PROTOTYPE Environnement de développement

Dans ce chapitre, nous expliquerons comment le cas pratique décrit au point précédent (5.3.5) peut être implémenté sur Google Cardboard. Le but est d'avoir un prototype de réalité virtuelle concret, fonctionnel et respectant au maximum le scénario d'apprentissage. Nous commencerons par détailler l'environnement de développement avant d'indiquer la marche à suivre afin d'installer les composantes nécessaires à la réalisation de ce prototype. Le point principal, le développement, sera ensuite exposé et commenté ; puis nous terminerons en présentant un bilan final de l'application créée.

Environnement de développement

Avant de commencer à développer, il convient de se demander comment et avec quel logiciel le faire. Il existe en effet de nombreux environnements de développement intégré (IDE) permettant de créer des applications Android (dans la mesure où la donnée requiert de développer une application pour ce système d'exploitation) : Android Studio, Eclipse, IntelliJ IDEA, etc.

Étant donné que nous sommes ici face à un cas particulier – développer pour Google Cardboard – il est logique de tout d'abord se renseigner sur le site Internet officiel du *device*[11] pour suivre les guides de démarrage destinés aux développeurs.

Google explique sur son site qu'il existe deux SDK différents pour développer sur Cardboard : le premier est un SDK Cardboard pour Android, le deuxième un SDK Cardboard pour Unity (Google, 2015).

Nous détaillerons ces deux kits de développement et leurs IDE respectifs dans les points suivants.

[11] https://developers.google.com/cardboard/overview

6.1.1. Android Studio

Dans un premier temps, nous avons téléchargé Android Studio. Il s'agit de l'environnement de développement intégré le plus courant, il s'agit de surcroît de celui que Google recommande pour le développement d'applications Android. Comme tout IDE récent, il aide au développement grâce au système d'auto-complétion, d'une hiérarchie de projet, de différentes couleurs selon le type de variables, de documentation ainsi que d'autres fonctionnalités fort utiles.

Nous nous sommes très vite retrouvés confrontés à un problème : contrairement au développement d'applications Android « classiques », nous avons ici besoin d'une interface visuelle bien plus poussée pour y importer des vidéos à 360 degrés et ajouter des interactions à celles-ci. Un simple aperçu du design de l'application tel que celui proposé par Android Studio ne suffit pas. Même si Google explique de manière succincte dans un tutoriel basique comment développer pour Cardboard sur Android Studio, nous nous rendons vite compte que nous manquons d'éléments théoriques et pratiques pour créer des objets en trois dimensions en Java[12] ou pour importer et lire des vidéos dans Android Studio.

Nous décidons donc de changer d'environnement de développement et de tester celui destiné aux créateurs de jeux vidéo et d'applications à contenu visuel pour lequel Google propose également un SDK : Unity 3D.

6.1.2. Unity 3D

Unity 3D (communément appelé Unity) est une plate-forme de développement pour la création de jeux et d'expériences interactives en 3D et en 2D (Unity - Game engine, tools and multiplatform, 2015). Ce moteur de jeu est souvent utilisé pour la réalisation de jeux vidéo par des professionnels. Le SDK Cardboard qui lui est destiné permet ainsi d'adapter une application Unity 3D existante pour la réalité virtuelle ou de créer, en partant de rien, une expérience de réalité virtuelle à l'aide de cette plate-forme. Grâce à lui, les applications, qui

[12] Java est un langage de programmation. Servant souvent de premier langage aux apprenants développeurs, il est le langage de base pour la création d'applications Android.

peuvent afficher du contenu en 3D avec rendu binoculaire, sont capables de réagir aux mouvements de la tête et de les enregistrer. De plus, le SDK adapte automatiquement l'application aux caractéristiques physiques de Cardboard, inclut la configuration stéréo automatique pour le modèle de Cardboard défini et corrige la distorsion par rapport aux lentilles intégrées (Google, 2015).

Figure 15 : logo d'Unity
Source : http://www.socialcubix.com/wp-content/uploads/portfolio/derby/unity-logo.png

6.2. Installation des composants

Avant de commencer, il est bien sûr essentiel d'avoir Java ainsi que le JDK (soit le SDK pour développer en Java) préalablement installés sur son ordinateur. Sans cela, il sera impossible de procéder à l'installation du SDK Android.

Depuis la page de téléchargement pour développeurs Android[13], il est possible de télécharger soit l'environnement Android Studio complet, soit uniquement le SDK (appelé à cette occasion *stand-alone SDK tools*). Ce dernier est suffisant pour notre cas, étant donné que nous avons fait le choix de ne pas travailler sur Android Studio, mais sur Unity. Une fois le téléchargement et l'installation terminés, le SDK Manager se lance automatiquement (pour autant que la case à cocher n'ait pas été désactivée).

Depuis l'Android SDK Manager, il est nécessaire de sélectionner et d'installer au minimum les éléments suivants :

- Android SDK Tools

[13] http://developer.android.com/sdk/installing/index.html

- Android SDK Platform-tools
- Android SDK Build-tools (de la dernière version)
- SDK Platform de la version d'Android désirée

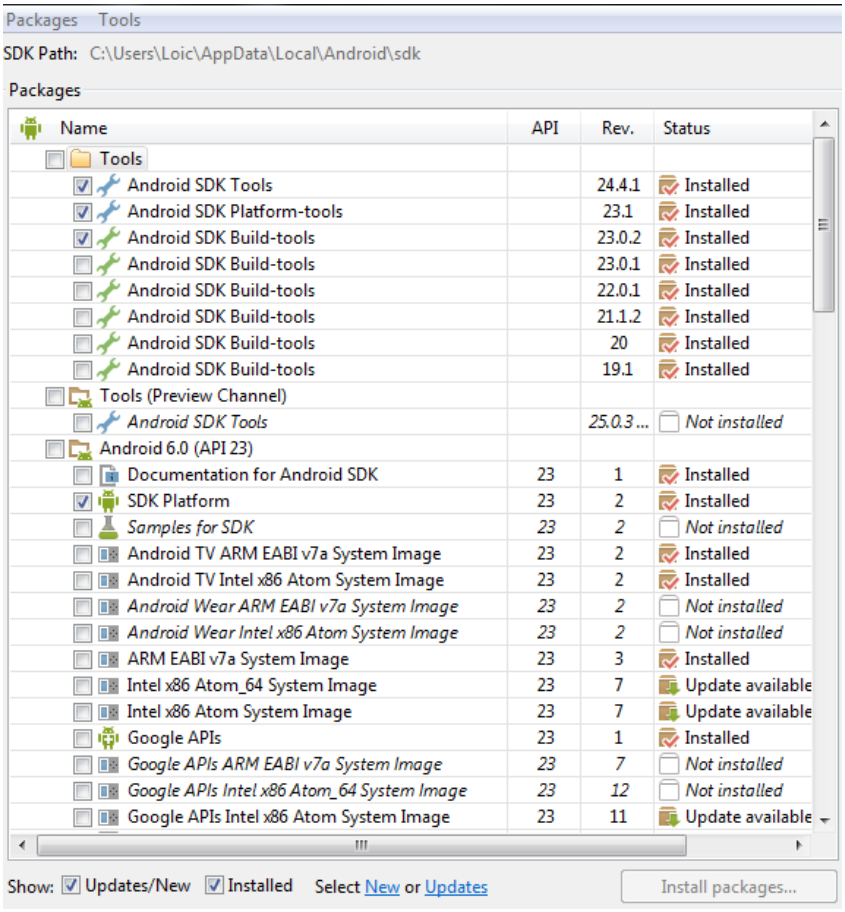

Figure 16 : Android SDK Manager
Source : capture d'écran réalisée par l'auteur

Passons maintenant au téléchargement d'Unity. Actuellement en version 5.3, le logiciel est disponible en deux éditions : personnelle (gratuite) et professionnelle (à partir de 75 dollars par mois). Nous opterons pour notre part pour l'édition personnelle, dont les fonctionnalités principales sont les mêmes que celles de la version professionnelle. Unity 5

est à télécharger directement depuis le site officiel d'Unity[14], puis à installer simplement en suivant les étapes les unes après les autres.

Enfin, il convient de télécharger le SDK Cardboard pour Unity. Le fichier avec l'extension .unitypackage[15] peut directement être téléchargé et ne nécessite aucune installation, puisqu'il ne sera importé dans le projet que plus tardivement.

Une fois tous les composants installés, nous pouvons lancer Unity et créer un nouveau projet 3D :

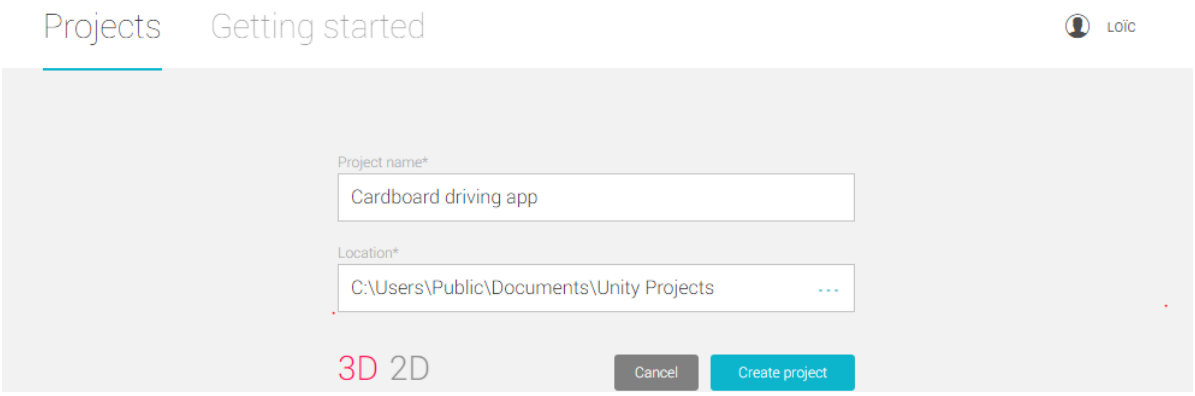

Figure 17 : fenêtre de création d'un nouveau projet Unity
Source : capture d'écran réalisée par l'auteur

6.3. Développement

Dans le but de détailler au maximum les principaux points du développement de notre prototype sur Google Cardboard, nous montrerons en premier lieu comment importer le SDK Cardboard pour Unity. En deuxième lieu, nous détaillerons le mode d'importation des vidéos à 360 degrés dans Unity. En troisième lieu, nous expliquerons comment changer de scène (et donc de vidéo) dans Unity. Enfin, en dernier lieu, nous exposerons comment chaque type d'interaction précédemment défini peut (ou ne peut pas) être ajouté à un projet Cardboard, à savoir pour notre cas pratique les *fuse buttons*, une interaction découlant de l'action de l'aimant latéral de Cardboard et le contrôle par la voix.

[14] https://unity3d.com/get-unity
[15] https://github.com/googlesamples/cardboard-unity/blob/master/CardboardSDKForUnity.unitypackage?raw=true

6.3.1. Importation du SDK Cardboard pour Unity

La première étape est d'importer le SDK Cardboard pour Unity téléchargé précédemment. Pour cela, dans le menu, nous choisissons *Assets* > *Import Package* > *Custom Package…* et sélectionnons le fichier CardboardSDKForUnity.unitypackage, tout en confirmant à l'aide du bouton *Import*.

Dans l'explorateur de projet (au bas de l'écran), un dossier Cardboard est désormais apparu parmi les assets[16]. Une fois ouvert, nous constatons qu'il est composé de six sous-dossiers, dont un appelé *Prefabs*. Dans ce dernier se trouve un fichier nommé *CardboardMain.prefab*. Il faut alors le glisser-déposer dans la hiérarchie (menu latéral sur la gauche) pour qu'Unity affiche les éléments à la manière de Cardboard, à savoir avec deux angles de vue similaires (un pour l'œil gauche et un pour l'œil droit). Le sous-dossier *CardboardMain/Head/Main Camera* affiche d'ailleurs ces deux caméras. L'ancienne *Main Camera*, à la racine du projet, peut désormais être supprimée.

L'environnement de travail affiché correspondra désormais à ceci :

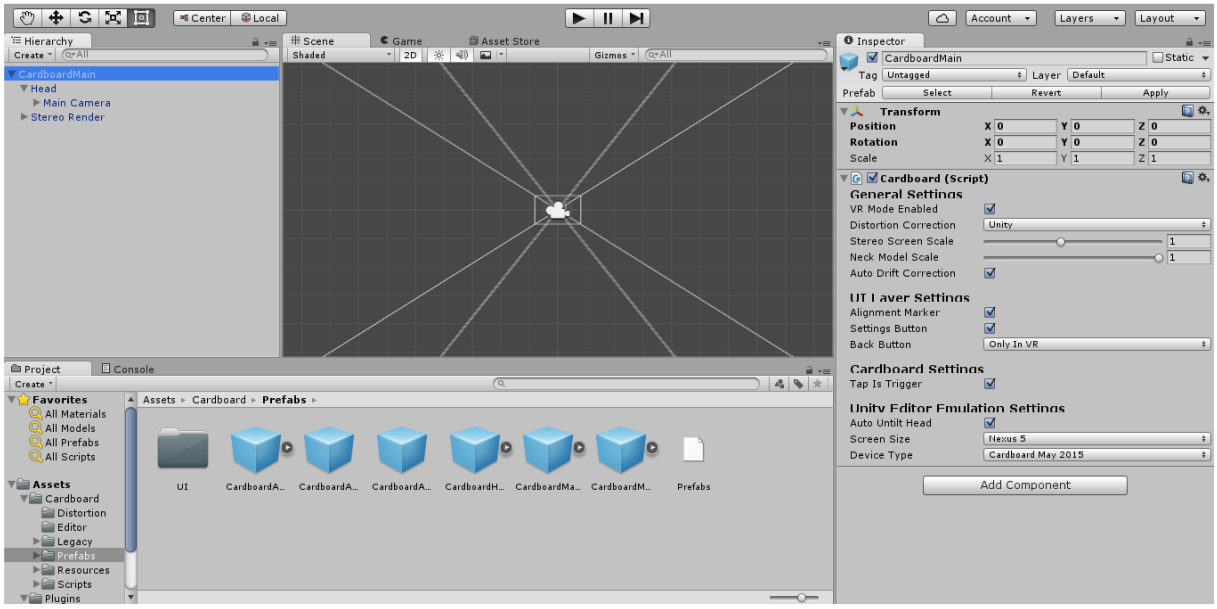

Figure 18 : environnement de travail Unity configuré pour Cardboard
Source : capture d'écran réalisée par l'auteur

[16] Les assets sont toutes les ressources audio, vidéo, fichiers de code et autres composantes du projet.

Il est possible d'ajouter toutes sortes d'éléments visuels en trois dimensions à notre application pour Cardboard. Néanmoins et puisque nous ne créerons pas d'arrière-plan, de fond, de sol ou d'objets 3D pour implémenter notre cas pratique, ce dont nous avons besoin un premier lieu, c'est d'importer notre première vidéo à 360 degrés.

6.3.2. Importation de vidéos à 360 degrés

L'un des aspects fondamentaux et *a priori* simple à implémenter s'est finalement révélé l'un des plus compliqués : importer une vidéo à 360 degrés et faire en sorte que celle-ci soit entièrement visible dans une application mobile de réalité virtuelle optimisée pour Cardboard, avec bien sûr la possibilité pour l'utilisateur de regarder autour de lui pour ne voir que la vidéo et aucun autre élément.

En effet, les problèmes suivants sont survenus :

- premièrement, l'importation de vidéos dans un projet Unity n'est possible qu'avec la version professionnelle du logiciel
- deuxièmement, la technique couramment utilisée pour lire des fichiers vidéo dans Unity, *MovieTexture*, ne fonctionne pas pour un déploiement mobile, mais uniquement pour un logiciel/jeu visant les plates-formes PC ou Mac (il existe cependant depuis peu la fonction *Handheld.PlayFullScreenMovie()* permettant de contourner ce problème)
- troisièmement, même la version professionnelle d'Unity ne permet pas de travailler avec des vidéos à 360 degrés.

Pour pallier à ces différents soucis, nous avons opté pour un plugin disponible au téléchargement sur l'Asset Store (la boutique d'assets d'Unity) : Easy Movie Texture[17]. Ce dernier ne nécessite pas la version professionnelle (pour autant que la version 5 ou plus récente d'Unity soit installée) et peut gérer les vidéos à 360 degrés visant à être déployées sur Android ou, dans une moindre mesure, sur iOS.

[17] Lien de téléchargement : https://www.assetstore.unity3d.com/en/#!/content/10032 (prix actuel : 55$)

Dans le but de représenter une vidéo à 360 degrés dans Unity, il ne suffit pas de l'importer dans le projet (attention à ne pas mettre d'accents ou de caractères spéciaux dans le nom du fichier !) et d'espérer qu'elle se lance automatiquement. En effet, afin de représenter cette impression d'immersion au cœur d'un environnement entourant le sujet, il est nécessaire de travailler avec une sphère, un objet tridimensionnel dont tous les points sont situés à équidistance d'un point central. La vidéo sera projetée sur la surface interne de la sphère (ce qui n'est possible qu'avec le plugin précédemment nommé), alors que le point central de celle-ci représentera le sujet. La distance entre ce dernier et la sphère étant toujours la même, il aura tout loisir de regarder partout autour de lui en ayant l'impression d'être dans un autre monde très réaliste, sans se douter qu'il se trouve en fait au centre d'une sphère virtuelle dont les parois internes projettent ce qu'il voit.

Une fois la sphère (objet Sphere) créée et la vidéo à 360 degrés importée dans le projet puis projetée sur les parois internes de la première nommée, il est nécessaire de placer l'objet *Main Camera* du SDK Cardboard au centre exact de celle-ci. Cet objet représente effectivement le point de vue de l'utilisateur et s'occupe d'afficher la vue en « mode réalité virtuelle », c'est-à-dire en doublant l'écran avec un léger décalage entre chaque image destinée à chaque œil, donnant ainsi une impression de stéréoscopie.

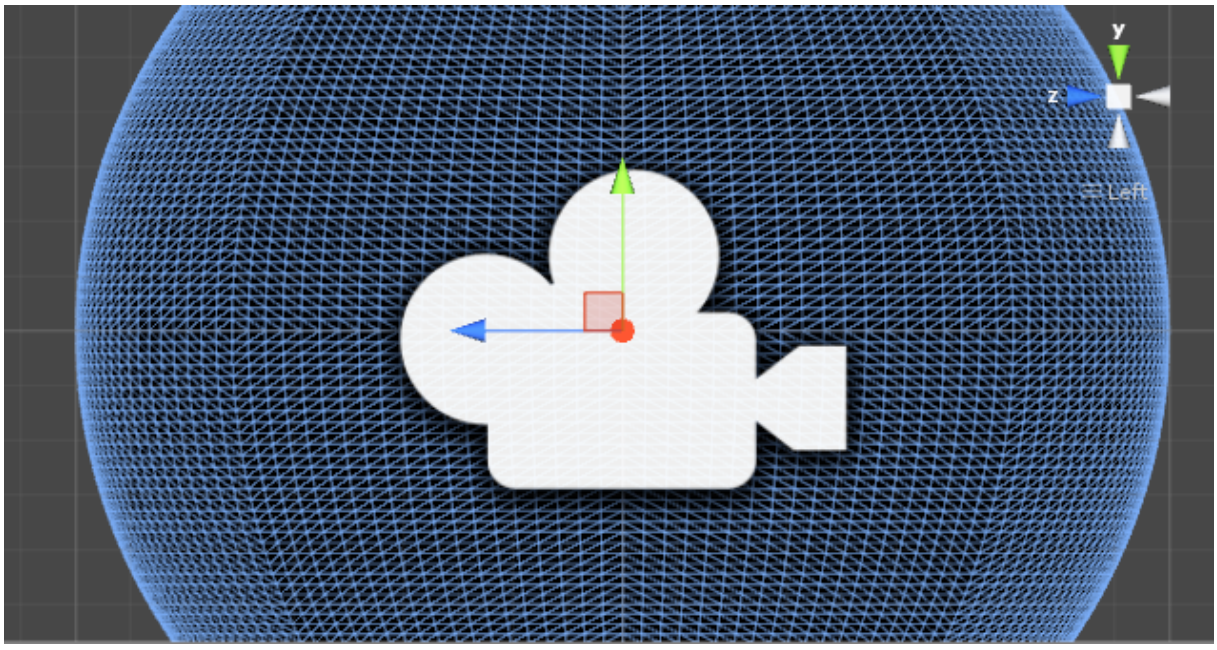

Figure 19 : objet *Main Camera* se situant au cœur d'un objet Sphere
Source : capture d'écran réalisée par l'auteur

Bien que visible depuis un smartphone Android en mode réalité virtuelle pour Cardboard – ce qui est le but final – notre vidéo à 360 degrés ne peut pas directement être lue avec le bouton de lecture d'Unity. Il faut en effet à chaque fois déployer l'application sur le smartphone Android (voir point 6.3.7) pour constater les changements effectués dans le projet. Ceci nous posera un problème majeur au regard de notre cas pratique (voir point 6.3.4).

Enfin, il est possible que la vidéo soit inversée lors de la lecture de celle-ci sur le smartphone. Dans notre exemple, cela avait pour désavantage de donner l'impression que le conducteur et le volant se trouvaient à droite, comme dans une voiture anglaise. Afin de pallier à ce problème, il a fallu retourner horizontalement chaque vidéo avant de l'importer successivement dans Unity 3D.

6.3.3. Changements de scènes

Dès lors que nous savons comment importer une vidéo et la jouer en mode Cardboard, il s'agit de savoir passer de l'une à l'autre (notre cas pratique est constitué de 13 vidéos ou images accompagnées de la voix off). Pour cela, nous travaillerons à l'aide de scènes. Une scène est constituée de tous les éléments visuels présents dans la hiérarchie du projet ; dans notre cas *Main Camera* et *Sphere*. Les assets ne sont cependant pas rattachés à une scène, mais au projet dans son ensemble.

En dupliquant la première scène de sorte à en créer une seconde, nous obtiendrons exactement les mêmes éléments. Notre caméra pour Cardboard et notre sphère seront donc déjà entièrement créées comme dans la première scène ; il faudra ainsi pour chaque nouvelle scène changer le nom de la vidéo à lire et gérer le changement de scène selon le type d'interaction. De plus, avant le déploiement, dans le menu *File/Build Settings*, il est nécessaire d'ajouter toutes les scènes qui seront intégrées à l'application finale, comme détaillé sur la figure suivante :

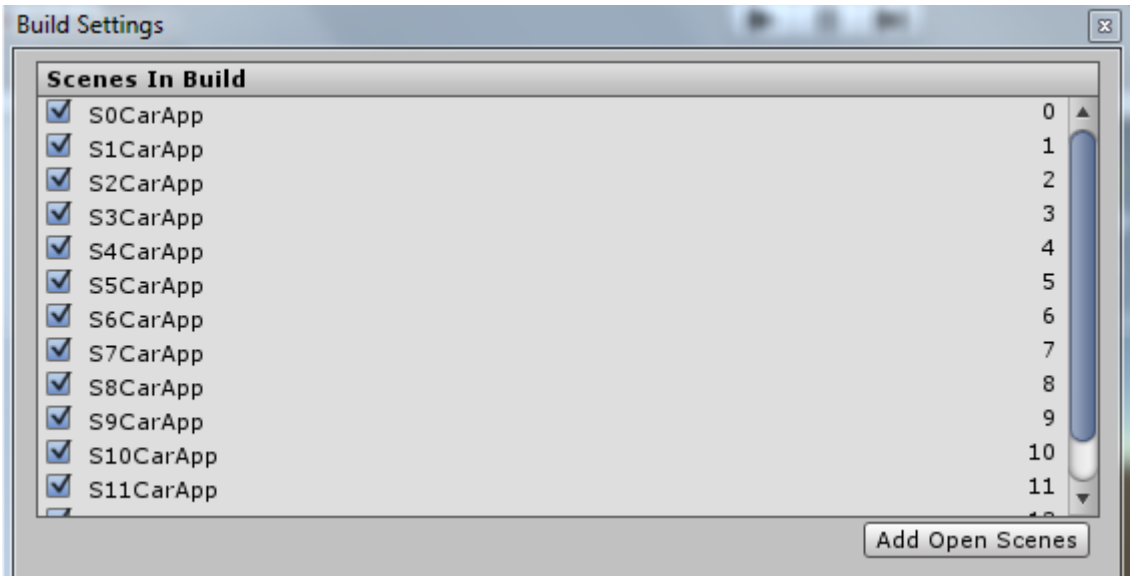

Figure 20 : obligation d'ajouter toutes les scènes du projet dans le menu *Build Settings*
Source : capture d'écran réalisée par l'auteur

Chaque scène possède non seulement un nom, mais également un numéro chronologique (la première scène est la numéro 0). Cela nous sera particulièrement utile pour indiquer la scène suivante, comme nous l'expliquerons plus loin dans ce travail.

Concernant la méthode appelant la scène suivante, son emplacement dépend avant tout du type d'interaction, comme nous le verrons dans les points suivants. Si le changement de scène doit être effectué directement à la fin de la vidéo et qu'il ne nécessite aucune interaction de la part du sujet, il suffit d'insérer l'instruction directement dans le script de lecture de la vidéo (*MediaPlayerCtrl*). Elle devra ensuite être placée à l'endroit précis où *m_CurrentState == MEDIAPLAYER_STATE.END* est valide, à savoir lorsque la vidéo courante a été entièrement lue et est arrivée à son terme. En revanche, si l'action de passer à une autre vidéo découle d'une interaction, c'est dans une autre classe qu'il faudra placer l'instruction en question, comme nous le verrons dans les points suivants.

Jusqu'à la version 5.2 d'Unity, cette instruction ne se composait que d'une seule ligne : *Application.LoadLevel("nomDeLaScene")*. Cependant, depuis Unity 5.3 (sorti en décembre 2015), l'instruction est *SceneManager.LoadScene("nomDeLaScene")* ; celle-ci devant en plus obligatoirement être précédée de l'ajout du namespace correspondant (*using UnityEngine.SceneManagement)* au sommet de la classe. Plutôt que de passer un argument

de type String à la méthode *LoadScene()*, nous pouvons également lui donner directement le numéro de la scène (int) correspondant au numéro indiqué sur la figure 20.

Voyons maintenant comment appeler cette méthode selon les interactions définies.

6.3.4. Interaction par le regard

Quand bien même l'interaction par le regard était destinée à être l'une des plus importantes de ce cas pratique, elle ne pourra pas être implémentée en raison du fait que nous travaillons avec des vidéos. En effet, et comme nous le voyons sur la figure 19, seule la sphère est visible lors du développement; la vidéo projetée sur ses parois ne s'affiche pas dans Unity, mais uniquement sur le smartphone, une fois l'application déployée.

Il est donc malheureusement impossible d'ajouter un *fuse button* étant donné que nous ne voyons pas la vidéo au moment où nous devrions le créer. Nous ne pouvons pas placer d'objet que le sujet doit regarder spécifiquement sans avoir un aperçu de ce qu'il voit, d'autant plus que ces boutons étaient dans notre cas pratique positionnés à des endroits-clés tels que sur la vitre arrière ou sur la clé du véhicule.

Si cela avait été possible avec des vidéos à 360 degrés, nous aurions procédé de la même manière que lorsque nous avons eu l'occasion de tester l'interaction par le regard avec des éléments 3D modélisés: en ajoutant tout d'abord des éléments Physics Raycaster et Event System, en écrivant ensuite le code qui renvoie vers une autre scène (de la même manière que nous le ferons pour l'interaction de type aimant latéral), puis en ajoutant ce code à un élément tridimensionnel en définissant que chaque paramètre renvoie à une scène différente. Le script *GazeInputModule* faisant partie du SDK Cardboard est spécifiquement conçu pour capturer la direction du regard de l'utilisateur ; il faut encore l'ajouter et le lier avec un composant Collider pour que le regard du sujet soit intercepté et renvoie à une autre scène.

Cette restriction qui empêche de procéder à l'ajout de cette interaction sur des vidéos à 360 degrés explique sans doute pourquoi, dans les applications que nous avons eu l'occasion de tester, les interactions liées au regard ne sont utilisées que dans des applications de jeux vidéo modélisés et jamais lors de la lecture de vidéos en mode réalité virtuelle.

En ce qui concerne notre prototype, l'interaction par le regard sera remplacée par l'actionnement de l'aimant latéral de Cardboard.

6.3.5. Interaction par l'aimant latéral

Afin de détecter si l'aimant latéral de la Cardboard faisant office de bouton a été actionné, nous avons besoin d'une classe très complexe (*CardboardMagnetSensor*) contenant différentes méthodes. Une fois celle-ci ajoutée aux assets, nous devons en créer une seconde qui appellera la méthode de la première, testant si l'aimant latéral a bien été tiré vers le bas (après l'avoir activé). Si c'est le cas, nous pourrons alors à nouveau appeler l'instruction précédemment nommée : *SceneManager.LoadScene(numeroDeScene)*. De plus, nous prendrons soin de définir le numéro (ou le nom) de la scène dans une variable qui officiera comme paramètre de la classe, de manière à pouvoir déclarer ce numéro directement dans l'appel du script (voir figure 21, le champ *Go To Scene*) et ne pas devoir créer une multitude de classes identiques où seul le numéro de la scène change. Cette classe est appelée *CardboardTriggerControlMono* et son contenu peut être consulté en annexe III. Elle devra être ajoutée à notre objet sphère afin d'être intégrée au projet, comme montré sur la figure suivante :

Figure 21 : appel du script lançant la scène dont le numéro correspond à celui entré dans *Go To Scene*
Source : capture d'écran réalisée par l'auteur

Désormais, une simple action de l'aimant permettra de passer à la scène suivante (pour autant que cela ait été géré pour la scène en question). Le sujet ne se rendra pas compte que le programme a changé de scène et de sphère ; il aura toujours l'impression de se trouver au centre du véhicule, avec de nouvelles instructions à écouter et des actions à effectuer. Le sentiment d'immersion virtuelle reste donc total, malgré le changement de scène.

6.3.6. Interaction par la voix

Intéressons-nous enfin au dernier type d'interaction défini dans notre cas pratique, le contrôle par la voix. À nouveau, différents obstacles se dressent sur notre chemin : outre tous les inconvénients précédemment nommés que possède le contrôle vocal (difficultés de compréhension, d'interprétation de langue, bruit environnant, micro placé à l'intérieur de la Cardboard), il se trouve qu'Unity 3D ne propose pas de support direct pour la reconnaissance vocale. Reprendre la bibliothèque Microsoft Speech Recognition serait une idée, mais elle n'est pas adaptée au déploiement pour mobile. Quant à celle de Google, elle n'est destinée qu'au développement Android standard.

Il existe tout de même une solution pour contourner ce problème sur Unity : l'achat d'un plugin spécifique. Tout comme pour les vidéos à 360 degrés, différents plugins sont capables de faire ce qu'Unity ne peut accomplir tout seul. Citons Speech Recognition for Android[18]ou Android SpeakNow[19], tous deux basés sur la bibliothèque Google Speech Recognition.

Étant donné que nous n'avons dans notre cas pratique qu'une seule instruction vocale et qu'il serait saugrenu d'acheter un plugin pour l'utiliser pendant une seule scène dont la durée n'excède pas dix secondes, nous n'avons pas pu tester celle-ci et la remplacerons dans par un actionnement de l'aimant latéral de Cardboard.

6.3.7. Déploiement

Afin de déployer l'application sur le smartphone, quelques étapes doivent être effectuées avant de la lancer pour la première fois.

Tout d'abord, il faut accéder au menu *File/Build Settings* et choisir *Android* dans la liste des plates-formes de développement. Attention à ne pas cliquer directement sur *Build And Run*, mais de tout d'abord sélectionner *Player Settings*, comme indiqué en vert dans la figure ci-dessous :

[18]Lien de téléchargement : https://www.assetstore.unity3d.com/en/#!/content/13882 (prix actuel : 10$)
[19] Lien de téléchargement : https://www.assetstore.unity3d.com/en/#!/content/16781 (prix actuel : 40$)

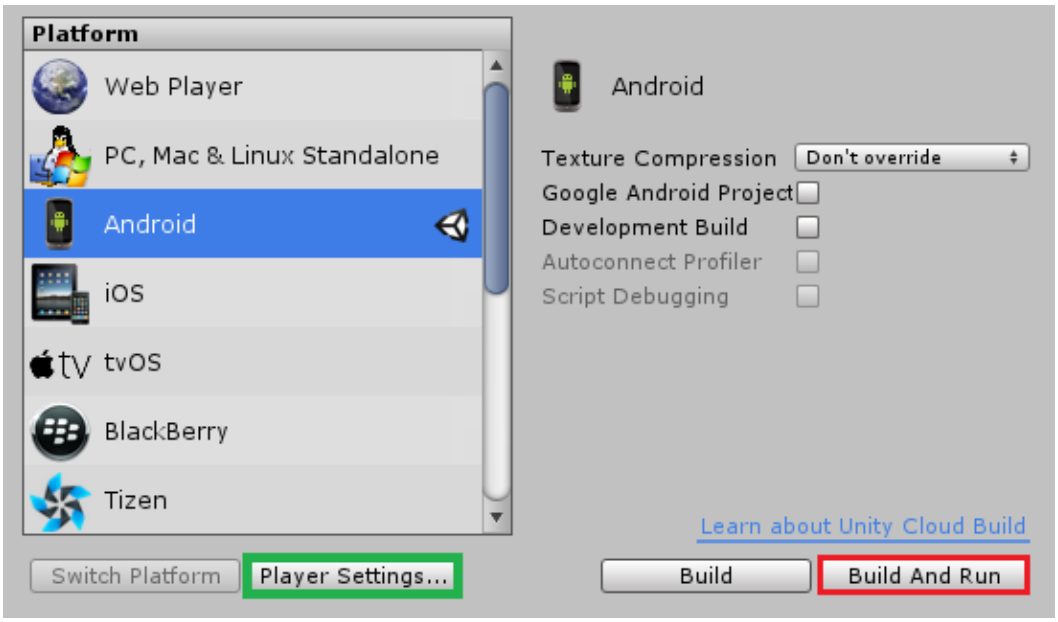

Figure 22 : menu *Build Settings* permettant le déploiement d'une application Unity3D
Source : capture d'écran réalisée par l'auteur

Un menu latéral s'ouvre alors sur la droite de l'écran. Le paramètre *Default Orientation* (sous-menu *Resolution and Presentation*) doit être changé à *Landscape Left* afin d'ordonner à Android de ne pas changer automatiquement l'orientation de l'écran selon la manière dont le téléphone est tenu (verticalement ou horizontalement). Sous *Other Settings*, dans la case *Bundle Identifier*, Android exige d'insérer un nom unique reliant l'application au système. Ce nom est généralement composé de deux lettres indiquant le pays de l'application (ou *com* pour une application commerciale) suivi du nom de l'entreprise ou de l'école et se terminant par le nom de l'application, le tout séparé par des points (exemple : ch.hevs.BachelorApp).

Une fois ces réglages effectués, *Build And Run* (bouton encadré en rouge sur la figure 22 ci-dessus) peut être pressé, pour autant qu'un smartphone Android soit relié au PC via un câble USB. Une fenêtre s'ouvre alors et permet de choisir où enregistrer le fichier .apk, à savoir l'application en elle-même qui peut directement être installée sur d'autres terminaux Android. Lorsque l'application aura été enregistrée sur l'ordinateur, Unity se chargera de la placer directement sur le téléphone connecté via USB puis de la lancer. Le câble peut alors être débranché et le smartphone placé dans la Cardboard afin de profiter de l'expérience de réalité virtuelle.

6.4. Résultat final et bilan

Nous avons donc développé notre prototype avec un succès mitigé. Le point positif le plus notable est sans aucun doute le fait que les vidéos sont entièrement visibles à 360 degrés sur Cardboard, donnant ainsi la possibilité d'apprentissage en immersion virtuelle. De plus, il y a une possibilité de navigation entre les différentes vidéos, une progression dans le jeu qui intervient soit automatiquement, soit lorsque l'utilisateur est appelé à le faire. En revanche, nous pouvons nous estimer déçus de l'interaction qui a lieu entre le sujet et le prototype, celle-ci se limitant à pousser l'aimant latéral vers le bas lorsqu'une action est demandée. Une fois de plus, comme dans deux nombreuses autres applications, un certain manque d'interaction se fait ressentir.

Par ailleurs, il arrive fréquemment que l'application *crashe* sans raison lors du changement de scène. Il est alors nécessaire de la redémarrer et de recommencer à la première scène pour tenter d'aller plus loin dans le jeu. Ces bugs sont totalement aléatoires (ils peuvent intervenir aussi bien après la première scène qu'à l'avant-dernière) et sont probablement causés par un manque de mémoire vive pour faire tourner l'application.

Nous arrivons par conséquent à la conclusion qu'il n'aurait peut-être pas fallu tenter de mélanger vidéos à 360 degrés et *serious game* (ou jeu vidéo d'une manière générale). C'est avec amertume que nous comprenons désormais pourquoi il existe deux grandes catégories distinctes d'applications de réalité virtuelle pour Google Cardboard, à savoir :

- les vidéos filmées à 360 degrés qui n'exigent aucune interaction de la part de l'utilisateur (excepté bouger la tête pour regarder autour de lui)
- et les jeux vidéo modélisés avec des textures et des objets 3D dans lesquels il y a une constante interaction entre le jeu et l'utilisateur, notamment à travers le regard de ce dernier ou encore grâce au bouton latéral de Cardboard, voire grâce au contrôle vocal.

Travailler avec des vidéos et penser y ajouter des éléments externes n'était probablement pas une bonne solution ; il aurait peut-être été plus judicieux d'inventer un cas pratique moins concret et de développer une application en la créant de toutes pièces avec des

objets modélisés tels qu'on en trouve sur l'Asset Store, sans filmer quoi que ce soit. Il aurait alors été possible d'ajouter un arrière-plan virtuel ainsi que des objets en 3D et de gérer le regard du sujet afin de pousser l'interaction un cran plus loin.

CONCLUSION

La réalité virtuelle n'en est qu'à ses débuts. Celle qui est souvent surnommée « *the next big thing* » s'apprête à débarquer cette année dans les salons des particuliers de manière massive, avec le lancement des trois principaux casques de réalité virtuelle que sont l'Oculus Rift, le PlayStation VR et le HTC Vive. Selon un rapport établi en 2015 par la Deutsche Bank, l'industrie de la réalité virtuelle pourrait valoir sept milliards de dollars en 2020 (James, 2015), principalement en raison de son développement sur le marché du jeu vidéo. Les plus grandes entreprises technologiques ont d'ailleurs presque toutes déjà investi ce marché : Facebook (via le rachat d'Oculus), Google, Samsung, HTC, Sony et Microsoft. Quant à Apple, l'annonce du lancement d'un produit en lien avec la réalité virtuelle ne saurait tarder si l'on se fie aux divers rachats et embauches opérés ces derniers mois par l'entreprise américaine (Hattersley, 2016).

Jusqu'à présent, ce sont les casques couplés à des smartphones comme le Samsung Gear VR ou la Google Cardboard qui ont fait office de pionniers de la réalité virtuelle. Le nombre toujours croissant d'applications disponibles sur le Play Store témoigne de l'intérêt grandissant des consommateurs. Néanmoins, il est important de souligner qu'une part considérable d'utilisateurs télécharge une ou plusieurs application(s) de réalité virtuelle dans le seul but de voir à quoi ressemble cette nouvelle forme de divertissement pendant quelques temps, avant de vite l'oublier. Ces utilisateurs passent totalement à côté des possibilités offertes par cette nouvelle technologie.

Il est cependant vrai que les applications possédant un réel potentiel font encore défaut. Une simple recherche sur le Play Store suffit pour constater que la plupart des applications de réalité virtuelle sont des jeux sans aucune interaction ou se contentant d'afficher des vidéos à 360 degrés. Or, s'il n'y pas d'interaction, comment maintenir l'attention de l'utilisateur davantage que quelques minutes ? Cela est d'autant plus vrai si celui-ci est coupé du monde et qu'il ne peut pas voir autre chose que ce que l'on veut lui montrer. L'apparition de *haptic devices* ou de joysticks compatibles avec les casques VR passifs ne pourra que faciliter la façon d'interagir avec la réalité virtuelle et la rendre plus populaire.

Définir un scénario d'apprentissage nous a montré à quel point il était difficile d'imaginer ce qui pourrait (ou au contraire ne pourrait pas) être développé par la suite. À trop vouloir mélanger les éléments entre jeux vidéo et vidéos à 360 degrés, nous nous sommes retrouvés confrontés à des limites que nous n'imaginions pas. Nous avons pu développer des fonctionnalités innovantes dans un but d'apprentissage, mais de manière limitée, et avons compris pourquoi la grande majorité des applications de réalité virtuelle Android disponibles présentaient toutes la même lacune, à savoir un déficit d'interaction. Programmer dans le but de créer de la réalité virtuelle requiert non seulement une certaine expérience dans la création de jeux vidéo, mais oblige également les développeurs à penser différemment.

Fort heureusement, il existe déjà des applications faisant bon usage de l'interaction certes limitée de Google Cardboard. Nous pensons notamment à celles dévolues à l'apprentissage immersif. La plupart d'entre elles prennent en compte le fait que montrer quelque chose d'inhabituel à quelqu'un ne doit pas se faire sans raison, dans un esprit purement démonstratif. Avec des explications claires, des interactions définies et la volonté d'enseigner d'une manière novatrice, l'apprentissage immersif pourrait se développer très rapidement et rencontrer un franc succès, au même titre que celui qui semble promis à l'industrie vidéoludique au printemps prochain.

La réalité virtuelle est à nos portes et elle pourrait bien changer notre façon de voir le monde. Encore relativement peu connue du grand public et réservée aux professionnels il y a quelques années, Google l'a rendue accessible à tous avec son casque Cardboard. Toutefois, ses buts et utilisations ne sont pas encore clairement définis dans l'esprit de la population. Le monde du jeu vidéo la démocratisera très prochainement. L'étape suivante devrait logiquement être d'y chercher d'autres usages, notamment éducationnels et didactiques. La tendance de l'apprentissage immersif amorcée sur les casques passifs pourrait bien se confirmer et ouvrir les portes virtuelles du savoir à tout un chacun.

RÉFÉRENCES

AFP. (2012, Décembre 6). *Berlin célèbre les 100 ans de la découverte du buste de Néfertiti*. Récupéré sur L'Express: http://www.lexpress.fr/actualites/1/culture/berlin-celebre-les-100-ans-de-la-decouverte-du-buste-de-nefertiti_1196024.html

Alvarez, J. (2007). *Du jeu vidéo au Serious game. Approches culturelle, pragmatique et formelle*. Toulouse: Université de Toulouse II et III.

Anzil, P. (2013, Avril 13). *E-santé et mesure de soi, la science-fiction à portée de main*. Récupéré sur Les Numériques: http://www.lesnumeriques.com/capteur-activite/e-sante-mesure-soi-science-fiction-a-portee-main-a1669.html

Apple. (2015). *Apple Watch - Acheter*. Récupéré sur Apple (CH): http://www.apple.com/chfr/watch/buy/

Apple. (s.d.). *WatchKit*. Récupéré sur Apple Developer.

Augmented Reality | Definition of augmented reality by Merriam-Webster. (s.d.). Récupéré sur Merriam-Webster: http://www.merriam-webster.com/dictionary/augmented%20reality

Bavor, C. (2016, Janvier 27). *(Un)folding a virtual journey with Google Cardboard*. Récupéré sur Official Google Blog: https://googleblog.blogspot.fr/2016/01/unfolding-virtual-journey-cardboard.html

Beal, V. (s.d.). *Waht is Software Development Kit (SDK)?* Récupéré sur Webopedia: http://www.webopedia.com/TERM/S/SDK.html

Berko, L. (2013, Juillet 15). *Could Living as a Virtual Cow Make You Go Vegan?* Récupéré sur Motherboard: http://motherboard.vice.com/blog/could-living-as-a-virtual-cow-make-you-go-vegan

Biri, V., Bouvier, P., de Sorbier de Pougnadoresse, F., Chaudeyrac, P., & Piranda, B. (2006). *Immersion dans un monde visuel et sonore en 3D*. Récupéré sur Université de Marne la Vallée: http://igm.univ-mlv.fr/~fdesorbi/missbiblio/BLCPdS06nat/BLCPdS06nat.pdf

Burnham, K. (2013, Septembre 18). *8 Wearable Tech Devices To Watch*. Récupéré sur InformationWeek: http://www.informationweek.com/mobile/mobile-devices/8-wearable-tech-devices-to-watch/d/d-id/1111596?page_number=6

Burns, M. (2015, Juillet 9). *Google Glass Is Alive*. Récupéré sur TechCrunch: http://techcrunch.com/2015/07/09/google-glass-is-alive/#.ywbtksk:VcG1

Burns, M. (2015, Janvier 19). *Today Is The Last Day To Buy Google Glass*. Récupéré sur TechCrunch: http://techcrunch.com/2015/01/19/today-is-the-last-day-to-buy-google-glass/

Burrowes, D. (2014, Décembre 8). *Baer's Odyssey: Meet the serial inventor who built the world's first game console*. Récupéré sur Ars Technica: http://arstechnica.com/gaming/2014/12/in-the-beginning-ralph-h-baer-and-the-birth-of-the-game-console/

Casques-VR. (s.d.). *Présentation HTC Vive*. Récupéré sur Casques-VR: http://casques-vr.com/htc-vive/

Castejon, M. (2015, Septembre 15). *Sony PlayStation VR : ne l'appelez plus Project Morpheus*. Récupéré sur FrAndroid: http://www.frandroid.com/marques/sony/310357_project-morpheus-casque-de-realite-virtuelle-de-sony-change-de-nom

Chapman, G. (2016, Janvier 9). *Vegas awash with altered realities*. Récupéré sur The Asian Age: http://www.asianage.com/technomics/vegas-awash-altered-realities-169

Charnay, A. (2015, Septembre 8). *GoPro et Google lancent Odyssey, leur plate-forme pour filmer à 360 degrés*. Récupéré sur 01net: http://www.01net.com/actualites/gopro-lance-odyssey-une-sphere-pour-filmer-a-360-degres-913237.html

Chièze, J. (2015, Octobre 6). *HoloLens + Project XRay: date de sortie et prix du kit dévoilés*. Récupéré sur Gameblog: http://www.gameblog.fr/news/53875-suivez-la-conference-microsoft-avec-hololens

Danilewsky, F. (2015, Mai 31). *Le casque Gear VR disponible pour les Galaxy S6 – vidéo*. Récupéré sur IDBOOX: http://www.idboox.com/smartphone/le-gear-vr-disponible-pour-les-galaxy-s6-video/

de Waal-Montgomery, M. (2015, Septembre 2). *Samsung has shown it's serious about Tizen, but is it yet 'The OS of Everything'?* Récupéré sur VentureBeat: http://venturebeat.com/2015/09/02/samsung-has-shown-its-serious-about-tizen-but-is-it-yet-the-os-of-everything/

DSFI Group. (2015, Mars 5). *communiqué de presse - DSFI Group présente sa division Events lors du Salon Heavent Meetings de Cannes*. Récupéré sur 24presse: http://www.24presse.com/cp.php?id=9916955

Ducros, M. (s.d.). *Principe de la vision stéréoscopique*. Récupéré sur Photo Stereo: http://photo.stereo.free.fr/stereoscopie/stereoscopie-principe.php

Durel, J. (2015, Juillet 21). *HTC Vive : la meilleure expérience de réalité virtuelle... pour l'instant*. Récupéré sur CNET: http://www.cnetfrance.fr/produits/htc-vive-la-meilleure-experience-de-realite-virtuelle-pour-l-instant-39820800.htm

Fove. (s.d.). *FOVE: The World's First Eye Tracking virtual reality headset*. Récupéré sur FOVE: http://www.getfove.com/

Fromentin, M. (2015, Mai 21). *Le casque FOVE, nouvel arrivant sur le marché de la réalité virtuelle*. Récupéré sur Übergizmo: http://fr.ubergizmo.com/2015/05/21/casque-fove-realite-virtuelle.html

Giraudon, J. (2015, Avril 8). *La réalité virtuelle pour l'enseignement de savoirs abstraits ou nécessitant la pratique du terrain*. Récupéré sur L'Agence nationale des Usages des TICE: http://www.cndp.fr/agence-usages-tice/que-dit-la-recherche/la-realite-virtuelle-pour-l-enseignement-de-savoirs-abstraits-ou-necessitant-la-pratique-du-terrain-82.htm

Google. (2015, Avril 16). *Cardboard Overview | Cardboard*. Récupéré sur Google Developers: https://developers.google.com/cardboard/overview

Google. (2015, Mai 28). *Cardboard SDK for Unity*. Récupéré sur Google Developers: https://developers.google.com/cardboard/unity/

Google. (2015). *Google Cardboard*. Récupéré sur Google: https://www.google.com/get/cardboard/

Google. (2015). *Jump - Google*. Récupéré sur Google Cardboard: https://www.google.com/get/cardboard/jump/

Google. (2015, Mai 22). *Platform Overview | Glass*. Récupéré sur Google Developers: https://developers.google.com/glass/develop/overview

Google. (s.d.). *A new dimension - Designing for Google Cardboard - VR design guidelines*. Récupéré sur Google: http://www.google.com/design/spec-vr/designing-for-google-cardboard/a-new-dimension.html#