Introduction

LVDTSS a été créé dans le cadre d'un projet universitaire pour mettre en application et approfondir nos connaissances en réalité augmentée. Au cours du projet, nous avons dû faire des concessions dues au matériel à notre disposition. Cependant, nous avons également fait des découvertes qui nous ont permis d'aller plus loin que ce que nous imaginions.

Description

À l'heure actuelle, cette application est capable de positionner des accessoires, bien positionnés sur le visage d'une personne, comme :

- Des chapeaux
- Des lunettes
- Des boucles d'oreilles

En passant sur la caméra arrière du téléphone vous pourrez vous servir d'un catalogue d'accessoires préconçus afin de faire défiler le menu sur l'accessoire voulu. Pour sélectionner l'objet, il y a deux choix :

1. Sélectionner l'objet dans le menu situé en bas de l'écran. Comme la sélection avec la caméra frontale.
2. Toucher l'objet qui apparaît en 3D dans votre écran. Il se mettra à faire une rotation puis viendra se positionner sur vous.

Les objets sont cumulables. Cela peut rendre la visualisation absurde ou merveilleuse, à vous de choisir.

Librairies

Nous avons choisi de partir sur des librairies de l'ARFoundation. C'est un choix qui a été dument réfléchi. En effet ce choix permettait d'éviter de devoir lier OpenCV à Vuforia pour répondre à notre besoin. De plus cette librairie et les plugins associés sont installables de façon extrêmement triviale. La seule contrainte est qu'il faut avoir en sa possession un Smartphone ou un iPhone assez récent pour pouvoir tester la réalisation.

Travail Réalisé

Formations

Afin de pouvoir répondre aux besoins de notre application j'ai suivi plusieurs tutoriaux pour comprendre ce que l'ARFoundation pouvait nous apporter. Les différentes sources qui m'ont aidées ponctuellement seront citées dans la dernière section du rapport. Voici les principaux:

1. Le premier tutoriel est disponible sur la plateforme de formation Unity : https://learn.unity.com/tutorial/tracking-faces-in-ar?uv=2020.2&projectId=5fc7bd28edbc2a001fc8b017. Ce tutoriel, qui a été fait en groupe, nous a permis d'acquérir les bases sur le tracking de la tête avec l'ARFoundation et plus précisément l'ARFaceManager.
2. Le deuxième tutoriel que j'ai suivi, https://www.youtube.com/watch?v=Ml2UakwRxjk, était plus pour découvrir comment créer et « build » un projet Unity ARFoundation de A à Z. En faisant ce tutoriel j'ai également pu découvrir comment fonctionnait les RayCast et m'en servir dans le projet LVDTSS pour toucher des objets et obtenir des informations sur ceux-ci. Je ne vous cache pas que j'ai passé des dizaines d'heures sur ce tutoriel. Il était plein d'erreurs à cause des évolutions de librairies. Cependant j'ai fini par obtenir le résultat escompté et apprendre pleins de choses en résolvant les erreurs.

3. Le troisième tutoriel, que j'ai suivi avec Salomé, fut sur le menu interactif de l'application : https://www.youtube.com/watch?v=RqomLumqwCk. Ce tutoriel nous a apporté les connaissances nécessaires à propos des canvas et de ce que l'on pouvait faire avec (comme positionner des boutons).
4. Le quatrième et dernier tutoriel fut sur le tracking d'images : https://youtu.be/I9j3MD7gS5Y. Ce tutoriel est une merveille pour apprendre à utiliser l' « AR tracked image manager » et son composant indispensable le « ReferenceImageLibrary » afin de détecter des images et d'obtenir des informations sur celle-ci comme sa position dans le monde réel.

Base de l'application

La base de l'application à été réalisée en groupe. C'est dans cette première phase que nous avons mis en place le tracking de la tête. Nous avons commencé par utiliser des masques puis Tony et Chloé se sont occupés du reste du travail pour pouvoir positionner les accessoires au bon endroit sur la tête notamment grâce à une position relative à l'ARFace. Par la suite l'ajout d'un « Mesh Filter » a permis de rendre le positionnement adaptable à n'importe quel visage. Grâce à cela les accessoires sont positionnés, dimensionnés et tourné en fonction de la mesh proportionnelle au visage.

Plus tard dans le projet nous avons décidé de retravailler le rendu des accessoires. En effet les accessoires ne prennent pas en compte la tête comme élément virtuel mais l'ARFace permet de représenter la tête du monde réel dans le monde virtuel. Pour les accessoires de base j'ai retenu la solution d'ajouter un matériel d'occlusion sur l'ARFace. Pour les accessoires plus complexes comme les chapeaux c'est Tony qui s'est chargé des occlusions.

Toujours en relation avec la base du projet je me suis permis de retoucher tous les prefabs d'accessoires importés et formatés (position relative etc.) par Tony. En effet les éléments ressemblaient à des jouets donc j'ai modifié leurs matériaux pour que les accessoires semblent plus réalistes. Ce simple changement a également permis une réverbération de la lumière qui fait penser que nous portons réellement l'accessoire. Dans le design j'ai aussi enlevé certaines parties des lunettes et des boucles d'oreilles qui gênaient l'occlusion. Cela permet un réalisme plus important.

Menu Interactif

Le menu interactif a été réalisé avec Salomé en « pair programming ». C'est le premier menu qui a été réalisé afin de pouvoir sélectionner un ou plusieurs accessoires à positionner. Les étapes de la production ont été les suivantes :

1. Création d'un Canvas. Le Canvas est un objet qui permet de détecter la fenêtre de visualisation et de poser des composants relativement à cette fenêtre de visualisation. C'est un composant Unity qui facilite énormément le travail pour pouvoir créer des boutons d'interaction.
2. Sur le Canvas il a fallu ajouter trois boutons. Un bouton permet de faire défiler le menu vers la gauche. Un autre bouton permet de faire défiler le menu vers la droite. Le bouton central est le bouton de sélection. Le bouton de sélection est ici le plus intéressant. L'image du bouton représente l'accessoire qu'on veut positionner ou enlever. Ce sont les deux premiers boutons, « droite » et « gauche », qui vont permettre de changer l'image affichée dans le bouton de sélection. Lorsqu'un objet est sélectionné le bouton de sélection correspondant à l'accessoire est verdâtre. Ainsi l'utilisateur sait si en appuyant il va sélectionner ou désélectionner l'accessoire.
3. L'étape la plus importante fu le script **« SwitchAccessoire »** positionné dans l'ARSession. Je détaillerai le code ci-dessous. Le script permet d'écouter les interactions avec les 3 boutons. En fonction des interactions il va modifier le bouton de sélectionner, positionner et/ou enlever des objets sur l'ARFace qui représente le visage du monde réel.

Explication du code de « Switch Accessoire »

Dans un premier temps il existe 3 variables publiques essentielles au bon fonctionnement du script :

1. ListOfImgs : Tableau de Sprite (images). Cette liste permet de connaître les images à afficher sur le bouton.
2. ListOfObjects : Tableau de GameObjects. Cette liste permet de savoir quel prefab instancier en fonction de l'accessoire sélectionné.
3. Selected : Tableau de booléens. Cette liste permet de savoir si un accessoire est sélectionné ou non (pour savoir si on ajoute ou supprime l'accessoire).

Ces trois variables sont extrêmement liées. L'index des éléments de chaque tableau doit correspondre à un seul et même accessoire.

Une dernière variable que j'estime importante à connaitre, cette fois privée, est « objInstantiates » qui va garder en mémoire les objets instanciés pour pouvoir les détruire au moment voulu.

Nous allons maintenant observer les fonctions dans les grandes lignes :

1. Start() : Elle va récupérer les objets nécessaires et appliquer des écouteurs d'évènements sur les différents boutons. De plus elle va créer l' ARSessionOrigin via la fonction makeFace().
2. nextImg() : Appelée au clic sur le bouton de droite elle va incrémenter l'index et afficher dans le bouton de sélection l'image correspondante.
3. prevImg() : Appelée au clic sur le bouton de gauche elle va décrémenter l'index et afficher dans le bouton de sélection l'image correspondante.
4. goToSpecObj(GameObject prefab) : Cette fonction est utilisée par le menu virtuel que nous verrons après. Elle permet de faire défiler le menu directement sur l'objet détecté.
5. selectObj() : Appelée au clic sur le bouton de sélection elle va dire qu'un accessoire vient d'être sélectionnée ou désélectionné. Par la suite elle va instancier ou supprimer un objet via la fonction placeOrRemoveObject().
6. colorImg() : Appelée dans chaque fonction qui écoute les clics des boutons cette fonction permet de colorier le bouton de sélection pour dire à l'utilisateur si l'accessoire est sélectionné ou non.
7. placeOrRemoveObject() : En fonction de l'état de l'accessoire (sélectionné ou non), cette fonction va instancier ou supprimer un accessoire en tant qu'enfant de l'ARFace. Ensuite elle va recharger l' ARSessionOrigin via la fonction makeFace().
8. makeFace() : Cette fonction va permettre de regénérer l' ARSessionOrigin. Si ce composant n'est pas de nouveau instancié à chaque modification les ajouts et les suppressions d'accessoires ne seront pas visibles. On a besoin de la recharger pour qu'elle prenne en compte les modifications apportées à l'ARFace.

Menu Virtuel – Caméra arrière

Le menu interactif a été réalisé avec Salomé en « pair programming ». Afin de rajouter une petite touche d'interaction virtuelle. Nous avons décidé de mettre en place une sélection des accessoires au travers d'un catalogue. Cette fonctionnalité est basée sur l'ImageTracking de l'ARFoundation. Nous avons fait quelques essaies avec OpenCV (pour détecter des expressions du visage). Cependant, OpenCV prenait le dessus sur la caméra. Nous avons aussi essayé de le faire avec le « face tracking » mais le résultat est trop dépendant du matériel utilisé (iOS ou Android). Finalement le plus accessible était de rester sur les modules présents et de se baser sur de la reconnaissance d'image avec la librairie ARFoundation déjà inclue.

Au scan d'un accessoire ce dernier apparaîtra devant l'utilisateur. En même temps que l'objet apparaît le menu interactif va défiler sur l'accessoire détecté. A partir de ce moment l'utilisateur à deux choix :

1. Utiliser l'icône du menu interactif pour sélectionner l'objet.
2. Toucher l'accessoire qui est apparu à l'écran. Ce dernier fera une rotation sur lui-même puis viendra se positionner sur le visage.

Peu importe le type de sélection, une fois l'objet sélectionné, la caméra passera en frontale pour positionner (ou enlever) l'accessoire sur le « face tracking ». Toutes les interactions du menu virtuel sont gérées à partir du script : **« followImage » (positionné en composant de l'objet ARSession).**

Pour utiliser le face tracking et l'image tracking il a fallu « découper » le programme en deux. La caméra frontale va gérer le face tracking tandis que la caméra arrière gère l'image tracking et donc le menu virtuel. C'est le bouton en haut à droite de l'écran qui permet de modifier la caméra. Le script associé est : **« switchCamera » (positionné en composant de l'objet ARSession)**.

Explication du code de « Follow Image »

Il y a seulement deux variables intéressantes à connaitre pour ce script :

1. placeablePrefabs : Tableau public de GameObjects. Cette liste permet de connaître tous les GameObject qui peuvent être appelés par le tracking d'image. Elle est différente de celle dans le script du menu interactif qui permet de positionner sur la tête. La raison pour laquelle ces listes sont indépendantes est que j'ai dû faire des prefabs personnalisés, comme pour la boucle d'oreille, afin que le rendu soit agréable à visualiser.
2. spawnedPrefabs : Dictionnaire de GameObjects identifiés par leur nom. Ce dictionnaire permet de savoir si les objets instanciés doivent être visibles ou non.

Nous allons maintenant observer les fonctions les plus intéressantes du script :

1. Awake() : Dès que le script est activé on va récupérer les objets dont on a besoin et instancier tous les accessoires (sans les rendre visibles).
2. ImageChanged(ARTrackedImagesChangedEventArgs eventArgs) : Cette fonction va permettre de suivre la détection des images en fonction de si l'image vient d'être ajoutée, modifiée ou supprimée. Dans le cas où elles sont ajoutées ou modifiées on va activer l'instance du GameObject qui y est liée et la placer devant la caméra via la fonction UpdateImage(ARTrackedImage trackedImage).
3. UpdateImage(ARTrackedImage trackedImage) : Cette fonction permet de montrer en face de la caméra l'objet qui a été détecté grâce au nom de l'image scannée (l'image détectée doit posséder le même nom que l'objet instancié à faire apparaitre).
4. Update() : dans cette fonction Unity qui s'exécute en continu il y a deux parties importantes :
    a. On va constamment regarder si un objet a été touché via un « RayCast ». Le RayCast permet de dire quel est le premier objet virtuel rencontré relativement à un clic de l'utilisateur. Une notion importante est que le RayCast détecte un objet grâce à ses collider (que ce soit une « mesh » ou une « box »).
    b. Une fois qu'on a détecté qu'un objet a été sélectionné/touché on va gérer sa rotation. Dès qu'on détecte que l'objet a fait un tour sur lui-même sa position est réinitialisée et on active le positionnement sur le visage de l'utilisateur via la fonction du menu interactif.

Explication du code de « Switch Camera »

Ce script est assez simple. Au clic sur le bouton de la caméra, en haut à droite, le script va simplement échanger la caméra (frontale (User) ⇔arrière (World)) et activer ou désactiver le tracking d'image.

Publication

Pour la publication de l'application j'ai simplement modifié le « splash screen » en me basant sur l'icône faite par Tony. De ce fait l'utilisateur est accueilli un peu moins brutalement.

Problèmes rencontrés et Solutions apportées

Problèmes	Pistes	Solutions
Occlusion des accessoires du monde virtuel relativement au visage du monde réel	Ajouter des matériaux d'occlusions sur le visage ou effectuer des calculs pour savoir quand et quoi cacher. Renseignement sur Depth API d'ARCore qui permet l'occlusion automatique.	Dans un premier temps et pour les accessoires assez simples il a suffi de rajouter un matériau d'occlusion sur l'ARFace. Pour améliorer le rendu j'ai aussi supprimé des éléments trop encombrants sur certains prefabs.
ARFace qui ne charge pas les accessoires ajoutés et supprimés	Recharger l'ARFace ; Recharger l'ARFaceManager ; Recharger l'ARSessionOrigin	La solution la plus appropriée et fonctionnelle après différent test fu de réinstancier l'ARSessionOrigin à chaque ajout ou suppression d'accessoire.
Menu virtuel	OpenCVSharp ; TanserFlowSharp ; Vuforia ; ARFoundation	La solution la plus adéquate dans un souci de respect de l'existant était d'utiliser la détection d'image par ARFoundation
Raycast inefficace	J'ai appris après quelques recherches que le RayCast détectait un objet grâce à ses colliders.	En fonction des objets j'ai soit ajouté un « mesh collider » soit un « box collider »
Objet du menu virtuel qui reste devant la caméra lors du retour sur la caméra frontale	L'image tracking ne renvoie jamais d'information lorsque l'image n'est plus détectée.	Désactivation de tous les objets instanciés pour l'image tracking à chaque fois que la caméra change de côté (frontale ⇔ arrière).

Evolutions prévues

1. Positionnement au touché sur le visage. En étudiant les « vertices » touchés avec le RayCast nous pourrions réussir à trouver la position relative de l'accessoire sur le visage.
2. Accessoires catégorisés. Remplacement automatique des accessoires d'une même catégorie en cas de choix multiple. Cela évitera de pouvoir mettre deux chapeaux en même temps.
3. Retravailler l'occlusion des accessoires. Ce n'est pas encore parfait, on pourrait retravailler nos composants d'occlusion.

Sources utiles au projet

1. https://free3d.com/fr/3d-models/ ; 2. https://learn.unity.com/tutorial/tracking-faces-in-ar?uv=2020.2 ; 3. https://www.youtube.com/watch?v=kIcvAi60qlI&t=1768s ; 4. https://www.youtube.com/watch?v=RqomLumqwCk ; 5. https://youtu.be/JQEovMKq2U0 ; 6. https://medium.com/@harivigneshjayapalan/arcore-cupcakes-5-augmented-faces-api-2ca78681228e (vulgarisation des termes) ; 7. https://youtu.be/6seuSOUf3OU ; 8. https://support.wikitude.com/support/discussions/topics/5000082243 ; 9. https://www.youtube.com/watch?v=lXvt66A0i3Q ; 10. https://youtu.be/I9j3MD7gS5Y ; 11. https://www.turbosquid.com/3d-models/ 12. https://developers.google.com/