COARV : Projet Unity

Développement d'un Puzzle game sous Unity en Réalité Virtuelle

Yicheng CHEN , Olivier CORNET , Alexis DUPUIS , Léa GINZBURG , Loïc HENTZ & Hugo PAUGAM

Option Réalité Virtuelle

Lien du Git cassé : `https://github.com/ViviFar/ProjetCONAV`

Lien du nouveau git avec le rendu final : `https://github.com/ViviFar/ProjetUnityV2`

Sommaire

1 Introduction & Objectif du projet

Lors de différents cours pendant l'année, nous avons eu l'occasion de manipuler et de nous familiariser avec Unity et Git, dans le cadre de différentes initiations et TP. Cependant, nous n'avons pas vraiment eu l'occasion de les utiliser dans le cadre d'un projet d'équipe plus concret, dans l'objectif de développer une application en Réalité Virtuelle (excepté lors du Hackathon de Laval). Ce projet a ainsi pour but de poser les bases d'un jeu de type *Puzzle* ou *Escape Game* avec au moins un niveau fonctionnel (incluant des interactions en Réalité Virtuelle). Afin de nous aider dans ce projet, nous avons suivi une courte formation sur les bonnes pratiques à adopter sur Git et Unity.

2 Présentation de l'application

2.1 Réflexions autour de l'application

Nous avons décidé de nous réunir en début de semaine suivant la présentation, afin de déterminer le plus précisément possible quelle application nous allions créer, avec quels objectifs, atteignables par quelles énigmes, faisant intervenir quelles interactions. Même si la réunion a duré plus de 2 heures, il nous semblait important de déterminer le plus tôt possible dans quelle direction allait partir le projet, et de définir le plus de tâches précises à effectuer pour assurer le développement du projet. De cette manière, cela nous permettait d'éviter les incompréhensions entre les différents membres du groupe.

2.2 L'application

2.2.1 L'environnement global

Nous avons choisi de créer un jeu à partir de deux idées principales :

— Le voyage dans le temps.

— Un joueur ayant un rôle de détective.

Le jeu sera composé de 3 scènes distinctes, correspondant à 3 époques différentes :

— Les années 80.

— Les années 2050.

— L'époque western.

Toutes ces scènes correspondent au bureau de notre héros. Il pourra changer de scène par l'intermédiaire d'un walkman, et de cassettes audio correspondant aux sons en vogue dans les différentes époques.

2.2.2 Les années 80 et le début du jeu

Au début du jeu, notre héros se trouve à son époque, dans les années 80, et arrive au bureau comme à son habitude, pour commencer une banale journée de travail. A sa grande surprise, il trouve un walkman sur son bureau, posé en évidence, un pot de fleur vide, ainsi qu'un mystérieux coffre, pour le moment verrouillé. En touchant ce walkman, son "double" du futur va apparaître, porteur d'une mauvaise nouvelle. Il s'adresse alors à notre protagoniste : "Bonjour à toi, moi du passé ! J'ai toujours rêvé de dire ça... bref !

Ne t'inquiète pas, je ne suis pas venu te tuer, mais pour demander ton aide. Vois-tu, notre espèce n'a plus la chance de vivre aux côtés d'arbres, ou ne serait-ce que d'un brin d'herbe. [Pause] Je m'attendais à ce que tu me dévisages, mais sache que mon corps a été victime de radiations, et non pas d'une modélisation lowpoly hasardeuse. Malgré de nombreux signaux d'alarme, vous n'avez pas été capables de nous sauver du réchauffement climatique et la pollution nous intoxique et nous asphyxie, à tel point que notre espèce est vouée à s'éteindre d'ici quelques années. Mais tu peux changer notre funeste destin en ramenant les arbres dans notre époque. Cette aventure te mènera à travers différents âges et époques, entre lesquels tu pourras voyager grâce à Boris, ton fidèle walkman et différentes cassettes audios. Utilise donc celle-ci qui t'amènera à notre époque pour constater les dégâts. Nous comptons sur moi, euh sur toi, Détective ! ". Le joueur devra ainsi voyager à travers les différentes époques pour rétablir l'ordre des choses, et faire revivre les végétaux !

Dans un premier temps, le joueur met la cassette dans le walkman et voyage dans le futur.

2.2.3 Le futur et le synthétiseur

Le voilà téléporté dans le bureau de son alter ego futuriste, absent des lieux. Une fois arrivé, le joueur constatera notamment la présence sur les lieux d'un énigmatique tiroir de "transport interdimensionnel", ainsi que d'un synthétiseur trônant au centre du bureau. En interagissant avec ce dernier il constatera dans un premier temps que lorsque l'on appuie avec une des manettes sur le clavier, celui-ci produit une note. Lorsqu'il enchaîne les différentes notes, une musique rappelant l'époque Western va commencer à jouer. Il faudra que le joueur joue du piano jusqu'à la fin de la musique pour qu'une cassette contenant l'enregistrement de sa prouesse apparaisse, et lui permette de se téléporter dans l'époque des cowboys, où se poursuivra son aventure. L'utilisateur utilisera donc cette nouvelle cassette pour un nouveau voyage temporel.

2.2.4 Le Western et la dynamite

Le détective arrive alors dans un bureau à l'aspect rustique, proche d'un bureau typique d'un shérif dans une ville agitée par des bandits et des cowboys.

Outre les différentes décorations présentes dans le bureau, le joueur remarquera rapidement la présence d'une fissure plus que suspecte sur un pan de mur du bureau, ainsi que de la dynamite. Lancer la dynamite sur le mur entraînera sa destruction, et le joueur pourra récupérer dans le trou béant ainsi créé une clé. A la vue de cette clé, le joueur se souviendra du coffre qu'il a rencontré en début de partie, dans son présent, et va chercher un moyen d'y retourner. Cependant, comme pour les autres époques, il lui faut une cassette correspondant aux années 80 pour retourner dans cette époque avec le walkman. C'est à ce moment-là qu'il se souviendra avoir aperçu dans le futur un lot de cassettes de rock posé sur le bureau, le joueur va ainsi se téléporter dans le futur pour récupérer une de ses cassettes.

2.2.5 Retour dans le futur et choix de la cassette

De retour dans le futur, le joueur trouvera à sa disposition une demi-douzaine de cassettes, mais seule l'une d'entre elles pourra lui permettre de se téléporter à son époque. Laquelle ? Il s'agit de celle qui correspond à la musique d'ambiance que le joueur a eu l'occasion d'écouter au début du jeu, lors de l'arrivée de son "double" du futur dans les années 80. Une fois cette cassette trouvée, le joueur constatera via son walkman que celle-ci

ne peut pas être lue par celui-ci, étant donné qu'elle doit être rembobinée. On utilisera pour cela un crayon présent dans la scène du Western, mais il se pose un problème : comment téléporter la cassette permettant de retourner dans le présent avec nous, pour pouvoir la remonter manuellement dans le passé ? C'est ici qu'intervient le tiroir dimensionnel ! Celui-ci permettra d'accomplir cette tâche : chaque objet placé dans un tiroir interdimensionnel voyagera sans altération d'une époque à l'autre, suivant les voyages accomplis par le joueur. On place ainsi la cassette dans le tiroir interdimensionnel et on retourne dans le Western.

2.2.6 Retour dans le passé et rembobinage de la cassette

Dans le Western, le joueur récupère la cassette, et la rembobine manuellement avec le crayon présent dans la scène. Une fois la cassette remontée, on placera dans le tiroir interdimensionnel la clé du coffre découverte plus tôt (si ce n'est pas déjà fait) et on retourne dans les années 80 avec la cassette rock cette fois-ci fonctionnelle.

2.3 Ultimes voyages et fin du jeu

Dans les années 80, on ouvre le coffre avec la clé stockée dans le tiroir dimensionnel. On trouve à l'intérieur de celui-ci une mystérieuse graine, que l'on s'empresse de planter dans le sol. On voyage ensuite dans le futur pour voir l'impact de notre action. On arrive dans un futur tout vert, où la graine issue du pot de fleurs est devenu un arbre majestueux. Le joueur a sauvé le futur et gagné la partie.

3 Organisation du projet

3.1 Répartition des tâches

On a décidé de désigner quelqu'un en charge du git, puis de séparer les tâches en 3 parties :

— Le design des scènes et objets

— Les scripts

— L'intégration du casque Vive (téléportation, grabbing...).

Voici comment nous nous sommes répartis les tâches :

	Design	Scripts	Intégration RV
Y. CHEN		X	
O. CORNET	X		
A. DUPUIS	X		
V. FARGETTE		X	
L. GINZBURG		X	
L. HENTZ			X
H. PAUGAM	X		

Table 1 – Répartition des tâches au sein du projet

3.2 Choix de design et quelques difficultés rencontrées

3.2.1 Des graphismes “fait-maison"...

Pour rester dans l'esprit d'une véritable gamejam, nous avons décidé de ne pas récupérer d'asset ni de scène toute faite. A la place, chaque élément du jeu que vous verrez a été imaginé puis créé exactement comme nous le désirions. L'avantage est que le jeu a un ressenti plus personnel, avec des scènes créées selon notre vision du jeu, des blagues cachées dans le décor, et une certaine cohérence des style de graphismes à travers les différentes scènes.

Vous pourrez par exemple trouver des références à la série The Office (cf. fig 1), à Doom le premier fps dont le dernier opus a eu droit à une sortie VR en décembre 2017, ou encore Beat Saber l'un des plus gros succès VR de 2018 ou encore au jeu Rez Infinite.

FIGURE 1 – La scène de la série américaine The Office dont le jeu contient une référence cachée.

Ce choix a également été motivé par la volonté de designer les 3 scènes avec un layout similaire afin que le joueur sente qu'il se trouve au même endroit à différentes époques. Les scènes ont donc été pensées avec un layout commun. Les dessins prototypes pour les scènes ont d'ailleurs été réalisés en superposant plusieurs calques afin de garder les mêmes éléments marquants malgré un changement de design radical

censé représenté des époques éloignées.

La première difficultée a donc été de se mettre d'accord sur les dimensions de la scène. Pour la création du décor, Alexis s'est occupé des scènes "années 80" et "futur 2050" sous Blender. Olivier a ensuite designé la scène "western" en prenant lui-aussi le même layout général. Toutefois, il a utilisé directement Unity pour la création. Yicheng s'est occupé de l'extérieur de la scène Western.

Les scènes créées sous Blender ont posées quelques problèmes d'importation dans Unity. On peut normalement utiliser soit un export en *.fbx*, soit directement le fichier .blend pour importer l'asset sous Unity. Malheureusement l'import en *.blend* cause encore beaucoup de soucis au niveau de la gestion des textures et des polices personnalisées, qu'il faut souvent refaire à la main (pas l'UV mapping fort heureusement, mais sur de grosses scènes le fait de devoir recaler tous les matériaux est tout de même fastidieux). Un autre problème très récurrent est l'inversion des normales. Le renderer de Blender affiche souvent les matériaux dans les deux sens, tandis que Unity utilise les normales. Cela crée des trous inattendus dans les meshs qu'il faut alors réparer en inversant les normales sous Blender... puis en réimportant tout le mesh. Pour un peu que vous vouliez importer en *.blend*, il va falloir recaler toutes les textures à nouveau !

Très vite nous avons identifié quelques bonnes pratiques :

— Utiliser des *fbx* autant que possible, qui posent moins de problèmes de textures.

— Éviter de faire un unique gros projet Blender par scène si possible. Même si c'est souvent pratique pour garder les décors à l'échelle, cela demande plus de travail pour réimporter si une modification sous Blender est obligatoire.

— Une fois une scène importée, commencer par vérifier tous les meshs pour être sûr qu'aucune normale n'est inversée. Une fois ceci fait, commencer à corriger les éventuels problèmes de textures.

Pour les animations, Blender permet de "bake" celles-ci dans le fbx. Toutefois, nous avons utilisé ici des animations gérées par des contraintes ("Follow path" pour les voitures volantes). Si Blender permet très facilement de gérer ce types d'animation, le baking est fastidieux et l'export a fini par ne pas fonctionner dans notre cas. En revanche, un bake de toutes les animations est exporté lorsqu'on utilise le *.blend* dans Unity. Pour la scène du futur, nous avions donc le problème d'avoir un import en *.fbx* qui contenait les textures mais pas les animations, et un import en *.blend* qui contenait les animations mais pas les textures. Nous avons donc importé la scène fois : une fois en *.fbx* avec tous les éléments texturés sauf les voitures volantes, et une fois en *.blend* avec uniquement les voitures (pour lesquelles il a donc fallu remettre les textures).

Enfin la difficulté principale a bien sûr été que ce choix de design est très consommateur en temps, puisque faire les décors à la main prend énormément de temps (compter au moins 7h par scène), ce à quoi s'ajoute la difficulté de l'import "à la main" là où télécharger un asset sur l'asset-store est immédiat et garantie sans bugs en général.

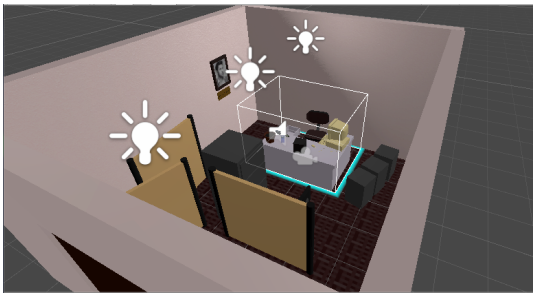
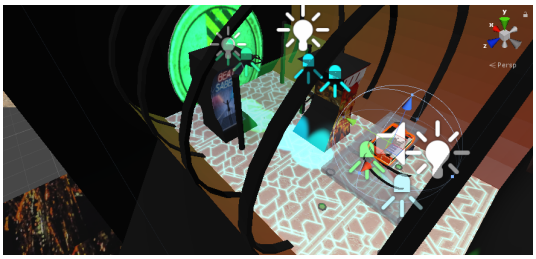


FIGURE 2 – Vues globales de la scène des années 80



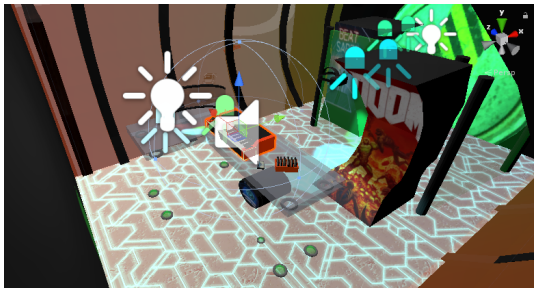


FIGURE 3 – Vues globales de la scène des années 2050s

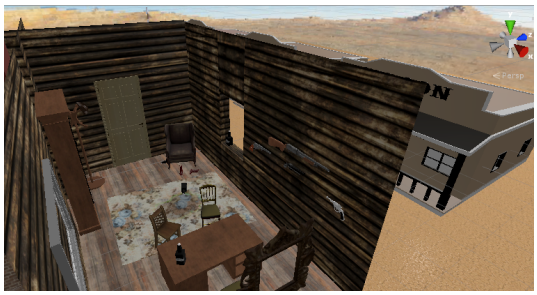


FIGURE 4 – Vues globales de la scène des années Western

3.2.2 Quelques problèmes avec Git

Nous avons fait en sorte d'être très prudents sur l'utilisation de Git. Toutefois, nous avons eu un raté lorsqu'une personne a supprimé par inadvertance des assets utiles à l'utilisation du casque dans la scène avec un push sur le master. Nous avons alors voulu réimporter les éléments manquants, mais Git avait créé une drôle de fusion où les éléments manquants étaient encore présents sous forme de "Missing prefabs" plutôt que clairement supprimés, et les réimporter ne corrigeait pas le problème. Nous avons décidé de revenir en arrière sur le master.

Nous sommes revenus avec succès à un commit antérieur mais quelques personnes avaient travaillé depuis le master "cassé" et n'ont donc pas pu push leur travail. Nous avons à nouveau perdu un peu de temps à

refaire ce travail depuis le commit fonctionnel. Pour récupérer le commit antérieur fonctionnel, nous avons créé une branche *temp_ back_ in_ time* 0 pqrtir de ce commit, en guise de nouveau master, la branche master n'étant plus fonctionnelle. Pour nous assurer de la bonne continuité du projet, nous avons créé un nouveau git, sur lequel se trouve notre rendu final.

3.3 Implémentation du Vive

L'implémentation du Vive a été réalisée à l'aide d'un tutoriel disponible sur `https://www.raywenderlich.com/9189-htc-vive-tutorial-for-unity`. Le joueur pourra se téléporter sur les objets avec le layout CanTeleport en appuyant sur le touchpad et en visant bien le sol, et pourra grab les objets avec le layout Grabbable en utilisant la gâchette à proximité de l'objet que l'on souhaite tenir.

3.3.1 Intégration de la caméra et des controllers à la scène

L'intégration du Vive à la scène nécessitait dans un premier temps d'importer le plugin SteamVR dans le projet.

La caméra a rapidement été mise en place, il suffisait simplement de remplacer la *MainCamera* intégrée par défaut dans le projet par le prefab *CameraRig*, disponible dans l'onglet *Prefabs* du plugin SteamVR. De cette manière, le joueur peut observer directement la scène, et ses controllers sont également affichés.

3.3.2 Intégration du grabbing des objets

Pour pouvoir effectuer le grabbing des objets avec les controllers, une première étape est d'ajouter aux 2 controllers :

— Un *rigidbody* avec l'option *isKinematic* cochée et la gravité non cochée, pour que ceux-ci ne soient pas soumis à la simulation physique, mais contrôlés directement par le joueur.

— Un *Box Collider* où doit être coché *isTrigger*, on modifie également le collider, par défaut beaucoup trop grand : *Center* à (0, -0.04, 0.02) et *Size* à (0.14, 0.07, 0.05).

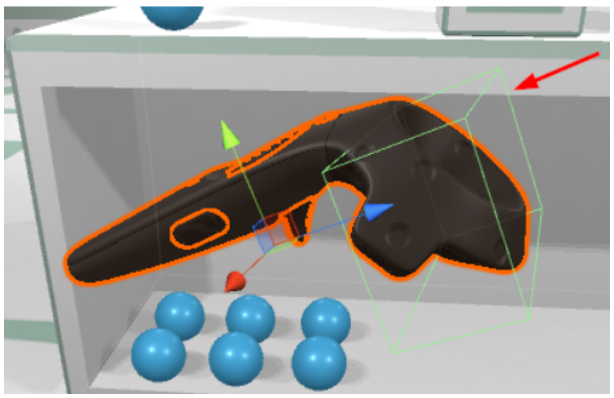


FIGURE 5 – Le collider de bonne taille sur l'un des controllers, adapté pour le grabbing d'objets

La gestion du grabbing d'objets est codée dans le script *ControllerGrabObject*, contenant deux variables différentes :

— *collidingObject*, GameObject avec lequel le trigger a été activé, pour que l'on puisse le prendre.

— *objectInHand*, en guise de référence à l'objet que le joueur est en train de tenir.

Ce script est également constitué de différentes méthodes :

— *SetCollidingObject*, qui permet, à partir d'un collider, de récupérer le GameObject associé pour le prendre et le relâcher, on l'appellera pour les fonctions suivantes.

— *OnTriggerEnter, OnTriggerStay, OnTriggerExit* qui permettent de gérer ce qu'il doit se passer quand le collider trigger entre en contact ou ne touche plus un autre collider.

— *GrabObject*, qui permet de grab un objet via les *FixedJoint*

— *releaseObject*, qui permet de relâcher un objet.

On appellera ces fonctions pour gérer le grabbing des objets dans la fonction Update().

3.3.3 Téléportation dans une pièce via les controllers

Avant de s'attaquer au script de la téléportation, une première étape clé est de définir le pointeur de notre outil de téléportation.

L'intégralité du script de téléportation sera géré dans le script *LaserPointer*. On utilisera la méthode *ShowLaser* dans ce script pour afficher le laser. On l'affichera si le joueur veut se téléporter, et que le rayon émit suite à son action touche un endroit sur la scène où il est possible de se téléporter. Dans ce cas, la position visée par le joueur est enregistrée pour effectuer une éventuelle téléportation, et montrée au joueur par le biais d'un réticule.

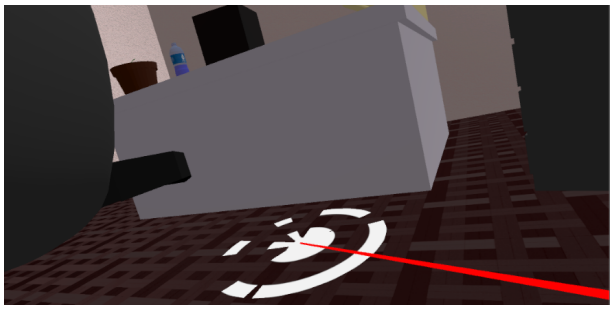


FIGURE 6 – La téléportation en cours avec le controller

On constate qu'il y a un problème d'affichage du curseur, qu'il faudra régler pour la suite du projet.

4 Rédaction des scripts nécessaires

4.1 Changement de scène

Pour gérer les différentes époques du jeu, nous avons choisi de placer trois zones correspondant à chaque époque dans une unique scène Unity. Ainsi, pour téléporter le joueur d'une époque à l'autre, nous effectuons simplement une translation selon l'axe x. Cette action sera effectuée dans le jeu lorsque certaines cassettes entreront en contact avec le walkman.

Dans un premier temps, pour faciliter l'écriture du script, nous avons créé une scène basique comportant :

— 3 plans : 1 pour chaque époque

— un pavé bleu : le futur walkman

— une sphère : le futur personnage

— 3 boutons dans un premier temps pour gérer les changements de zones

— un pavé blanc : une des futures cassettes, créé dans un second temps.

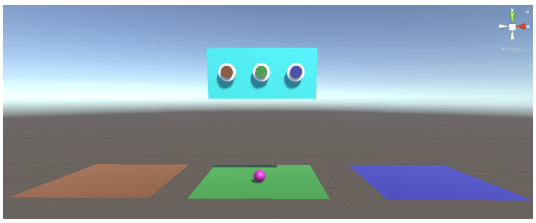


FIGURE 7 – Scène pour la translation avec boutons

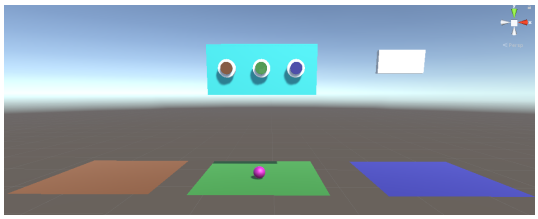


FIGURE 8 – Scène pour la translation avec cassette

La première étape a donc été la gestion des déplacements en appuyant sur des boutons. Pour cela nous avons attaché un script, *ZoneManager*, au pavé bleu. Celui-ci calcule les translations à effectuer entre les différentes zones et comprend notamment 3 fonctions : *GoToA()*, *GoToB()*, *GoToC()*. Ces fonctions gèrent les déplacements dans les zones en prenant bien en compte la position actuelle de l'objet à translater (dans le jeu : le joueur).
Pour la gestion avec les boutons, nous avons créé un script *BigButton*, attaché à chaque bouton. Celui-ci permet de déplacer l'objet dans la scène correspondant au bouton grâce à un clic gauche de souris. Les figures suivantes montrent le déplacement quand on appuie sur le bouton marron puis sur le bouton bleu.

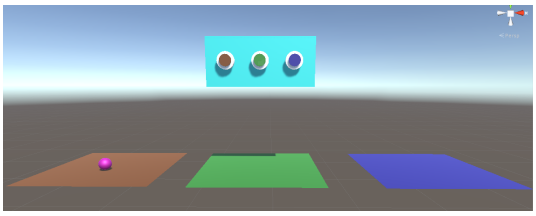


FIGURE 9 – Translation vers la zone de gauche

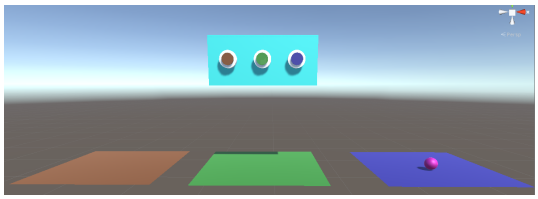


FIGURE 10 – Translation vers la zone de droite

Une fois cette scène basique créée, nous avons voulu mieux l'adapter au jeu en ajouter une cassette dans la scène ainsi qu'un script *Cassette* qui lui est associé. Lorsque la cassette touche le pavé bleu, l'objet est translaté dans la zone voulu grâce à la mise en collision des deux pavés. Nous implémentons également le lancement de la musique associée à la cassette lors de la collision. Dans le script *Cassette*, nous gérons également l'interaction avec les différentes cassettes et le walkman. En effet, nous avons prévu que si une cassette est déjà présente dans le walkman, celle-ci est remplacée par la nouvelle et l'ancienne se retrouve dans le tiroir inter-dimensionnel. La figure suivante représente l'effet de la collision entre la cassette et le walkman, pour une configuration d'une translation dans la zone de gauche.

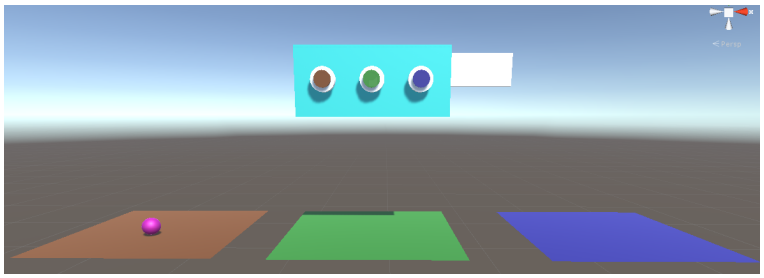


FIGURE 11 – Translation dans la zone de gauche grâce à une cassette

4.2 Synthétiseur

Pour passer de la scène du futur à la scène du passé, le joueur devra utiliser un synthétiseur afin de récupérer une cassette avec une musique évoquant le Western. Nous avons donc associé au synthétiseur un script *PlayingSynthesizer* qui permet de faire cela. Lorsque le joueur touche le synthé avec un controller, une unique note de musique est jouée. S'il poursuit en touchant plusieurs fois d'affilée le synthé avec les deux controllers, une musique continue se lance, la musique correspondant au Western. On fixe un temps à atteindre pour que la cassette apparaisse.

FIGURE 12 – Synthétiseur dans les années 2050

4.3 Ouverture du coffre

Nous allons ici parler de l'implémentation de l'ouverture du coffre pour le final du jeu. Malheureusement cette partie n'a pas pu être finalisée par manque de temps. Le but était ici de programmer le lancement d'une animation sur le coffre présent dans les années 80 lorsqu'on en approchait la clé trouvée dans le passé. Pour cela nous avons à nouveau créé une scène basique avec simplement un cube en guise de coffre et une capsule aplatie pour la clé. Nous avons attaché un script *OuvrirCoffre* au coffre. Celui-ci permet de lancer une animation *Animator* lorsque la clé entre en contact avec lui. Nous n'avons pas eu le temps de créer l'animation pour la scène finale.

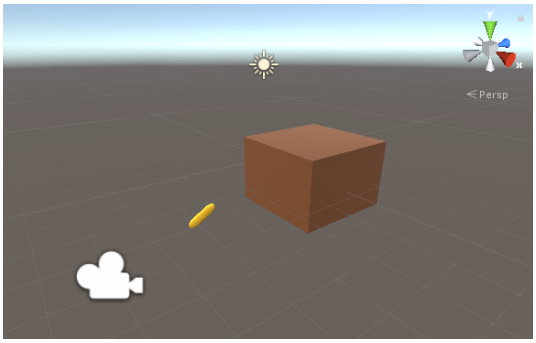


FIGURE 13 – Scène pour l'ouverture du coffre avec une clé

5 Conclusion

Comme toujours dans ce genre de projet, les attentes ne sont pas les mêmes que le produit final. Ici, nous avons globalement réussi à tenir nos exigences en terme de graphismes. En revanche nous sommes allé à l'essentiel sur certains points du scénario :

— Le double du futur n'apparaît pas pour expliquer le scénario. Nous avions trouvé une solution pour animer le personnage en motion capture via une Kinect, mais le temps nécessaire à enregistrer la voix, chorégraphier et jouer les mouvements et bake les animations sous Blende puis les exporter a été jugé trop grand par rapport à l'impact réel de cet élément sur la jouabilité.

— Nous avions imaginé de jouer différentes musiques pour les mauvaises cassettes, suivi d'un son indiquant qu'elles sont abîmées par exemple, mais nous avons mis cette idée de côté.

— L'énigme du rembobinage via un crayon a été abandonnée car nous n'avions plus le temps de réaliser les scripts nécessaires.

— Nous aurions aimé faire un fade-to-black durant la téléportation.

— Nous n'avons pas de scène finale correspondant aux attributs "un futur tout vert"... Donc pas d'écran de fin à proprement parler.

Malgré cela, nous restons assez satisfait du travail effectué. Nous aurions tout de même aimé avoir un peu plus de temps pour rendre un travail fini et un produit jouable.