Rapport de projet

Sujet : Extreme-VR

BARBERON Léa

ESMIEU Michaël

ROUFF Sébastien

2021

INFO 4A

1) INTRODUCTION

a) Présentation du projet

Le projet Extreme-VR a pour finalité la réalisation d'une simulation en réalité virtuelle d'une plateforme de travail permettant d'effectuer des analyses de risques en milieu radiologique. Cela consiste à créer des scénarios dans un environnement virtuel dans un but pédagogique. Il est par exemple question de simuler les procédures d'habillage précédant une entrée en zone contrôlée, simuler la prise en main d'appareils et d'outils nécessaires au bon déroulement de ces analyses ainsi que l'intervention sur un problème.

b) Éléments à disposition

Il nous a été fourni :

- **le code source du projet existant** (disponible à cette adresse : https://github.com/elysa-lacot/EXTREME-VR) : il s'agit d'un répertoire contenant l'ensemble des fichiers du projet à ouvrir sur Unity afin d'apporter des modifications
- **une démonstration du projet** (https://vimeo.com/536528051/40776b162f) : cette vidéo nous amène directement dans la scène principale et nous présente différentes interactions possibles telles que l'ouverture des portes et la manipulation de valves.
- **2 casques Oculus Quest 1 et 2 :** ce sont les casques de réalité virtuelle sur lesquels nous devions développer le projet.

Le projet proposait déjà un environnement immersif, il nous était alors proposé plusieurs objectifs afin d'améliorer celui-ci.

2) OBJECTIFS

a) Faire Fonctionner le projet sur Oculus Quest 2

Le premier objectif du projet consistait à faire fonctionner la simulation sur le casque Oculus Quest 2. En effet, il nous a été indiqué que celle-ci se lançait correctement sur le casque Oculus Quest 1 mais ne fonctionnait pas sur l'Oculus Quest 2, le problème étant l'impossibilité de lancer les scénarios. Il s'agissait donc de trouver d'où venait le problème et de réussir à le résoudre.

b) Ajout d'éléments au projet

Le second objectif était l'ajout d'extensions au projet. On nous a laissé libres de faire ce que l'on souhaite tant que cela restait formel. On nous a par exemple proposé d'ajouter des lumières à la scène ainsi que d'autres objets.

3) MISE EN PLACE DES OUTILS

a) Installation et paramétrages des outils, et lancement du projet sur le casque

Pour pouvoir travailler sur ce projet, il est nécessaire de posséder, en plus du casque, de différents logiciels correctement paramétrés afin de pouvoir modifier le projet existant mais aussi pour pouvoir le lancer sur un des casques à notre disposition.

Les explications sur le bon déroulement de ces tâches se trouvent en annexes.

4) RÉALISATION

a) Prise en main des outils

La réelle première étape du projet, après l'installation de tous les outils nécessaires, a été la prise en main des logiciels, plus particulièrement du logiciel **Unity** permettant la réalisation de jeux-vidéos qui était le principal outil de développement pour le projet.

Pour cela nous avons chacun suivit un premier tutoriel permettant de découvrir les principales fonctions de Unity avec par exemple la créations et la modifications d'objets 3D dans une scène, ainsi que l'écriture de scripts permettant d'animer certains objets de la scène continue, de déplacer notre "joueur" ou encore de modifier, en fonctions des actions de ce derniers, des informations qui s'afficheraient à l'écran.
Une fois ceci fait, nous avons pu nous plonger plus aisément dans le projet pour en comprendre un peu mieux le fonctionnement.

b) Tests et débogage sur Oculus Quest 2

Pour pouvoir lancer l'application il était tout d'abord nécessaire d'avoir correctement installé et paramétré les différents logiciels et drivers indispensables au fonctionnement du projet sur les casques Oculus comme indiqué en annexe. Une fois tout ceci fait, il est normalement possible de "build" le projet pour pouvoir ensuite le lancer sur un casque.

Grâce à une connaissance, nous avons découvert tardivement qu'il était possible également de lancer le projet depuis Unity grâce à Oculus Air Link (cf : annexes), permettant ainsi d'utiliser les ressources de l'ordinateur et d'éviter de "build" le projet à chaque test, ce qui permet de gagner énormément de temps sur le développement. Il a fallu faire néanmoins attention à cela dans le cadre de notre projet par rapport aux problèmes de chemins que cela créé dans l'arborescence du projet lors de son installation sur casque.

Tout d'abord, l'arborescence du projet contient de nombreux dossiers un peu désordonnés mais on finit par s'y retrouve, pour avoir accès aux différentes scènes du projet (*MainMenu* et

DefaultScene), il faut aller dans « *Assets/PFE/Scenes* ».

Nous avons compris que *MainMenu* amène l'utilisateur sur une première scène dans une salle avec un menu affiché sur le mur d'en face afin de sélectionner un scénario en utilisant les manettes. Une fois ce scénario sélectionné, le joueur est censé être envoyé dans la *DefaultScene* afin d'y effectuer le scénario choisi. Cependant lors du premier lancement de l'application sur les casques, nous avons pu observer un message d'erreur à l'emplacement où devaient apparaître les scénarios :

« Could not find a part of the path '/mnt/sdcard/scenario' »

Nous avons pu ainsi comprendre qu'il y avait un problème de chemin et après avoir correctement étudié le projet nous avons trouvé la partie de code où le problème apparaissait.

Le script en question a pour objectif d'ouvrir un dossier qui contient des scénarios écrits à l'avance dans des fichiers .txt et de les afficher afin de permettre au joueur de choisir celui qu'il souhaite effectuer. Se lance ensuite un autre script qui permet la lecture et l'interprétation de ces fichiers de scénario.

Le chemin *'/mnt/sdcard/scenario'* ainsi affiché ne permettant pas d'ouvrir les scénarios qui se trouvent dans *'Assets/PFE/Resources/Scenario'* dans l'arborescence du projet, nous avons alors testé avec d'autres chemins qui étaient affichés en commentaire dans le code ainsi qu'en nous aidant des informations que l'on a pu trouver sur internet. Malheureusement aucun n'a fonctionné affichant toujours le même message d'erreur. Nous avons donc conclu que l'arborescence dans le dossier source du projet n'est pas la même que celle du projet une fois installé dans le casque.

A l'aide des méthodes *Application.dataPath()* et *Application.persistantDataParth()* et à l'aide d'affichage de messages à l'écran nous avons cherché s'il était possible de trouver les fichiers de scénario dans l'arborescence du casque. Cependant il se trouve que les fichiers de scénarios étant appelés par un script et non directement par un objet présent dans la scène, ceux-ci n'étaient donc pas reconnus comme utilisés par la scène lors de l'installation dans le casque.
A partir de là plusieurs solutions se sont offertes à nous, la première étant d'indiquer dans le script le chemin menant au dossier scénario situé sur le PC (par exemple *D:/Projet/ExtremeVR/Assets/PFE/Resources/Scenario')* et de lancer la scène avec Air Link. Cela a bien permis d'afficher le choix des scénarios mais cette solution a pour contrainte d'avoir un PC à disposition pour pouvoir effectuer la simulation.

La deuxième solution était d'utiliser les méthodes méthodes ***Application.dataPath()*** ou ***Application.persistantDataParth()*** précédentes afin de copier manuellement les scénarios dans un dossier du casque pour pouvoir les ouvrir ensuite. La première méthode renvoie le chemin courant dans lequel se trouve l'application, il semble que ce soit un dossier temporaire, nous avons donc utilisé

la seconde méthode qui elle renvoie un dossier permanent permettant de stocker des données de l'application. Pour cela nous avons dû installer l'application **SideQuest** sur PC qui offre un gestionnaire de fichier permettant d'avoir accès aux données stockées sur l'Oculus. Nous avons donc copié le dossier *Scenario* et indiqué le chemin à l'endroit suivant : */storage/emulated/0/Android/data/com.DefaultCompany.PFE/files/Scenario*

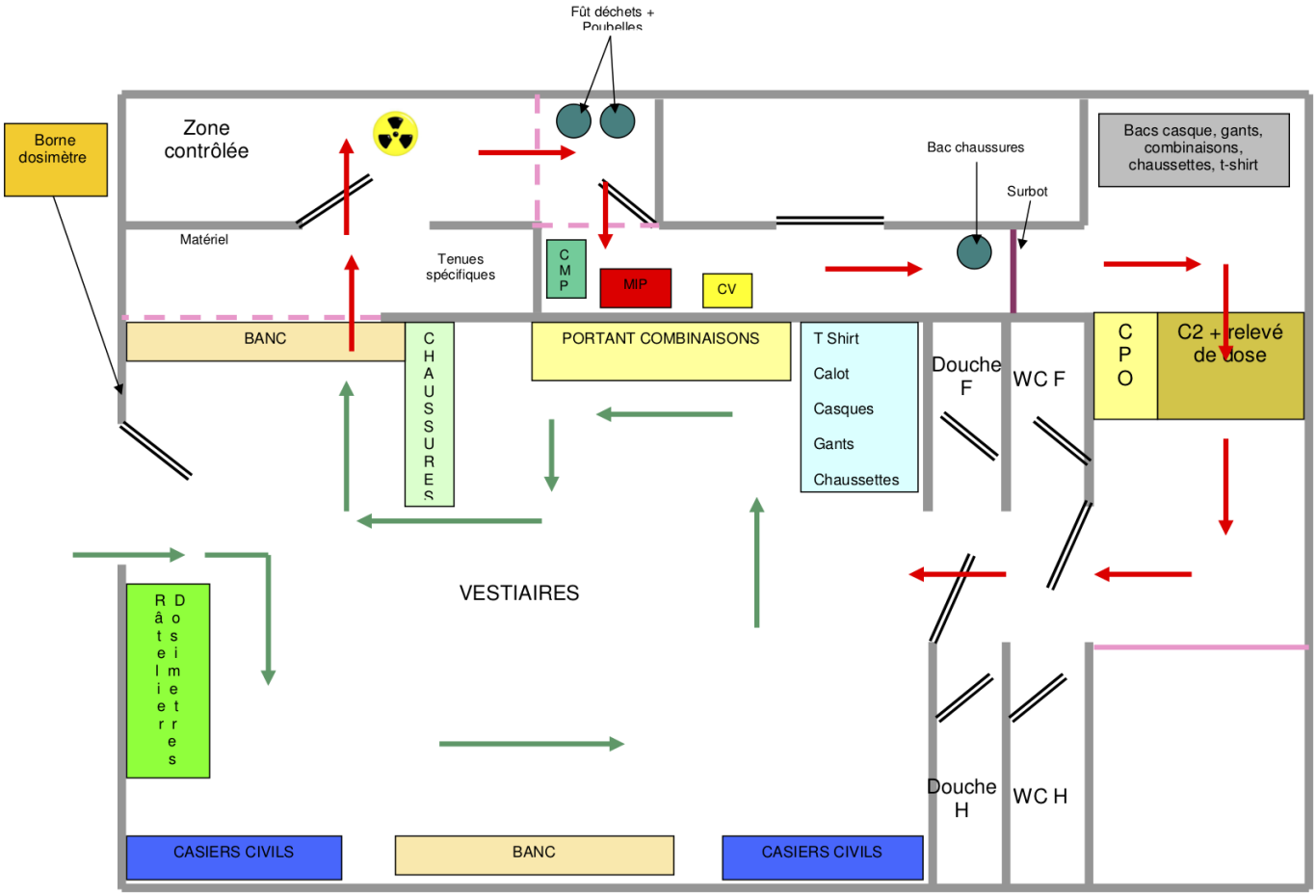
Cette solution n'est pas non plus très pratique mais nous a permis de répondre au premier problème qui était de trouver et d'afficher les scénarios dans le menu. Cependant lors de la sélection de l'un de ces scénarios il ne se passe rien et le joueur reste dans le menu principal.

Nous avons cherché à résoudre également ce problème cependant la seule solution que nous avions était de revoir l'ensemble du système de lancement des scénarios mais nous ne disposions pas de suffisamment de temps pour cela.

Il est cependant possible de lancer la scène principale *'DefaultScene'* sans lancer de scénario, pour cela il faut tout d'abord activer un joueur dans la scène en sélectionnant un des 2 objets 'player' dans la partie *'Hierarchy'* à gauche de la fenêtre puis en cochant la case en haut dans la partie *'Inspector'* à droite de la fenêtre, à côté du nom du 'player', cela permet d'interagir avec certains objets mais pas de remplir les objectifs ni de les afficher. (Le lancement de cette scène avec Air Link provoque un problème d'affichage, la musique de scène est audible mais seul un écran noir s'affiche).

c) Nouveaux éléments ajoutés à la scène

Pour ce qui est de l'ajout de nouveaux éléments au projet, nous avons constaté que des panneaux étaient présents dans la scène du projet et que ceux-ci affichaient un plan de la scène. Il y a certains éléments qui figurent sur le plan mais qui sont absents de la scène. Pour rester dans le thème du projet nous avons choisi d'essayer de "remplir" la scène avec certains de ces éléments.



Plan de la scène principale "DefaultScene"

Malgré que nous n'ayons aucune connaissance dans le domaine, par des recherches nous avons pu par exemple découvrir par exemple à quoi semblait correspondre l'indication « **C2** » sur le plan. Cela s'apparenterait à un scanner ressemblant à un portique de sécurité, qui aurait pour fonction d'analyser un taux de contamination d'une personne qui travaille sur un site sensible.

Nous avons donc imaginé son fonctionnement à partir des photos que nous avons trouvé sur internet

et nous pensions donc mettre en place dans le scénario une interaction avec ce scanner C2 comme suit :

1. - Le joueur arrive devant le “C2” et appuie sur un bouton, cela ouvre une première barrière du scanner qui fonctionne comme un « sas », ce qui permet au joueur de se placer à l’intérieur, la barrière se referme derrière lui par la suite (il faut faire attention à ne pas refermer la barrière lorsque le joueur se trouve en dessous).
2. - Le joueur attend sans bouger un certain délai (par exemple 5 secondes)
3. - Un affichage en « tête haute » ou bien sur un écran placé dans le C2 afficherait les consignes au joueur ainsi que l’avancée de la tâche, puis il lui indiquerait lorsque celle-ci est terminée (on peut également faire s’allumer des diodes rouges et vertes pour informer le joueur)
4. - La seconde barrière s’ouvre ensuite et le joueur peut continuer le scénario.

Il est possible d’importer dans Unity des objets réalisés avec le logiciel gratuit de modélisation **Blender** qui est assez facile à prendre en main (plus simple et plus utile que le module **ProBuilder** fournit dans Unity qui permet d’apporter quelques modifications sur la structure d’un objet).

Nous avons donc réussi à modéliser sur Blender un objet simple pouvant s'apparenter à un scanner “C2” avec deux barrières et un interrupteur “modulables” ce qui permettra ensuite de les animer pour pouvoir effectuer les tâches décrites juste précédemment.

N’ayant pas réussi à lancer le scénario, la mise en place de cette tâche ne s’est pas faite.

Nous avons également modélisé et ajouté dans une pièce une diode rouge ainsi qu'un haut-parleur, la première émet une lumière rotative à la manière d’un gyrophare et le second émet en boucle un son d’alarme, ce qui permet de rendre la simulation plus immersive. Il sera possible à l’avenir d’écrire un script qui désactive cette alarme une fois le scénario terminé. Enfin, à l’aide d’Assets disponibles gratuitement sur Oculus nous avons pu rajouter des bancs et une poubelle comme indiqué sur le plan. Nous avons aussi rajouté de la physique aux casiers et mis des lumières là où il n’y en avait pas dans les endroits un peu sombres de la scène

Pour continuer le projet, il est également possible de réaliser de manière analogue le C.P.O. indiqué sur le plan qui correspondrait à un coffre dans lequel on peut déposer des objets pour que ceux-ci soient analysés de la même manière que le scanner C2. Il faudra ensuite finir de compléter la scène avec les autres meubles ou objets indiqués sur le plan et d’ajouter les tâches souhaitées aux scénarios.

5) ANNEXES

a) Lien vers les fichiers du projet

Code : https://github.com/MichaelEsmieu/EXTREME-VR

Vidéo de démonstration : https://vimeo.com/558885055

b) Installation et paramétrage des logiciels

(Ces informations sur l'installation et le paramétrage du casque et des différents logiciels sont trouvables sur le site d'Oculus ou ailleurs sur internet, cependant la plupart utilisent d'anciennes versions donc la manipulation peut être légèrement différente.)

Pour lancer l'application il est nécessaire d'avoir installer au préalable ***Unity*** (avec les builds Android) et ***Oculus*** sur son PC ainsi que l'application *Oculus* sur mobile.

***(Inutile** depuis les versions 2019.X.XX de Unity : Il était également nécessaire d'installer **Android Studio** et d'aller chercher dans les paramètres « File/Settings » « Appearance & Behavior/System Settings/Android SDK » pour installer les **Android SDK** versions 4.4 (Kitkat) et éventuellement les versions ultérieures. Vérifier au passage le dossier d'installation des SDK et aller dans « Préférences/External Tools » dans Unity pour régler le bon chemin vers les Android SDK et Android NDK.)*

Il est faut se rendre sur le site d'Oculus afin de créer un compte utilisateur puis se « créer » une compagnie afin de pouvoir passer son compte en **mode développeur**. On connecte ensuite le casque VR à l'application mobile par Bluetooth puis on va dans les paramètres de celle-ci et on active le mode développeur.

Ensuite, installer les **drivers Android adb** depuis le site d'Oculus (*https://developer.oculus.com/downloads/package/oculus-adb-drivers/*), puis connecter le casque à son PC avec un câble USB et lancer l'application Oculus. Activer le **débugage USB** sur le casque et confirmer.

Pour ouvrir le projet sur Unity :

Lancer Unity hub, une fenêtre s'ouvre, cliquer sur « ***ADD*** » afin d'ajouter un projet et sélectionner le **dossier contenant le projet**. (Ici il est possible soit d'installer la version *2020.1.17f1* d'Unity qui est la version sur laquelle le projet a précédemment été développé afin de lancer directement le projet, soit d'ouvrir le projet avec une version plus récente ce qui va demander un temps de chargement plus ou moins long afin que Unity vérifie et rende compatible l'ensemble du projet avec la nouvelle version.

Une fois le projet ouvert, aller dans l'onglet « ***Package Manager*** » au-dessus de la fenêtre de visualisation de la scène, cela ouvre la fenêtre de gestion des packages, vérifier dans l'onglet « *Packages* » section « *My Assets* » la présence de l'asset « ***Oculus Integration*** » (cela correspond au dossier */Asset/Oculus* situé dans le dossier du projet) s'il n'est pas reconnu, il faut de nouveau le « *download* » puis « *import* » dans le projet. De même dans la section « All » (ou « Unity Registry ») vérifier les packages **Open VR** et **XR Legacy Input Handlers** (attention les noms sont différents sur les versions récentes d'Unity).

Il faut ensuite aller vérifier les réglages des paramètres de **build**, pour cela onglet *"File/Build Settings".* Vérifier qu'Android est sélectionné, si le projet n'est pas correctement réglé, un bouton "*Switch platform"* doit être visible en bas de la fenêtre des paramètres de build, cliquer dessus et patienter. Vérifier ensuite que "Texture Compression" est réglé sur *"ASTC"*, que *"Run Device"* détecte bien le casque Oculus connecté au PC sinon le chercher dans le menu déroulant.

Aller ensuite dans *"Player Setting"* en cliquant sur le bouton en bas à droite de la fenêtre de build. Régler les paramètres suivants pour qu'ils correspondent à votre projet : *"Company Name", "Product Name", "Version" et "Package Name"* en conséquence.
Vérifier que *"Minimum API Level"* est réglé sur Android 6.0 Marshmallow pour les Oculus Quest 1 et 2.

Aller dans l'onglet *XR-Plug-in Management* à gauche et vérifier qu'il est installé sinon cliquer sur *"Install XR...",* puis s'assurer que la case *"Oculus"* est bien cochée.

Il est ensuite proposé de régler différents paramètres graphiques sur le site d'Oculus afin d'avoir un meilleur rendu.

Normalement après cela il est possible de Build le projet et de le lancer sur un casque.

c) Lancement d'une scène sur un casque

Il est possible avec **AirLink** de lancer le projet sur son PC mais de l'afficher dans le casque lorsque celui-ci est branché sans avoir à build.
Pour cela, vérifier que l'application Oculus est lancée sur le PC et activer Air Link dans les *Paramètres/Beta.* Aller ensuite dans les *paramètres/expérimental* du casque et cliquer une fois sur AirLink, il devrait s'afficher une première fenêtre d'informations, attendre 2-3 secondes puis une autre fenêtre devrait s'ouvrir en demandant d'accepter la connexion AirLink.

Cela permet de tester le projet sans avoir à le build à chaque fois ce qui permet de gagner du temps (attention cependant cela utilise les ressources de l'ordinateur, celui-ci doit donc être suffisamment puissant).

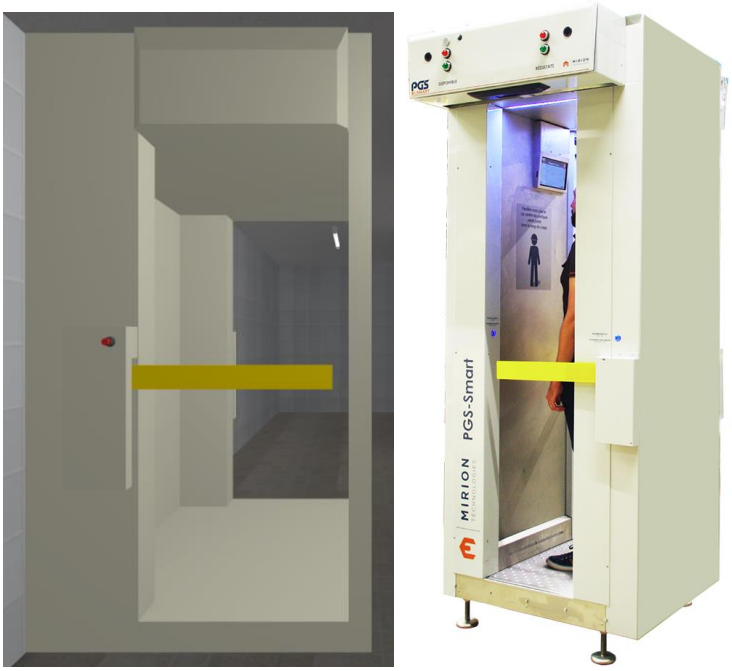
d) Partie du code à déboguer

*Dans **MainMenu.cs***

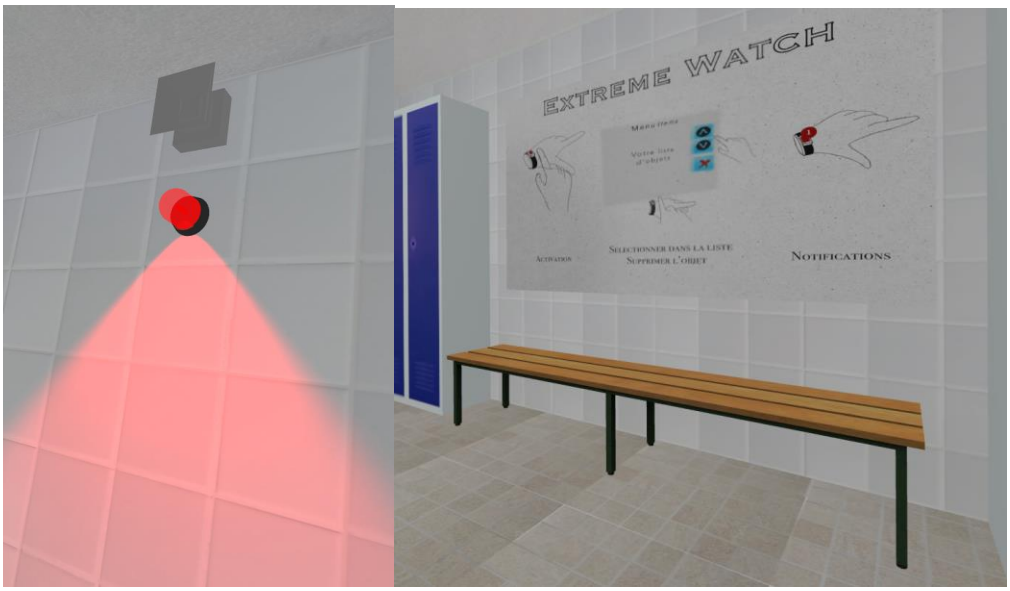
```
namespace ExtremeVR
{
	public class MainMenu : MonoBehaviour
	{
	private int selectedIndex;
	public Text mainText;
	public Text levelFileText;
	public GameObject LeftArrowImage;
	public GameObject RightArrowImage;
	private bool axisReleased;
	private List<string> scenarioFiles;

	// Start is called before the first frame update
	void Start()
	{
		askPermission() ;
		try
		{
			//DirectoryInfo dir = new
DirectoryInfo("Assets/PFE/Resources/Scenario");
			//DirectoryInfo dir = new
DirectoryInfo("/mnt/sdcard/Scenario");
			//DirectoryInfo dir = new
DirectoryInfo("/storage/emulated/0/Android/data/com.DefaultCompany.PFE/file
s/Assets/PFE/Resources/Scenario");

			DirectoryInfo dir = new
DirectoryInfo("/storage/emulated/0/Android/data/com.DefaultCompany.PFE/file
s/Scenario");
			FileInfo[] info = dir.GetFiles("*.txt");
			scenarioFiles = new List<string>();
			UnityEngine.SceneManagement.Scene currentScene =
SceneManager.GetActiveScene();
			GameObject[] currentObj =
currentScene.GetRootGameObjects();
			/*for (int i = 0; i < currentObj.Length; i++)
			{
			if (currentObj[i].GetComponent(typeof(OVRCameraRig)) !=
null || currentObj[i].GetComponent(typeof(OVRPlayerController)) != null)
				DontDestroyOnLoad(currentObj[i]);
			}*/
			foreach (FileInfo f in info)
			{
			scenarioFiles.Add(f.Name.Replace(".txt",""));
			}

			if(scenarioFiles.Count == 0)
			{
			levelFileText.text = "ERROR : NO SCENARIO FILE FOUND";
			return;
			}
			selectedIndex = 0;
			LeftArrowImage.SetActive(false);
			if(scenarioFiles.Count > 1)
RightArrowImage.SetActive(true);
```

```
                levelFileText.text = scenarioFiles[0];
                //Application.persistentDataPath
                axisReleased = true;
            }
            catch(Exception e)
            {
                levelFileText.text = e.Message ;
            }
        }

        // Update is called once per frame
        void Update()
        {
            if (Input.GetButtonDown("Fire1"))
            {
                Debug.Log("Loading...");
                mainText.text = "Chargement...";

                //sceneFile = scenarioFiles[selectedIndex];
                SceneLoader.SceneToLoad = scenarioFiles[selectedIndex];
                StartCoroutine(SceneLoader.LoadScene());
            }
            if (Input.GetButtonDown("Fire2") && selectedIndex <
(scenarioFiles.Count - 1))
            {
                selectedIndex++;
                levelFileText.text = scenarioFiles[selectedIndex];
                if(scenarioFiles.Count -1 <= selectedIndex)
RightArrowImage.SetActive(false);
                LeftArrowImage.SetActive(true);
                axisReleased = false;
            }
            if (Input.GetButtonDown("Jump") && selectedIndex > 0)
            {
                selectedIndex--;
                levelFileText.text = scenarioFiles[selectedIndex];
                if(selectedIndex == 0) LeftArrowImage.SetActive(false);
                RightArrowImage.SetActive(true);
                axisReleased = false;
            }
            if (Input.GetAxis("Horizontal") >= -0.25 &&
Input.GetAxis("Horizontal") <= 0.25)
            {
                axisReleased = true;
            }
        }
```

e) Captures d'écran des éléments ajoutés à la scène



Un "C2" à droite et celui que nous avons essayé de modéliser à gauche



Ajout d'une alarme, de bancs et de lumières