Développement Unity pour Hololens

ADNET, BELMONT, GERMONVILLE-BELLET, RUDONI

Années scolaires : 2021-2022

Tuteurs : ROEGEL Denis / BINET Julien

Sommaire

1.Introduction

a. Objet du document

Le document a pour but de synthétiser les différents points importants du projet, nous le structurons en plusieurs parties distinctes dans un souhait de compréhension et de lisibilité. A la demande des enseignants relatifs au Projet Tutoré (U.E. 41 - M4106), nous rédigeons ce rapport afin d'apporter une étude complémentaire au travail réalisé durant ces derniers mois, permettant ainsi de retracer tout ce qui a été fait, et éventuellement la projection future du projet.

b. Présentation du projet

Le projet 5 : Développement Unity pour Hololens est un projet visant à importer l'horloge de Notre Dame de Paris dans le casque de réalité mixte de Microsoft, Hololens 2 par l'intermédiaire du logiciel de développement Unity pour ce qui est relatif à la création du projet. D'une autre part, nous avons utilisé le logiciel Microsoft Visual Studio s'axant sur la partie code et importation du projet (sa création) dans le casque.

Le casque Hololens 2 est la deuxième itération du casque de réalité mixte Hololens de Microsoft, sorti en 2019, celui-ci permet d'afficher en surcouche de la réalité, et cela via des projecteurs sur les verres, des informations ou des images. En complément, grâce à ses capteurs, le casque permet d'interagir avec lesdites informations grâce à des mouvements des mains, des yeux, ou même grâce à la voix.

Le modèle de l'horloge nous a été fourni par notre tuteur, ROEGEL Denis. Ce modèle se compose de 359 modèles 3D représentant chacun une pièce individuelle de l'horloge.

Unity est un logiciel de création de jeux ainsi qu'un moteur graphique, celui-ci nous a servi à centraliser et mettre en relation nos scripts et nos assets dans une scène 3D.

Enfin, Microsoft Visual Studio est un environnement de développement intégré, il nous a servi à rédiger les scripts du projet, dans le langage C#, ainsi qu'à importer notre build Unity dans le casque Hololens 2.

c. L'équipe

L'équipe est composée de quatre personnes, ADNET Paul, BELMONT Tristan, GERMONVILLE-BELLET Alexandre et RUDONI Antonin. Celle-ci s'est formée au départ car nous étions tous dans la même classe durant le troisième semestre.

Alexandre, membre le plus qualifié en termes de développement Unity, a proposé en premier le sujet numéro 5, il a été un élément indispensable dans la mise en place des différentes fonctionnalités du projet.

Paul, orienté vers l'aspect esthétique de la globalité du développement Hololens, s'est associé à Tristan dans le but de mettre en place un système de Pop-Ups pour définir les informations des pièces du modèle.

Antonin quant à lui a permis un enregistrement plus qualitatif que les autres afin de proposer une visualisation claire et précise de la scène projetée à travers le prisme du casque Hololens 2. L'ensemble de l'équipe a su définir les objectifs à l'aide des tuteurs afin de réaliser un travail optimal, la phase d'apprentissage sur Unity Learn fut plus longue pour certains que pour d'autres, mais les tâches ont été revues afin de pallier cette contrainte.

2.Analyse

a. Contraintes

Durant la réalisation de notre projet tutoré, nous avons dû confronter quelques contraintes telles que les performances du casque, elles sont assez limitées, le projet doit être assez léger dans son contenu pour que le casque puisse supporter le projet et ainsi fonctionner.
Ensuite il y a la disponibilité du casque, le casque est disponible uniquement au Charly Lab et dans le cadre de l'aménagement de nos cours uniquement le jeudi après-midi, mais aussi par la taille du projet qui met entre 40 minutes et 1 heure 30 minutes à s'exporter après une modification et qui nous a fait perdre un temps considérable sur le lieu de test.
La documentation Microsoft par rapport à l'utilisation du casque est obsolète et ne nous a pas facilité la tâche lorsque nous devions debugger le code ou démarrer une nouvelle fonctionnalité.

b. Gestion du temps

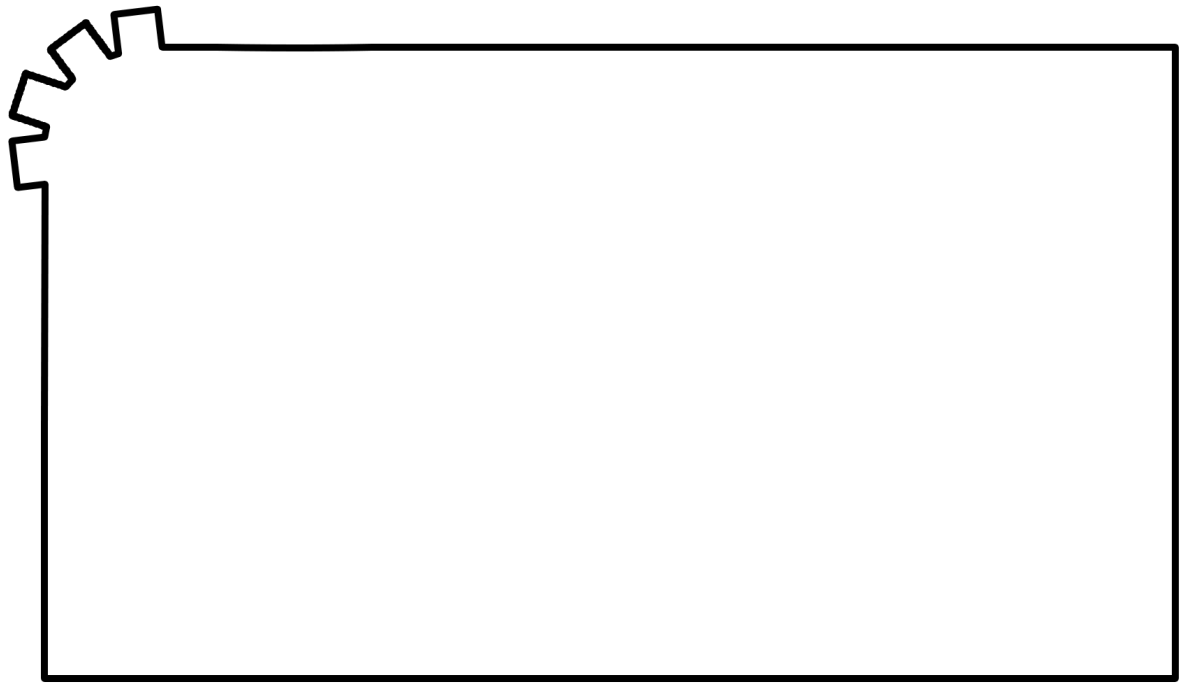
Une de nos principales contraintes, était le fait que le casque ne nous était pas accessible quand nous le voulions, il a donc fallu gérer notre temps. Nous avions donc accès au casque le jeudi après-midi, lors de l'ouverture du CharlyLab , nous nous sommes donc concentrés sur l'importation du build dans Hololens, ce qui prenait également beaucoup de temps, ainsi que sur les tests et les captures d'images pour les vidéos. Le reste du temps, nous nous contentions de l'émulateur proposé par Unity, et faisions les scripts et autres assets et configurations chez nous.

3.Réalisation

a.Tâches réalisées

Lors de la première itération, nous avons commencé par ajouter et régler le modèle 3D dans une scène simple Unity, puis nous avons procédé à l'intégration de cette scène dans le casque Hololens, en ajoutant toutes les dépendances de Microsoft Mixed Reality. Les dépendances se trouvent ici : Bienvenue dans Mixed Reality Feature Tool. Il suffit de télécharger le MRFT. Une fois le logiciel lancé, il faut choisir son projet Unity, sélectionner tout ce qui se trouve dans Mixed Reality Toolkit, sélectionner OpenXR Plugin dans Platform Support et sélectionner Mixed Reality Input dans Other Features. Nous avons aussi réalisé une simplification avec Blender de certaines pièces de l'horloge qui contenait trop de polygones. Pour finir cette itération, nous avons animé quelques pièces pour vérifier si le casque supporte les animations. Attention : Ces animations n'ont pas pour but d'être réalistes car aucun membre du groupe a des connaissances en horlogerie et elles servent juste a montrer qu'il est possible de réaliser des animations dans le casque.

Lors de la seconde itération, nous avons commencé par ajuster les différents réglages dont la taille et la hauteur pour donner une expérience plus confortable avec le casque. Nous avons choisi une hauteur permettant à un utilisateur d'environ 1 mètre 80 d'avoir la meilleure expérience. Nous avons ajouté des informations d'une pièce lors d'une interaction avec cette dernière. Dans cette itération, l'information affichée est uniquement le nom du modèle 3D de la pièce. Une fenêtre graphique a été réalisée pour habiller les informations données mais cette fenêtre a été retirée car elle était désagréable et trop encombrante dans le casque.



Une commande vocale a été réalisée afin de déployer l'horloge dans le casque. Lors du lancement de la scène, en prononçant le mot "Afficher", la scène charge l'horloge et l'affiche. Cette fonctionnalité n'est pas restée dans la version finale car la commande vocale ne fonctionnait pas de manière consistante dans le casque. Nous avons réalisé le premier enregistrement à la première personne lors de cette itération.

Pour la dernière itération, nous avons cherché comment pouvoir gérer les pièces après l'interaction. Nous avons d'abord pensé à bloquer la pièce mais ce choix était assez contre-intuitif, une pièce que l'on attrape mais qui ne bouge pas est désagréable. Nous avons donc décidé de replacer la pièce au moment où la pièce est relâchée. Nous avons aussi amélioré l'affichage de l'information des pièces en affichant le nom exact ou la description de son utilité dans l'horloge. Une optimisation de la scène a été obligatoire car les performances étaient insuffisantes pour le confort dans le casque. Une suppression de tous les calculs de lumières a été instaurée pour l'optimisation. Pour finir, cette dernière itération est marquée par la réalisation d'un build en WebGL de la scène du casque. Ce build est agrémenté d'un systeme de deplacement avec les touches classiques Z,Q,S,D, d'un système d'affichage d'informations similaire à celui du casque, mais fonctionnant avec un rayon invisible au centre de la caméra et qui affiche au contact d'une pièce.

b.Difficultés rencontrées

Lors de la réalisation, nous avons rencontré de nombreux problèmes. Les principaux sont directement liés au casque. Le premier est le manque de documentation pour Mixed Reality. Attention, il existe une documentation mais elle n'est pas à jour, et donc partiellement obsolète. Par exemple, la réalisation des interactions nous a pris beaucoup de temps car il n'y a aucune documentation dessus. Nous avons dû regarder des projets indépendants pour voir comment c'était réalisable. Le second était notre temps avec le casque. Le casque, appartenant à l'iut, n'était utilisable que lorsque nous allions au CharlyLab. Si on couple ce problème a un autre souci, qui est le temps des builds sur le casque, qui pouvait durer entre 20 minutes et 1 heure et demi, nous avions peu de temps pour travailler avec le casque.

c.Explication du Code

Le script principal est le script qui gère l'entièreté de l'horloge, il est dédié à ajouter le modèle à la scène et de lui ajouter les composants a chaque pièce.

```
using System.Collections;
using System.Collections.Generic;
using System.Globalization;
using System;
using System.IO;
using UnityEngine;
using Microsoft.MixedReality.Toolkit.UI;
using Microsoft.MixedReality.Toolkit.Input;

public class Horloge: MonoBehaviour {
    //Objet de chaque piece
    private GameObject g1;
    //Texture de chaque piece
    private Mesh m1;
    //Rendu des textures de chaque piece
    private MeshRenderer meshRenderer;
    //Liste de String comprenant toutes les informations de chaque piece
    private List<string> readText;



    //methode qui se lance au demarrage
    void Start()
    {
        //On cree un objet parent Horloge
        GameObject parent = new GameObject("Horloge");

        //On declare la liste de string
        readText = new List<string>();
        //on ajoute a la liste les infos des pieces
        DonnerCoordonnées();

        //On crée un quaternion pour les angles des pieces
        Quaternion newRotation = new Quaternion();
        //On modifie l'angle
        newRotation.eulerAngles = new Vector3(-90,0,0);

        //GameObject scriptgetter = GameObject.Find("PieceScriptGetter");
        //var script = scriptgetter.GetComponent<Piece>();
```

```
//Pour chaque piece dans la liste
foreach (string s in readText)
{
   //On cree un tableau en coupant la ligne au moment des virgules
        string[] words = s.Split(',');

        string s1="i="+words[0]+"name="+words[1];

   Console.WriteLine(s);

   //On cree l'objet avec le nom de la piece
   g1 = new GameObject(words[1]);
   //On ajoute la piece au parent horloge
   g1.transform.SetParent(parent.transform);
   //On ajoute a la piece le composant MeshFilter
   g1.AddComponent<MeshFilter>();
   //On ajoute a la piece le composant MeshRenderer
   g1.AddComponent<MeshRenderer>();
   //On ajoute a la piece le composant BoxCollider
   //g1.AddComponent<BoxCollider>();
   //On charge la texture correspondant au nom de la piece
   m1 = (Mesh)Resources.Load(words[1],typeof(Mesh));
   //On ajoute la texture de la piece au MeshFilter
   g1.GetComponent<MeshFilter>().mesh = m1;

   //On recupere le MeshRenderer de la piece
   meshRenderer = g1.GetComponent<MeshRenderer>();

   //On ajoute a la piece le composant BoxCollider
   BoxCollider bc = g1.AddComponent<BoxCollider>();

   //recuperation des limites des meshs
   Bounds b = m1.bounds;
   //redimension du box collider avec la taille du mesh
   bc.size = new Vector3(b.size.x,b.size.y,b.size.z);
   //On fait une rotation a la piece de l'angle modifie
   g1.transform.Rotate(newRotation.eulerAngles);
   //On cree lobjet CultureInfo en le clonant
   var culture = (CultureInfo)CultureInfo.CurrentCulture.Clone();
   //On precise le separateur des nombres qui est un point pour les decimales
   culture.NumberFormat.NumberDecimalSeparator = ".";
   //On bouge la piece a ses coordonnees en utilisant lobjet CultureInfo
```

```
g1.gameObject.transform.Translate(float.Parse(words[2],culture),-float.Parse(words[
3],culture),float.Parse(words[4],culture));
            //On fait une rotation de la piece de 180 degres sur laxe Z
            g1.transform.Rotate(new Vector3(0,0,180));
            //On modifie la couleur du rendu des textures en utilisant lobjet CultureInfo
                                                meshRenderer.material.SetColor("_Color",new
Color(float.Parse(words[5],culture),float.Parse(words[6],culture),float.Parse(words[7],
culture)));

            //ajout du composant cree qui gere les interactions
            g1.AddComponent<EveHand>();


        }

        //On change léchelle parent pour réduire l'horloge complète
        parent.transform.localScale = new Vector3(0.0008f, 0.0008f, 0.0008f);
        //On bouge lhorloge en la baissant sur l'axe y et l'avancant sur l'axe z
        parent.transform.Translate(new Vector3(0, -1f, 1f));

        Debug.Log("End");

    }




    //methode pour toutes les coordonnees de chaque pièce (359 pieces)
    void DonnerCoordonnées()
    {
        readText.Add("0,Z,0.0,350.0,-937.1,0.8,0.8,0.8");
        //etc...
    }

}
```

Le second script PopUpCreate sert à la création de popup sur la caméra principale.

```
using System.Collections;
using System.Collections.Generic;
using System.Globalization;
using System;
```

```
using System.IO;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class PopupCreate : MonoBehaviour
{
    //creation de popup
    private TextMesh textmesh;
    private GameObject cam;
    private GameObject pop;

    void Start()
    {
        //recuperation de la camera
        this.cam = GameObject.Find("Main Camera");
        //creation du gameobject conteneur du texte
        this.pop = new GameObject();
        //objet renomme
        this.pop.name = "PopUp";
        //objet camera parent du popup
        this.pop.transform.SetParent(this.cam.transform);
        //position du popup par rapport a la cam
        this.pop.transform.position = new Vector3(0, 1, 3);
        //modification de son echelle
        this.pop.transform.localScale = new Vector3(0.1f, 0.1f, 0.1f);
        //asjout d'un meshrenderer
        this.pop.AddComponent<MeshRenderer>();
        //ajout du textmesh
        this.textmesh = this.pop.AddComponent<TextMesh>();
        //on regle le texte vide
        this.textmesh.text = "";
        //changement de la taille de police
        this.textmesh.fontSize = 500;
        this.textmesh.characterSize = 0.05f;
        //on bloque le texte au centre du popup
        this.textmesh.anchor = TextAnchor.MiddleCenter;
    }

    //methode qui change le texte du popup avec le texte en parametre
    public void changerTextePopUp(string txt)
    {
        this.textmesh.text = txt;
    }
```

```
    //methode qui supprime le texte
    public void supprimerTextePopUp()
    {
        this.textmesh.text = "";
    }

}
```

Le troisième script est le composant qui gère les interactions avec chaque pièce.

```
using System.Collections;
using System.Collections.Generic;
using System.Globalization;
using System;
using System.IO;
using UnityEngine;
using Microsoft.MixedReality.Toolkit.UI;
using Microsoft.MixedReality.Toolkit.Input;

public class EveHand : MonoBehaviour
{

    private GameObject popUp;

    //pointeur obligatoire mais inutile (sert a faire comprendre au casque que c'est un
event listener)
    private IMixedRealityPointer _pointer;
    // Start is called before the first frame update
    void Start()
    {
        //on recupere le popup
        this.popUp = GameObject.Find("PopUpCreator");
        //on ajoute un manager de contrainte pour la gestion des interactions
        this.gameObject.AddComponent<ConstraintManager>();
        //on ajoute un manipulateur d'objet
        ObjectManipulator om =
this.gameObject.AddComponent<ObjectManipulator>();
        //om.ManipulationType = 0;
        //on ajoute la methode de manipulation au depart de l'interaction
        om.OnManipulationStarted.AddListener(HandleOnManipulationStarted);
        //on ajoute la methode de manipulation a la fin de l'interaction
        om.OnManipulationEnded.AddListener(HandleOnManipulationEnded);
    }
```

```
    // Update is called once per frame
    void Update()
    {

    }
    //methode de manipulation au depart de l'interaction
    private void HandleOnManipulationStarted(ManipulationEventData eventData)
    {
        //on modifie le popup en ajoutant le nom de la piece
        this.popUp.GetComponent<PopupCreate>().changerTextePopUp(this.name);


    }
    //methode de manipulation a la fin de l'interaction
    private void HandleOnManipulationEnded(ManipulationEventData eventData)
    {
        //on modifie le popup en retirant le texte
        this.popUp.GetComponent<PopupCreate>().supprimerTextePopUp();
    }
}
```

Le dernier script gère le mouvement de certaines pièces.

```
using System.Globalization;
using System;
using System.IO;
using UnityEngine;
using UnityEngine.UI;

public class Mouvement : MonoBehaviour
{
    private GameObject go,go1,go2,go3,go4,go5,go6,go7;
    private GameObject pop;

    void Start()
    {
        //On recupere les objets qui vont etre en mouvement
        go = GameObject.Find("EB1");
        go1 = GameObject.Find("EB2");
        go2= GameObject.Find("EE1");
        go3 = GameObject.Find("EE2");
        go4 = GameObject.Find("EF2");
        go5 = GameObject.Find("EF1");
```

```
        go6 = GameObject.Find("IC1");
        go7 = GameObject.Find("IC2");
    }

    void Update()
    {
        //Rotation de la piece sur elle meme
        go.transform.Rotate(Vector3.up *Time.deltaTime* 50,Space.Self);
        go1.transform.Rotate(Vector3.up * Time.deltaTime * 50, Space.Self);
        go2.transform.Rotate(Vector3.up * Time.deltaTime * 50, Space.Self);
        go3.transform.Rotate(Vector3.up * Time.deltaTime * 50, Space.Self);
        go4.transform.Rotate(Vector3.up * Time.deltaTime * 50, Space.Self);
        go5.transform.Rotate(Vector3.up * Time.deltaTime * 50, Space.Self);
        go6.transform.Rotate(Vector3.up * Time.deltaTime * 50, Space.Self);
        go7.transform.Rotate(Vector3.up * Time.deltaTime * 50, Space.Self);
    }

}
```

4.Conclusion

Pour conclure ce rendu et ce projet, nous avons travaillé sur un projet avec un fort potentiel, qui peut être amélioré évidemment mais les nombreuses contraintes ne nous ont pas permis de réaliser tout ce qui était prévu. Le fait que ce soit un sujet exploratoire nous a fait comprendre l'importance d'une bonne documentation mais aussi que toutes les idées ne sont pas réalisables avec peu de temps. Le projet a donc permis à toute l'équipe de se développer de manière individuelle et collective, que ce soit dans l'appréhension d'un travail conséquent, l'organisation générale en fonction des compétences de chacun et également dans le cadre de mener à bien un projet commun. Nous avons également pu bénéficier d'un cadre de développement unique, notamment grâce au casque de réalité mixte Hololens 2, une réelle surprise pour l'ensemble de l'équipe. La globalité du projet tutoré nous a apporté connaissances et applications méthodologiques. Il nous a permis de développer notre sens logique en travaillant sur un projet en relation avec une manipulation de modèle 3D dans l'espace, une expérience non négligeable pour toute l'équipe.

Nous finissons ce rapport par des remerciements, vis-à-vis de l'établissement dans lequel nous étudions, l'IUT Nancy Charlemagne, mais aussi à nos enseignants référents sur le projet tutorés : Messieurs BINET Julien et ROEGEL Denis, sans qui cela n'aurait pas pu être possible, nous remercions également l'ensemble du corps enseignant et l'IUT pour nous avoir permis de développer dans un tel cadre, alliant ainsi technologies innovantes tels que les casques Hololens et environnements de développement tels que le moteur de jeu Unity et Visual Studio.