Projet: conception d'un musée virtuel en BabylonJS

Luis Felipe MONTOYA

Josué SAVY

Professeur :

Éric MAISEL

RV – Réalité Virtuelle

Brest
Juin 2022

Contenido

Table de figures

1 – CONTEXTE

Dans le cadre du projet du module « Réalité Virtuelle », nous devions réaliser un environnement 3D qui puisse permettre à un utilisateur de se déplacer dans la scène et d'interagir avec les différents objets de cette dernière. Pour cela, nous avons utilisé le moteur 3D en temps réel : Babylon.JS. Cet outil nous permet de créer de façon simple plusieurs éléments d'une scène, tel qu'une caméra, des objets ou des animations, avec les nombreuses méthodes déjà implémentées.

Pour la réalisation de notre musée virtuel, nous avons du suivre quelques indications par rapport à la répartition de la surface et la construction du bâtiment comme la présence de deux étages ou une surface de 30 x 30 mètres. Tous les règles à respecter se trouvent dans le document *enonce.pdf*.

Une première étape a donc consisté au design de notre musée et à un choix de la thématique pour pouvoir introduire des tableaux ou des statues en accord avec notre sujet. Notre musée a pour objet de montrer **l'évolution de la Science à travers les différentes cultures de l'Histoire**.

Les *Figures 1* et *2* montrent un premier exemple de design fait, dans lesquels nous pouvons voir les différentes salles et espaces des deux étages. Pour présenter brièvement notre idée initiale, nous avons sur le premier étage :

- Un accueil, en bas à gauche ;
- Un distributeur automatique, représenté par la lettre « M » ;
- Des statues, indiquées par des cercles orange ;
- Une fontaine d'eau, marquée en bleu ;
- Des toilettes, en bas à droite ;
- Un ascenseur, représenté par l'espace avec la croix ;
- Une boutique de souvenirs, à gauche de l'ascenseur ;
- Et trois salles avec des peintures.

Le second étage était composé de :

- Une salle de cinéma, en haut à gauche ;
- Une salle avec une statue dynamique, en haut à droite ;
- Un espace de relax et jeux, en bas à gauche ;
- Et une zone pour prendre un café, représenté par le rectangle à gauche des toilettes.

Ce premier design a été légèrement modifié avec le début de son développement.

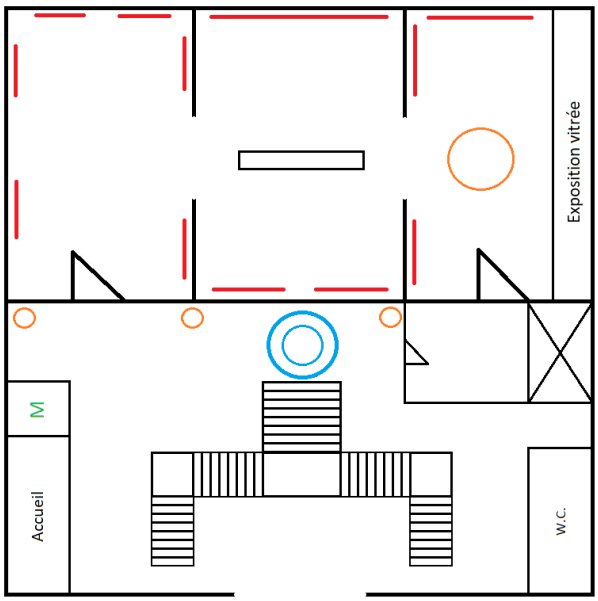

Figure 1. Design initial du premier étage

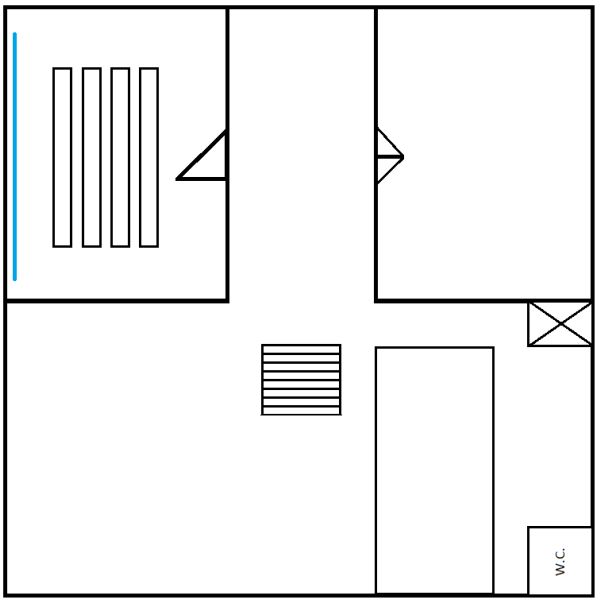

Figure 2. Design initial du second étage

2 – DÉVELOPPEMENT

Structure du musée

Le projet est divisé en six classes, chacune traitant une tâche différente. Dans le fichier *create_structure.js*, nous trouvons toutes la structure du musée : les murs, ici nommés cloisons de chacune des sous-parties du musée. Nous avons également créé des panneaux pour guider l'utilisateur dans son parcours dans la fonction *create_panels* (*Figure 3 et 4).* Pour cela, chaque panneau a été fait sur le site Canva.

```
function create_panels(mat_base) {
    var mat_welcoming_panel = creerMateriauSimple("mat-welcoming_panel", {texture:"assets/textures/panels/welcoming_panel.jpg"}, scene);
    var mat_entry_panel = creerMateriauSimple("mat-entry_panel", {texture:"assets/textures/panels/first_floor_panel.jpeg"}, scene);
    var mat_secondFloor_panel = creerMateriauSimple("mat-secondFloor_panel", {texture:"assets/textures/panels/second_floor_panel.jpeg"}, scene);
    var mat_cinema_panel = creerMateriauSimple("mat-cinema_panel", {texture:"assets/textures/panels/cinema_panel.jpeg"}, scene);
    var mat_room_1_panel = creerMateriauSimple("mat-room_1_panel", {texture:"assets/textures/panels/room_1_panel.jpg"}, scene);

    var welcoming_panel = createPanel("Welcome Panel", {material:mat_welcoming_panel, largeur:1.5}, scene, mat_base);
    welcoming_panel.position.set(18,1.55,-7);
    welcoming_panel.rotation.y = Math.PI / 2 ;

    var panel_1 = createPanel("Panel Entry 1", {material:mat_entry_panel}, scene, mat_base);
    panel_1.position = new BABYLON.Vector3(17, 1.55, 3.5);

    var panel_2 = createPanel("Panel Second Floor", {material:mat_secondFloor_panel}, scene, mat_base);
    panel_2.position = new BABYLON.Vector3(17, 6.55, 15.75);

    var panel_3 = createPanel("Panel Second Floor End", {material:mat_cinema_panel}, scene, mat_base);
    panel_3.position = new BABYLON.Vector3(15, 6.55, 29.5);

    var panel_4 = createPanel("Panel Room 1", {material:mat_room_1_panel}, scene, mat_base);
    panel_4.position = new BABYLON.Vector3(1.75, 1.55, 16.0);
    panel_4.rotation.y = 4*Math.PI/3 ;
}
```

Figure 3. Creation des panneaux

Figure 4. Exemple d'un panneau

Nous avons placé quatre panneaux dans les zones clés pour que l'utilisateur connaisse les différents espaces qu'il peut découvrir.

Dans ce fichier, nous trouvons également le code pour construire les escaliers. Il est important de faire la remarque que la caméra n'est pas capable de monter par les escaliers à cause de la gravité et de la collision avec chaque marche donc nous avons dû mettre un rectangle invisible sur les escaliers de telle sorte que la caméra se déplace sur cet objet au lieu des marches (*Figure 5*).

```
115     var invisibleStair = creerCloison("invisibleStair",{hauteur:12.2, largeur:2.5, epaisseur:0.01, materiau:invisibleMat},scene) ;
116     invisibleStair.rotation.x = 1.05;
117     invisibleStair.position = new BABYLON.Vector3(15.0, -0.8, 1.1);
118     invisibleStair.checkCollisions = true;
```

Figure 5. Code réalisé pour pouvoir monter les escaliers

Création des tableaux

Dans le fichier *create_painting.js*, nous trouvons tous les tableaux affichés sur les trois salles du premier étage. La première salle montre les anciennes cultures qui ont collaboré à l'avancée de la Science (*Figure 6*), dans la deuxième salle nous avons des représentations de moments historiques importants et finalement, la troisième salle nous présente des faits récents comme la découverte des trous noirs ou de la quantique.

Figure 6. Exemple de tableaux

Création des objets

Afin d'éviter perdre beaucoup de temps avec la création de certains objets avec Babylon, voir même avec un outil comme Blender, nous avons choisi d'importer plusieurs objets 3D qui étaient disponibles en ligne. Ce code se trouve dans le fichier *import_objects.js* et utilise la classe AssetsManager de Babylon pour les importer. Au total, vingt objets ont été utilisés. Pour avoir un format commun, les objets étaient sauvegardés sur Blender sur le format « .glb », par la suite, ils étaient placés sur la scène et avaient ces textures ajoutées (*Figure 7*).

```
// ----- BENCH ----- //
var bench = loader.addMeshTask("bench", "", "assets/objects/", "bench_formated.glb");
bench.onSuccess = function(t) {
    t.loadedMeshes.forEach(function(m) {
        m.position.x = 18;
        m.position.z = 27;
        m.scaling = new BABYLON.Vector3(0.16,0.16,0.16);

        if(Object.values(m)[24] == "node1" || Object.values(m)[24] == "node2" ) {
            m.material = new BABYLON.StandardMaterial("mat", scene);
            m.material.diffuseTexture  = new BABYLON.Texture( "assets/textures/railing.jpg",  scene);
        }
        if(Object.values(m)[24] == "node0" || Object.values(m)[24] == "node3" ) {
            m.material = new BABYLON.StandardMaterial("mat", scene);
            m.material.diffuseTexture  = new BABYLON.Texture( "assets/textures/dark_wood.jpg",  scene);
        }
        if(Object.values(m)[24] == "node4" || Object.values(m)[24] == "node5" || Object.values(m)[24] == "node6") {
            m.material = new BABYLON.StandardMaterial("mat", scene);
            m.material.diffuseTexture  = new BABYLON.Texture( "assets/textures/carpet.jpg",  scene);
        }

    });
};
```

Figure 7. Exemple d'import d'objet

Pour certains de ces objets, il a été long de travailler sur les textures sachant qu'il fallait identifier à quel nom correspondait chaque partie de l'objet. C'est le cas de l'humain qui était divisé en plusieurs meshes, correspondant à chaque partie du corps. (*Figure 8*)

Figure 8. Différentes textures de l'humain virtuel

De même, certains objets ont dû être importés deux fois pour représenter la rotation sachant qu'il n'était pas possible de le faire avec la propriété « rotation » présente dans Babylon. Pour cela, la rotation était faite directement sur l'objet sur Blender. C'est le cas des objets « railings » ou de la bibliothèque.

Figure 9. Objets importés pour la zone de relax

Afin de gérer les collisions de la caméra avec les objets importés, il a fallu créer des boîtes invisibles autour de l'objet en question. Ceci est lié au fait que certains objets possèdent énormément de faces donc une collision avec la caméra ralenti la simulation ou bloque la caméra. Ces boîtes invisibles sont créées dans la fonction *addColliders.*

Animations

Les animations présentes dans le musée représentent l'interaction de l'utilisateur avec l'environnement virtuel. Parmi ces animations, nous avons des ouvertures et fermetures de portes, l'activation de lumières sur des tableaux, l'activation de rotation de sphères et le démarrage d'un film dans la salle de cinéma. Ces animations sont présentes dans le fichier *animations.js* et utilisent la classe « actionManager » de Babylon pour pouvoir les intégrer toutes à la scène. Pour qu'une animation soit déclenchée, il faut tout d'abord que la caméra perçoive une collision avec une boîte invisible à l'aide de la fonction « OnIntersectionEnterTrigger ».

Deux statues dynamiques sont présentes dans le musée, une dans le premier étage qui symbolise l'orbite de satellites par rapport à une planète ; et l'autre dans le second étage, qui représente les sphères de Newton. (*Figure 10 et 11*)

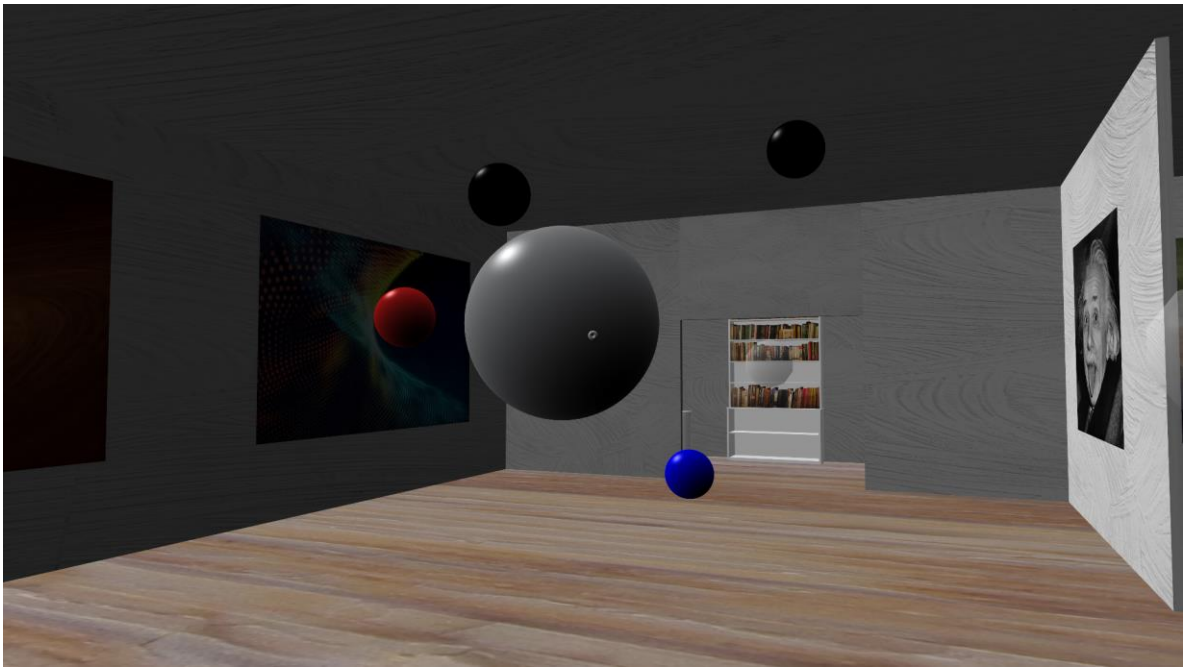

Figure 10. Orbite de satellites

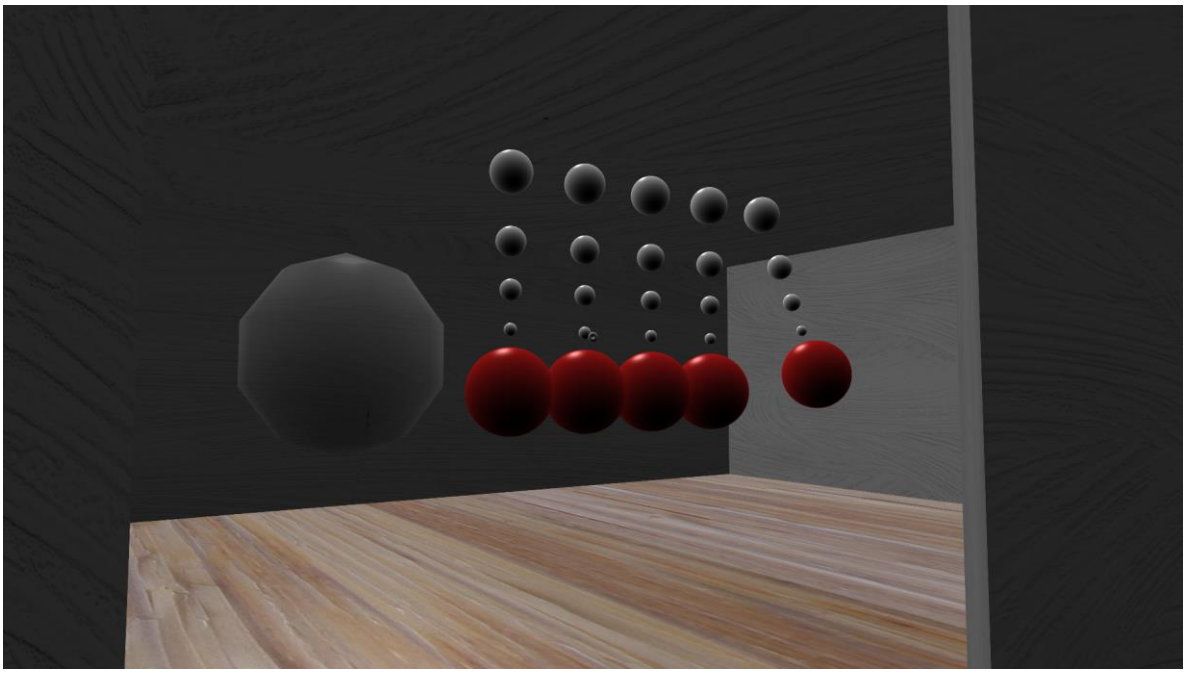

Figure 11. Sphères de Newton

Nous retrouvons également une fonction permettant à l'utilisateur de lancer ou de stopper un film dans la salle de cinéma. Pour la lancer, il suffit d'appuyer sur la barre d'espace du clavier et l'animation est déclenchée. Cette animation a été possible grâce à l'utilisation d'une texture vidéo, mise sur l'objet importé de l'écran vidéoprojecteur. (*Figure 12 et 13*)

```
function playVideo() {
    // Implementation of the video texture
    var video_plane = BABYLON.MeshBuilder.CreatePlane("video_texture", {height:2.33, width:6}, scene);
    video_plane.position = new BABYLON.Vector3(0.65,7.43,23);
    video_plane.rotation.y = -Math.PI / 2 ;
    var video_planeMat = new BABYLON.StandardMaterial("m", scene);
    var video_planeTex = new BABYLON.VideoTexture("vidtex","assets/textures/video/movie_texture.mp4", scene);
    video_planeMat.diffuseTexture = video_planeTex;
    video_planeTex.video.pause();
    video_planeMat.roughness = 1;
    video_planeMat.emissiveColor = new BABYLON.Color3.White();
    video_plane.material = video_planeMat;

    scene.onKeyboardObservable.add((evt) => {
        switch (evt.type) {
            case BABYLON.KeyboardEventTypes.KEYDOWN:
                switch (evt.event.key) {
                    case " ":
                        if(video_planeTex.video.paused) video_planeTex.video.play();
                        else video_planeTex.video.pause();
                        console.log(video_planeTex.video.paused? "paused":"playing");
                    break
                }
            break;
        }
    });
}
```

Figure 12. Code pour lancer la vidéo

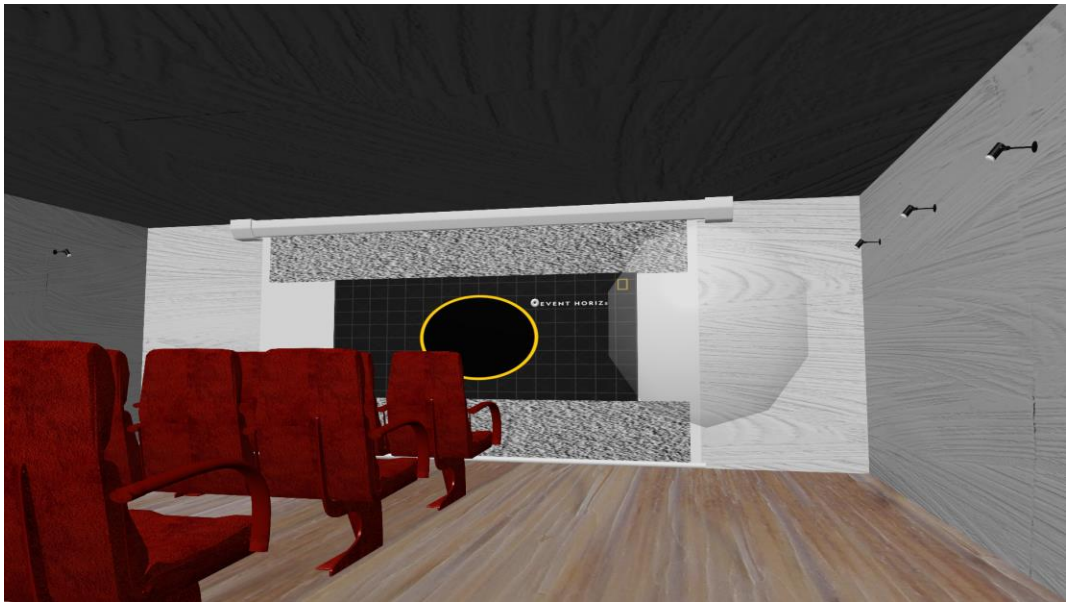

Figure 13. Salle de cinéma

Des animations ont également été développées pour les portes (celle d'entrée et celles des toilettes du premier étage). Nous avons choisi de représenter les deux types d'ouvertures possibles, un déplacement latéral et une ouverture sous un axe de gravitation (*Figure 14)*.

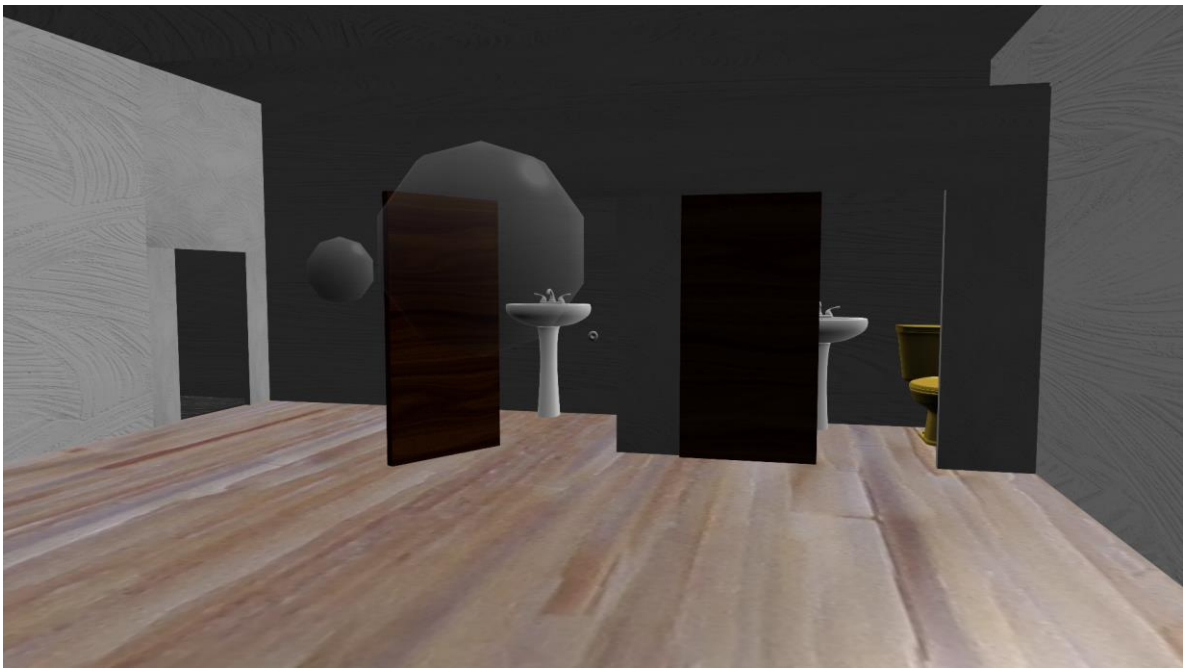

Figure 14. Différentes ouvertures de portes possibles

Points de déplacements

Il est possible de suivre des points pour se déplacer à l'intérieur de la scène : ce sont les checkpoints. Ces checkpoints sont créés dans le fichier *prims.js* mais sont placés dans le fichier *musee.js* (*Figure 15*). Au total, 17 checkpoints ont été créés pour permettre à l'utilisateur de se déplacer sur la scène.

```
function createTeleportationMark() {
    points = [];

    // -- Outside points
    points.push(new BABYLON.Vector3(15, 2.2, -20));
    points.push(new BABYLON.Vector3(15, 20, -40));
    points.push(new BABYLON.Vector3(15, 2.2, -6));

    // -- FirstFloor points
    points.push(new BABYLON.Vector3(15, 2.2, 2));
    points.push(new BABYLON.Vector3(6, 2.2, 7));
    points.push(new BABYLON.Vector3(5.5, 2.2, 22.5));
    points.push(new BABYLON.Vector3(20, 2.2, 22.5));
    points.push(new BABYLON.Vector3(25, 2.2, 13));
    points.push(new BABYLON.Vector3(16, 2.2, 13));
    points.push(new BABYLON.Vector3(22, 2.2, 4));
    points.push(new BABYLON.Vector3(29, 2.2, 9));

    // -- SecondFloor points
    points.push(new BABYLON.Vector3(15, 7.2, 14));
    points.push(new BABYLON.Vector3(9, 7.2, 14));
    points.push(new BABYLON.Vector3(22, 7.2, 14));
    points.push(new BABYLON.Vector3(15, 7.2, 27));
    points.push(new BABYLON.Vector3(10, 7.2, 27));
    points.push(new BABYLON.Vector3(20, 7.2, 27));

    var i = 0;
    points.forEach(function(position){
        checkPoints("checkpoint_" + i, { position: position }, scene);
        i++;
    });
}
```

Figure 15. Fonction de création des checkpoints

Il ne faut pas oublier le checkpoint qui se situe à l'extérieur du musée qui permet d'avoir un point de vue général du projet. Pour le trouver, il suffit de tourner la caméra vers l'arrière, lors de lancer le projet, et pointer vers le haut (*Figure 16 et 17*).

Figure 16. Checkpoint permettant la vue entière du musée

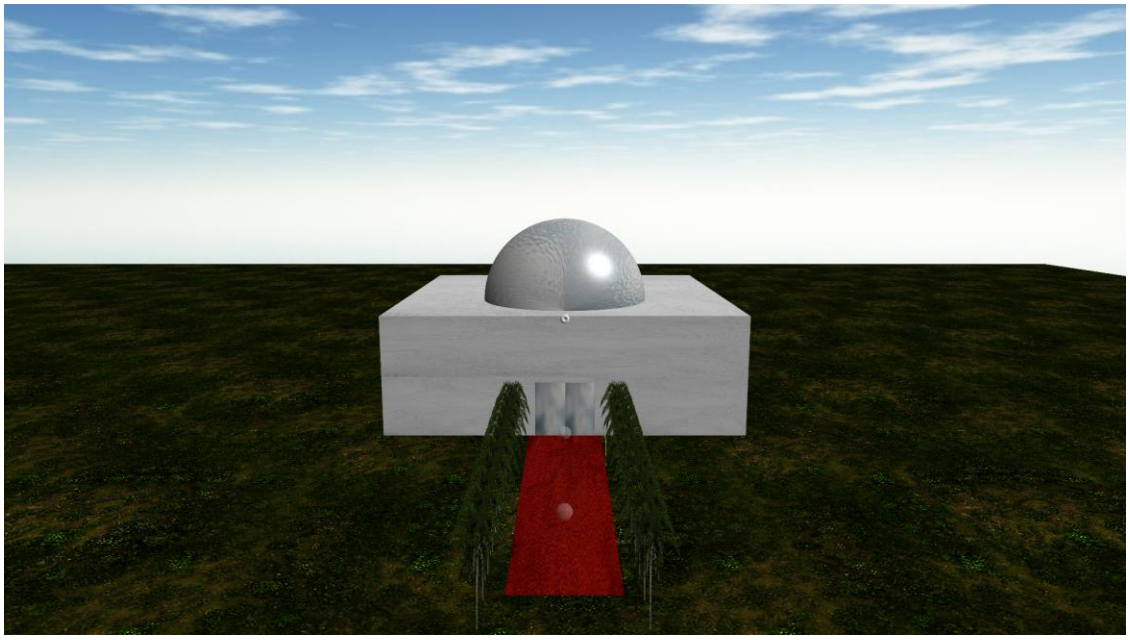

Figure 17. Vue entière du musée

Lors de faire click sur un des checkpoints, l'animation de télétransportation est déclenchée, qui permet donc de replacer la caméra à sa nouvelle position (*Figure 18*).

```
function startTeleportation(camera, x, y, z) {
    new_position = new BABYLON.Vector3(x, y, z);
    const animation = new BABYLON.Animation("teleportation", "position", 60 ,BABYLON.Animation.ANIMATIONTYPE_VECTOR3,BABYLON.Animation.ANIMATIONLOOPMODE_CYCLE);
    keys = [];
    keys.push({
        frame: 0,
        value: camera.position
    });
    keys.push({
        frame: 5,
        value: new_position
    });
    animation.setKeys(keys);
    camera.animations.push(animation);
    scene.beginAnimation(camera, 0, 5, false, speedRatio=0.2);
}
```

Figure 18. Animation pour le déplacement de la caméra

Bonus

Tous les modules sont compilés depuis le fichier *musee.js* et depuis le *index.html*. En effet, comme c'est le module principal de l'application, nous chargeons toutes les principales fonctionnalités depuis ce dernier. Dans le fichier principal, nous avons crée la décoration externe du musée (les arbres et le tapis d'accueil, *Figure 17*). Nous avons créé également le pointeur permettant à l'utilisateur de voir la direction de visualisation de la caméra (*Figure 19*).

```
function createVisualDot() {
    dot = BABYLON.Mesh.CreateTorus("visualDot", 0.015, 0.01, 10, scene, false);
    dot.parent = camera;
    dot.position.z = 2.0;
    dot.rotation.x = Math.PI / 2;
}
```

Figure 19. Fonction de création du pointeur

Dans les *Figures 20 et 21*, nous pouvons observer une vue des deux étages depuis un angle élevé. Nous pouvons ainsi apprécier la distribution des espaces et des salles à l'intérieur de notre musée virtuel.

Par rapport au design initial, certains espaces ont changé : nous nous sommes adaptés aux défis présents lors du développement des éléments de la scène pour ne pas surcharger notre application et réduire son rendement.

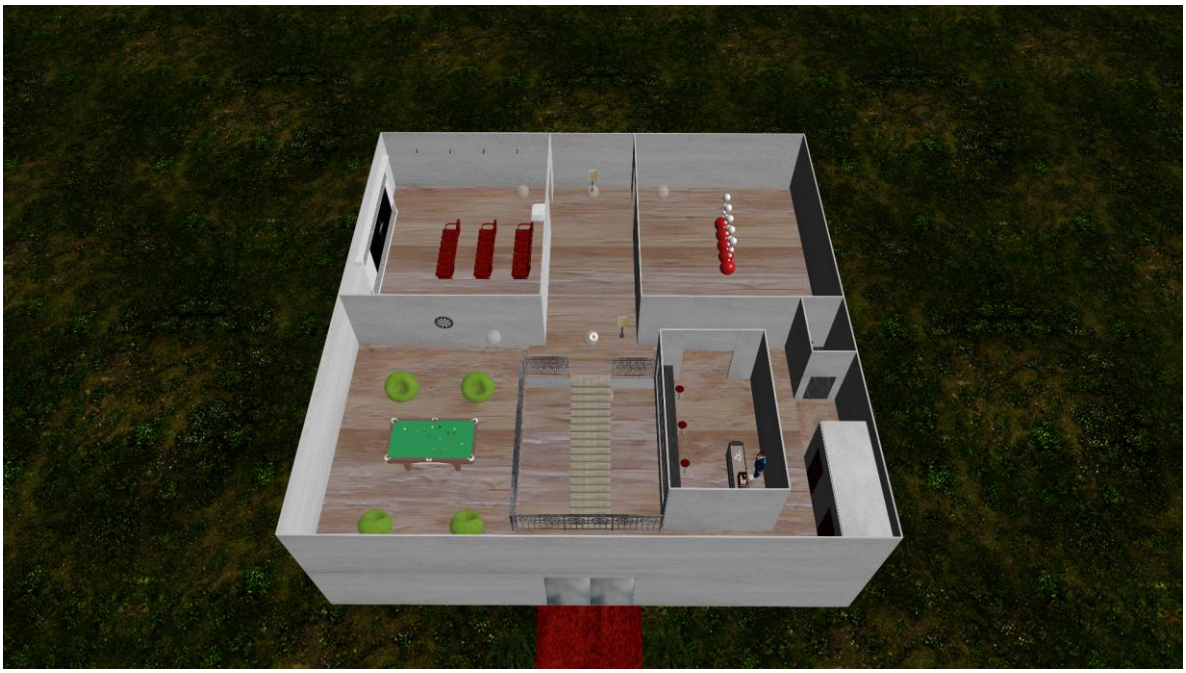

Figure 20. Vue du second étage

Figure 21. Vue du premier étage