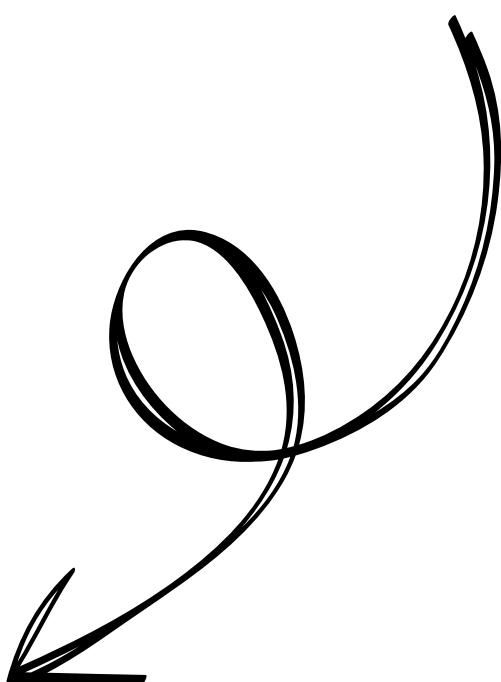


Figure 4 : Figure illustrant l'activation du son.

Pour implémenter ces fonctionnalités, nous avons utilisé les détecteurs d'utilisateur de Babylon.JS pour déclencher les sons et les affichages de textes. Voici un exemple de code montrant comment créer un tableau avec ces interactions :

```
/** detecteur (evenement afficher le nom de l'oeuvre) */
let watchingTableau = false;
collider.actionManager = new BABYLON.ActionManager(scene);
let tableauDesc = new BABYLON.ExecuteCodeAction(
{
    trigger: BABYLON.ActionManager.OnIntersectionEnterTrigger,
    parameter: {
        mesh: box,
    },
},
(evt) => {
    watchingTableau = true;
    light.setEnabled(true);
    music.play();
    var i = 15; // ms

    /** timer nom de tableau */
    var handle = window.setInterval(() => {
        i--;
        UItext.text = tableauName;

        if (i === 0) {
            window.clearInterval(handle);
            UItext.text = "";
        }
    }, 100);

    /** timer description */
    var j = 30; // ms
    var handle2 = window.setInterval(() => {
        j--;

        if (j === 0) {
            window.clearInterval(handle2);
            if (watchingTableau)
                UItext.text = tableauDescription;
        }
    }, 100);
});
```

```

let tableauOut = new BABYLON.ExecuteCodeAction(
{
    trigger: BABYLON.ActionManager.OnIntersectionExitTrigger,
    parameter: {
        mesh: box,
    },
},
(evt) => {
    watchingTableau = false;
    light.setEnabled(false);
    UItext.text = "";
    music.stop();
}
);
collider.actionManager.registerAction(tableauDesc);
collider.actionManager.registerAction(tableauOut);

// Set position and rotation
group.parent = cloison;
group.position = new BABYLON.Vector3(x, y, z);
group.rotation.y = rotation;

return group;
}

```

Figure 5: Code pour initialiser les détecteurs de lampes et de son, ainsi que pour déclencher l'affichage des images et du texte.

### 3. Cration des portes :

Pour **les portes coulissantes**, lorsque le visiteur s'approche d'une porte, celle-ci **s'ouvre automatiquement**. Quand l'utilisateur s'loigne de la porte, qu'il l'ait franchie ou non, **la porte se referme**. Les portes sont dotes **d'une texture raliste** qui renforce l'impression de qualit et d'authenticit.



Figure 6 : Figure reprsentant l'ouverture et la fermeture de la porte automatiquement.

Voici le code qui détecte l'utilisateur et ouvre ou ferme la porte en conséquence : **OpenDoor()**

```
function Opendoor(scene, porte, camera, largeur) {
    var box = BABYLON.MeshBuilder.CreateBox("detecteur", { width: 3, height: 5, depth: 10 })
    box.position = new BABYLON.Vector3(0, 0, 0);
    box.useAlphaFromDiffuseTexture = true;
    box.parent = porte;
    var invisibleMaterial = new BABYLON.StandardMaterial("stairsmat");
    invisibleMaterial.alpha = 0;
    box.material = invisibleMaterial;
    let collidercam = BABYLON.MeshBuilder.CreateBox(
        "collider",
        { size: 0.5 }
    );
    // création mesh camera
    collidercam.parent = camera;
    collidercam.isPointerBlocker = false;
    collidercam.actionManager = new BABYLON.ActionManager(scene);
    let actionOpenDoor = new BABYLON.ExecuteCodeAction(
        {
            trigger: BABYLON.ActionManager.OnIntersectionEnterTrigger,
            parameter: {
                mesh: box,
            },
        },
        (evt) => {
            var i = 30;
            movedoor(porte, scene, false, largeur);
            var handle = window.setInterval(() => {
                i--;
                if (i === 0) {
                    window.clearInterval(handle);
                    movedoor(porte, scene, true, largeur);
                }
            }, 100);
        }
    );
    collidercam.actionManager.registerAction(actionOpenDoor);
}
```

Figure 7 : Code pour ouvrir et fermer la porte après détection.

#### 4. Création des status :

Pour la partie **des statues tournantes**, nous avons ajouté **deux statues** dans le hall et deux autres au deuxième étage. Ces statues sont des objets de type .obj avec leur propre texture. Pour les importer, nous utilisons la fonction pré définie **ImportMesh** de Babylon.js, en spécifiant **les propriétés** de nos statues. Ensuite, nous faisons appel à la fonction **rotateStatue**, qui permet de faire **tourner** les statues **en temps réel** d'un angle donné.

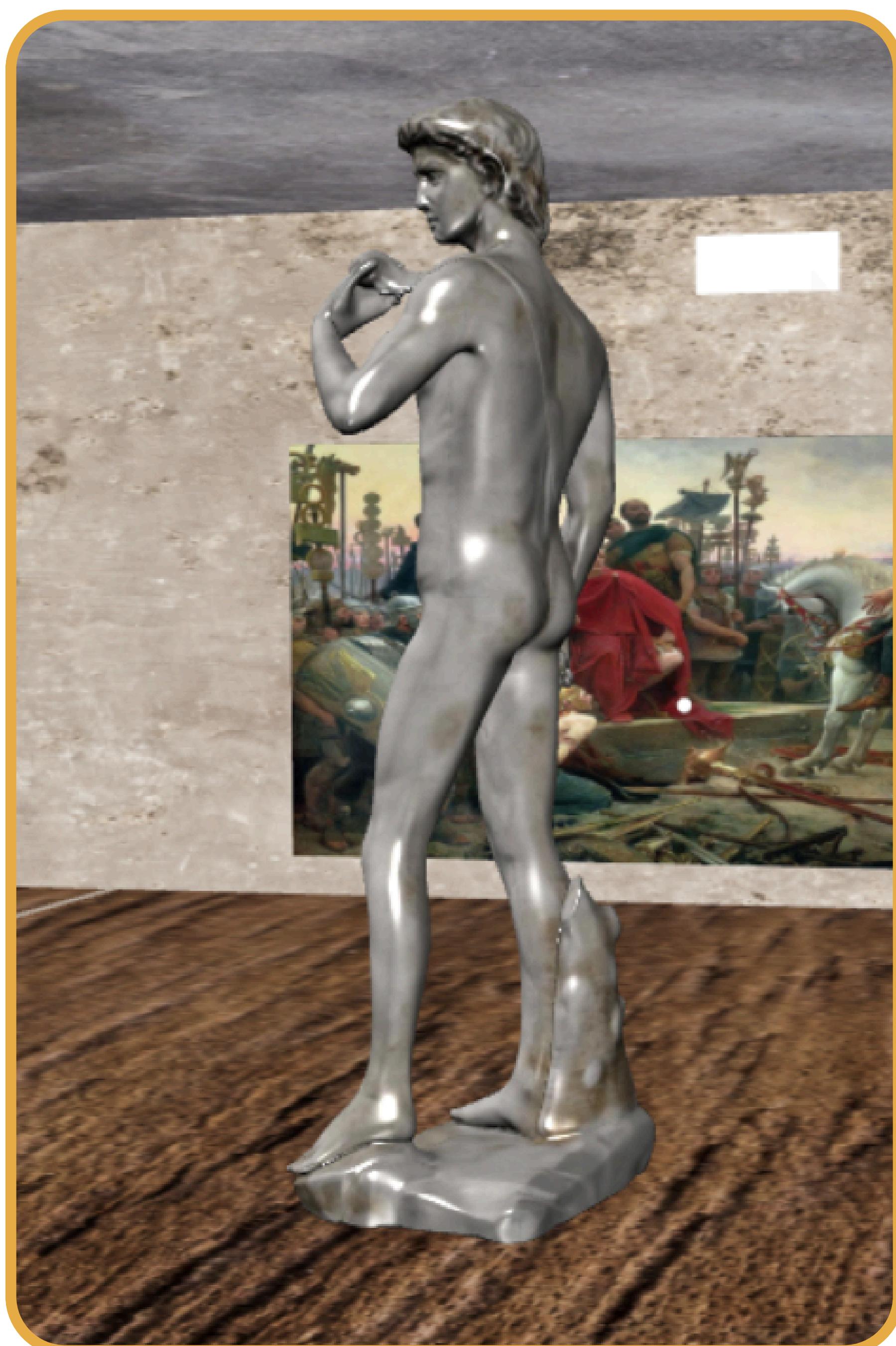


Figure 8 : Figure représentant un état avec rotations en cours.

Voici le code qui importe la mesh et appelle la fonction **rotateStatue()** .

```
// Status
BABYLON.SceneLoader.ImportMesh("", "assets/status/", "Venus.obj", this.scene, (meshes) => {
    const statue_material = PRIMS.standardMaterial("mat-venus", {
        texture: "assets/status/Venus.jpg",
        uScale: 1,
        vScale: 1
    }, this.scene);

    const m = meshes[1];
    m.scaling = new BABYLON.Vector3(1, 1, 1);
    m.position = new BABYLON.Vector3(7.5, 0, 0);
    m.rotation.y = Math.PI / 2;
    m.material = statue_material;
    m.checkCollisions = true;

    rotateStatue(m, this.scene);
});
```

Figure 9 : Code qui importe les objets et fait tourner les états.

## 5. Points de déplacement :

Nous avons mis en place **des sphères pour permettre à l'utilisateur de se déplacer dans le musée**. Pour se déplacer, il suffit de placer le curseur sur une sphère, qui change de couleur en rouge pour indiquer qu'elle est **sélectionnable**. L'utilisateur peut alors cliquer pour se déplacer à l'emplacement de la sphère. Ce mécanisme offre une meilleure expérience utilisateur en permettant de choisir précisément où se déplacer dans le musée.



Figure 10 : Figure représentant des sphères de téléportation activé.

## 5. Bonus : (Ascenseur)

Nous avons ajouté **un bonus** intéressant sous la forme d'un **ascenseur**, permettant à l'utilisateur de **se déplacer vers le deuxième étage**. Le design de l'ascenseur est **classique** : il comporte une **porte** qui **s'ouvre** lorsque l'utilisateur **s'en approche**, **se fermant automatiquement** après un **certain temps**. Un **bouton** est également présent pour permettre à l'utilisateur de monter vers le haut. Cette fonctionnalité enrichit l'expérience en offrant une navigation fluide et intuitive à travers les différents niveaux du musée virtuel.

Et une **notification** qui prévient nos visiteurs que l'ascenseur est temporairement **hors service**.



Figure 11 : Figure représentant un ascenseur.



Figure 12 : Figure représentant un ascenseur après ouverture automatique, détection et affichage de messages.

### **III. Conclusion**

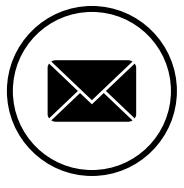
En conclusion, le module de Réalité Virtuelle, et plus particulièrement le projet de création d'un musée virtuel, nous a offert une formidable opportunité d'explorer le monde du 3D grâce à Babylon.js.

En nous documentant et en expérimentant, nous avons non seulement appris à créer des objets et des environnements interactifs, mais nous avons aussi découvert le potentiel créatif et ludique de la réalité virtuelle.

Ce projet nous a permis de repousser les limites de notre imagination tout en acquérant des compétences précieuses dans le domaine de la conception et du développement d'applications en 3D.



a1elfino@enib.fr



e1haddad@enib.fr