Rapport projet musée

Noé Paillet, Tanguy Ohron

Sommaire :

I- Introduction :

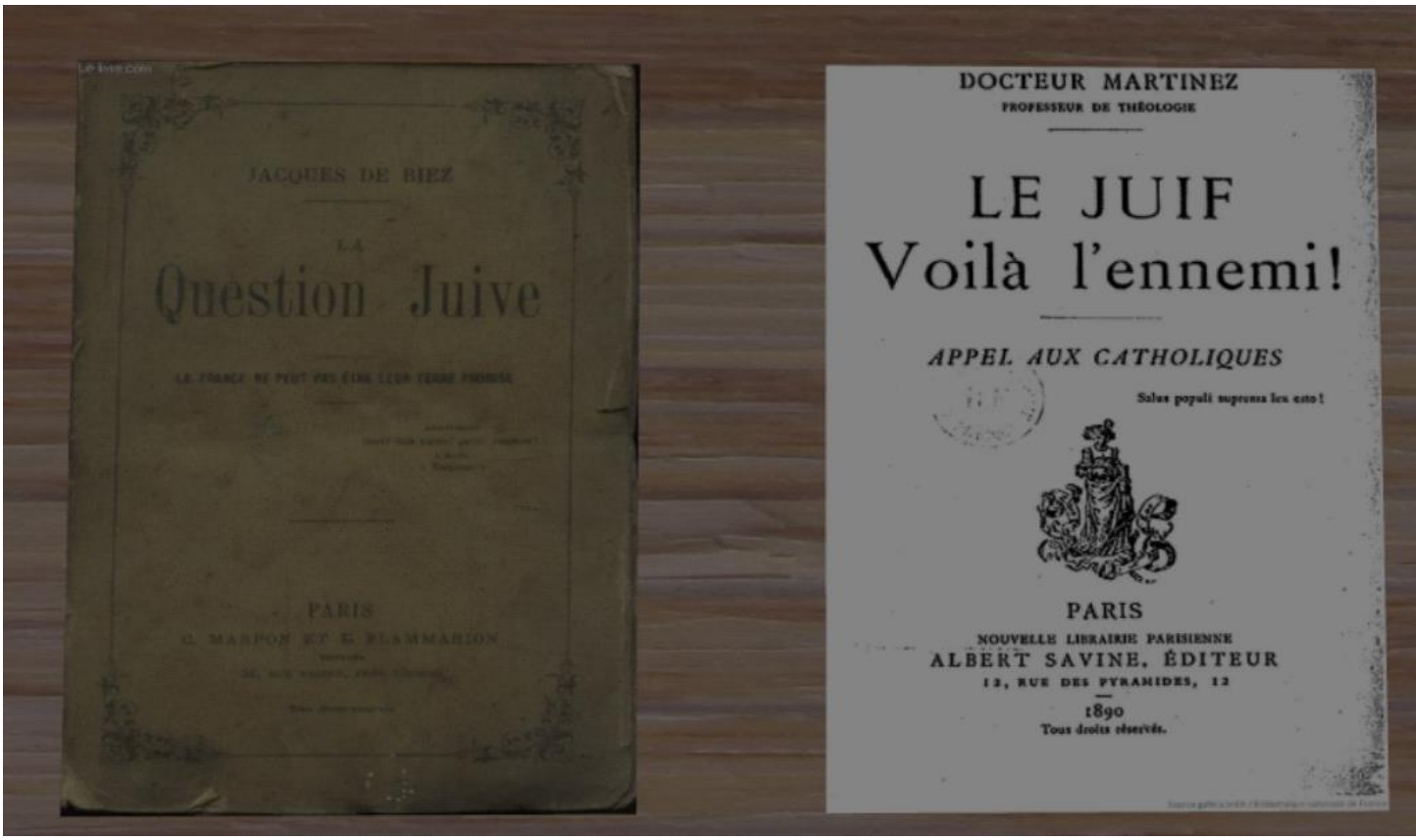
Le sujet du projet est la réalisation d'un musée en 3D basé sur BabylonJs, ayant pour thème la Belle Époque, période de 1880 à 1914, durant laquelle la France connaît d'importants progrès techniques et sociaux. Nous avons décidé de présenter la face glorieuse de cette période, mais aussi ces faces plus sombre, qui ont parfois permis ces progrès.
Le musée se décompose en 5 salles :

La première, le grand hall, présente différents tableaux montrant la vie bourgeoise parisienne, ses luxes, et manigances.

La seconde est une salle de transition. Rien n'y est exposé, elle permet d'accéder aux deux autres salles du rez de chaussé, ainsi qu'à la mezzanine.

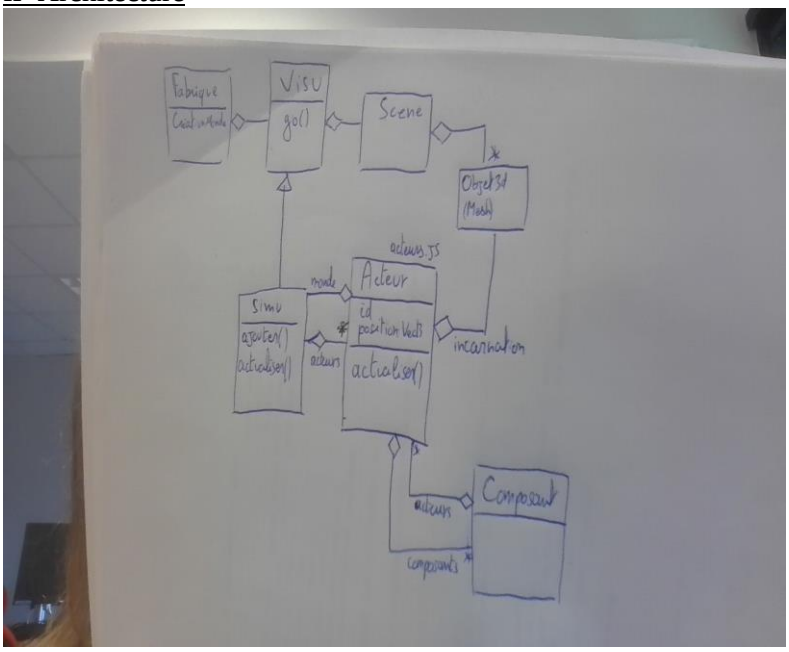
A droite de celle ci se trouve la salle des colonies. Elle présente des œuvres montrant le pillage et les répressions coloniales Française de l'époque.

Et à sa gauche, la salle sur l'antisémitisme présente différentes œuvres montrant la montée en puissance et la normalisation de l'antisémitisme en France, ayant eu les conséquences que nous connaissons.



Enfin, la mezzanine montre l'extraction des ressources, notamment le pétrole, qui ont en partie permis les progrès de l'époque.

II- Architecture



Pour ce projet, nous avons décidé de réaliser l'architecture ci-dessus, présentée en cours. Dans celle ci-chaque objet est un acteur, a qui un ou plusieurs composants sont associés. Les composants peuvent appeler la création un objet 3D ou 2D via les primitives, ou appliquer une rotation, une translation, ou encore une attache à un autre acteur. La classe Simu, elle, permet la création de ces acteurs, et la classe Visu s'occupe de la création de la caméra. Enfin, la classe Fabrique (World02 dans notre cas), appelle ces différents éléments.

Cette architecture permet une grande modularité, et une bonne division des différentes taches, afin que le fichier World02 ne soit pas trop surchargé. Cependant, elle nous à parfois posé des difficultés pour implémenter de nouveaux éléments, souvent à cause du fait que le mesh des Acteurs ne soit pas accessibles dans World02.

III- Fonctionnalités

1) Les tableaux

Pour créer les tableaux, nous avons créé un fichier json (json_files/monde.json) dans lequel sont décrits tous les tableaux.
Une fonction asynchrone présente dans world nous permet de lire le json et creer les tableaux avec les informations voulue :

```
async function init() {
    try {
        var fileData = await MTH.loadJSON("./json_files/monde.json");
        for (var cle in fileData) {
            const elt = fileData[cle];
            const typeActor = ACTORS[elt.type];
            const dataActor = elt.data;
            const composants = elt.composants;
            const actor = this.createActor(cle, typeActor, dataActor);
            for (var comp of composants) {
                const typeComp = COMPS[comp.type];
                const dataComp = comp.data;
                actor.add(typeComp, dataComp);
                if(dataComp.parent){
```

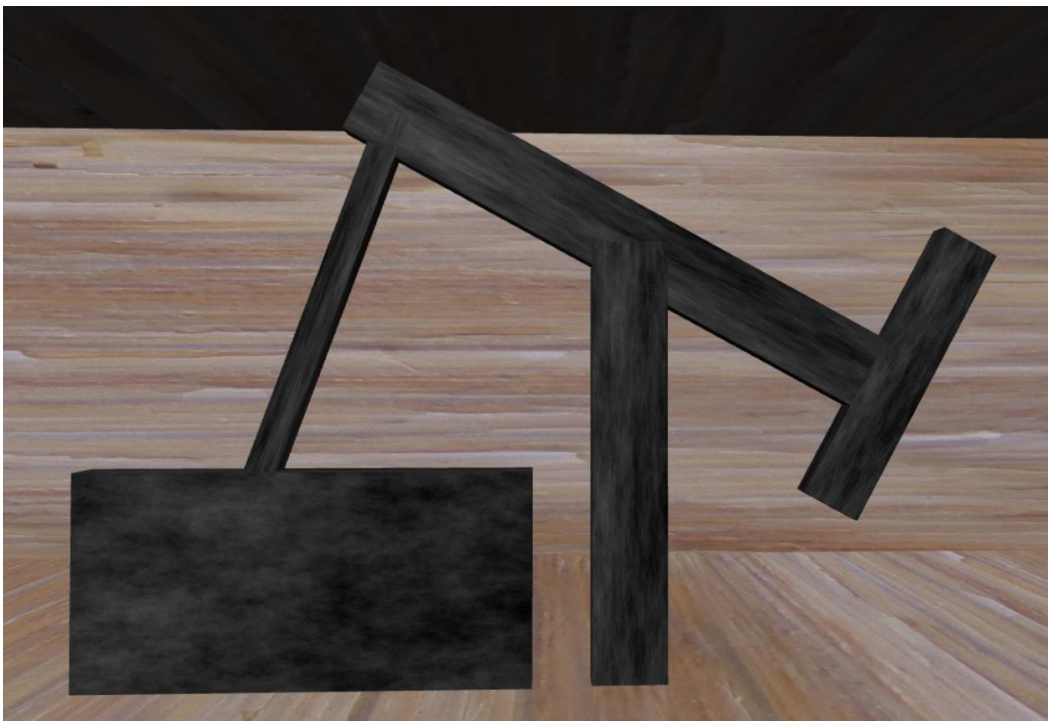
```
        actor.add(COMPS.anchoredTo,{parent:dataComp.parent});

        actor.add(COMPS.position,{x:dataComp.position.x, y:dataComp.position.y,
z:dataComp.position.z});
                                                }
                                                if(dataComp.rotation==true){
                                                        actor.add(COMPS.rotation,{y:Math.PI});
                                                }
                                        }
                                }
                        } catch (error) {
                                console.error('Erreur lors du chargement du JSON:', error);
                        }
                }

                // Utilisation de la fonction fléchée pour préserver le contexte `this`
                const initFunction = async () => {
                        await init.call(this);
                };
```

La fonction MTH.loadJSON a été créédans le fichier MTH et permet de récupérer les informations du json sous forme de dictionnaire.

2) Les objets 3D animés



Sur la mezzanine, vous trouverez des objets 3D animés représentant des puits de pétroles. Ceux qui sont définit entièrement dans prims.js dans la fonction creerPuitpetrole().
Pour créer cet objet animé, il y a plusieurs étapes :

- Création des maillages
BABYLON.MeshBuilder.CreateBox(...)

- Positionnement et parentée des maillages
upperArm.setParent(lowerArm);
upperArm.position.y = 1.5;

- Création du squelette
const skeleton = new BABYLON.Skeleton(...)

- Création des os
const upperArmBone = new BABYLON.Bone(...)

- Liaison maillages/os
upperArmBone.linkTransformNode(upperArm);

- Animation
const upperAnimation = new BABYLON.Animation(...)
upperKeys.push({ frame: 0, value: Math.PI / 2 });
upperAnimation.setKeys(upperKeys);

3) <u>Les portes coulissantes</u>

Les portes coulissantes sont également définies dans prims.js
Tout d'abord, l'objet porte est créé avec BABYLON.MeshBuilder.CreateBox(). Ensuite les positions ouvert et fermé sont définit via des vector3. Les animations d'ouverture et de fermeture sont définies avec BABYLON.Animation.CreateAndStartAnimation(…).
Enfin, la détection de proximité se fait grâce à un Vector3.Distance entre la position de la caméra et de l'objet porte. La position de la porte est obtenue via la fonction getAbsolutePosition(), ce qui permet de prendre en compte les translations effectuées sur l'objet dans world02.

4) Les amers

Nous avons fait 2 amers qui se trouvent dans la pièce principale, se sont deux sphères et lorsqu'on clique dessus, on se téléporte en face d'elle (à 2 mètres de leur centre).
Pour cela, nous avons utilisé la fonction addEventListener et avons creer un attribut type pour les amers qui fait que lorsqu'on on détecte un mesh avec un type amer on active la fonction qui modifie les atributs pos et target de la camera.

```
        window.addEventListener('click', function (event) {
    var pickResult = scn.pick(event.clientX, event.clientY);
    if (pickResult.hit) {
       if (pickResult.pickedMesh.type === "amer") {
          var pos = pickResult.pickedMesh.getAbsolutePosition();
                                    camera.position.x = pos.x;
                                    camera.position.z = pos.z-2;
                                    camera.setTarget(pos);

       }
    }
```

5) Description textuelle des tableaux

Dans un premier temps, nous avons voulu utiliser la classe Cartel (dans Component) pour afficher la description textuelle des tableaux. N'ayant pas réussi à faire fonctionner cette classe, nous avons implémenté cette fonction directement dans la fonctions prims.creerPoster(). Pour cela, la description du tableau est passée en paramètre de la fonction.
D'abord, nous créons le panneau d'affichage :

```
let descpition = document.createElement('div');
descpition.innerText = descript;
```

Nous créons ensuite les observables :

```
tableau1.onPointerOverObservable = new BABYLON.Observable();
tableau1.onPointerOutObservable = new BABYLON.Observable();
```

Enfin, nous ajoutons des fonctions à ces observables, permettant le display ou nous du panneau d'affichage. La détection de proximité se fait de la même manière que pour les portes.

Lorsqu'on regarde un tableau à travers un mur, la description d'affiche aussi. Pour palier à cela, nous avons essayé de rajouter une condition d'angle entre la caméra et le tableau afin qu'elle ne s'affiche que si l'on est en face de celui-ci. Nous n'avons pas réussi à réaliser cela car il n'existe pas de fonction équivalente à getAbsolutePosition() pour les rotations. Les fonctions getRotation et getRotationToRef renvoient donc toujours (0,0,0), car elles ne prennent pas en compte les rotations effectuées sur l'objet dans world02.

6) Description orale des tableaux

2 tableaux sont dotés de description sonore :
Dessin_dreyfus et Decapite qui se trouvent dans la salle gauche et droite du musée.
Nous avons enregistré les description orale en mp3 et nous les avons utilisé à l'aide de la fonction Sound de Babylon dans la Prims de poster:

```
const music = new BABYLON.Sound("Music", sound, scn, null, {
```

```
            loop: true,
            autoplay: true,
            spatialSound: true,
            maxDistance: 5 // distance maximale d'écoute
        });
    MTH.SoundToMesh(music, group);
```

La description est jouée en continu (même lorsque nous ne l'entendons pas) et elle tourne en boucle.