

Travail de Bachelor 2022

Learning Astronomy with augmented reality



Etudiant : Bastien Ferrari

Professeur : Antoine Widmer

Résumé

La réalité augmentée peut-elle se mettre au service de l'enseignement dans les écoles ? Cette question se pose depuis l'essor de cette technologie ces dernières années. La réalité augmentée offre de nouvelles opportunités quant à l'apprentissage d'une nouvelle matière. Un contenu animé en réalité augmentée peut motiver les élèves à étudier des nouveaux concepts. L'interactivité que propose les applications en réalité augmentée est susceptible de rendre l'apprentissage d'une nouvelle matière plus ludique.

Différentes sciences sont abordées très tôt lors du cursus scolaire. Certaines d'entre elles comme les mathématiques ou l'astronomie possèdent des concepts abstraits difficilement visualisables. La réalité augmentée peut répondre à cette problématique. En effet, l'astronomie comporte beaucoup de concepts 3D qui ne sont pas évidents à se représenter, comme le fonctionnement du système solaire, par exemple.

Ce travail a pour objectif de créer un prototype d'aide à la compréhension des concepts de l'astronomie. Ce dernier passe par une application développée grâce à la méthodologie agile. Ce prototype sera ensuite repris par un comité de recherche afin de continuer le travail.

L'application veut aider les étudiants à la compréhension du fonctionnement de notre système solaire. Elle permet aussi la découverte et l'apprentissage des différentes planètes du système solaire.

Mots-clés : Réalité augmentée, apprentissage, astronomie, système solaire

Avant-propos

Ce travail a été effectué dans le cadre de mon travail de Bachelor en informatique de gestion à la Haute Ecole Spécialisée de Suisse Occidentale Valais-Wallis.

La HES-SO Valais-Wallis, en collaboration avec l'école de formation d'enseignants valaisanne effectue une recherche concernant la création et la fourniture d'applications en réalité augmentée afin d'aider à l'apprentissage des sciences.

Ce travail, soumis par le Dr. Prof. Antoine Widmer, a pour but de développer une application aidant à l'apprentissage de l'astronomie, à travers notre système solaire. Afin de réaliser ce travail, nous avons effectué des recherches sur des applications déjà existantes afin de ressortir leurs points forts et faibles pour, par la suite, construire une application unique qui réponde à nos besoins. Puis, nous avons détaillé notre résultat et discuté des problèmes rencontrés ainsi que des améliorations qui pourraient être apportées à celui-ci.

Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidé d'une manière ou d'une autre à la réalisation de ce travail de Bachelor, en particulier :

- Monsieur Antoine Widmer, le professeur en charge, pour son accompagnement durant toute la durée du projet.
- Messieurs Dario Zenhausern et Romain Roduit pour leur feedback concernant l'application.
- Mademoiselle Nadège Ferrari, pour la relecture du document.

Table des matières

Table des illustrations	vi
Liste des abréviations	viii
1 Introduction	1
1.1 Contexte	1
1.2 Problématique	2
1.3 Méthodologie	3
2 État de l'art	5
2.1 Solutions existantes	5
2.1.1 Planets AR	5
2.1.2 AstroReality	6
2.1.3 Cosmos Creator – AR Universes	6
2.2 Technologies	7
2.2.1 Unity	7
2.2.2 Vuforia	8
2.2.3 AR Foundation	9
2.2.4 Lightship	11
2.2.5 Placenote	12
2.3 Choix technologique	13
3 Résultats	14
3.1 Installation du SDK et mise en place de l'environnement	14
3.2 Placement d'objets	16
3.3 Multijoueur	21
4 Discussion	33
4.1 Problèmes rencontrés	33
4.1.1 L'astronomie	33
4.1.2 Lightship et Unity	34
4.2 Améliorations	35
5 Conclusion	37
6 Références	38

7	Annexes.....	41
7.1	Annexe I : Product Backlog.....	41
7.2	Annexe II : Journal de bord.....	42
7.3	Annexe III : Guide d'utilisation.....	43
8	Déclaration de l'auteur	49

Table des illustrations

FIGURE 1 : APPLICATION <i>IKEA PLACE</i>	2
FIGURE 2 : APPLICATION PLANETS AR.....	5
FIGURE 3 : APPLICATION ASTROREALITY	6
FIGURE 4 : APPLICATION COSMOS CREATOR.....	7
FIGURE 5 : PLACEMENT D'UNE CHAISE GRÂCE À VUFORIA	9
FIGURE 6 : LES CLOUD ANCHORS SELON ARCORE	10
FIGURE 7 : POKEMON GO	11
FIGURE 8 : LES SPATIAL NOTES UTILISÉES PAR PLACENOTE.....	12
FIGURE 9 : LOGO DU SDK LIGHTSHIP	13
FIGURE 10 : CRÉATION DE L'ARDKAUTHCONFIG	14
FIGURE 11 : CLÉ API DU PORJET	15
FIGURE 12 : ARBORESCENCE DU PROJET UNITY.....	15
FIGURE 13 :EMPLACEMENT DU JDK	16
FIGURE 14 : SECTION À AJOUTER DANS LE FICHIER ANDROIDMANIFEST.XML	16
FIGURE 15 : COMPARATIF DE LA QUALITÉ DE DÉTECTION DE L'IMAGE	17
FIGURE 16 : CONFIGURATION DE L'AR PLANE MANAGER	17
FIGURE 17 : VISUALISATION DU PLAN GRÂCE À L'AR PLANE MANAGER	18
FIGURE 18 : CONFIGURATION DE L'AR HIT TESTER.....	18
FIGURE 19 : ATTRIBUTION DE LA VARIABLE PLACEMENTOBJPF	19
FIGURE 20 : GESTION DU PREMIER TOUCHER DE L'ÉCRAN	19
FIGURE 21 : MÉTHODE TOUCHBEGAN	20
FIGURE 22 : INSTANCIATION DE L'OBJET	20
FIGURE 23 : AJUSTEMENT DE LA HAUTEUR DE L'OBJET INSTANCIÉ	21
FIGURE 24 : CONFIGURATION DE L'ARNETWORKINGSCENEMANAGER.....	21
FIGURE 25 : LES DIFFÉRENTS ÉTATS DES PARTICIPANTS LORS D'UNE SESSION MULTIJOUEUR.....	23
FIGURE 26 : CONFIGURATION DU GAMEMANAGER	24
FIGURE 27 : INITIATION DE LA SESSION MULTIJOUEUR	24
FIGURE 28 : MÉTHODE GÉRANT LES CONNEXIONS	25
FIGURE 29 : DÉCLARATION DE VARIABLES	25
FIGURE 30 : SÉRIALISATION D'UN VECTEUR	26
FIGURE 31 : ENVOI DES DONNÉES VIA BROADCASTDATA	26
FIGURE 32 : RÉCEPTION DES DONNÉES ENVOYÉES À TRAVERS LE RÉSEAU	27
FIGURE 33 : GESTION DU TOUCHER D'ÉCRAN	27
FIGURE 34 : INSTANCIATION DU SYSTÈME SOLAIRE.....	28
FIGURE 35 : MÉTHODE GETRANDOMPLANET	28
FIGURE 36 : ENVOI D'UN NOM DE PLANÈTE VIA BROADCASTDATA.....	29

FIGURE 37 : RÉCEPTION DES DONNÉES AVEC LE TAG 1	29
FIGURE 38 : FONCTIONNEMENT D'UN RAYON DANS UNITY.....	30
FIGURE 39 : IMPLÉMENTATION DU RAYCAST	30
FIGURE 40 : ENVOI D'UN BIT VIA BROADCASTDATA	31
FIGURE 41 : GESTION DU SCORE	32

Liste des abréviations

AR	Augmented reality – Réalité augmentée
VR	Virtual reality – Réalité virtuelle
SDK	Software Development Kit – Kit de développement
JDK	Java Development Kit – Kit de développement Java
ARDK	Augmented Reality Development Kit – Kit de développement de réalité augmentée
API	Application Programming Interface – Interface de programmation applicative
LLAPI	Low Level API – API de bas niveau
HLAPI	High Level API – API de haut niveau

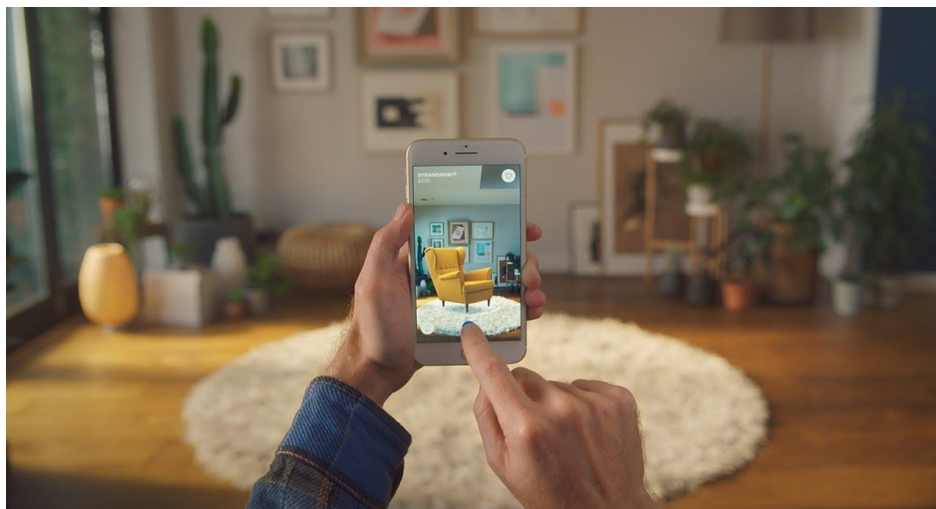
1 Introduction

1.1 Contexte

La technologie fait aujourd'hui partie de nos mœurs. Des enfants aux adultes en passant par les adolescents ou encore les personnes âgées, tout le monde est concerné par l'avènement des nouvelles technologies. Bien que celles-ci ne fassent pas toujours l'unanimité, elles trouvent bien souvent leur place dans différents domaines tels que la santé, l'industrie, l'agriculture ou encore l'enseignement.

Ce que l'on nomme aujourd'hui « révolution technologie » n'est pas vu du même œil par tout le monde. Certains y voient une opportunité immense qui permettra de créer des emplois alors que d'autres, plus pessimistes, pensent que ces technologies supprimeront des postes de travail. Que nous soyons dans un camp ou l'autre, nous ne pouvons pas négliger l'impact, souvent positif, de ces nouvelles technologies. Celles-ci nous ont déjà grandement aidés par le passé et vont encore nous faciliter la tâche à l'avenir, soit en automatisant beaucoup d'éléments, soit en offrant un nouveau point de vue sur le monde qui nous entoure.

Une des technologies qui a connu un réel essor ces dernières années est la réalité virtuelle (VR). Cette dernière se définit comme une expérience immersive où l'utilisateur se retrouve plongé dans un environnement virtuel grâce à un casque prévu à cette effet. Une autre technologie proche de la VR connaît aussi un franc succès : c'est la réalité augmentée (AR), définie comme une technologie qui enrichit le monde réel par du contenu numérique. En d'autres termes, l'AR permet d'avoir un contenu virtuel mixé avec notre réalité. L'AR a déjà conquis quelques domaines comme l'immobilier où elle permet d'afficher un modèle 3D à partir d'un plan en 2D. Plus récemment, le géant suédois de la vente de mobilier IKEA a sorti *IKEA Place*, une application qui aide à visualiser, grâce à la réalité augmentée, les futurs meubles de notre maison.

Figure 1 : Application *IKEA Place*

Source : <https://www.ikea.com/ch/fr/customer-service/mobile-apps/>

La figure 1 nous montre l'application *IKEA Place*. L'utilisateur fait usage de celle-ci afin de placer virtuellement un fauteuil dans son salon. Il décidera par la suite d'aller se procurer ce bien au magasin s'il est convaincu par celui-ci.

Un domaine encore trop peu étudié mais très prometteur pour l'AR est celui de l'éducation et de la culture. En effet, l'AR peut parfaitement répondre aux besoins de ce domaine en y rajoutant une plus-value. Nous allons aborder dans la suite de ce travail comment l'AR pourrait fonctionner dans le domaine de l'éducation.

1.2 Problématique

Le secteur de l'éducation essaie toujours de se renouveler afin de trouver la meilleure méthode d'enseignement. On peut notamment noter l'arrivée des tableaux blancs interactifs qui remplacent les tableaux noirs, ou encore l'implémentation de capsules vidéo reprenant le contenu du cours. Ces capsules ont prouvé, par exemple, leur plein potentiel durant la pandémie du COVID-19. Mais qu'en est-il de l'utilisation de l'AR dans l'éducation ?

Le potentiel de l'AR concernant l'enseignement est immense. L'AR nous offre de nouvelles possibilités d'apprentissage qui peuvent permettre aux cours d'être plus attrayants, et à certaines informations d'être plus compréhensibles. De nos jours, les jeunes possèdent, pour la plupart, un smartphone. L'utilisation de l'AR en est donc simplifiée puisque ces derniers sont habitués à employer des appareils où l'AR y est déjà bien présente. Celle-ci peut donc rendre le cours plus interactif. En effet, pour un cours d'anatomie, par exemple, l'AR permet d'explorer le corps humain d'une manière totalement inédite. Dès lors, toutes les parties du corps sont

modélisées en 3D. La visualisation de notre corps en est donc grandement améliorée et détaillée.

Pour de jeunes étudiants, il est souvent difficile de se projeter dans l'espace. Les concepts 3D posent parfois des problèmes car on n'arrive pas à se visualiser les choses que l'on ne peut pas voir. L'AR permet de résoudre une partie de ces problèmes en modélisant ces concepts 3D. En pouvant les manipuler sur une application, ces concepts sont plus faciles à appréhender.

Pour ce travail, il a été demandé de chercher et d'explorer les opportunités d'une application en réalité augmentée touchant le domaine des sciences. Après quelques discussions, nous avons trouvé que l'astronomie était un domaine intéressant pour apprendre tout en utilisant l'AR. En effet, il est difficile de se visualiser des concepts infiniment grands comme notre système solaire. En modélisant celui-ci sur une application utilisant l'AR comme technologie, les étudiants peuvent en apprendre davantage sur le fonctionnement de notre système solaire. Nous allons donc nous pencher sur l'élaboration d'une application qui nous permettra de visualiser notre système solaire.

1.3 Méthodologie

Pour ce travail, nous allons utiliser la méthodologie Agile qui correspond parfaitement à ce dont nous avons besoin. Pour la conception de l'application, nous utilisons le framework SCRUM. Ce dernier a la particularité de découper un projet en sprints. Un sprint est un cycle de développement d'un produit. Il est décomposé en quatre étapes : la première est la planification du sprint, dans laquelle nous énonçons les objectifs à atteindre à la fin de celui-ci ; la deuxième est une réunion intermédiaire, appelée « daily meeting », qui réunit les développeurs du projet afin de faire le point sur les tâches effectuées la veille, et celles qui sont planifiées pour la journée. La troisième tâche, appelée « sprint review », est une démonstration des nouvelles fonctionnalités de l'application qui aide à faire le point sur l'avancement du projet ; enfin, la dernière étape est la rétrospective du sprint qui permet aux développeurs d'échanger afin de ressortir les problèmes rencontrés pendant le sprint dans le but d'améliorer la cohésion de l'équipe. En raison du temps très court pour la conception de l'application, le travail a été divisé en deux sprints de deux semaines chacun pour un total de quatre semaines de travail sur l'application.

Pour la bonne réalisation du projet, et avant de commencer les sprints, nous avons dû réaliser un product backlog. Ce dernier se définit comme une liste de fonctionnalités qu'un produit doit contenir. On peut le voir comme une « to-do list ». Le product backlog a été réalisé

par nos soins. Il est important de noter que le product backlog est modifiable à n'importe quel moment lors de la réalisation de l'application. Tout au long du projet, il y a la possibilité de rajouter des précisions sur les fonctionnalités. Ces dernières peuvent être totalement modifiées à mesure que les intérêts changent au cours du développement.

En parallèle à cette méthodologie, un journal de bord hebdomadaire contenant les tâches accomplies chaque semaine ainsi qu'une estimation de temps ont été réalisés. Ce journal permet d'avoir un suivi sur l'avancement de l'application mais aussi du rapport écrit.

2 État de l'art

2.1 Solutions existantes

Il existe déjà plusieurs applications qui permettent de visualiser le système solaire et/ou chacune des planètes individuellement en réalité augmentée. Nous allons survoler quelques-unes de ces applications en dégagant leurs forces et leurs faiblesses.

2.1.1 Planets AR

Planets AR est une application gratuite qui permet de visualiser des planètes en AR. Disponible sur Google Play, Amazon Appstore et Galaxy Store, cette application donne le droit à l'utilisateur de voir les planètes de deux façons différentes : soit dans notre environnement grâce à la réalité augmentée, soit dans la nuit étoilée grâce à une vue en 3D. L'application se veut comme une expérience de réalité augmentée immersive pour aider les enfants à découvrir et à étudier les planètes du système solaire. L'application propose d'en apprendre davantage sur chaque planète grâce à un texte qui ajoute des informations sur la planète. (Planets AR, s.d.)

Figure 2 : Application Planets AR



Source : <https://play.google.com/store/apps/details?id=com.AgrMayank.PlanetsAR>

La simplicité d'utilisation de l'application en fait une de ses grosses forces. Cette application détient de nombreuses fonctionnalités intéressantes, comme le redimensionnement des planètes, par exemple. Cependant, elle ne possède pas de mode multijoueur comme prévu dans notre prototype.

2.1.2 AstroReality

AstroReality est une application qui fusionne des produits physiques avec la réalité augmentée. AstroReality nous permet d'explorer l'espace grâce à un très grand réalisme. Elle propose des modèles miniatures de planètes qu'il faudra scanner avec l'application afin de récolter une quantité astronomique d'informations concernant la planète scannée. AstroReality a pour mission d'inspirer autant de personnes que possible dans la découverte de l'espace grâce à la technologie innovante qu'est la réalité augmentée.

Figure 3 : Application AstroReality



Source : <https://www.realite-virtuelle.com/astroreality-nasa-contenu-ar/>

Le réalisme de cette application en fait son gros point fort. La qualité des modèles miniatures ainsi que celle de l'application permettent d'avoir des informations précises concernant l'astre ainsi qu'un niveau de réalisme quasi inégalable. Cependant, tout cela a un prix. En effet, si l'on veut acquérir un modèle de la planète terre, il faudra dépenser une somme de 99 dollars. Comme pour Planets AR, cette application ne possède pas de mode multijoueur. De plus, elle touche un public déjà averti et ne correspond pas à des enfants qui souhaiteraient découvrir le système solaire. (AstroReality, s.d.)

2.1.3 Cosmos Creator – AR Universes

Cosmos Creator est une application gratuite qui permet de créer son propre système solaire en AR. Cette application est disponible uniquement sur iOS. Cosmos Creator propose deux fonctionnalités principales. La première est l'apprentissage des planètes et des étoiles grâce à des textes détaillés disponibles sur l'application. La deuxième fonctionnalité, elle, nous permet de créer notre propre système solaire en y ajoutant l'astre que nous choisissons. Grâce

à cette application nous pouvons apprendre l'impact de la gravité sur les astres du système solaire. (AppAdvice, 2018)

Figure 4 : Application Cosmos Creator



Source : <https://appadvice.com/app/cosmos-creator-ar-universes/1274468408>

La grande force de cette application réside dans la création de systèmes. Avoir la possibilité de créer nous-même notre propre système solaire est idéal pour apprendre tout en restant ludique. Cette application n'est cependant pas très facile à prendre en main. Un long tutoriel doit être lu avant de l'utiliser. Bien qu'elle n'implémente pas de mode multijoueur, l'application n'en reste pas moins ludique, un aspect que nous recherchons aussi dans notre projet.

2.2 Technologies

Afin de mener à bien notre travail, nous allons analyser les différentes technologies existantes liées à la réalité augmentée. Cette analyse va nous aider à choisir la bonne technologie qui répondra au mieux à nos critères.

Pour ce travail, nous cherchons à travailler avec une technologie qui nous permet de visualiser, en réalité augmentée, des planètes ainsi qu'un système solaire complet. Cette technologie devra donc nous donner le droit de placer des objets ainsi que de se déplacer autour d'eux. Un des points les plus importants, si ce n'est le plus, est l'intégration d'une partie multijoueur. Il est souhaité, dans notre travail, qu'un mode multijoueur soit créé. La technologie choisie aura donc besoin d'avoir un support multijoueur.

2.2.1 Unity

Unity est une plateforme de développement, proposée par Unity Technologies. Elle est l'un des principaux programmes de conception de jeux vidéo. Ce logiciel permet notamment de créer des applications en VR et en AR. Il est sorti en 2005 et laisse la possibilité de créer des applications compatibles avec Windows, Mac, Linux, PlayStation, Xbox, Android, etc.

Unity utilise principalement le langage de programmation C#, qui a été appris pendant ce Bachelor. Ce logiciel est gratuit pour un usage personnel tant que l'on ne dépasse pas un revenu de cent milles dollars les douze derniers mois. Unity convient donc parfaitement à notre travail, car nous avons déjà pu l'utiliser dans notre cursus et que nous connaissons son langage de programmation. (Unity Technologies , 2022)

2.2.2 Vuforia

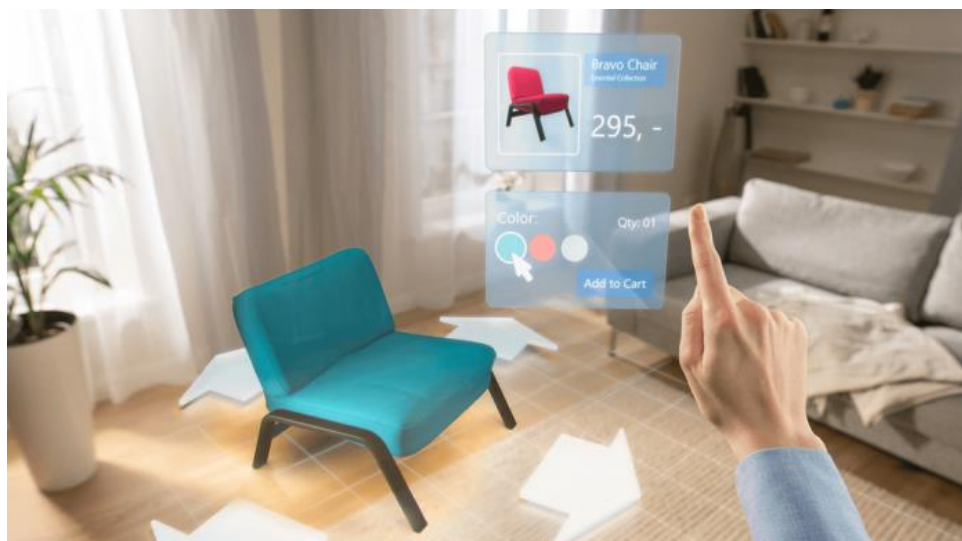
Vuforia Engine est l'un SDK les plus utilisés pour le développement en réalité augmentée. Un des avantages est la compatibilité avec la majorité des téléphones, des tablettes ou même des lunettes connectées. (PTC Inc., 2011)

Vuforia comporte une documentation détaillée et complète pour la création d'applications. Cette documentation est toujours mise à jour en intégrant des guides pratiques et des informations concernant les dernières fonctionnalités. De plus, l'outil est facilement intégrable à Unity via le gestionnaire de paquets de celui-ci. (PTC Inc., 2011)

Vuforia nous propose un plan gratuit BASIC contenant une licence qui débloque l'utilisation des fonctionnalités et des services Vuforia. On peut notamment compter parmi ses fonctionnalités, la détection grâce à des cibles d'image et des cibles multiples ou cylindriques ou même via le plan de sol. Cette licence gratuite nous permet aussi l'accès aux cibles d'image instantanée, à l'AR foundation (voir section 2.2.2) ou même à une reconnaissance d'images (jusqu'à 1000) dans le -Cloud. (PTC Inc., 2011)

Il existe une version premium qui contient notamment la possibilité de publier une application utilisant des Model Targets ou Area Targets. Les Model Targets permettent de créer une expérience de réalité augmentée pour de grands objets. Les Area Targets, quant à elles, offrent la possibilité de créer une expérience de réalité augmentée en réalisant un scan 3D d'un environnement intérieur. (PTC Inc., 2022)

Vuforia est un SDK intéressant car il propose le placement d'objets sur un plan horizontal comme le sol ou une table après l'avoir détecté. Cette fonctionnalité nous est utile dans notre

Figure 5 : Placement d'une chaise grâce à Vuforia

Source : <https://library.vuforia.com/environments/ground-plane>

projet, car nous voulons que notre application détecte le sol ou une table afin de pouvoir déposer une planète dessus. (PTC Inc, 2011)

Vuforia ne comporte pas directement l'implémentation du multijoueur. Il est nécessaire d'installer le framework PUN 2 (Photon Unity Network). Grâce à ce framework, nous sommes capables de développer du multijoueur dans notre application, mais il n'est pas directement intégré dans Vuforia. Il existe néanmoins une bonne documentation ainsi que des tutoriels concernant l'implémentation de ce package dans une application. (Photon, s.d.)

2.2.3 AR Foundation

AR Foundation est un framework conçu par Unity pour le développement en réalité augmentée. Très facilement implémentable dans Unity via le gestionnaire de paquets, ce framework comporte les fonctionnalités principales d'AR Core (section 2.2.2.1) et d'AR Kit (section 2.2.2.2). Il est compatible avec Android et IOS, ainsi qu'avec les lunettes connectées Magic Leap et HoloLens. (Unity, 2022)

Il est important de saisir que l'AR n'implémente pas elle-même des fonctionnalités de réalité augmentée, mais qu'elle nous présente une interface pour Unity. Les fonctionnalités seront décrites lorsque nous parlerons d'ARCore et d'ARKit.

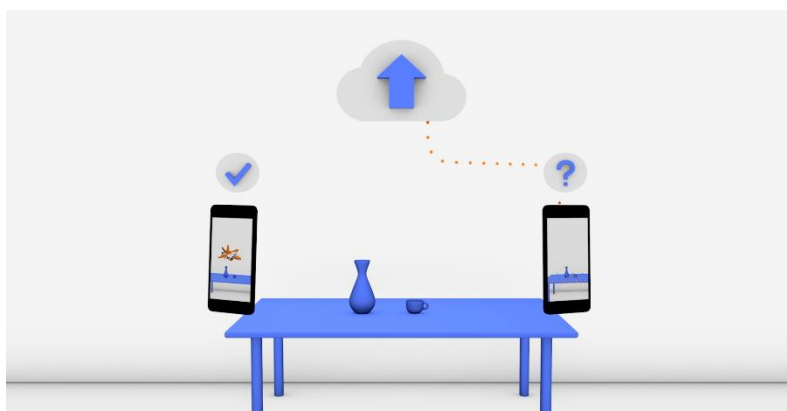
2.2.3.1 ARCore

ARCore est la plateforme de réalité augmentée conçue par Google. Grâce à différentes API, elle permet à nos téléphones de détecter notre environnement et d'interagir avec. Cette plateforme nous propose trois fonctionnalités importantes pour intégrer la réalité augmentée

dans nos applications : le suivi de mouvement qui permet au téléphone de se situer par rapport au monde, la compréhension de l'environnement qui offre la possibilité de détecter tout type de surfaces qu'elles soient horizontales, verticales ou même inclinées, et l'estimation de la lumière qui de déterminer avec plus ou moins de précision la quantité de lumière dans l'environnement. (Google, 2022)

ARCore répond donc à notre besoin concernant le placement dans l'environnement, mais qu'en est-il du multijoueur ? ARCore propose une expérience multijoueur grâce à l'API Cloud Anchor. Cette dernière est une ancre qui conserve les expériences de réalité augmentée dans le monde réel. Elle permet d'ancrer des couches d'informations pouvant contenir des objets à des emplacements réels, par exemple. Cet ancrage est stocké dans le Cloud et peut être ensuite récupéré par plusieurs utilisateurs en même temps. Pour utiliser cette API, il suffit de l'activer dans un projet Google Cloud Platform et dans la configuration de session de réalité augmentée. Il nous faudra enfin nous authentifier afin d'avoir accès à l'hébergement dans le Cloud. ARCore intègre donc l'expérience multijoueur. Il est nécessaire néanmoins d'exécuter quelques étapes avant de l'intégrer complètement. (Google, 2022)

Figure 6 : Les Cloud Anchors selon ARCore



Source : <https://developers.google.com/ar/develop/cloud-anchors>

2.2.3.2 ARKit

ARKit est très proche d'ARCore et vient de sortir sa nouvelle version, l'ARKit 6. Ce framework, créé par Apple, introduit notamment la vidéo 4K, qui nous permet de capturer des vidéos à très haute résolution. ARKit propose une API de profondeur qui donne la possibilité à notre application d'utiliser au mieux les informations concernant la profondeur d'un environnement. Cette API rend le placement d'objet encore plus réaliste que jamais. Cette fonctionnalité requiert le scanner LiDAR, équipé sur les iPad Pro 11 pouces de deuxième génération, les iPad 12,9 pouces de quatrième génération, les iPhone 12 Pro ainsi que les 12 Pro Max. L'ARKit 6 permet également la capture de mouvement. En effet, cette nouvelle génération du framework capture le mouvement d'une personne en temps réel. Ces

mouvements peuvent être pris comme une entrée pour les expériences de réalité augmentée. ARKit 6 offre la possibilité de créer une carte topologique de notre environnement en étiquetant les sols, murs plafonds, portes, etc. Cette cartographie nous donne plus d'informations concernant l'environnement scanné afin d'améliorer notre expérience de réalité augmentée. Ce framework propose de nombreuses autres améliorations comme l'Instant AR ou la People Occlusion que nous n'allons pas détailler dans ce rapport. (Apple, 2022)

ARKit ne contient pas d'intégration multijoueur sur Unity, tout comme AR Core. Cependant, il existe un SDK du nom de Placenote qui permet l'implémentation du multijoueur sur iOS grâce à ARKit. Nous allons décrire le SDK Placenote dans la section 2.2.2 plus tard durant ce rapport. (Mathew, 2018)

2.2.4 Lightship

Lightship est un kit de développement de réalité augmentée créé par Niantic. Ce dernier est connu pour avoir développé Pokémon GO, l'application de réalité augmentée considérée comme une révolution, qui a animé l'été 2016. C'est un des premiers jeux grand public à utiliser la réalité augmentée couplée à la géolocalisation. (Niantic, 2021)

Figure 7 : Pokemon GO



Source: <https://pokemongolive.com/>

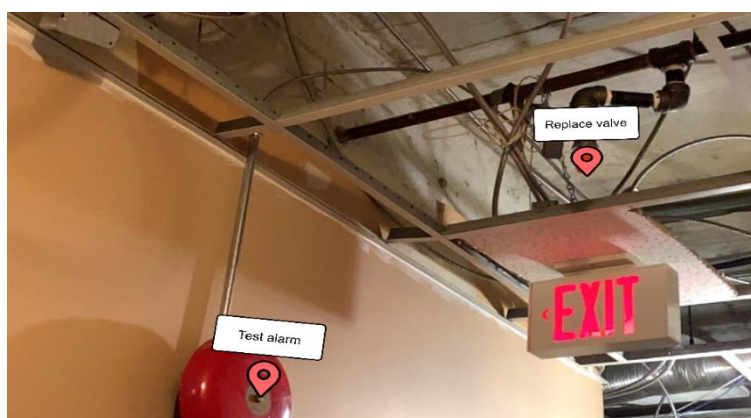
Lightship offre trois améliorations notables : la cartographie en temps réel ; l'amélioration de la segmentation sémantique qui permet de dissocier le plus précisément possible les différentes surfaces ou couches de notre monde ce qui accorde à la réalité augmentée la possibilité d'interagir avec des surfaces spécifiques ; et enfin, la fiabilité du multijoueur. Niantic met un point d'honneur à améliorer l'expérience multijoueur. Lightship propose des API de miss en réseau qui vont permettre la synchronisation du contenu et des joueurs ainsi que des sessions allant jusqu'à 5 joueurs simultanés. (Niantic, 2021)

Ce SDK convient parfaitement à nos attentes. Grâce à sa partie multijoueur bien développée, Lightship répond à notre besoin concernant notre partie multijoueur. Il fonctionne autant bien sur Android que sur iOS et est facilement importé dans Unity. Il en fait donc un SDK de premier choix pour la réalisation de notre travail. L'offre gratuite de Lightship comprend toutes les fonctionnalités du SDK. La seule contrainte se situe au niveau de l'expérience multijoueur où le transfert de données pendant une session multijoueur est limité à 50 Mo. Cette contrainte ne devrait pas affecter notre travail. Notons aussi que Lightship propose une documentation complète ainsi que des tutoriels afin de bien prendre en main le SDK. (Niantic, 2022)

2.2.5 Placenote

Placenote est un SDK gratuit disponible uniquement sur iOS. La grande force de ce dernier est sa capacité à scanner l'environnement. En effet, l'utilisateur de Placenote peut scanner son appartement afin d'avoir un modèle 3D complet de celui-ci. Ce SDK permet aussi le partage en réseau de différents objets. Une des fonctionnalités que Placenote met en avant est l'utilisation de « Spatial notes », des notes que l'on pose à n'importe quel endroit de notre environnement. Ces notes peuvent être ensuite partagées à d'autres utilisateurs. (Placenote, s.d.)

Figure 8 : Les Spatial notes utilisées par Placenote



Source : <https://placenote.com/annotations>

Ce SDK est une solution pour le développement d'une application multijoueur basée sur iOS. Placenote est cependant moins complet que les précédents SDK que nous avons analysés. Il contient néanmoins une bonne documentation ainsi que quelques tutoriels. Un de ses gros points forts est la présence de quelques projets d'exemples open source qui permettent de bien se familiariser avec le SDK. (Placenote, 2020)

2.3 Choix technologique

Notre choix de technologie se porte sur le SDK de Niantic, Lightship. Ne possédant pas d'ordinateur tournant sur iOS, ni de smartphone ou tablette de ce même système d'exploitation, nous laissons de côté, pour ce projet, les SDK ARKit et Placernote. Comme nous voulons un SDK qui intègre directement la fonctionnalité de multijoueur, nous abandonnons Vuforia, qui nécessite un framework afin de faire fonctionner le multijoueur. Il nous reste donc le choix entre Lightship et ARCore. Les deux se valent, mais par conseil de notre professeur, nous allons utiliser le SDK Lightship. L'avantage de ce kit de développement est sa nouveauté. En effet, Lightship a débarqué tout fraîchement sur le marché de l'AR. Il sera donc intéressant de découvrir ce SDK à travers ce projet pour en dégager ses forces et ses faiblesses pour de potentiels futurs projets.

Figure 9 : Logo du SDK Lightship



Source : <https://nianticlabs.com/news/lightship/?hl=fr>

3 Résultats

Cette partie du travail traitera de tout ce qui concerne l'application que nous devons créer. Nous allons notamment survoler ses principales fonctionnalités.

L'application dispose de trois écrans avec des fonctionnalités différentes. Le premier offre la possibilité de placer n'importe quelle planète du système solaire dans notre environnement et de la regarder dans tous les sens possibles. Le deuxième nous montre le système solaire complet. Enfin, le dernier est un jeu multijoueur où deux joueurs s'affrontent dans le système solaire.

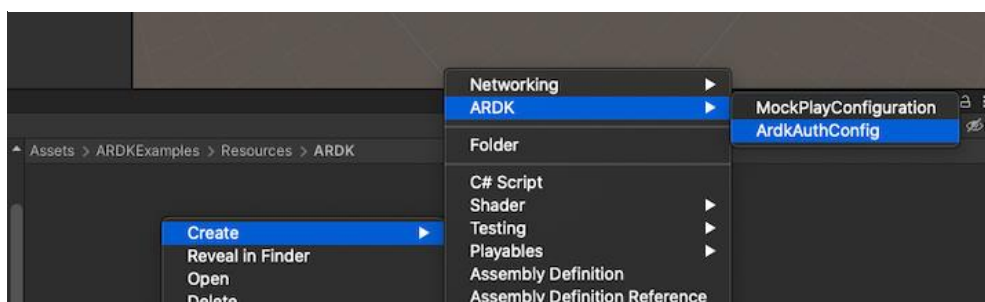
3.1 Installation du SDK et mise en place de l'environnement

Pour ce projet, nous utilisons la version 2021.3.4f1 de Unity ainsi que Visual Studio 2022 pour la conception des scripts de notre application.

Afin de bien commencer le projet, il est nécessaire d'installer et de configurer le SDK. Nous avons donc, dans un premier temps, téléchargé la version 2.0.0 du SDK.

Pour utiliser l'entièreté des fonctionnalités de l'ARDK, nous devons générer une clé API de l'ARDK. Pour ce faire, nous avons créé un compte sur le site de Lightship afin de récupérer la clé. Ensuite, nous avons créé un fichier Resources/ARDK dans notre projet. C'est dans ce dossier que nous allons retrouver la clé. Pour utiliser et activer cette clé, nous devons ajouter un ArdAuthConfig dans ce dossier. C'est dans ce ArdAuthConfig que nous mettons notre clé API.

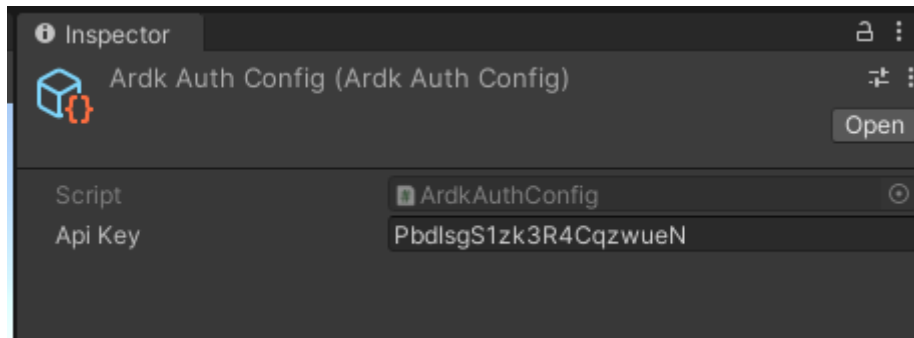
Figure 10 : Création de l'ArdAuthConfig



Source : Données de l'auteur

Cette clé nous permettra d'utiliser toutes les fonctionnalités utiles pour le multijoueur. Attention, les noms ainsi que la clé sont sensibles à la casse. Il est donc nécessaire de vérifier que l'arborescence des fichiers soit correctement écrite et il faut bien contrôler que la clé ne contienne aucun espace avant ou après celle-ci.

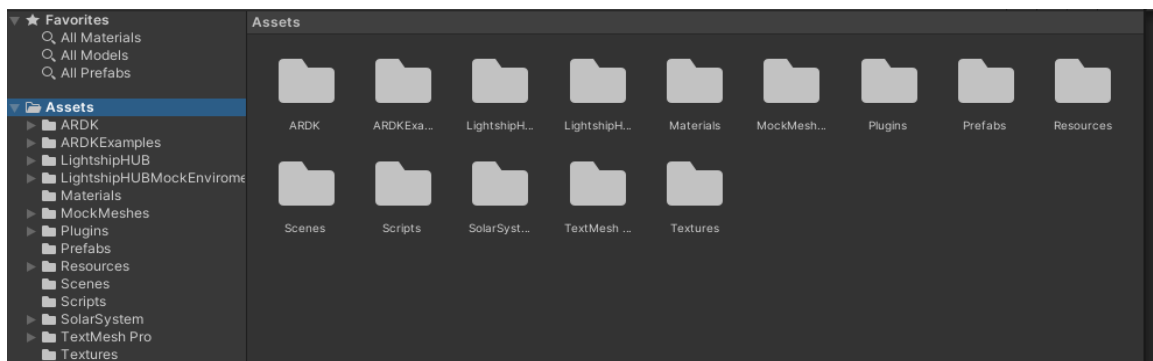
Figure 11 : Clé API du projet



Source : Données de l'auteur

Il n'y a besoin que d'une seule et unique clé pour l'application. Nous avons dû supprimer un ArdkAuthConfig déjà existant sous le dossier ARDKExamples.

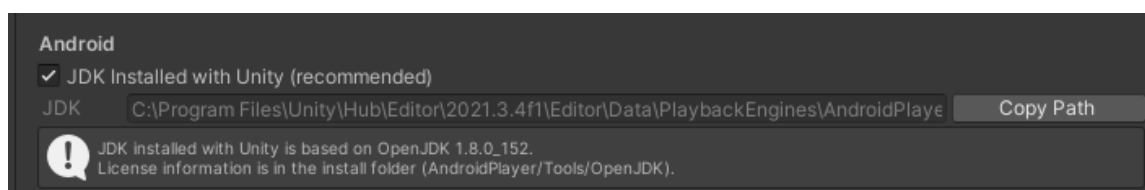
Figure 12 : Arborescence du projet Unity



Source : Données de l'auteur

Voici l'arborescence de notre projet. ARDK, ARDKExamples, LightshipHUB et LightshipHUBMockEnvironnement sont des assets tirés du SDK de Niantic. SolarSystem est un asset récupéré sur le magasin d'asset d'Unity, créé par Adam Bielecki (Bielecki, 2021). Ce sont les cinq assets que nous avons téléchargés.

Afin de pouvoir tester notre application sur notre smartphone, nous effectuons quelques changements. Premièrement, nous spécifions le JDK d'Android dans les préférences d'Unity.

Figure 13 :Emplacement du JDK

Source : Données de l'auteur

Par la suite, nous modifions les Player Settings. Nous changeons le Company Name au nom de la HES-SO Valais-Wallis. Ensuite, nous supprimons Vulkan des Graphics APIs, choisissons IL2CPP en ce qui concerne le Scripting Backend, et enfin nous cochons ARMv7

Figure 14 : Section à ajouter dans le fichier AndroidManifest.xml

```
<queries>
  <package android:name="com.google.ar.core" />
</queries>
```

Source : Données de l'auteur

et ARM64 pour les Target Architectures. Comme nous travaillons avec un téléphone utilisant Android 12, nous faisons une modification en plus. Pour tous les téléphones qui utilisent Android 11 ou plus, nous devons ajouter un AndroidManifest.xml personnalisé. Pour ce faire, nous devons, sous Publishing Settings toujours dans les Player Settings, activer l'option Custom Main Manifest. Cette option crée un AndroidManifest.xml dans notre projet dans le dossier Assets/Plugins/Android. Nous modifions ce fichier en y ajoutant cette section à la fin du fichier.

Cette section doit se situer juste avant la balise </manifest>. Après avoir modifié ce fichier, nous pouvons « build » notre application sur notre téléphone. Notre environnement de travail est désormais en place.

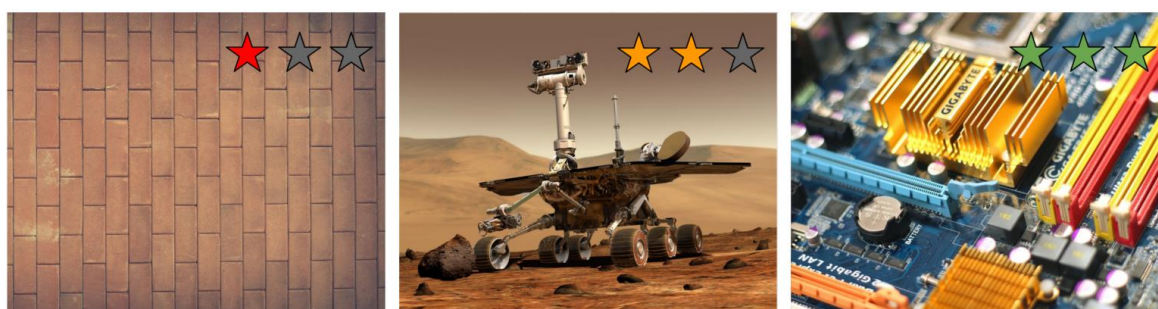
3.2 Placement d'objets

La première étape de ce projet est de pouvoir poser un objet virtuel comme une planète ou le système solaire dans notre environnement réel. Pour ce faire, Lightship nous propose quelques scripts déjà existants afin de réaliser au mieux cette fonctionnalité.

Afin de placer un objet virtuel dans un monde réel, Lightship offre deux techniques différentes. Il y a tout d'abord la technique d'apparition de l'objet via une image que l'on détecte grâce à notre caméra. Cette technique contient quelques limites. En effet, il est nécessaire de choisir une image qui soit de bonne qualité pour Lightship. Pour que ce soit le cas, il est

préférable qu'elle ne contienne pas de motifs répétitifs et qu'elle possède de forts contrastes de couleur.

Figure 15 : Comparatif de la qualité de détection de l'image

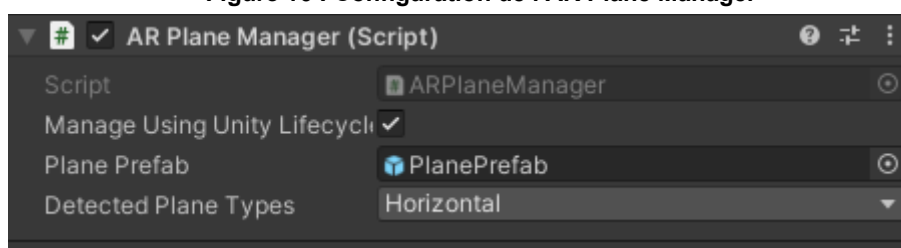


Source : <https://www.wikitudo.com/blog-image-recognition-tracking-best-practices-and-target-guidelines/>

La figure 14 nous montre de bonnes et de mauvaises images en ce qui concerne la détection d'image. Comme nous voulons nous déplacer autour des planètes ou du système solaire, ce mode de placement n'est pas adapté à notre situation, car si nous nous déplaçons, nous risquons de voir notre objet virtuel disparaître. Nous utilisons donc la deuxième méthode pour placer un objet virtuel dans notre monde.

Cette méthode, appelée Detecting Planes, permet de détecter et d'interagir directement avec le monde réel afin d'obtenir un rendu d'AR plus immersif que jamais. Cette technologie utilise les ancres (ancres). Une anchor peut correspondre à tout ce que l'on trouve dans notre environnement physique. Notre caméra va donc tout scanner afin d'obtenir et de créer un champ d'action pour placer notre contenu virtuel. De plus, nous pouvons spécifier le type de plan que notre application détecte. Ce type peut être soit vertical, soit horizontal, soit les deux ou soit aucuns. Pour cette application, nous décidons d'appliquer la détection uniquement horizontalement. De ce fait, nous pouvons placer nos planètes ou notre système solaire sur une surface plane comme le sol ou une table. Il sera plus judicieux pour les élèves de placer leurs objets devant eux sur un bureau plutôt que sur un mur. Ainsi, ils pourront plus facilement se déplacer afin de voir tous les angles des objets placés.

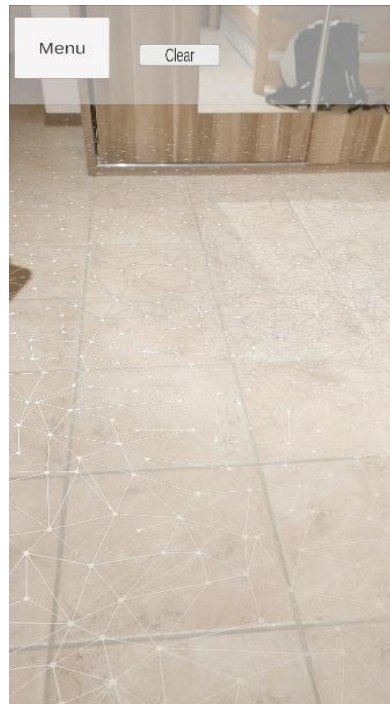
Figure 16 : Configuration de l'AR Plane Manager



Source : Données de l'auteur

La figure 15 nous montre le script fourni par Lightship qui concerne la détection de plan. Nous voyons que nous avons sélectionné uniquement le type horizontal. De plus, le script demande d'ajouter un prefab pour le plan. Cet objet nous sera utile pour visualiser les anchors créées par l'application afin de pouvoir placer au bon endroit notre objet virtuel.

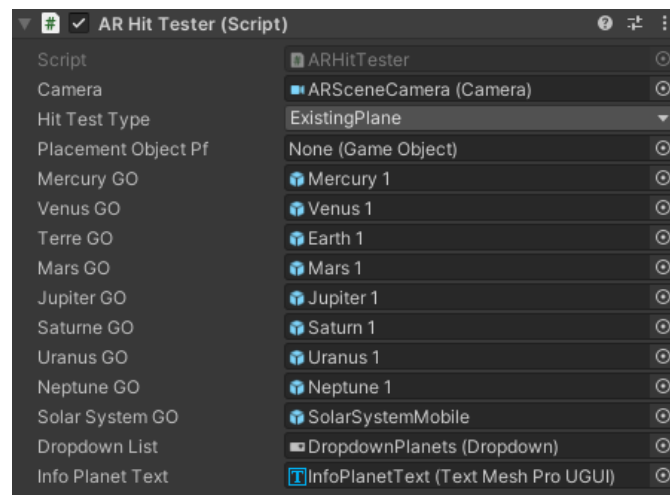
Figure 17 : Visualisation du plan grâce à l'AR Plane Manager



Source : Données de l'auteur

Maintenant que nous avons vu comment notre application détecte le plan afin de poser nos objets virtuels, nous allons regarder comment fonctionne le placement d'objet virtuel avec cet ARDK. Afin de placer un objet virtuel dans notre environnement, Lightship nous propose un script afin de nous faciliter la tâche. Ce script se nomme AR Hit Tester et permet de placer n'importe quel objet sur notre surface détectée auparavant.

Figure 18 : Configuration de l'AR Hit Tester



Source : Données de l'auteur

La figure 17 décrit la configuration du script AR Hit Tester. Tout d'abord, ce script nécessite la caméra d'AR. Le Hit Test Type est directement en corrélation avec le plan que nous avons détecté auparavant. Ici, nous choisissons l'option ExistingPlane, qui correspond à un plan d'anchor déjà présent dans la scène. Ce plan est le plan existant détecté par notre caméra, comme nous pouvons le voir sur la figure 16. Ensuite, nous attribuons nos GameObjects correspondants à chaque variable. La variable Placement Object Pf prendra une valeur différente dépendant de la scène où nous nous situons mais aussi de la planète sélectionnée dans la Dropdown List. Si nous sommes dans la scène où nous pouvons voir une seule planète, nous instancierons mercure comme objet de base. Si nous nous situons dans la scène où nous observons le système solaire, ce sera celui-ci qui sera défini comme objet de base.

Figure 19 : Attribution de la variable placementObjPf

```
Scene scene = SceneManager.GetActiveScene();
if (scene.name == "SinglePlanet")
    _placementObjectPf = mercuryGO;
else if (scene.name == "SolarSystemScene")
    _placementObjectPf = solarSystemGO;
```

Source : Données de l'auteur

Pour placer notre objet, il faut que notre application détecte le moment et l'endroit de l'écran que nous touchons avec notre doigt ou notre stylet. Pour détecter le toucher, l'ARDK utilise une classe nommée PlatformAgnosticInput. Cette classe va simplement détecter toute touche sur notre écran que ce soit avec la souris sur navigateur ou avec notre doigt sur notre téléphone. Dès que nous détectons une touche, nous allons appeler une méthode qui va s'occuper de la gestion de cette touche.

Figure 20 : Gestion du premier toucher de l'écran

```
var touch = PlatformAgnosticInput.GetTouch(0);
if (touch.phase == TouchPhase.Began)
{
    if (EventSystem.current.IsPointerOverGameObject())
        return;

    if (!_isPlanet)
    {
        TouchBegan(touch);
    }
}
```

Source : Données de l'auteur

La méthode `TouchBegan` va s'occuper de la gestion de la touche. Dans un premier temps, nous allons récupérer la frame sur laquelle la touche a été faite. Nous appelons une méthode `HitTest` sur cette frame pour trouver la position exacte du toucher. Cette méthode prend en paramètre la hauteur et la largeur de notre caméra, la position de la touche dans l'espace de l'écran en pixels et le `HitTestType` défini plus haut. Cette méthode nous retourne un tableau des touches du plus proche au plus loin du point d'impact de l'écran.

Figure 21 : Méthode `TouchBegan`

```
var currentFrame = _session.CurrentFrame;
if (currentFrame == null)
{
    return;
}

var results = currentFrame.HitTest
(
    _camera.pixelWidth,
    _camera.pixelHeight,
    touch.position,
    hitTestType
);
```

Source : Données de l'auteur

Ensuite, nous allons récupérer la première valeur de ce tableau, celle qui correspond à l'endroit le plus proche du toucher. Ce point va être transformé en une position, c'est-à-dire que ce point va être transformé afin d'être placé dans un monde en trois dimensions. Une fois fait, il nous reste plus qu'à appeler la méthode `Instantiate` de Unity, qui nous permet d'instancier notre objet à la position récupérée.

Figure 22 : Instanciation de l'objet

```
// Get the closest result
var result = results[0];

var hitPosition = result.WorldTransform.ToPosition();

_placedObjects.Add(Instantiate(_placementObjectPf, hitPosition, Quaternion.identity));
_isPlanet = true;
```

Source : Données de l'auteur

L'objet que nous avons sélectionné est maintenant visible à l'endroit où nous avons touché notre écran. Cependant, un problème subsiste. Les planètes étant rondes, la moitié de celles-ci se situe dans le sol, ce qui n'est pas très pratique ni esthétique pour pouvoir bien les observer. Pour pallier ce problème, nous avons rajouté cette ligne à la figure 21 juste avant d'instancier l'objet.

Figure 23 : Ajustement de la hauteur de l'objet instancié

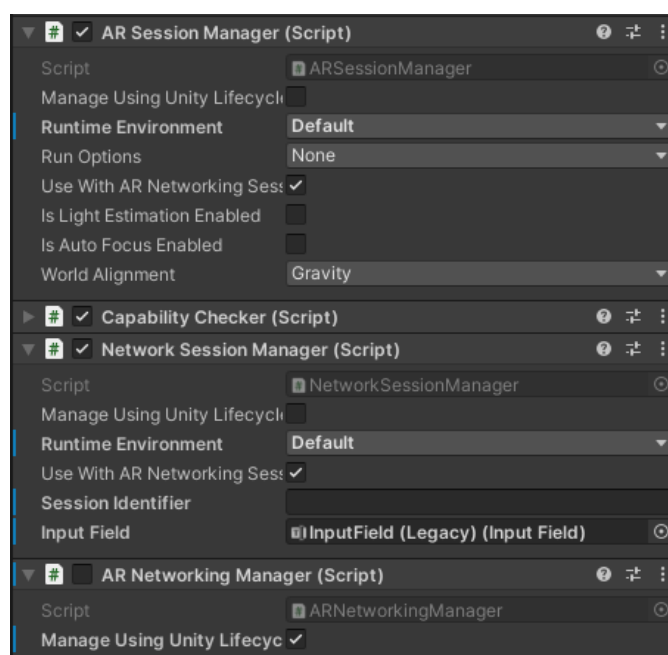
```
hitPosition.y += _placementObjectPf.transform.localScale.y;
```

Source : Données de l'auteur

Cette ligne nous permet de rajouter, pour la coordonnée y, la taille de la planète sélectionnée. De ce fait, la planète se situe correctement sur notre plan et non pas coupée en deux.

3.3 Multijoueur

Le multijoueur est une des fonctionnalités importantes que prône Lightship. Grâce à leur ARDK, nous sommes capables de construire une application possédant un espace multijoueur afin d'augmenter l'interactivité entre les joueurs. Comme pour le placement d'objet, l'ARDK nous propose plusieurs scripts existants qui vont nous aider à la conception du multijoueur. Afin de bien commencer la réalisation de cette fonctionnalité, il faut dans un premier temps ajouter le manager ARNetworkingSceneManager. Ce manager contient tous les scripts utiles à la conception du réseau multijoueur ainsi que la caméra qui sera utilisée. L'ARNetworkingSceneManager contient quatre scripts : l'ARSessionManager, le CapabilityChecker, le NetworkSessionManager ainsi que l'ARNetworkingManager. Ces scripts sont nécessaires pour la mise en réseau ainsi que pour la localisation.

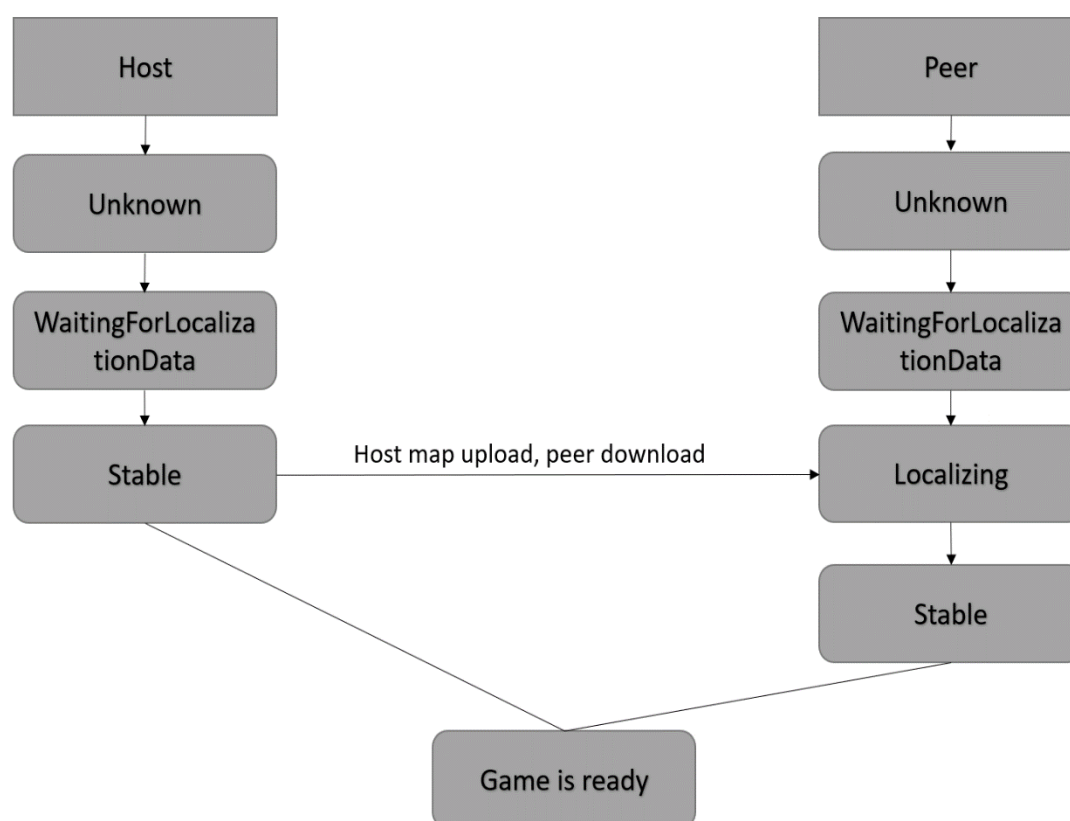
Figure 24 : Configuration de l'ARNetworkingSceneManager

Source : Données de l'auteur

Il y a quelques points à noter dans cette figure. Dans le script `ARSessionManager`, il est important de vérifier que la case `Use With AR Networking Session` soit cochée. Cette coche assure que la session d'AR communique bel et bien avec le réseau contenu dans le script `NetworkSessionManager`. Sous celui-ci, nous attribuons le champ `Input field` à notre champ d'entrée. Ce champ représente l'identifiant qui sera utilisé lors de la mise en réseau de l'application. Lorsque l'hôte de la partie choisira cet identifiant, il faudra qu'il le communique à son partenaire de jeu afin qu'il puisse rejoindre la même session de jeu, sans quoi les deux personnes créeront deux sessions multijoueur différentes.

Pour que les deux joueurs interagissent dans le même monde de réalité augmentée, ils doivent partager les informations de leur session. La connexion entre les deux joueurs est gérée par le manager utilisé plus haut. Nous pouvons posséder deux rôles lors de la création d'une session de mise en réseau : le rôle d'hôte (`host`) ou celui de pair (`peer`). Lorsque l'on crée une session pour la première fois, nous sommes désignés comme hôte. Lorsqu'au contraire, nous rejoignons une session déjà existante, nous sommes désignés comme pair. Pour rejoindre la même session, nous devons utiliser le même identifiant de session ainsi que la même clé API décrite sous la section 3.1. Une fois que la connexion est faite entre les deux utilisateurs, notre application localise notre environnement afin de vérifier l'emplacement des deux joueurs et de pouvoir, par la suite, placer un objet virtuel au même endroit pour les deux utilisateurs. Pour ce faire, `Lightship` nous offre deux possibilités, la localisation grâce à un environnement commun ou grâce à des marqueurs visuels tel qu'un code QR par exemple. Nous utilisons, dans ce projet, la localisation grâce à un environnement commun, afin de rester en adéquation avec la méthode de détection décrite dans la section 3.2.

Afin de synchroniser l'environnement entre les deux joueurs, l'hôte doit scanner un objet statique dans l'environnement. Il apparaîtra sous le statut de `WaitingForLocalizationData`. Une fois que l'objet aura correctement été scanné par l'application, il sera envoyé au Niantic backend et l'hôte aura le statut de `Stable`. Le pair, de son côté, aura le statut de `WaitingForLocalizationData` tant que l'hôte n'aura pas fini de scanner son environnement. Une fois les données envoyées à Niantic, le pair les téléchargera et passera sous le statut de `Localizing`. Il devra à son tour scanner l'objet que l'hôte aura scanné. Une fois l'objet scanné et reconnu par la session, le pair aura le statut de `stable` et la session multijoueur pourra démarrer correctement. Les deux utilisateurs partageront le même environnement et pourront se localiser dans le monde partagé entre les deux joueurs.

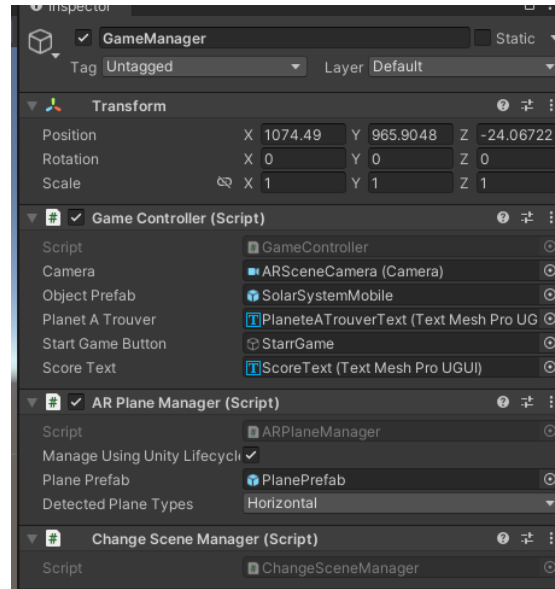
Figure 25 : Les différents états des participants lors d'une session multijoueur


Source : Données de l'auteur, inspiré de la vidéo
https://www.youtube.com/watch?v=mjB4_KNjXkw

Maintenant, nous allons nous pencher sur la question du partage et de l'interaction d'un objet tel que le système solaire entre les deux utilisateurs. Il faut que lorsqu'un utilisateur effectue une action lors de la session, l'autre utilisateur la reçoive. Pour cette étape de communication entre les joueurs, l'ARDK propose deux APIs : la High-Level API (HLAPI) et la Low-Level API (LLAPI). La HLAPI est utilisée si nous avons besoin d'une création et d'une suppression d'objets chez tous les utilisateurs ou alors d'une manipulation de transformations d'objets. HLAPI est aussi employée lorsque nous voulons définir des droits de suppression ou d'ajout aux utilisateurs présents dans la session multijoueur. La LLAPI est quant à elle utilisée principalement pour prioriser une messagerie efficace. Cette API est employée lorsque nous voulons créer un seul objet. LLAPI offre de meilleures performances au niveau du réseau, ce qui peut nous être utile afin d'avoir une expérience multijoueur la plus performante possible. Dans ce projet, nous manipulons uniquement un seul objet, à savoir le système solaire, nous allons donc utiliser la LLAPI. Il est important de noter que l'ARDK possède un serveur en backend qui fournit la gestion des sessions, l'apparition des pairs et toute l'architecture concernant l'acheminement des messages.

Afin d'utiliser la LLAPI, nous créons un script qui contient toute la logique du jeu multijoueur. Ce script se situe dans un GameObjet comportant notamment le script de PlaneManager que nous avons déjà utilisé auparavant.

Figure 26 : Configuration du GameManager



Source : Données de l'auteur

Le script GameController s'occupera d'établir la logique de la session multijoueur. De plus, un script nommé MessageManager est créé. Ce dernier, s'occupera principalement de la communication entre les joueurs. Pour commencer, nous allons créer une session multijoueur.

Figure 27 : Initiation de la session multijoueur

```
// Start is called before the first frame update
Message Unity | 0 références
private void Start()
{
    startGameButton.SetActive(false);
    ARNetworkingFactory.ARNeworkingInitialized += OnAnyARNetworkingSessionInitialized;
}
```

Source : Données de l'auteur

Dès que nous avons initialisé une session d'AR en réseau, nous appelons une autre méthode qui nous permettra de notifier différents aspects.

Figure 29 : Déclaration de variables

```
private IARNetworking _arNetworking;  
  
private MessageManager _messagingManager;
```

Source : Données de l'auteur

Figure 28 : Méthode gérant les connexions

```
private void OnAnyARNetworkingSessionInitialized(AnyARNetworkingInitializedArgs args)  
{  
    _arNetworking = args.ARNetworking;  
    _arNetworking.PeerStateReceived += OnPeerStateReceived;  
  
    _arNetworking.Networking.Connected += OnDidConnect;  
  
    _messagingManager = new MessageManager();  
    _messagingManager.InitializeMessagingManager(args.ARNetworking.Networking, this);  
}
```

Source : Données de l'auteur

Cette méthode fait appel à plusieurs méthodes. OnDidConnect nous dira notamment si nous sommes l'hôte de la session. OnPeerStateReceived permettra de récupérer l'état (voir figure 24) de notre connexion et d'activer le bouton pour commencer le jeu une fois que tous les utilisateurs se situeront à l'état stable. Enfin, nous initialiserons notre manager en ce qui concerne les futurs messages communicants entre les deux utilisateurs. Une fois l'état stable atteint pour les deux utilisateurs, cela veut dire que nous partageons le même environnement et que nous pouvons instancier notre système solaire.

Pour instancier notre système solaire, il faut que notre application communique entre les différents utilisateurs. Cette communication est principalement gérée par le script MessageManager. Le point principal de la communication via le réseau est centré autour des messages que nous envoyons et que nous recevons. Avec la LLAPI, les messages doivent contenir un uint, qui correspond au tag du message, et un tableau de byte contenant les données du message à envoyer. Afin de transmettre les données à tous les utilisateurs, nous utilisons la méthode BroadcastData.

Figure 31 : Envoi des données via BroadcastData

```
internal void SpawnGameObjects(Vector3 position)
{
    var data = SerializeVector3(position);
    Debug.Log("Position : " + position);

    _networking.BroadcastData
    (
        0,
        data,
        TransportType.ReliableOrdered
    );
}
```

Source : Données de l'auteur

Dans la figure 29, les données que nous voulons envoyer sont un vecteur3 correspondant à une position à trois dimensions. Cette position est celle à laquelle notre système solaire sera instancié via le GameController. Le message que nous envoyons a pour tag 0. Le type de transport est défini à ReliableOrdered. Cela signifie que le message que nous envoyons est garanti d'arriver à tous les utilisateurs dans l'ordre dans lequel il a été envoyé. Ce type de transport n'est pas le plus rapide mais il offre la garantie que notre position soit bel et bien reçue. Enfin, afin que notre message passe par le réseau avec les bonnes données, il est nécessaire de transposer nos données dans un tableau de bytes afin qu'il puisse traverser le réseau. Pour ce faire, l'ARDK nous offre des éléments qui nous permettent de sérialiser facilement notre position.

Figure 30 : Sérialisation d'un vecteur

```
GlobalSerializer.Serialize(_builderMemoryStream, vector);
return _builderMemoryStream.ToArray();
```

Source : Données de l'auteur

La figure 30 nous montre la sérialisation d'un vecteur en 3D (vector) en un tableau de bytes.

Une fois le message envoyé dans le réseau avec succès, nous devons informer les pairs de la venue d'un message. L'ARDK contient un évènement lorsqu'un pair reçoit n'importe quelles données. Celui-ci se nomme PeerDataReceived et va ajouter les éléments reçus via le réseau grâce à une méthode appelée OnDidReceiveDataFromPeer. Cette méthode va gérer tous les messages reçus lors de notre session multijoueur.

Dans cette méthode, nous récupérons les données du message qui nous a été envoyé et nous stockons sa structure dans la variable args. De cette variable, nous pouvons en sortir le tag et bien sûr le tableau de bytes, l'essence même du message, grâce à un CopyData. Comme notre application doit gérer plusieurs messages, nous utilisons un switch case. Dans ce cas, nous l'avons vu précédemment, le message envoyé a comme tag 0. Nous récupérons donc le message si son tag est zéro en désérialisant le message. La variable data ici est un

tableau de bytes. Nous avons de base un vecteur 3D. Il faut donc passer à cette variable la méthode `DeserializeVector3` qui va désérialiser notre tableau de bytes en vecteur 3D. De ce fait, tous les pairs ont reçu la position envoyée auparavant.

Figure 32 : Réception des données envoyées à travers le réseau

```
private void OnDidReceiveDataFromPeer(PeerDataReceivedArgs args)
{
    Debug.Log("Received message with tag: " + args.Tag);

    var data = args.CopyData();
    switch (args.Tag)
    {
        case 0:
            Debug.Log("Creating game objects");
            _controller.InstantiateObjects(DeserializeVector3(data));
            break;
    }
}
```

Source : Données de l'auteur

C'est la méthode `InstantiateObjects` qui nous permet d'instancier notre système solaire. Le placement s'effectue comme dans la section 3.2 à quelques détails près. Tout d'abord, nous autorisons la touche de l'écran pour obtenir la position future de notre système solaire une fois que tous les utilisateurs ont été stables et nous limitons la capacité de création du système solaire uniquement à l'hôte.

Figure 33 : Gestion du toucher d'écran

```
if (_isSynced && !_isGameStarted && _isHost)
{
    if (PlatformAgnosticInput.touchCount <= 0)
        return;

    var touch = PlatformAgnosticInput.GetTouch(0);
    if (touch.phase == TouchPhase.Began)
    {
        OnTapScreen(touch);
    }
}
```

Source : Données de l'auteur

De plus, le jeu ne doit pas être démarré afin de pouvoir instancier notre système solaire. Une fois la première touche de l'écran détectée, nous appelons la méthode `OnTapScreen` qui est la même que dans la section 3.2 à un détail près. Nous n'instancions pas l'objet dans cette méthode. Celle-ci nous sert à récupérer l'endroit où nous touchons notre écran afin de placer notre objet. L'instance de notre système solaire s'effectue dans la méthode `InstantiateObjects`. Avant d'instancier l'objet, nous devons envoyer notre position aux autres utilisateurs grâce à notre méthode du `MessageManager` écrite dans la figure 29.

Figure 34 : Instanciation du système solaire

```
if (_solarSystem != null)
{
    if (_isHost) {
        Debug.Log("Position of host : " + position);
        _messagingManager.SpawnGameObjects(position);
    }
    return;
}

scoreText.text = "Score : 0 - 0";

// Instantiate the solar system
Debug.Log("Instantiating the solar system");
_solarSystem = Instantiate(objectPrefab, position, Quaternion.identity);
```

Source : Données de l'auteur

Dans cette figure, nous nous assurons qu'uniquement l'hôte puisse partager sa position afin d'instancier le système solaire. Notre système solaire est maintenant correctement instancié pour les deux joueurs.

Passons à la logique du jeu. Le principe de celui-ci est d'aller toucher la planète du système solaire correspondant à la planète affichée sur l'écran. Pour afficher aux deux utilisateurs la même planète à rechercher, nous avons dû créer un nouveau message afin que les deux planètes à trouver soient les mêmes. Nous avons tout d'abord une méthode très simple qui nous retourne une planète au hasard parmi les planètes du système solaire.

Figure 35 : Méthode GetRandomPlanet

```
internal string GetRandomPlanet()
{
    return planetes[Random.Range(0, planetes.Length)];
}
```

Source : Données de l'auteur

Dès que nous allons commencer le jeu en appuyant sur le bouton correspondant, nous appelons une méthode `GetPlanet` du `MessageManager`. Cette méthode est plus ou moins similaire à celle pour envoyer la position de notre système solaire. La différence se situe au niveau du type d'objet que nous voulons transmettre aux utilisateurs. Ici, nous transmettons un `string` et non pas un `vector3`. Comme pour le `vector3`, nous devons transformer notre `string` en tableau de bytes afin de pouvoir la récupérer par la suite, car la méthode `BroadcastData` ne peut envoyer uniquement un tableau de bytes.

Figure 36 : Envoi d'un nom de planète via `BroadcastData`

```
internal void GetPlanet()
{
    byte[] data = Encoding.ASCII.GetBytes(_controller.GetRandomPlanet());

    _networking.BroadcastData
    (
        1,
        data,
        TransportType.ReliableUnordered,
        true
    );
}
```

Source : Données de l'auteur

Dans la figure 35, nous récupérons les bytes du `string` grâce à la méthode `GetBytes`. Ensuite, nous l'envoyons à travers le réseau avec le tag 1, le tag 0 étant déjà utilisé pour la position de notre système solaire. Nous pouvons remarquer le booléen `true` qui spécifie au message qu'il doit être envoyé également à l'expéditeur du message. Une fois le message envoyé, nous le récupérons grâce à la méthode `OnDidReceiveDataFromPeer` déjà existante.

Figure 37 : Réception des données avec le tag 1

```
case 1:
    Debug.Log("Planete à trouver");
    string message = Encoding.ASCII.GetString(data);
    Debug.Log(message);
    _controller.planetATrouver.text = message;
    break;
```

Source : Données de l'auteur

Cette fois-ci, le tag doit être égal à 1. Nous retransformons notre tableau de bytes en un `string` grâce à la méthode `GetString`, puis nous attribuons ce `string` au texte contenant la planète à trouver. De ce fait, les deux utilisateurs possèdent la même planète à trouver, ce qui rend le jeu plus attrayant et avec un aspect orienté vers la compétition.

Maintenant que nous savons quelle planète nous devons trouver, il nous reste à implémenter la partie technique du jeu. Nous voulons pouvoir toucher la planète du système solaire correspondante à la planète écrite sur notre application. Pour implémenter cette fonctionnalité nous utilisons un rayon (ray).

Figure 38 : Fonctionnement d'un rayon dans Unity



Source : Données de l'auteur

Un ray est un rayon qui traverse l'environnement. Lorsque ce rayon touche un objet, il le touche en un point que l'on appelle RaycastHit. Nous implémentons donc ce rayon pour qu'il traverse notre application et pour que lorsque nous touchons une planète, nous créons un RaycastHit afin de pouvoir interagir avec.

Figure 39 : Implémentation du Raycast

```
if (Input.touchCount > 0 && Input.touches[0].phase == TouchPhase.Began)
{
    Ray ray = _camera.ScreenPointToRay(Input.touches[0].position);
    RaycastHit hit;

    if (Physics.Raycast(ray, out hit))
    {
        if (hit.collider != null)
        {
            if (hit.collider.gameObject.name.Equals(planetATrouver.text))
            {
                Debug.Log("Planet Found");
                _messagingManager.PlanetFound(_isHost);
                _messagingManager.GetPlanet();
            }
        }
    }
}
```

Source : Données de l'auteur

Lorsque nous effectuons un toucher sur notre écran, nous allons créer un rayon qui part de notre caméra jusqu'au point de toucher. Ce rayon est créé grâce à la méthode ScreenPointToRay. Nous regardons ensuite si ce rayon entre en collision avec un objet, dans notre cas avec une planète. Attention, il est nécessaire et obligatoire que nos planètes comportent un composant collider. Pour nos planètes nous utilisons le composant sphere collider, afin que l'on puisse détecter une collision entre le toucher et la planète. Une fois un objet, ici une planète, touché, nous comparons le nom de l'objet touché au nom de la planète à trouver. Si les noms correspondent, nous appelons de nouveau notre méthode GetPlanet de notre MessagingManager afin de montrer une autre planète aux utilisateurs.

Il nous reste plus qu'à implémenter le système de scoring de points. Le score doit aussi passer par le réseau afin que les deux joueurs puissent l'observer. Il est donc géré par le `MessageManager`. La figure 38 nous montre l'appel d'une méthode `PlanetFound` qui prend comme paramètre un booléen. Le booléen, sous cette figure, est représentatif du statut du joueur. Si la valeur du booléen est vraie, le joueur est donc l'hôte de la partie ; sinon, c'est le second joueur. Afin de pouvoir transférer les informations aux joueurs, nous devons utiliser un tableau de bytes comme pour les messages précédents. Comme nous n'avons que deux joueurs possibles, nous créons un tableau de bytes de taille égale à 1. L'information que nous recevons dans notre méthode est un booléen, nous attribuons donc la valeur 0 si le joueur est l'hôte de la partie. Si, au contraire, le joueur n'est pas l'hôte, nous attribuons au tableau la valeur 1.

Figure 40 : Envoi d'un bit via `BroadcastData`

```
2 références  
internal void PlanetFound(bool isHost)  
{  
    var data = new byte[1];  
  
    if (isHost)  
        data[0] = 0;  
    else  
        data[0] = 1;  
  
    _networking.BroadcastData  
    (  
        2,  
        data,  
        TransportType.ReliableUnordered,  
        true  
    );  
}
```

Source : Données de l'auteur

Enfin, lorsque nous recevons le message, nous avons plus qu'à comparer si le premier élément du tableau est égal à 0 ou 1. S'il est égal à 0, nous ajoutons un point au score de l'hôte. S'il est égal à 1, nous ajoutons ce point au score du pair. Nous mettons ensuite à jour le score avec les nouveaux points.

Figure 41 : Gestion du score

```
_controller.scoreText.text =  
    string.Format  
    (  
        "Score: Host : {0} - Guest : {1}",  
        _controller.hostScore,  
        _controller.guestScore  
    );
```

Source : Données de l'auteur

4 Discussion

4.1 Problèmes rencontrés

Ce projet contenait, au début, plusieurs défis qui ont su être résolus au fil de l'avancement de l'application.

4.1.1 L'astronomie

Le premier défi concerne l'astronomie et le système solaire. Afin de créer une application intéressante et utile pour l'éducation, il est nécessaire de construire notre projet de manière précise. Étant donné que nous abordons le domaine de l'astronomie, et donc de l'infiniment grand, nous avons fait face à un problème d'adaptabilité à un format d'application mobile. En effet, comme nous le savons, les planètes qui composent notre système solaire sont gigantesques. Le soleil est une étoile qui possède un rayon équatorial de plus de 696'342 km. Comparé à la terre, qui elle a un rayon équatorial de 6'378km, le soleil est plus de 100 fois plus grand que la terre. Nous avons dû faire quelques concessions afin de garder un aspect qui colle à la réalité, tout en étant lisible sur une application mobile.

En ce qui concerne la partie où nous pouvons observer une planète, nous avons drastiquement diminué la taille des planètes afin qu'elles soient visibles dans notre application. Nous avons, cependant, gardé la proportion des planètes entre elles. Cela se traduit par une Terre qui est deux fois plus grande que Mars, mais plus de dix fois plus petite que Jupiter, la planète la plus grande du système solaire. En gardant la proportion entre les planètes, les élèves ou autres utilisateurs de l'application se rendent compte de la différence gargantuesque de taille entre les différentes planètes.

Pour la partie du système solaire, le défi de l'adapter à un format beaucoup plus petit nous amène à faire des concessions. Outre la différence de taille des différents astres, la distance entre eux rentre en compte lors de la conception du système solaire. Il a fallu trouver des solutions. Le premier but de cette partie de l'application était de pouvoir visualiser l'entier du système solaire. Comme nous avons déjà pu voir la différence de taille entre les planètes, nous avons donc décidé de mettre toutes les planètes à la même taille afin qu'elles soient toutes visibles. En ce qui concerne la distance entre les planètes, nous avons choisi de la réduire. La distance qui sépare le soleil d'Uranus, la planète du système solaire la plus lointaine, en est donc rendu à environ 3m. De ce fait, le système solaire reste visible pour tous les utilisateurs de l'application.

4.1.2 Lightship et Unity

L'application possède plusieurs problèmes qui, pour certains, ont été résolus durant le projet. Ces problèmes touchent notamment la prise en main de SDK de Lightship.

Dans le premier écran de notre application, lorsque nous pouvons visualiser une planète, nous avons rencontré des problèmes de texture. En effet, sur certaines versions de l'application, après l'avoir build sur notre smartphone, il s'est avéré que les textures des planètes ne s'affichaient plus. En testant depuis l'éditeur Unity, les textures s'affichaient correctement, à contrario de celles de l'application. Nous ne savons pas l'origine, ni la cause du problème, mais nous avons trouvé une manipulation qui résout ce problème. Nous devons totalement supprimer l'application de notre téléphone et générer un nouveau build depuis Unity. En supprimant totalement l'application, puis en la réinstallant, les textures des planètes ne disparaissent pas.

Le problème suivant a mis le plus de temps à être résolu, bien que ce soit le plus simple. Afin d'utiliser les pleines capacités de l'ARDK, nous devons rentrer une clé API dans notre projet. Cette clé doit se situer dans le dossier Ressources/ARDK. Notre problème a été d'appeler le dossier Ressources au lieu de Resources. Il faut donc bien vérifier que les noms de dossier soient les bons, et que la bonne clé API soit rentrée, auquel cas la fonctionnalité du multijoueur ne fonctionnera pas.

L'apprentissage de l'ARDK ne s'est pas fait sans problèmes. La plupart d'entre eux ont été résolus grâce à l'excellente documentation de l'ARDK ainsi qu'à leurs multiples vidéos explicatives. L'un des aspects les plus compliqués a été l'apprentissage de la partie multijoueur. L'échange des messages nous a posé quelques problèmes. Nous ne savions pas, au début, comment transférer nos données, comme la position d'un toucher sur l'application ou un string. C'est en approfondissant la lecture de la documentation que nous avons trouvé comment sérialiser et désérialiser un vecteur 3D afin qu'il traverse le réseau. Nous n'avons pas réussi à sérialiser notre string via les classes disponibles sur Lightship. Nous avons donc dû chercher sur la documentation de .NET afin de trouver une alternative fonctionnelle.

Un problème, cependant, subsiste dans notre application. Il s'agit d'un problème lors du placement du système solaire lors de la session multijoueur. Les deux systèmes sont censés se placer au même endroit ; or, lorsque nousinstancions notre système solaire sur notre téléphone hôte, le système apparaissait, sur l'autre téléphone, en hauteur où nous étions ou à un autre endroit aléatoire autour de nous. Nous ne connaissons pas la cause de ce problème, mais nous pouvons émettre quelques hypothèses : la première est le modèle du

second téléphone, un Samsung Galaxy S7, qui est le plus vieux Galaxy S à être compatible avec AR Core, le produit qui permet le fonctionnement d'application en réalité augmentée sur Android. La position pourrait être différente selon le modèle de téléphone choisi. La deuxième est un potentiel problème dans le code. Nous avons essayé plusieurs variantes dans le code, mais aucune d'entre elles étaient concluantes. Nous avons traqué la valeur de notre position pour voir si elle changeait durant l'exécution, elle semble rester exactement la même. Ce problème reste donc ouvert et est sujet à amélioration.

4.2 Améliorations

D'après les quelques retours des personnes qui ont pu tester l'application, il en résulte plusieurs points qui pourraient améliorer l'application à l'avenir.

Tout d'abord, il nous est venu l'idée d'implémenter un système de multi langues. Afin que l'application réponde aux besoins, dans un premier temps du canton du Valais, il serait judicieux d'implémenter l'application en allemand ou en haut valaisan. Par la suite, si l'application se veut être abordée sur le plan national, nous devons la traduire en italien.

Dans la partie où nous visualisons le système solaire, nous pourrions ajouter de nouvelles fonctionnalités afin d'en apprendre encore davantage sur le système solaire et plus facilement. Nous suggérons l'implémentation d'un bouton qui nous permet d'afficher directement sur les différentes planètes du système solaire le nom de celles-ci, afin de bien pouvoir les reconnaître lorsqu'elles sont en orbite autour du soleil. De plus, nous pourrions ajouter les distances réelles qui séparent le soleil des différents astres. Ces distances nous aideraient à bien visualiser la grandeur de notre système solaire, qui n'est pas représentatif sur notre application.

Une autre amélioration serait de fiabiliser la partie multijoueur du projet. Pour le moment, notre jeu multijoueur comporte encore quelques bugs, notamment dans le placement du système solaire. Ce dernier ne s'implémente pas au même endroit pour les deux joueurs. De plus, nous n'avons pas encore implémenter de fin de partie. Comme notre application est encore en développement, nous n'avons pas jugé nécessaire d'avoir une fin de partie. Cela permet notamment d'effectuer davantage de tests et de ne pas être limité. Pour cette fin de partie, nous pourrions avoir deux méthodes qui mettent fin à notre session : soit une fin quand un score est atteint, soit une fin après un chronomètre.

Notre jeu multijoueur peut être décliné sous d'autres formes. Nous pouvons penser à un jeu coopératif, plutôt qu'à un duel. Les jeux coopératifs sont un excellent moyen de s'amuser tout en développant des qualités d'entraide et de communication. Nous pourrions penser à un

jeu où une planète est montrée à un élève et celui-ci doit indiquer à son coéquipier quelle planète il doit chercher dans le système afin de marquer un point. Plusieurs duos pourraient s'affronter dans la classe afin de voir quel duo obtient le meilleur score. Enfin, notre jeu est actuellement limité à deux joueurs. Cependant, nous pourrions monter jusqu'à cinq utilisateurs en même temps, selon les recommandations de Lightship. Lightship recommande par ailleurs de limiter ces sessions de multijoueur à moins de cinq minutes par session. Il sera très intéressant de voir le futur développement de ce très jeune SDK afin de voir les améliorations que Niantic pourra apporter.

Enfin, comme mentionné plus haut, notre application est encore en stade de développement, nous n'avons donc que très peu touché au design de celle-ci. Afin que cette application puisse toucher professeurs comme élèves, il faudra que son UI soit facile à utiliser.

Nous avons aussi cherché des articles ou thèses qui développent le sujet de l'apprentissage de l'astronomie grâce à l'AR. Nous conseillons de jeter un œil à deux articles qui traitent de ce sujet. Le premier article, *An Augmented Reality Environment for Astronomy Learning in Elementary Grades: An Exploratory Study* (Fleck & Simon, 2013) traite d'une recherche comparative entre l'apprentissage physique contre et celui grâce à l'AR. Le deuxième, *Creative Situated Augmented Reality Learning for Astronomy Curricula* (Chen, Chen, & Wang, 2022) traite des bénéfices de l'apprentissage grâce à une application de réalité augmentée. Ces deux articles sont très intéressants car ils offrent des résultats après un test sur des élèves. Ils en arrivent à la même conclusion : les élèves ayant utilisés l'AR ont de légers meilleurs résultats. Ces articles montrent donc que l'AR possède un potentiel encore trop peu exploité en matière d'apprentissage.

5 Conclusion

Nous avons, durant ce travail, exploré le domaine de la réalité augmentée. Nous avons analysé les technologies résultantes de l'AR, afin de dégager laquelle serait la mieux adaptée à notre application. Cette dernière consiste à apporter un support technologique, basé sur la réalité augmentée, qui permet l'apprentissage de l'astronomie et plus précisément la découverte du système solaire.

Après plusieurs recherches, nous avons retenu un SDK qui nous a permis de développer notre application.

La principale fonctionnalité de notre application est la visualisation du système solaire. A cela, nous avons ajouté la fonctionnalité de voir une seule planète et d'avoir des informations complémentaires grâce à un lien qui nous dirige sur la page Wikipédia de la planète. Enfin, nous avons développé une expérience multijoueur pour l'application, où nous devons retrouver les planètes dans le système solaire.

Nous avons ensuite relevé les différents problèmes rencontrés pendant le développement et les améliorations possibles pour le futur de l'application.

Nous espérons que ce travail sera un premier point d'ancrage pour la venue de la réalité augmentée dans le domaine de l'éducation. Nous pensons que l'AR peut être grandement bénéfique lors d'apprentissage de nouvelles matières. Nous serions intéressés à observer le futur de ce projet et l'évolution de la réalité augmentée dans nos écoles.

6 Références

- AppAdvice. (2018, Mars 27). *Cosmos Creator - AR Universe*. Consulté le 07 15, 2022, sur <https://appadvice.com/app/cosmos-creator-ar-universes/1274468408>.
- Apple. (2022). *ARKit 6*. Consulté le Juin 15, 2022, sur <https://developer.apple.com/augmented-reality/arkit/>.
- AstroReality. (s.d.). *Augmented Reality Planet Earth Regular Model*. Consulté le Juin 14, 2022, sur <https://www.astroreality.com/augmented-reality-planet-earth-regular-model>.
- Bielecki, A. (2021, Octobre 06). *Solar System*. Récupéré sur <https://assetstore.unity.com/packages/templates/packs/solar-system-16139>.
- CDCP. (2021, Août 31). *Utilisation de la Réalité Augmentée dans l'éducation*. Consulté le 07 06, 2022, sur <https://www.cdc-tn.com/realite-augmentee-au-service-de-leducation/#:~:text=La%C3%A9alit%C3%A9%20augment%C3%A9e%20dans%20l%C3%A9ducation%20affectera%20bient%C3%B4t%20le%20processus,et%20les%20informations%20plus%20compr%C3%A9hensibles>.
- Chen, C.-C., Chen, H.-R., & Wang, T.-Y. (2022, Avril). Creative Situated Augmented Reality Learning for Astronomy Curricula. (E. T. Society, Éd.) *Educational Technology & Society*, 25(2), 148-162.
- Fleck, S., & Simon, G. (2013). An Augmented Reality Environment for Astronomy Learning in Elementary Grades: An Exploratory Study. *IHM '13: Proceedings of the 25th Conference on l'Interaction Homme-Machine* (pp. 14-22). Talence: Association for Computing Machinery.
- Google. (2022, Juin 06). *Extension AR Core - Cloud Anchors*. Consulté le Juin 15, 2022, sur <https://developers.google.com/ar/develop/unity-arf/cloud-anchors/developer-guide-android>.
- Google. (2022, Mars 23). *Présentation AR Core*. Consulté le Juin 15, 2022, sur <https://developers.google.com/ar/develop>.
- IKEA. (s.d.). *Applis IKEA*. Consulté le 07 01, 2022, sur <https://www.ikea.com/ch/fr/customer-service/mobile-apps/>.

- Liliia, Y., Midak, L., Kravets, I. V., Kuzyshyn, O. V., Berladyniuk, K. V., Buzhydhan, K., & Bazuik, L. V. (2021, Mars). Augmented reality in process of studying astronomic concepts in primary school.
- Mathew, N. (2018, Avril 25). *Building a Multiplayer ARKit Game with Placernote SDK*. Consulté le Juin 15, 2022, sur <https://hackernoon.com/building-a-multiplayer-arkit-game-with-placernote-sdk-6bc736ee6760>.
- Microsoft. (s.d.). *Encoding Class*. Consulté le Juillet 08, 2022, sur <https://docs.microsoft.com/en-us/dotnet/api/system.text.encoding?view=net-6.0>.
- net, J. d. (2020, Novembre 13). *Sprint: définition, du planning à la rétrospective*. Consulté le 07 21, 2022, sur <https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1443836-sprint-definition-planning-review-retrospective-backlog/>.
- Niantic. (2021, Mai 11). *Niantic Lightship*. Consulté le Juin 20, 2022, sur <https://nianticlabs.com/blog/lightship/?hl=fr>.
- Niantic. (2022). *Guides*. Consulté le Juin 06, 2022, sur <https://lightship.dev/learn/guides/>.
- Niantic. (2022). *Pricing*. Consulté le Juin 20, 2022, sur <https://lightship.dev/products/pricing/>.
- Niantic. (2022). *Tutorials*. Consulté le Juin 06, 2022, sur <https://lightship.dev/docs/ardk/tutorials.html>.
- Niantic. (2022). *User Manual*. Consulté le Juin 06, 2022, sur https://lightship.dev/docs/ardk/user_manual.html.
- Niantic. (s.d.). *Lightship developer community*. Consulté le Juin 15, 2022, sur <https://community.lightship.dev/>.
- Photon. (s.d.). *Introduction PUN*. Consulté le Juin 15, 2022, sur <https://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro>.
- Placernote. (2020). *Welcome to Placernote SDK*. Consulté le Juin 20, 2022, sur <https://docs.placernote.com/>.
- Placernote. (s.d.). *Explore Placernote*. Consulté le Juin 20, 2022, sur <https://placernote.com/about-the-sdk>.

Planets AR. (s.d.). *Page d'accueil*. Consulté le Juin 13, 2022, sur <https://planetsar.agrmayank.com/>.

PTC Inc. (2011). *Ground Plane*. Consulté le Juin 15, 2022, sur <https://library.vuforia.com/environments/ground-plane>.

PTC Inc. (2011). *Getting Started*. Consulté le Juin 15, 2022, sur <https://library.vuforia.com/>.

PTC Inc. (2011). *Pricing and Licensing Options*. Consulté le Juin 15, 2022, sur <https://library.vuforia.com/faqs/pricing-and-licensing-options>.

PTC Inc. (2022). *Pricing Plans*. Consulté le Juin 15, 2022, sur <https://www.ptc.com/en/products/vuforia/vuforia-engine/pricing>.

Unity. (2022, Mai 02). *About AR Foundation*. Consulté le Juin 15, 2022, sur <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/manual/index.html>.

Unity. (s.d.). *Physics.Raycast*. Consulté le Juillet 10, 2022, sur <https://docs.unity3d.com/ScriptReference/Physics.Raycast.html>.

Unity Technologies . (2022). *Page d'accueil*. Consulté le 06 21, 2022, sur <https://unity.com/fr>.

Unity. (s.d.). *Toggle*. Consulté le Juillet 05, 2022, sur <https://docs.unity3d.com/ScriptReference/UIElements.Toggle.html>.

7 Annexes

7.1 Annexe I : Product Backlog

US N°.	As an/a ...	Je veux	Afin de	Critères d'acceptance	Story Points	Sprint	US accepté	MosCoW
1	Dev	Préparer le git	Sauvegarder le projet	Le git doit être créer	1	1	Done	Must
2	Dev	Configurer Android sur Unity	Tester l'application avec le téléphone	L'application doit se lancer sur le téléphone	2	1	Done	Must
3	Utilisateur	Sélectionner une planète	pouvoir la visualiser	Je peux sélectionner n'importe quelle planète à l'aide d'une liste	3	1	Done	Must
4	Utilisateur	Toucher une surface plane	afficher la planète en AR	Une surface plane est détectée	3	1	Done	Must
5	Utilisateur	voir des informations supplémentaires concernant la planète	découvrir la planète	Un bouton info est visible Il affiche un lien wikipedia qui mène vers la planète en question	1	1	Done	Must
6	Utilisateur	Toucher une surface plane	Afficher le système solaire	Une surface plane est détectée	3	1	Done	Must
7	Utilisateur	Voir le fonctionnement du système solaire	Apprendre le fonctionnement du système solaire	Le système solaire doit être visible	5	1	Done	Must
8	Utilisateur	Rejoindre une session multijoueur	De jouer avec un camarade	Une case est disponible pour entrer l'ID de la session Un bouton pour rejoindre la session est visible	13	2	Done	Must
9	Utilisateur	Placer le système solaire	Commencer la partie	Le système solaire doit être visible Le bouton pour commencer la partie est disponible	8	2	Done	Must
9	Utilisateur	Changer la langue de l'application	Mettre l'application en allemand	Un bouton pour changer les langues doit être visible Les textes doivent être traduits	3			Could

Source : Données de l'auteur

7.3 Annexe III : Guide d'utilisation

Ce guide va nous aider à utiliser les différentes fonctionnalités de l'application.

Menu

Le menu de l'application comporte trois boutons. Le premier nous amène à l'écran pour visualiser une planète. Le deuxième nous redirige sur l'écran pour visualiser le système solaire. Enfin, le dernier permet de commencer une partie de multijoueur.



Planète unique

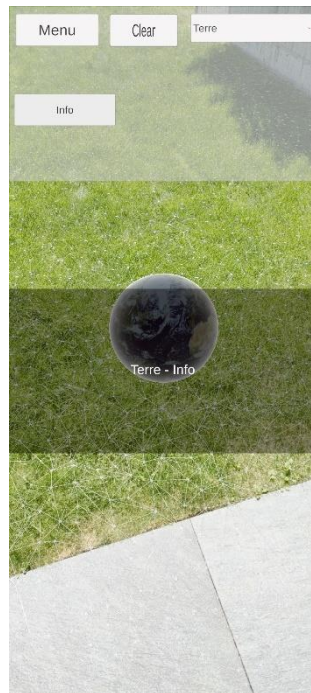
Dans cet écran, nous pouvons visualiser une planète de notre choix. Pour ce faire, il faut d'abord scanner notre environnement pour que l'application trouve une surface plane afin de poser une planète en AR.



Nous pouvons choisir quelle planète nous voulons voir en la sélectionnant grâce à la liste déroulante. Afin de la poser virtuellement, il suffit de toucher notre écran à l'endroit où nous voulons la voir.



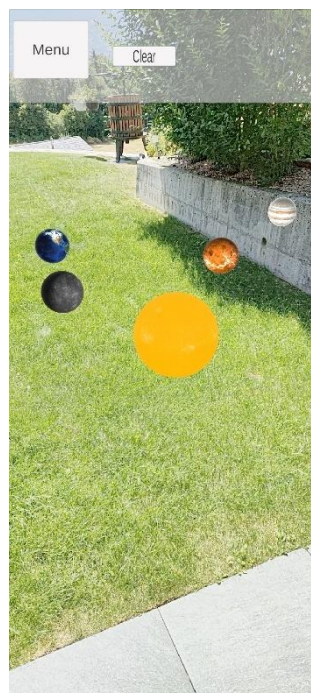
Le bouton Info nous montre un texte avec le nom de la planète ainsi qu'un texte « Info » qui nous redirige vers la page Wikipédia de la planète en question.



Le bouton Clear est utilisé pour retirer la planète visible afin d'en choisir une nouvelle et de la placer.

Système solaire

Cet écran reprend le même principe que l'écran précédent. Il faut dans un premier temps scanner l'environnement afin de trouver une surface plane. Après l'avoir trouvé, nous pouvons tout simplement placer notre système solaire à l'endroit souhaité.



Multijoueur

Pour commencer une session multijoueur, il faut rentrer un ID dans la zone prévue à cette effet.



Ensuite, nous pouvons rejoindre la session grâce au bouton Join Session. Une fois la session rejointe, il faudra communiquer l'ID au deuxième utilisateur afin qu'il puisse rejoindre la même session.



L'hôte de la partie va devoir scanner son environnement pour initialiser la partie. Pour cela, il est préférable de scanner un objet fixe qui ressort de l'environnement afin d'accélérer le

processus. Une fois que l'hôte a atteint le statut stable, le deuxième utilisateur pourra scanner le même objet afin d'être reconnu dans le même environnement.



Dès que les deux joueurs sont stables, l'hôte devra placer le système solaire en touchant une partie de l'écran scanné comme dans les deux écrans précédents.



Une fois le système solaire instancié pour les deux joueurs, l'hôte pourra lancer la partie grâce au bouton Start Game, fraîchement apparu.



Le jeu peut alors commencer. Pour marquer des points, il suffit de toucher la planète du système solaire qui correspond à la planète affichée à l'écran.



8 Déclaration de l'auteur

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du responsable de filière et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

- Antoine Widmer

Ayent, le 27 juillet 2022

Bastien Ferrari