Similarités avec un prototype existant

Au début de l'année 2015, Toyota a annoncé un partenariat avec Oculus afin de lancer un simulateur dont le but est d'apprendre aux jeunes les dangers que représentent les distractions durant la conduite (Smith, 2015). Ce programme, baptisé TeenDrive365, n'est pour l'heure disponible que dans des salons de l'automobile aux États-Unis et requiert de nombreuses ressources : une véritable voiture (qui restera stationnaire), un Oculus Rift, un casque audio, un ordinateur, une caméra et plusieurs capteurs (placés sur le volant, sur le levier de boîte de vitesses et dans l'habitacle) reliés aux deux derniers cités qui vont retranscrire les mouvements des utilisateurs. Cela offre des conditions quasiment réelles avec notamment un volant à manipuler qui va tourner de la même manière dans l'application de réalité virtuelle qu'il le fait en vrai et des pédales répondant aussi bien au véhicule présent dans l'Oculus qu'à la voiture physique. Ainsi, dans ce cas spécifique, seuls le paysage et le déplacement du véhicule seront 100% virtuels, le reste de la voiture existant autant réellement que virtuellement. Durant le trajet virtuel, plusieurs distractions interviendront : des piétons surgiront soudain sur la route, le passager (également virtuel) montrera au conducteur un message sur son smartphone, une autre voiture ne respectera pas la priorité ou le son de la radio ne sera pas bien réglé (Smith, 2015). Le but de ce simulateur n'est donc pas d'apprendre directement à conduire, mais plutôt de sensibiliser les jeunes et de leur donner une leçon afin qu'ils adoptent dans le futur une conduite sûre et responsable.

5.3. Détail du scénario

Avant de décrire précisément le scénario du cas pratique, nous prendrons le temps de commenter l'origine de l'idée ayant conduit à la réalisation de celui-ci, puis d'analyser les éléments pratiques qui en font partie, et enfin d'exposer comment il a été divisé en plusieurs niveaux de jeu.

5.3.1. Origine de l'idée

Contrairement au simulateur TeenDrive365 (voir point 5.2) qui s'axe principalement sur les distractions qui pourraient se produire pendant un trajet en voiture, le scénario du cas

pratique de ce travail traite de l'apprentissage de la conduite et de la connaissance du véhicule.

C'est en réfléchissant aux défis que la vie nous impose et à comment les simplifier qu'a surgi cette idée. Le permis de conduire constitue en effet une étape importante dans la vie d'une personne non domiciliée en ville.

Il n'est jamais évident de se retrouver face à des dizaines de commandes sans les connaître et sans savoir que faire. Pourtant, lorsque quelqu'un prend place pour la première fois derrière le volant d'une voiture, c'est exactement ce qu'il vit : assis dans le siège conducteur, il voit face à lui de nombreux boutons et outils qu'il ne connaît pas. Le cas pratique s'adresse ainsi essentiellement à des gens n'ayant encore jamais pris le volant, avec pour but de leur faire prendre confiance et d'être moins craintifs lorsqu'ils le feront pour la première fois grâce au fait d'avoir expérimenté quelque chose de similaire via cette application de réalité virtuelle. Il rend ainsi plus facile l'acclimatation à la place de conducteur. Le début du cas pratique constitue essentiellement une introduction au véhicule. Par la suite, les règles de circulation basiques sont expliquées, puis le sujet se retrouve confronté à des situations qu'un possesseur de permis de circulation se doit de savoir gérer.

5.3.2. Éléments intégrés au scénario

Le sujet lance l'application depuis l'écran du smartphone et place ce dernier dans la Google Cardboard. L'expérience commence alors : l'utilisateur se retrouve dans l'habitacle d'une voiture automatique et entend les instructions fournies par une voix off. Si, dans la plupart des jeux vidéo en réalité virtuelle, le personnage principal n'est pas visible à l'écran (jeux à la première personne), ce n'est pas le cas dans notre prototype, pour des raisons évidentes de tournage du cas pratique (il aurait en effet été compliqué de démarrer et de conduire une voiture sans personne derrière le volant).

Étant donné que l'interaction entre l'utilisateur et le véhicule virtuel est limitée et qu'il aurait été impossible de tester l'embrayage, le débrayage et les changements de vitesses sans aucun support physique, une voiture automatique a été préférée à une voiture manuelle dans ce cas pratique.

Les dispositifs suivants d'un véhicule automatique font partie intégrante du cas pratique et seront visibles en tout temps par le sujet :

- Emplacement/clé pour démarrer le véhicule
- Volant
- Ceinture de sécurité
- Levier de boîte de vitesses (quatre positions possibles : P, R, N, D)
- Trois rétroviseurs (gauche, centre, droite)

5.3.3. Types d'interaction

Comme évoqué au point 3.3.2 de ce travail, il peut exister au sein d'une application destinée à la Cardboard trois principaux types d'interaction : les *fuse buttons*, le contrôle par la voix et l'actionnement du bouton-aimant se situant sur le côté gauche de la Cardboard.

Afin de rendre l'application plus intéressante et variée pour l'utilisateur, ces trois sortes d'interactions sont utilisées dans le scénario. Lorsque différents types de commandes font partie de la même application, il est d'usage d'établir un certain nombre de règles afin que l'utilisateur ne se sente pas perdu et puisse associer que « telle commande » égale « telle action à effectuer ». Ainsi, dans ce cas pratique, nous avons défini de la manière suivante quelles sont les actions qui découlent de chacune des interactions :

- actionnement de l'aimant latéral : changer de vitesse à l'aide du levier de boîte de vitesses
- commandes vocales : déclencher une action qui engendrera un mouvement et aura une incidence majeure sur le scénario / répondre à une question oralement
- fixation sur un élément : action ciblant un objet précis

Ces choix ne sont pas anodins : dans une voiture automatique, changer de vitesse via le sélecteur de vitesses requiert un mouvement du bras vers le haut ou vers le bas et s'apparente dans ce sens plus à tirer un bouton (l'aimant de la Cardboard) vers le bas qu'à poser longuement son regard sur un élément. De même, les commandes vocales qui

engendrent un mouvement donnent une sensation de pouvoir à l'utilisateur, qui ressentira une certaine satisfaction à ce que ses ordres soient respectés : il aura ainsi tendance à les exprimer de manière claire et distincte. Une commande vocale pour une simple action telle que regarder derrière n'aurait pas la même logique et le même effet escompté qu'une commande forte dégageant une véritable action. Au contraire, porter son regard sur un objet précis et s'y attarder témoigne du sens du détail nécessaire à des actions précises requises par les *fuse buttons*.

5.3.4. Division du scénario en niveaux de jeu

Le prototype a été créé sur la base d'un *serious game* : dans le but de lui donner un côté ludique et afin de créer une progression échelonnée sur toute la durée du jeu, il a été divisé en plusieurs niveaux. Il est possible de passer à un niveau supérieur uniquement après avoir validé le niveau actuel en respectant les instructions fournies.

Le découpage entre les niveaux s'est fait d'une manière logique, chacun d'entre eux concernant un cas particulier. Bien que le détail de chaque niveau soit exposé plus tard, nous pouvons résumer ainsi chacun d'entre eux :

- Niveau 1 : introduction à l'environnement et démarrage du véhicule
- Niveau 2 : marche arrière, sortie d'une place de parking et marche avant
- Niveau 3 : feu de circulation et respect de celui-ci
- Niveau 4 : panneau STOP et direction à choix
- Niveau 5 : test d'attention et limite de vitesse
- Niveau 6 : parcage et arrêt du véhicule

Dans un premier temps, seuls les deux premiers niveaux ont été implémentés afin de pouvoir tester tous les types d'interaction et de se concentrer sur l'essentiel. La suite (niveaux trois à six) reprend les mêmes interactions, mais ajoute plusieurs nouveautés, notamment l'échec du niveau qui pourrait intervenir si le sujet venait à ne pas respecter les instructions.

5.3.5. Description du scénario

Au début du jeu, le véhicule est stationné dans une place de parking, de sorte qu'il est impossible de sortir en marche avant. Le sélecteur de vitesses se trouve en position P. Tous les éléments décris au point 5.3.2 sont visibles et le sujet a tout le loisir de regarder autour de lui afin de les localiser (voir figure 14).

Une information s'affiche par-dessus l'image, signalant au sujet qu'il s'agit du premier niveau. Ce dernier consiste principalement en une introduction à l'environnement à l'aide d'une description de l'habitacle de la voiture et de diverses instructions fournies oralement. Le texte suivant sera retransmis en voix off, alors que se lance la vidéo *01_45_secondes_intro_injected.mp4* :

« Bonjour et bienvenue dans cette application d'apprentissage à la conduite.

Vous allez maintenant être sensibilisé aux différents dispositifs du véhicule que vous pouvez observer autour de vous. Ouvrez bien vos oreilles et grand vos yeux. L'interrupteur de démarrage et l'emplacement pour insérer la clé se trouvent à droite du volant. Le volant avec lequel vous allez pouvoir tourner à gauche ou droite se situe devant vous. Trois miroirs appelés rétroviseurs sont à votre disposition : un à votre gauche et un à votre droite en dehors de la voiture, ainsi qu'un en haut au centre à l'intérieur de la voiture. Ce sont des éléments importants, car avant de tourner, vous devez bien vérifier qu'il n'y a pas un vélo ou un cyclomoteur qui essayerait de vous dépasser et que vous risqueriez de collisionner. »

Figure 14 : extrait du cas pratique enregistré par une caméra filmant à 360 degrés
Source : séquence filmée par Vincent Praz

S'ensuit automatiquement une image fixe pendant une trentaine de secondes (*02_ceinture_injected.mp4*) durant laquelle la voix off donne une première instruction :

« Commencez par attacher votre ceinture de sécurité en mimant le mouvement à l'aide de votre bras droit, depuis votre épaule gauche vers votre hanche droite, et ordonnez à Cardboard d'attacher votre ceinture en prononçant le mot belt *».*

Le smartphone écoute alors et attend le mot-clé que le sujet doit prononcer. À chaque fois qu'une commande vocale est en attente et si la reconnaissance vocale n'a pas été en mesure de reconnaître le son attendu au bout d'un laps de temps de dix secondes, l'instruction est répétée (ici : « *attachez votre ceinture en prononçant le mot* belt », *02_Instruction_ceinture_repetee_injected.mp4*). Au bout de quinze nouvelles secondes suit une nouvelle répétition. Finalement, quinze secondes plus tard, la voix off avertit l'utilisateur qu'il pourra presser le bouton latéral de la Cardboard pour remplacer l'instruction vocale.

Une fois l'interaction accomplie, la voix off annonce, par-dessus la vidéo *02_Video_ceinture_injected.mp4* :

« Ceinture de sécurité attachée. »

Puis elle continue d'expliquer l'environnement (*03_BoiteAvitesse_injected.mp4*) :

« En bas entre les deux sièges se trouve le levier de boîte à vitesses qui indique différentes positions, soit P (parking), R (marche arrière), N (neutre) et D (conduite). Vous êtes maintenant prêt à démarrer. Concentrez votre regard sur la clé qui se trouve dans l'interrupteur de démarrage du véhicule durant plusieurs secondes pour commencer votre voyage. »

Cette dernière action fait appel aux *fuse buttons* : lorsque l'utilisateur fixe un élément pouvant être déclenché, un compte à rebours s'active et ce n'est qu'au bout de trois secondes sans détourner le regard du bouton que l'action est validée. Si l'utilisateur n'entreprend aucune action dans les quinze secondes qui suivent l'instruction, la phrase suivante est répétée (*03_Intruction_cle_repetee_injected.mp4*) :

« Vous ne trouvez pas la clé ? Elle se trouve à droite du volant. Fixez-la pendant plusieurs secondes sans bouger pour démarrer. »

Une fois le bouton validé, le véhicule virtuel démarre (la vidéo *04_Demarrage_injected.mp4* est lancée et la voix off informe : *« Véhicule démarré. »*) et un label indique visuellement au sujet qu'il passe désormais au second niveau.

Le but de cette deuxième étape sera de sortir de la place de parking où le véhicule est actuellement stationné. La voix off énonce alors (*05_sortieparking_image_injected.mp4*) :

« Pour éloigner votre véhicule et pouvoir rouler, il va tout d'abord falloir reculer. Pour effectuer une marche arrière, commencez par vérifier que personne ni aucun objet ne se trouve derrière votre véhicule. Jetez un coup d'œil dans vos trois rétroviseurs, puis tournez-vous et fixez la vitre arrière pendant plusieurs secondes afin d'activer le bouton numérique. »

A nouveau un *fuse button* se déclenche selon les mêmes conditions que les autres. L'instruction « *fixez la vitre arrière pendant plusieurs secondes »* sera répétée (*05_Instruction sortie de parking_repetee-_injected.mp4)* si rien n'est fait au bout de quinze secondes.

Le scénario peut continuer (*06_Marchearriere_image_injected.mp4*) :

« La voie est libre. Pressez la pédale de frein, mettez le levier de boîte de vitesses en position R et relâchez la pédale. Pour placer le sélecteur de vitesses en position R, actionnez l'aimant latéral de Google Cardboard. »

Une fois l'aimant pressé, le véhicule change de vitesse et recule jusqu'à être en mesure de prendre la route (*06_Marchearriere_video_injected.mp4*). Le sujet entend alors (*07_Position D_injected.mp4*) :

« À présent, pressez la pédale de frein et passez en position D afin de permettre la marche avant. Il suffira de relâcher la pédale pour que le véhicule avance et d'appuyer sur la pédale d'accélération pour aller plus vite. Pour placer le sélecteur de vitesses en position D, actionnez le bouton latéral de Google Cardboard. »

Dès que l'interaction a été réalisée, c'est-à-dire dès que l'aimant latéral de Cardboard a été tiré vers le bas, la voiture avance (*08_Marcheavant_injected.mp4*) et le niveau deux est terminé.

Compte tenu de la relative longueur du cas pratique déjà décrit jusqu'ici et surtout de tous les défis auxquels nous allons devoir faire face lors du développement, notamment pour le passage d'une vidéo à une autre et pour gérer les différents types d'interactions, nous développerons pour le moment un prototype le plus fonctionnel possible avec ces deux premiers niveaux.

La suite du cas pratique tel que définie initialement, à savoir les niveaux trois à six (qui reprennent les mêmes types d'interactions sur la route, avec toutefois des éléments visuels nouveaux), peut être consultée en annexe (annexe II).

6. PROTOTYPE

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.

6.1. 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

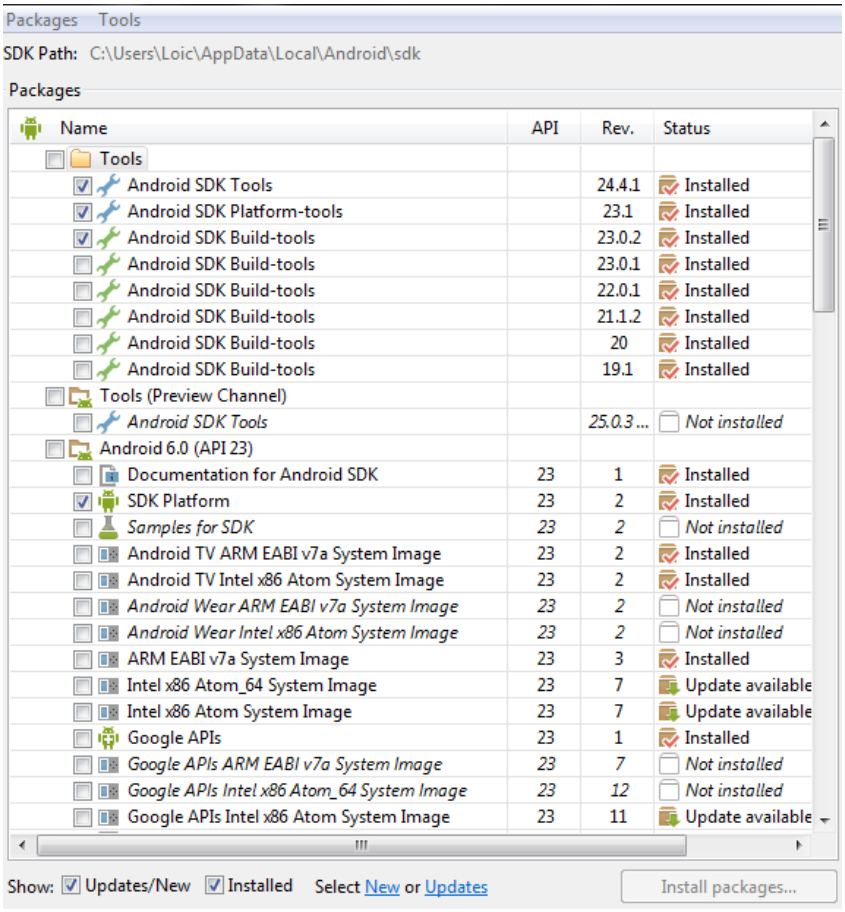

Figure 16 : Android SDK Manager
Source : capture d'écran réalisée par l'auteur

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 :

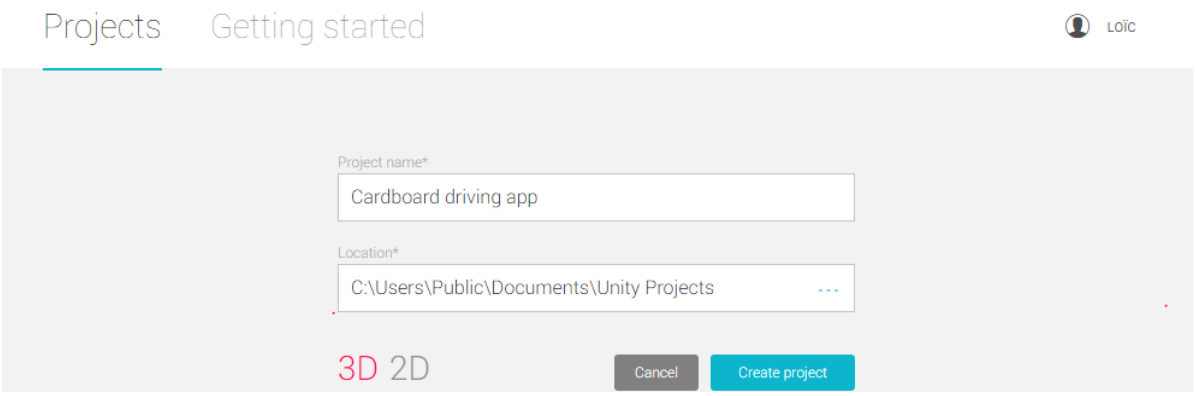

Figure 17 : fenêtre de création d'un nouveau projet Unity
Source : capture d'écran réalisée par l'auteur

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 :

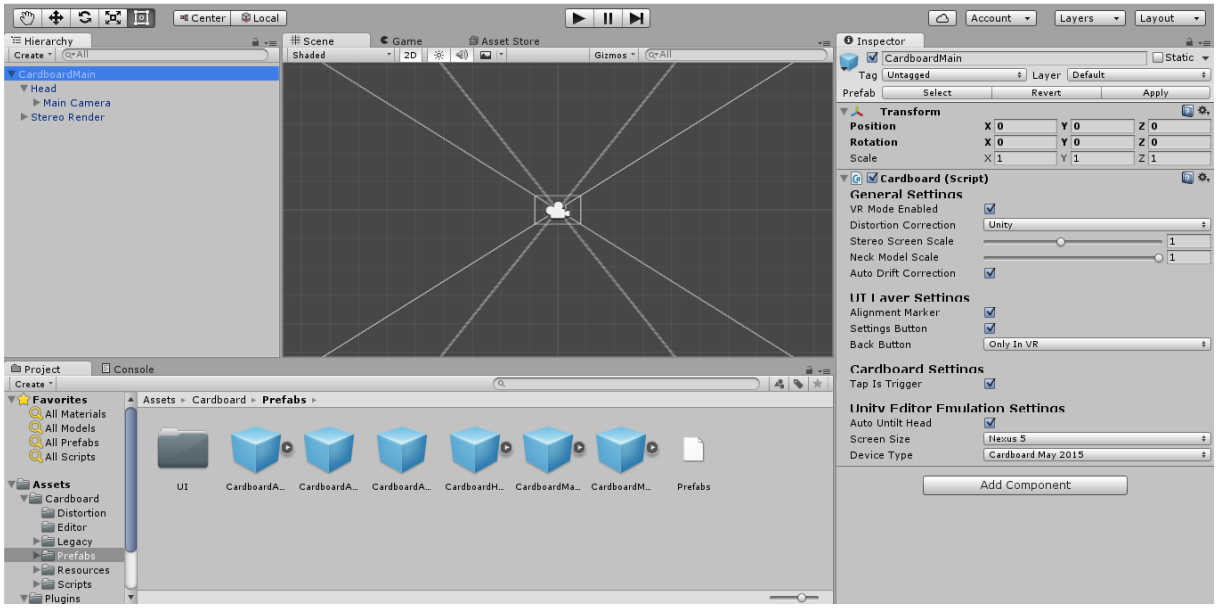

Figure 18 : environnement de travail Unity configuré pour Cardboard
Source : capture d'écran réalisée par l'auteur

[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.

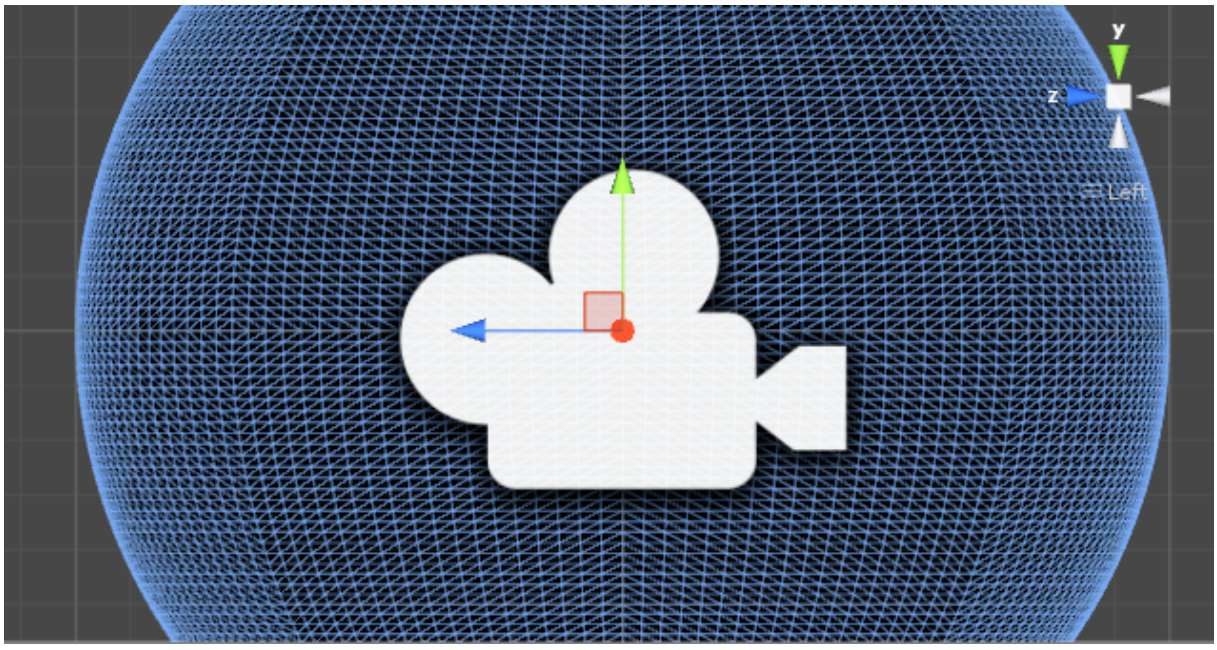

Figure 19 : objet *Main Camera* se situant au cœur d'un objet Sphere
Source : capture d'écran réalisée par l'auteur

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 :

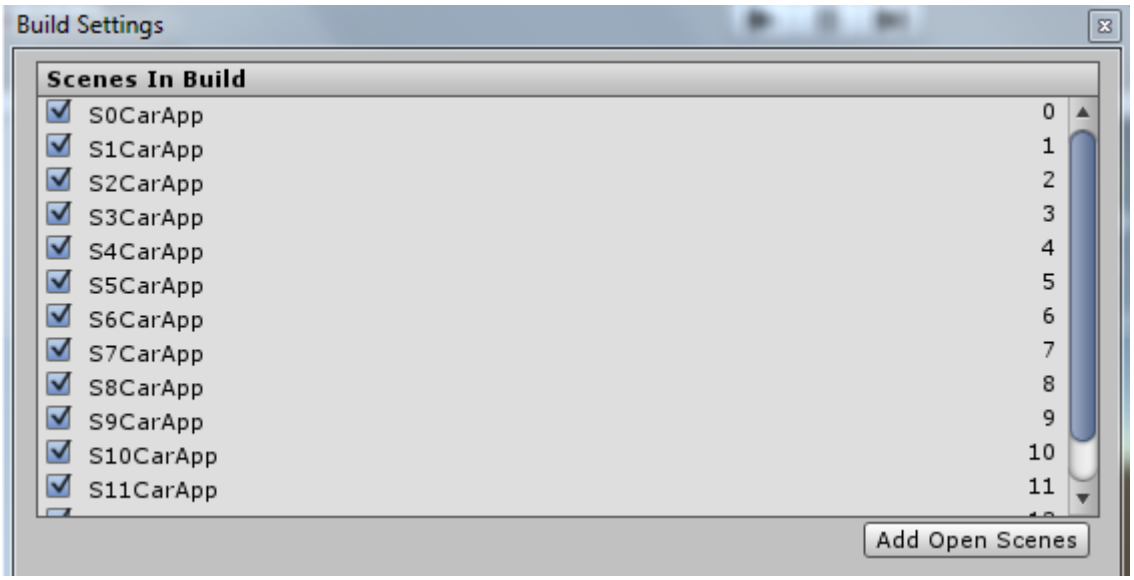

Figure 20 : obligation d'ajouter toutes les scènes du projet dans le menu *Build Settings*
Source : capture d'écran réalisée par l'auteur

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