



REPUBLIQUE DU BENIN

MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA
RECHERCHE EN INFORMATIQUE

INSTITUT DE FORMATION ET DE RECHERCHE EN INFORMATIQUE
IFRI-UAC

Filière : Internet et Multimédia (IM)

UC : Technologie Immersive

Projet 3

GALERIE D'ŒUVRES D'ART INTERACTIVES EN WEBXR

Membre du Groupe 3

- 1- DJOSSOU Carmel (Code, apk)
- 2- DJOUBALE Paterne (Image arts, audio, Présentation)
- 3- DOSSOUNON Jude -Ronel (Code)

supervisé par

M^r AHOUANDJINOU

Lémeç

ANNÉE – ACCADEMIQUE : 2024 -2025

Introduction

Notre travail pratique a pour but de créer une galerie d'art virtuelle en 3D accessible directement depuis un navigateur web, sans avoir besoin d'installer une application. Grâce à la technologie WebXR, les utilisateurs pourront explorer cette galerie de manière immersive, comme s'ils se déplaçaient réellement dans un musée.

Dans cet espace virtuel, ils pourront :

- Se déplacer librement pour découvrir différentes œuvres (peintures, sculptures, objets 3D).
- Interagir avec les œuvres en affichant des informations, en zoomant ou en écoutant une description audio.
- Naviguer facilement dans l'environnement grâce à un système de téléportation ou de déplacement libre.

Pour réaliser ce projet, nous utiliserons des outils comme A-Frame ou Three.js, qui permettent de créer des scènes en réalité virtuelle directement accessibles via un navigateur. Mais nous avons choisi d'utiliser A-Frame comme outils pour créer notre scène en réalité virtuelle. Ce TP est une excellente occasion d'apprendre les bases du développement en WebXR et de concevoir des expériences immersives en ligne.

I- La Modélisation dans Blender (Gérer par DJOUBALE Paterne)

Pour commencer je dirai que la **modélisation en Blender** consiste à créer des objets et environnements en 3D à l'aide du logiciel **Blender**, qui est un outil puissant et gratuit pour la conception 3D. Avec Blender, on peut **sculpter, texturer et animer** des modèles 3D pour les intégrer dans des projets de jeux vidéo, d'animation ou encore de réalité virtuelle (VR). Et il existe plusieurs techniques de modélisation sa dépend de comment tu sais comment utiliser les outils d blender.

Quel est le rôle de Blender dans notre projet WebXR ?

Dans le cadre de notre projet WebXR, Blender servira à **créer et préparer les œuvres d'art et l'environnement de la galerie virtuelle**. Voici les rôles principaux qu'il jouera :

1. **Modélisation des œuvres d'art**
 - Crédit de **peintures en 3D** (tableaux encadrés) et **sculptures** avec des détails réalistes.
 - Ajout de textures pour donner une apparence authentique aux œuvres.
2. **Conception de l'espace d'exposition**
 - Crédit des murs, sols et lumières pour reproduire une **ambiance immersive** de galerie d'art.
 - Ajout d'éléments décoratifs pour rendre l'environnement plus réaliste.
3. **Optimisation pour le web**
 - Réduction du nombre de **polygones** pour que les objets soient **légers et fluides** sur un navigateur.
 - Exportation des modèles en **GLTF/GLB**, un format compatible avec WebXR et A-Frame/Three.js.
4. **Animations et interactions**
 - Ajout d'effets comme une **mise en valeur des œuvres** lorsqu'un utilisateur s'approche.
 - Intégration d'animations simples pour rendre l'expérience plus dynamique.

Je ne pourrai pas dire que j'ai respecter tout ces étapes que le temps est peu et je ne maîtrise pas beaucoup de chose avec blender. Mais néanmoins j'ai faire des étapes et d'autres non.

J'ai fait la **modélisation** de certains **œuvres d'art** comme (les chaussures des hommes et des femmes, une table de jeux (dame), le chien avec chapeau debout, l'épée, ...etc). J'ai également fait un bâtiment que j'ai nommé musée et c'est en rentrant dedans qu'on pourra voir les œuvres d'arts. J'ai fait l'exportation de ce fichier en. glb qui est l'extension adapté pour le WebXR.

Au cours de cette modélisation pour aller vite j'ai utilise **Archimesh** pour faire le musé. **Archimesh** est un **addon** (extension) de Blender conçu pour **faciliter la création d'éléments architecturaux** en 3D. Il permet de générer rapidement des structures comme des **murs, fenêtres, portes, escaliers et meubles**, ce qui est très utile pour la modélisation d'intérieurs et d'extérieurs.

Cet outil est particulièrement apprécié pour les **projets d'architecture, de design d'intérieur ou de réalité virtuelle**, car il permet de créer des environnements détaillés sans avoir à modéliser chaque élément à la main.

Ensute j'ai utilisé aussi BlenderKit pour les objets d'art. **BlenderKit** est un **addon** (extension) aussi pour Blender qui permet d'accéder à une **bibliothèque en ligne de modèles 3D, matériaux, brosses et HDRI** directement depuis Blender. Il offre des ressources gratuites et payantes que les utilisateurs peuvent **télécharger et insérer rapidement** dans leurs projets. **BlenderKit** est particulièrement utile pour gagner du temps en modélisation, car il permet d'ajouter des objets déjà prêts et bien optimisés sans devoir tout créer à partir de zéro.

Fonctionnalités clés

- ✓ **Téléchargement rapide** de modèles 3D (meubles, sculptures, objets divers).
- ✓ **Matériaux réalistes** pour améliorer le rendu des surfaces.
- ✓ **HDRI** pour un éclairage réaliste.

Utilité pour WebXR

- Ajouter rapidement des œuvres et décors** sans modélisation.
- Utiliser des textures optimisées** pour le web.
- Gagner du temps** avec des modèles prêts à l'emploi.

Mais il y a Certains modèles qui sont un peu lourds pour WebXR.

- **Difficultés rencontrer**

Après l'exportation du fichier nous ne voyons pas les rendus de blender dans le webxr.

Après nous avons vu que ce n'était pas réellement la solution parce que quand on exporte le fichier on ne voit plus le rendu donc nous avons fait la construction du musée avec le A-Frame. Nous sommes partis télécharger des œuvres d'art sur internet et nous avons faire la description en audio de chaque œuvre

II- **Le code (généré par Carmel, Jude et Paterne (image et audio))**

Au cours de ce projet nous avons utilisé plusieurs bibliothèques importantes.

Nous avons :

A-Frame (pour la VR)

```
<script src="https://aframe.io/releases/1.4.2/aframe.min.js"></script>
```

C'est le framework principal qui permet d'afficher et d'interagir avec des objets en **3D** dans le navigateur.

📌 **A-Frame Environment Component** (*pour les décors*)

```
<script src="https://unpkg.com/aframe-environment-component@1.3.3/dist/aframe-environment-component.min.js"></script>
```

Cette bibliothèque ajoute **des environnements prêts à l'emploi** comme une galerie, un désert, une forêt, etc.

A-Frame Extras (*pour les animations et interactions*)

```
<script src="https://cdn.jsdelivr.net/gh/donmccurdy/aframe-extras@v6.1.1/dist/aframe-extras.min.js"></script>
```

Ajoute **des animations avancées et des contrôleurs de mouvement** pour l'expérience.

Howler.js (*pour le son*)

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/howler/2.2.3/howler.min.js"></script>
```

- Permet de **gérer le son** et de jouer de la musique ou des effets sonores dans la galerie.

Feuille de Style (CSS)

Cette partie définit **l'apparence** de la page.

Empêcher le défilement

```
body {  
    margin: 0;  
    padding: 0;  
    overflow: hidden;  
}
```

- Supprime les marges et le défilement** pour que l'expérience soit **en plein écran**.

Superposition d'accueil (overlay)

```
.overlay { position: fixed;  
    top: 0;  
    left: 0;  
    right: 0;  
    bottom: 0;  
    background-color: rgba(0, 0, 0,
```

- ❑ Superpose un écran noir semi-transparent au début.
- ❑ Affiche un **titre et un message de bienvenue** au centre.
 - ❑ opacity: 1; signifie que l'overlay est **visible au départ**, mais pourra disparaître avec une **transition fluide**.

Bouton "Démarrer l'expérience"

```
.start-button {  
    padding: 12px 30px;  
    background: linear-gradient(135deg, #6e8efb, #a777e3);  
    border: none;  
    border-radius: 50px;  
    color: white;  
    font-size: 1.2em;  
    cursor: pointer;  
    transition: transform 0.2s, box-shadow 0.2s;  
}
```

- ❑ Bouton **arrondi et coloré** avec un effet de **dégradé moderne**.
- ❑ Change d'apparence lorsqu'on passe la souris dessus.

Effet au survol du bouton (Hover)

```
.start-button:hover {  
    transform: translateY(-3px);  
    box-shadow: 0 7px 20px rgba(0, 0, 0, 0.3);  
}
```

transform: translateY(-3px) → Quand on passe la souris sur le bouton, il monte légèrement pour donner un effet dynamique.

box-shadow: 0 7px 20px rgba(0, 0, 0, 0.3); → Ajoute une ombre plus marquée pour un effet de relief.

Résultat : Le bouton paraît réactif et élégant quand l'utilisateur le survole.

Bloc d'Instructions

```
.instructions {  
    position: fixed;  
    bottom: 20px;  
    left: 20px;  
    color: white;  
    background-color: rgba(0, 0, 0, 0.5);  
    padding: 10px 15px;  
    border-radius: 5px;  
    font-family: 'Segoe UI', Tahoma, Geneva,  
    Verdana, sans-serif;  
    z-index: 100;  
    pointer-events: none;  
}
```

Explication :

Ce bloc affiche des instructions fixes en bas à gauche de l'écran.

position: fixed; → Il reste toujours visible, même si l'utilisateur bouge.

background-color: rgba(0, 0, 0, 0.5); → Un fond semi-transparent pour qu'il reste discret.

pointer-events: none; → Empêche l'utilisateur de cliquer dessus (juste pour information).

Résultat : Un texte explicatif qui reste en bas de l'écran et aide l'utilisateur.

Écran de Chargement

```
.loading {  
    position: fixed;  
    top: 0;  
    left: 0;  
    width: 100%;  
    height: 100%;  
    background-color: #000;  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    z-index: 2000;  
    color: white;  
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana,  
    sans-serif;  
}
```

Explication :

- C'est un **écran noir de chargement** qui couvre **toute la page** avant que la galerie ne s'affiche.
- **display: flex; justify-content: center; align-items: center;** → Place le texte ou l'animation de chargement **au centre de l'écran**.
- **z-index: 2000;** → S'assure qu'il est **au-dessus de tout**.

Résultat : Un écran qui indique à l'utilisateur que l'expérience est en train de se charger.

Animation de Chargement (Spinner)

```
.spinner {  
    width: 50px;  
    height: 50px;  
    border: 5px solid rgba(255, 255, 255, 0.3);  
    border-radius: 50%;  
}
```

- **Cercle de chargement blanc** qui tourne.
- **border: 5px solid rgba(255, 255, 255, 0.3);** → Ajoute un **effet de transparence** pour donner un look plus fluide.
- **border-radius: 50%;** → Le rend **circulaire**.
- **border-top-color: #fff;** → La partie supérieure est blanche, ce qui crée l'illusion de rotation.

Animation "Spin"

```
@keyframes spin { to { transform:  
rotate(360deg); } }
```

Explication :

- Définit une **animation de rotation** pour le cercle de chargement.
- **rotate(360deg);** → Fait tourner l'élément en **une seconde** et recommence **indéfiniment**.

Résultat: Un **petit cercle blanc qui tourne** au centre de l'écran pendant le chargement.

" <script>

```
// Gestion des interactions et fonctionnalités avancées
AFRAME.registerComponent('artwork-interaction', {
  schema: {
    title: {type: 'string', default: 'Sans titre'},
    artist: {type: 'string', default: 'Artiste inconnu'},
    year: {type: 'string', default: 'Date inconnue'},
    description: {type: 'string', default: 'Aucune description disponible.'},
    audioSrc: {type: 'string'}
  },
  init: function() {
    var el = this.el;
    var data = this.data;
    var camera = document.querySelector('#camera');
    var scene = document.querySelector('a-scene');

    // Création de l'interface de zoom/modal avec fond noir complet
    this.modalEl = document.createElement('a-entity');
    this.modalEl.setAttribute('id', 'artwork-modal');
    this.modalEl.setAttribute('visible', false);

    // Fond complètement noir qui occupe tout le champ de vision
    var bgPanel = document.createElement('a-plane');
    bgPanel.setAttribute('color', '#000');
    bgPanel.setAttribute('opacity', '1'); // 100% opaque
    bgPanel.setAttribute('width', '20'); // Plus grand pour couvrir tout
    bgPanel.setAttribute('height', '15'); // Plus grand pour couvrir tout
    bgPanel.setAttribute('position', '0 0 -0.1');
    this.modalEl.appendChild(bgPanel);
  }
});
```

```
// Image agrandie (plus grande pour un vrai zoom)
var artwork = document.createElement('a-image');
artwork.setAttribute('src', el.getAttribute('src'));
artwork.setAttribute('width', '10'); // Beaucoup plus grand qu'avant
artwork.setAttribute('height', '8'); // Beaucoup plus grand qu'avant
artwork.setAttribute('position', '0 0.25 0');
this.modalEl.appendChild(artwork);

// Texte d'information
var textEl = document.createElement('a-entity');
textEl.setAttribute('text', {
  value: data.title + '\n' + data.artist + ', ' + data.year + '\n\n' + data.description,
  color: 'white',
  align: 'center',
  width: 6
});
textEl.setAttribute('position', '0 -4.5 0'); // Plus bas pour s'adapter à la taille
this.modalEl.appendChild(textEl);

// Bouton de fermeture plus visible
var closeBtn = document.createElement('a-entity');
closeBtn.setAttribute('geometry', {
  primitive: 'plane',
  width: 0.8,
  height: 0.8
});
closeBtn.setAttribute('material', {
  color: '#ff5555',
  opacity: 0.9
});
```

```
closeBtn.setAttribute('text', {  
    value: 'X',  
    color: 'white',  
    align: 'center',  
    width: 4  
});  
  
closeBtn.setAttribute('position', '4.9 4 0');  
closeBtn.setAttribute('class', 'clickable');  
closeBtn.addEventListener('click', this.closeModal.bind(this));  
this.modalEl.appendChild(closeBtn);  
  
scene.appendChild(this.modalEl);  
  
// Initialisation audio  
if (data.audioSrc) {  
    this.audio = new Howl({  
        src: [data.audioSrc],  
        html5: true  
    });  
}  
  
// Ajout effet de survol  
el.addEventListener('mouseenter', function() {  
    el.setAttribute('scale', '1.05 1.05 1.05');  
    el.setAttribute('animation', {  
        property: 'components.material.material.color',  
        type: 'color',  
        to: '#fff',  
        dur: 300  
    });  
});
```

});

```
el.addEventListener('mouseleave', function() {  
    el.setAttribute('scale', '1 1 1');  
    el.setAttribute('animation', {  
        property: 'components.material.material.color',  
        type: 'color',  
        to: '#b8b8b8',  
        dur: 300  
    });  
});
```

Ce composant **artwork-interaction** ajoute des interactions avancées à une œuvre d'art affichée dans une scène VR en A-Frame. Il permet notamment :

- Un **effet de survol** sur l'œuvre.
- Un **zoom/modal** pour voir l'image en grand avec des détails (titre, artiste, description...).
- Un **bouton de fermeture** pour quitter le mode zoom.
- Une **lecture audio** optionnelle si une source est fournie.

➤ Principales Fonctionnalités

- **Définition des Propriétés**

Le composant prend plusieurs paramètres (**title**, **artist**, **year**, **description**, **audioSrc**) pour personnaliser chaque œuvre d'art.

- **Création d'un "Modal" (Zoom sur l'Œuvre)**

Quand on clique sur une œuvre :

- Une **fenêtre modale** s'ouvre avec :
 - Un **fond noir** pour masquer le reste.
 - **L'image en grand** au centre.
 - Une **description en texte** en dessous.
 - Un **bouton de fermeture** en haut à droite.
- **Effet de Survol**

Quand l'utilisateur pointe une œuvre :

- L'image **s'agrandit légèrement** (scale).
- La couleur de l'image **change progressivement** pour donner un effet lumineux.
- **Ajout d'un Son Optionnel**

Si une **source audio** est fournie, le code initialise un lecteur (**Howl** de la bibliothèque **Howler.js**) pour jouer une narration ou un son lié à l'œuvre.

- **Ce Qui se Passe Concrètement**

Quand un utilisateur **regarde** une œuvre : Elle change de couleur et grossit légèrement.

Quand un utilisateur **clique** sur une œuvre : Une fenêtre s'ouvre en **grand format** avec son titre, l'artiste, et une description.

Si un **fichier audio** est lié : Une narration peut être jouée en même temps.

Quand il clique sur le **bouton "X"** : La fenêtre **se ferme** et on revient à la scène normale.

```
// Ouverture du modal au clic
el.addEventListener('click', this.openModal.bind(this));
},

openModal: function() {
    // Sauvegarder position/rotation actuelle
    var camera = document.querySelector('#camera');
    this.originalPosition = camera.getAttribute('position').clone();
    this.originalRotation = camera.getAttribute('rotation').clone();

    // Positionner modal directement devant caméra
    var cameraDirection = new THREE.Vector3(0, 0, -1);
    cameraDirection.applyQuaternion(camera.object3D.quaternion);
    var modalPosition = new THREE.Vector3().copy(camera.object3D.position).add(
        cameraDirection.multiplyScalar(3) // Plus proche pour sensation d'immersion
    );
}
```

```
this.modalEl.setAttribute('position', modalPosition);
this.modalEl.setAttribute('rotation', camera.getAttribute('rotation'));
this.modalEl.setAttribute('visible', true);

// Animation de zoom
camera.setAttribute('animation', {
    property: 'position',
    easing: 'easeInOutSine',
    dur: 800,
    to: modalPosition.x + ' ' + modalPosition.y + ' ' + (modalPosition.z + 0.5)
});

// Lire audio
if (this.audio) {
    this.audio.play();
}

// Désactiver contrôles pendant zoom
document.querySelector('#camera').setAttribute('look-controls', 'enabled', false);
document.querySelector('#player').setAttribute('movement-controls',      'enabled',
false);

// Cacher le reste de la galerie (option renforcée)
var allArtworks = document.querySelectorAll('.frame, .clickable');
allArtworks.forEach(function(artwork) {
    if (artwork !== this.el) {
        artwork.setAttribute('visible', false);
    }
}.bind(this));
```

},

```
closeModal: function() {
    this.modalEl.setAttribute('visible', false);

    // Animation de retour
    var camera = document.querySelector('#camera');
    camera.setAttribute('animation', {
        property: 'position',
        easing: 'easeInOutSine',
        dur: 800,
        to: this.originalPosition.x + ' ' + this.originalPosition.y + ' ' + this.originalPosition.z
    });

    // Réactiver contrôles après animation
    setTimeout(function() {
        document.querySelector('#camera').setAttribute('look-controls', 'enabled', true);
        document.querySelector('#player').setAttribute('movement-controls', 'enabled', true);
    }, 800);

    // Rendre à nouveau visible le reste de la galerie
    var allArtworks = document.querySelectorAll('.frame, .clickable');
    allArtworks.forEach(function(artwork) {
        artwork.setAttribute('visible', true);
    });
}, 800);

// Arrêter audio
if (this.audio && this.audio.playing()) {
    this.audio.stop();
```

```
    }  
}  
});
```

```
// Gestionnaire de chargement des ressources  
window.addEventListener('load', function() {  
    var scene = document.querySelector('a-scene');  
    var loadingEl = document.querySelector('.loading');  
  
    if (scene.hasLoaded) {  
        loadingEl.style.display = 'none';  
    } else {  
        scene.addEventListener('loaded', function() {  
            loadingEl.style.display = 'none';  
        });  
    }  
  
    document.getElementById('start-btn').addEventListener('click', function() {  
        document.querySelector('.overlay').style.opacity = '0';  
        setTimeout(function() {  
            document.querySelector('.overlay').style.display = 'none';  
        }, 500);  
    });  
});  
</script>  
</head>  
  
<body>  
<!-- Écran de chargement -->
```

```
<div class="loading">  
  <div class="spinner"></div>  
</div>  
  
<!-- Écran de bienvenue -->  
  
<div class="overlay">  
  <h1>Bienvenue dans la Galerie d'Art Virtuelle</h1>  
  <p>Explorez notre collection d'œuvres d'art dans un espace immersif. Naviguez librement, approchez-vous des œuvres et cliquez dessus pour en découvrir plus et entendre leurs descriptions.</p>  
  <button id="start-btn" class="start-button">Commencer l'expérience</button>  
</div>  
  
<!-- Instructions sur l'interface -->  
  
<div class="instructions">  
  <p>Utilisez ZQSD/WASD pour vous déplacer | Cliquez sur une œuvre pour zoomer | ESC pour revenir</p>  
</div>  
  
<a-scene loading-screen="enabled: false">  
  <!-- Assets préchargées -->  
  <a-assets>  
    <!-- Textures de haute qualité -->  
      
      
    
```

```
  
  
<!-- Images d'œuvres d'art de qualité muséale -->  
  
  
  
  
  
  
  
  
  
  
  
  
  
<!-- Fichiers audio pour les descriptions -->  
<audio id="audio1" src="assets/audio/1.aac" preload="auto"></audio>  
<audio id="audio2" src="assets/audio/2.aac" preload="auto"></audio>  
<audio id="audio3" src="assets/audio/3.aac" preload="auto"></audio>  
<audio id="audio4" src="assets/audio/4.aac" preload="auto"></audio>  
<audio id="audio5" src="assets/audio/5.aac" preload="auto"></audio>  
<audio id="audio6" src="assets/audio/6.aac" preload="auto"></audio>  
<audio id="audio7" src="assets/audio/7.aac" preload="auto"></audio>  
<audio id="audio8" src="assets/audio/8.aac" preload="auto"></audio>  
<audio id="audio9" src="assets/audio/9.aac" preload="auto"></audio>  
<audio id="audio10" src="assets/audio/10.aac" preload="auto"></audio>  
<audio id="audio11" src="assets/audio/11.aac" preload="auto"></audio>
```

<audio id="audio12" src="assets/audio/12.aac" preload="auto"></audio>

Ouverture du Modal au Clic sur l'Œuvre

Le code commence par écouter un **clic** sur l'œuvre (el.addEventListener('click', this.openModal.bind(this));). Lorsqu'un utilisateur clique sur l'œuvre :

- Il appelle la fonction **openModal** qui gère l'animation de zoom et l'affichage des informations détaillées.
-

Fonction openModal (Affichage du Modal Zoom)

1. **Sauvegarde des positions actuelles** : La position et la rotation de la caméra sont enregistrées pour pouvoir revenir à la même position après le zoom.
2. **Calcul de la position du Modal** : Le **modal** (fenêtre pop-up) est placé directement **devant la caméra**, à 3 unités de distance, pour créer une sensation d'immersion.
3. **Déplacement de la caméra** : La caméra se déplace progressivement vers le modal, avec une animation de zoom de 800 ms.
4. **Lecture audio** : Si un fichier audio est lié à l'œuvre, il commence à jouer, offrant une description sonore de l'œuvre.
5. **Désactivation des contrôles de la caméra** : Les **contrôles de mouvement** sont désactivés pendant l'animation de zoom pour empêcher l'utilisateur de bouger accidentellement pendant cette phase.
6. **Cacher les autres œuvres** : Toutes les autres œuvres de la galerie sont **cachées** temporairement pour focaliser l'attention sur l'œuvre sélectionnée.

Fonction closeModal (Fermeture du Modal)

1. **Cacher le Modal** : La fenêtre modale est rendue invisible (this.modalEl.setAttribute('visible', false);).
2. **Retour à la position initiale** : La caméra revient à sa position initiale, avec une animation fluide de 800 ms.
3. **Réactivation des contrôles** : Les contrôles de la caméra et du mouvement sont **réactivés** après l'animation, permettant à l'utilisateur de reprendre le contrôle.
4. **Rendre visibles les autres œuvres** : Toutes les œuvres cachées précédemment sont **rendre visibles** à nouveau.
5. **Arrêter l'audio** : Si un audio était en cours de lecture, il est **arrêté** à la fermeture du modal.

Gestion du Chargement de la Scène

1. **Écran de chargement** : Lorsque la scène VR est en train de se charger, un **écran de chargement** est affiché avec un spinner (animation de chargement).
2. **Cache de l'écran de chargement** : Dès que la scène est **complètement chargée**, l'écran de chargement disparaît.

3. **Affichage de l'écran de bienvenue** : Un écran d'accueil (overlay) s'affiche, avec un bouton pour **commencer** l'expérience. Lorsque l'utilisateur clique sur "Commencer", cet écran d'accueil s'estompe.

Préchargement des Ressources

Le code précharge plusieurs ressources pour l'expérience VR :

- **Textures** : Différentes images pour le sol, les murs, le plafond, et les cadres des œuvres.
- **Œuvres d'art** : Images de haute qualité des œuvres à afficher.
- **Audio** : Des fichiers audio associés à chaque œuvre, préchargés pour être lus lors de l'interaction avec l'œuvre.

En résumé on peut dire que :

- ❑ L'utilisateur peut cliquer sur une œuvre pour la voir en **zoom** avec une animation et entendre une description audio.
- ❑ Le reste de la galerie est caché pour mieux se concentrer sur l'œuvre sélectionnée.
- ❑ Lors de la fermeture du modal, tout revient à la normale avec une **animation fluide**.

Sons d'ambiance :

```
<audio id="ambient-sound" src="https://cdn.glitch.global/29e07830-2317-4c15-a044-135e73c7f814/museum-ambience.mp3?v=1643783746372"
preload="auto" loop></audio>
```

Description : Ce code définit un fichier audio qui sera utilisé comme ambiance sonore dans la scène. Le fichier audio est chargé depuis une URL externe et est configuré pour se répéter en boucle. L'attribut **preload="auto"** indique que le fichier audio sera préchargé pour une lecture rapide lorsqu'il sera nécessaire.

Système de caméra amélioré :

```
<a-entity id="player" movement-controls="speed: 0.15; camera: #camera;
constrainToNavMesh: false"> <a-entity id="camera" camera position="0 1.6 0" look-
controls="pointerLockEnabled: true"> <a-entity cursor="rayOrigin: mouse; fuse: false"
raycaster="objects: .clickable" position="0 0 -1" geometry="primitive: ring; radiusInner:
0.01; radiusOuter: 0.015" material="color: #ffffff; shader: flat; opacity: 0.8"></a-entity>
</a-entity> </a-entity>
```

Description : Cette section définit un joueur avec un contrôleur de mouvement pour déplacer la caméra dans l'environnement. La caméra est placée à une hauteur de 1.6 unités et est contrôlée par les mouvements de la souris (pointerLockEnabled: true bloque

la souris pour une meilleure immersion). Un curseur est également ajouté pour interagir avec les objets cliquables de la scène, comme des œuvres d'art.

Ambiance sonore

```
<a-entity sound="src: #ambient-sound; autoplay: true; loop: true; volume: 0.3"></a-entity>
```

Description : Ce composant ajoute l'ambiance sonore à la scène en utilisant l'élément `<audio>` défini précédemment. Il démarre immédiatement (`autoplay: true`), est en boucle, et le volume est réglé à 0.3.

Éclairage sophistiqué

```
<a-entity light="type: ambient; color: #bbb; intensity: 0.5"></a-entity>
<a-entity light="type: directional; color: #fff; intensity: 0.7; castShadow: true" position="-1 4 3"></a-entity>
<a-entity light="type: spot; color: #fff; intensity: 0.5; angle: 45; penumbra: 0.3; castShadow: true" position="1 4 -3" rotation="-45 0 0"></a-entity>
```

•

Description : Ces éléments ajoutent plusieurs types d'éclairage à la scène :

- **Lumière ambiante** : une lumière douce qui éclaire uniformément la scène (`intensity: 0.5`).
- **Lumière directionnelle** : imite la lumière du soleil, projetant des ombres.
- **Lumière spot** : fournit un éclairage directionnel plus concentré, idéal pour illuminer des objets spécifiques comme des œuvres d'art.

Structure de la galerie et sol avec texture

```
<a-entity environment="preset: default; ground: flat;groundColor: #222;groundColor2: #333; dressing: none; skyType: gradient; skyColor: #1a1a1a; horizonColor: #333"></a-entity> <a-plane position="0 0 0" rotation="-90 0 0" width="30" height="30" material="src: #floor-texture; repeat: 15 15; normalMap: #floor-normal; metalness: 0.2; roughness: 0.1"></a-plane>
```

Description :

- La première entité crée un environnement avec un sol plat et un ciel dégradé. Les couleurs sont configurées pour donner un effet de profondeur.
- Le second élément est un plan représentant le sol, qui est texturé avec un motif de marbre (répété 15 fois en largeur et en hauteur). Il utilise également une carte de relief (`normalMap`) pour ajouter des détails de texture.

Murs avec des œuvres d'art :

Les murs de la galerie sont construits avec des boîtes <a-box>, qui utilisent des textures pour simuler des surfaces sophistiquées. Ensuite, des images représentant des œuvres d'art sont placées sur les murs, avec des informations interactives associées à chaque tableau, telles que le titre, l'artiste, l'année, et une description. Ces éléments sont interactifs, permettant aux utilisateurs de cliquer sur les œuvres pour obtenir des détails supplémentaires, comme de l'audio ou des informations supplémentaires.

Chaque œuvre d'art est placée dans un cadre défini par une boîte (box) avec une texture de cadre, et l'interaction des utilisateurs est gérée par des attributs comme artwork-interaction.

Le reste du code à partir de la ligne 543 à 666

② Indications de navigation :

- Trois entités (<a-entity>) sont créées à des positions spécifiques pour indiquer la direction de navigation (gauche, droite, haut).
- Chaque entité contient un cercle (<a-circle>) avec une flèche (\leftarrow , \rightarrow , \uparrow) pour aider l'utilisateur à naviguer dans l'espace.
- Le cercle a une légère opacité et une couleur sombre, et les flèches sont centrées et blanches pour la visibilité.

② Système de détection pour interaction améliorée :

- Le code utilise la bibliothèque **Howl** pour jouer un son d'ambiance en boucle dans la scène (un bruit de musée).
- Il y a un gestionnaire d'interaction pour afficher un message d'aide à l'utilisateur (invitation à approcher les œuvres et à cliquer pour plus d'informations).
- Un setTimeout affiche le message d'aide après 8 secondes et le supprime après 5 secondes.
- Une logique de performance est mise en place avec un contrôle de la fréquence des mises à jour de l'animation à 60 images par seconde pour une performance optimale.

② Gestion des erreurs et résilience :

- **Composant "error-handler"** : Si une ressource (comme une image ou un modèle 3D) échoue à se charger, un message d'erreur apparaît en haut à droite pour informer l'utilisateur.
- **Gestion des erreurs JavaScript** : Si une erreur JavaScript se produit, elle est consignée dans la console pour le développeur.
- Ces mécanismes visent à informer l'utilisateur en cas de problème tout en essayant de maintenir une expérience stable.

Conclusion

En conclusion, ce projet de galerie d'art virtuelle en 3D permet de plonger les utilisateurs dans une expérience immersive accessible directement depuis un navigateur. En utilisant la technologie WebXR et l'outil A-Frame, nous avons créé un environnement interactif où les utilisateurs peuvent se déplacer librement, explorer des œuvres d'art variées et interagir avec elles de manière intuitive. L'intégration de fonctionnalités telles que la téléportation, l'affichage d'informations contextuelles et les descriptions audio améliore encore l'expérience de l'utilisateur. Ce travail pratique nous a non seulement permis de comprendre les fondamentaux du développement WebXR, mais il a également mis en lumière le potentiel des technologies immersives pour offrir des expériences culturelles innovantes et accessibles à un large public.