

Projet de développement

**VI50**

---

# Rapport de projet

## EXPERIENCE MUSICALE ET VISUELLE INTERACTIVE

# INTRODUCTION

## Vision et Réalité Virtuelle

Le projet dont ce rapport fait l'objet a été réalisé par un groupe de deux étudiants du Département Informatique de l'UTBM en filière I2RV, dans le cadre de l'UV VI50: vision et réalité virtuelle.

Le large choix de sujets proposés et leur liberté d'interprétation nous a permis de travailler avec une motivation certaine produite par le choix d'un sujet très intéressant à nos yeux.

Nous groupe a donc retenu le sujet suivant :

*“Sujet 6 : Expérience musicale et visuelle interactive”*

## Outils

Afin de nous guider dans la réalisation du projet, les premières séances de travaux pratiques du semestre ont été consacrées à la prise en main des différents matériels à notre disposition.

Le travail a été entièrement réalisé à travers le moteur de jeu Unity3D.

Pendant ces séances de travaux pratiques, nous avons en l'occurrence appris à développer pour Unity et intégrer les flux de données de plusieurs medium de réalité virtuels dans ce logiciel dont Kinect for Windows que nous utilisons dans notre solution.

## I. Concept

### I.I Objectifs du projet

Le sujet choisi laisse libre place à l'imagination quand à l'application qui va être développée. Une première réflexion quand aux objectifs visés nous a permis de cerner les besoins et la direction à suivre. Dans cette partie sont détaillés les besoins que nous avons retenus, mis en lien avec les directives imposées par le sujet. Notre application vise principalement à relaxer et distraire, en installant diverses ambiances musicales pour l'utilisateur tout en lui laissant la possibilité de manipuler le son.

#### I.I.I Expérience musicale

Le projet cherche en premier lieu à faire vivre à l'utilisateur une expérience auditive. La musique dans l'application prend deux aspects différents, représentés respectivement par deux modes qui peuvent être utilisés alternativement.

##### Écoute

D'abord, l'utilisateur peut simplement lancer l'écoute d'un morceau qu'il affectionne et observer l'univers dans lequel il est plongé réagir autour de lui.

##### Manipulation

Ensuite, l'utilisateur peut créer sa propre musique à partir de sons proposés et représentés de façon concrète afin qu'il puisse les manipuler. Un retour vers le mode d'écoute lui permet une fois les manipulations terminées d'entendre le résultat de ses expériences..

#### I.I.II Expérience visuelle

Pour accompagner cette expérience musicale arrive en second lieu la notion d'expérience visuelle. En effet, le regard est lui aussi sollicité à plusieurs points de vue.

##### Ambiance

Dès son immersion dans l'application, l'utilisateur découvre une ambiance très différente du monde extérieur. Pendant une brève période de découverte, il peut prendre connaissance de l'intégralité de l'univers qui l'entoure par simple mouvement de la tête, possible dans toutes les directions, ou encore s'amuser à constater la présence ses bras et ses jambes reproduits dans l'application et qui suivent ses mouvements réels.

## Objets animés sensibles au son

Dans un second temps, l'utilisateur sera amené à vivre un autre genre d'expérience visuelle, qui consiste cette fois à observer le mouvement d'objets du monde ayant la particularité de réagir au son produit. Ces objets contribuent à l'ambiance globale de l'univers du fait de leur adéquation avec le rythme de la musique écoutée : un morceau lent produira des mouvements calmes et aura un effet relaxant voir hypnotisant sur l'utilisateur tandis que des mouvements énergiques seront provoqués par un morceau plus rapide ou utilisant des fréquences plus perçantes.

### I.III Expérience interactive

L'interactivité se situe dans le séquenceur que nous avons choisi de créer.

L'utilisateur a dans cette application d'activer une note donnée au moment qu'il choisit pour ainsi créer des boucles musicales qui feront réagir l'environnement comme cité précédemment.

Ces changements dans l'environnement prendront alors effet immédiatement, reliant les actions de l'utilisateur à des résultats visuels et sonores.

## I.II Immersion utilisateur

### I.II.I Métaphore : le vinyle

Vinyle + pistes + notes représentées visuellement donc manipulable

### I.II.II Interactions et schèmes

Deux sens mis à contribution : ouïe et vue

Manipulation à l'aide de geste naturels (levée et descente) et sélection avec le regard

### I.II.III Quasi-absence de déplacement

Univers abstrait, plongée dans autre chose, découverte par le regard et l'expérimentation des actions et réactions de l'application. Absence de déplacement qui pourrait donner lieu à un retour à la réalité (ah, la kinect t'as perdu ....)

## I.III Matériel utilisé

### I.III.I Kinect

La Kinect nous a semblé un médium évident pour faire passer les commandes de l'utilisateur dans notre application sachant que le sujet comporte les termes "le corps comme instrument", la représentation du corps sous forme d'un avatar était évidente.

Nous avions eu l'occasion de nous faire la main avec ce matériel en travaux pratiques, et la disponibilité de plusieurs exemplaires du dispositif nous a permis de passer beaucoup de temps à l'étudier. De plus, un asset très pratique disponible gratuitement sur le Store Unity nous a permis une intégration très facile après quelques recherches.

Nous nous servons de la Kinect pour capter des mouvements amples de la main pour sélectionner les différents modes de notre application.

### I.III.II Oculus Rift

Nous avons décidé l'Oculus Rift pour plusieurs raisons.

Tout d'abord son import dans Unity est très aisée, son attrait technologique est important, et plus important, il fournit un sentiment d'immersion impressionnant.

Nous utilisons donc le Rift pour donner un point de vue à la première personne sans limites de bordures pour l'utilisateur ce qui permet de créer une grande scène explorable par le regard sans avoir à se déplacer mais en gardant une impression de volume.

Le regard de l'utilisateur est donc couplé avec celui de son avatar dans notre application, de plus la visée ( sélection des objets ) se fait avec le regard.

### I.III.III Clic

Nous avions tout d'abord pensé au clic classique d'une souris sans fil, puis le professeur nous a proposé d'utiliser un gant de données ce qui permettrait une immersion plus grande avec l'utilisation de la main et de gestes précis pour les doigts.

Toutefois ce gant n'étant pas disponible, nous nous sommes rabattus sur l'utilisation d'une simple télécommande de présentation. Présentant un clic simple, elle ne rappelle pas la forme d'une souris et est plutôt ergonomique concernant la prise en main.

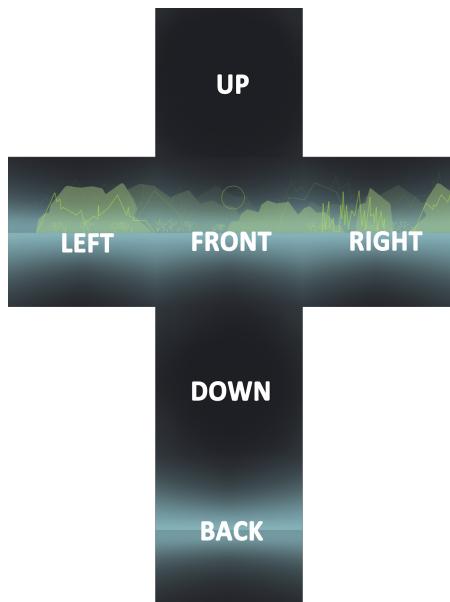
Cela peut être important dans le sentiment de dépaysement que l'on recherche dans une application de réalité virtuelle, et précisément dans cette application.

## II. Réalisation

Le projet est réalisé sous Unity, moteur 3D vers lequel nous avons été fortement orienté lors des travaux pratiques. Unity permet un interfaçage simple entre la machine, et plus particulièrement l'application, et le matériel périphérique nécessaire au fonctionnement de celle-ci. Le logiciel offre une prise en main rapide grâce à son interface intuitive, de plus, il permet une grande liberté dans la création à travers l'ajout de scripts qui peuvent être rattachés à chaque objet de l'univers. Les scripts existants utilisés ainsi que ceux créés pour les besoins du projet sont codés en C#.

### II.I Graphismes

#### II.I.I Skybox



L'horizon, le ciel et le sol du monde sont dessinés à l'aide d'une Skybox, à savoir un cube de très grande dimension qui place son centre au niveau de la caméra et dont les faces intérieures sont couvertes de six textures qui lui sont rattachées, comme le montre la figure ci-contre.

Cet outil correspond au besoin dans le sens où la position de l'utilisateur est globalement fixe et où l'on souhaite composer une ambiance surréaliste dès l'immersion.

Notre skybox a été créée manuellement sous Photoshop, la difficulté étant d'annuler l'effet de coins en trouvant la juste déformation de l'image.

#### II.I.II Avatar

L'avatar utilisé provient du package Kinect with MS-SDK disponible sur le Unity Asset Store et utilisé en séance de travaux pratiques.

Etant donné que l'application utilise une vue à la première personne, une faible importance a été accordée à la modélisation du tronc et de la tête du personnage : en effet seuls ses bras et jambes sont visibles lors de



l'utilisation. Une texture très simple transparente et ne réagissant pas à la lumière lui est appliquée.

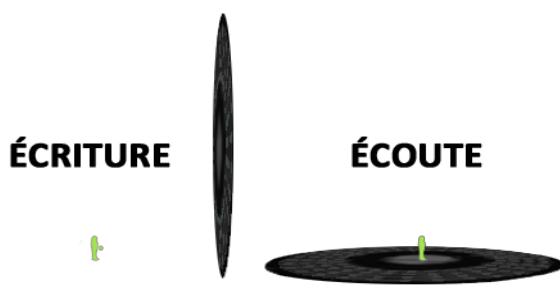
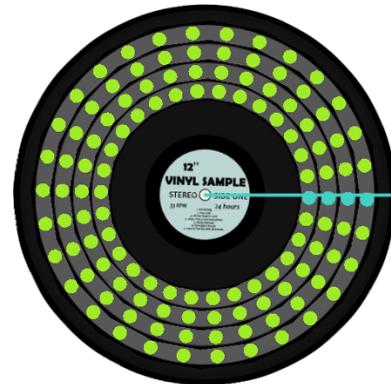
### II.I.III Vinyl

Le vinyl est un cylindre plat contenant quatre pistes. Au lancement de l'application, chacune des pistes génère un ensemble de notes réparties sur sa surface.

Lors de la lecture, le vinyl entre en rotation selon l'axe y.

Une aiguille fixe, appelée *Needle* et représentée en bleu sur la figure ci-contre, entre alors en collision avec certaines notes ayant pour effet de jouer le son qui leur est attribué.

Les notes sont générées en début de programme respectivement par chacune des pistes à l'aide du script *GenerateNotes*.



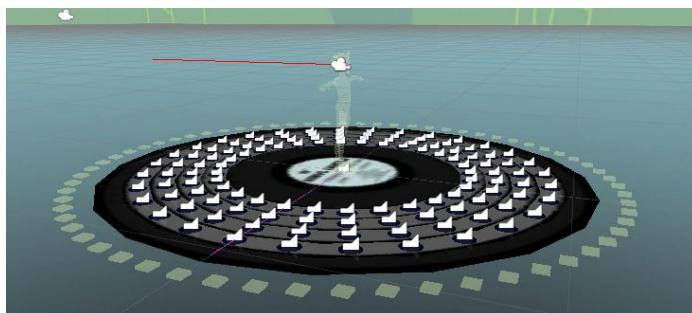
Le vinyl peut également subir un déplacement représentant le passage de la fonction d'écoute à la fonction d'écriture comme illustré à gauche. La position de l'avatar est repérable grâce à la petite silhouette verte. Ce déplacement est adouci par son exécution sur plusieurs Frame selon un facteur d'adoucissement.

### II.I.IV Objets animés

Retenant le concept d'interaction à la fois musicale et visuelle, plusieurs objets sont animés de manière à attirer le regard de l'utilisateur. Notre application étant à la fois un lecteur de musique et un séquenceur, nous nous sommes concentrés sur la visualisation de la musique pour rester cohérent à la fois par rapport à notre sujet que qu'à notre solution.

#### Niveaux de fréquences

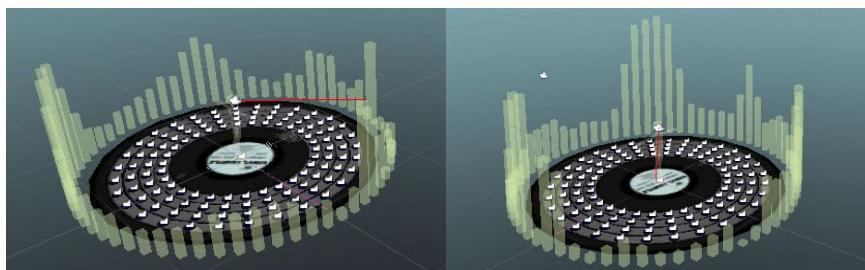
Les niveaux de fréquence sont affichées à l'utilisateur par le biais d'un ensemble de cubes circulaires dont la hauteur (composante Y) varie à chaque instant selon la fréquence qui lui est attribuée par rapport à l'ensemble de fréquences jouée à l'instant t.



Au départ les cubes ont tous une hauteur négligeable et apparaissent plats pour signifier qu'aucun son n'est capté.

Ils sont générés au lancement de

l'application et encerclent le vinyl pour créer un espace de confinement autour de l'utilisateur.

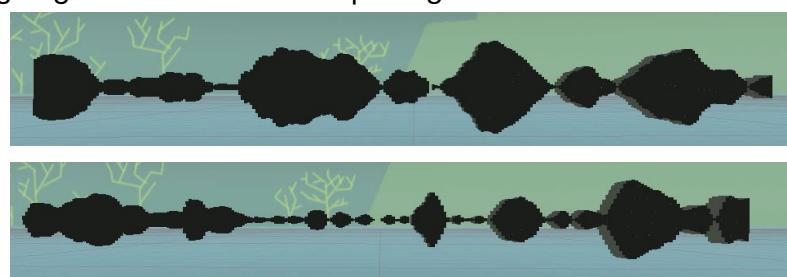


Ensuite, chaque cube réagit à la fréquence qui lui est propre et ce à chaque instant, ce qui permet de créer une certaine visualisation de volumes de la musique.

## Spectre audio

Autre visualisation concernant la fréquence, un spectre audio est dessiné à hauteur de vue de l'utilisateur, lui aussi réagissant à la musique à chaque instant.

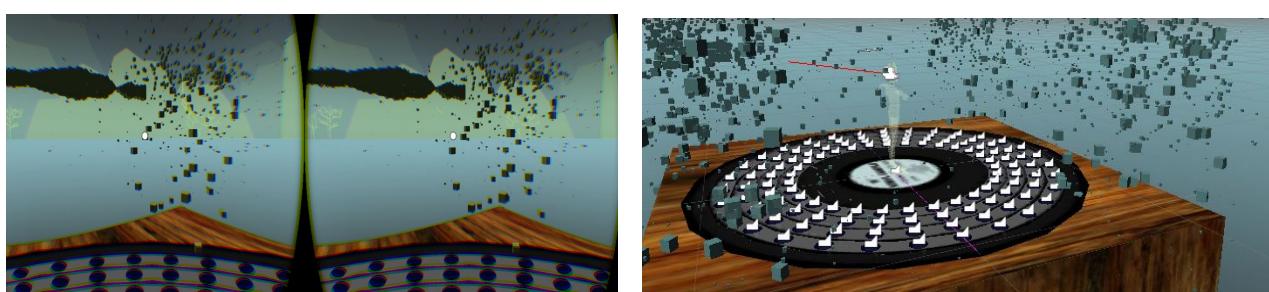
Sa couleur change également au fil du temps et graduellement avec la musique.



## Systèmes de particules

Quatre systèmes de particules sont placés au coin du support du vinyl et produisent en continu des particules éphémères qui changent de couleur avec la musique et le spectre cité précédemment, et changent également de forme avant de mourir: elles éclatent en une sous-émission de particules plus fines et de longévité moindre.

De plus, elles sont animées en taille plusieurs fois par seconde, ce qui donne une illusion de battement, et par ce biais, une visualisation du rythme.



Les systèmes de particules remplissent l'espace sans l'obstruer ce qui permet d'attirer le regard de l'utilisateur et d'accroître le sentiment d'immersion dans la simulation.

## II.II Audio

Unity est un logiciel très pratique concernant l'importation de sons puisque plusieurs formats sont acceptés (mp3, ogg) et le GameObject AudioSource est déjà implémenté.

Il suffit donc de glisser ses fichiers de musique dans une scène pour qu'ils soient instantanément joués et inclus dans l'architecture du projet.

Toutefois la prise en main des outils sonores d'Unity fut longue car de nombreuses options sont disponibles.

### II.II.I Séquenceur

Dans notre solution finale, les sons ne sont pas synthétisés comme nous l'avions souhaité au début. Un manque de temps nous a contraint à abandonner le portage d'application telle que C# Synth Project vers Unity.

Finalement, nous stockons des SoundFount, des petits fichiers mp3 très courts (0 à 2 secondes) qui joue l'enregistrement d'une note donnée d'un instrument donné.

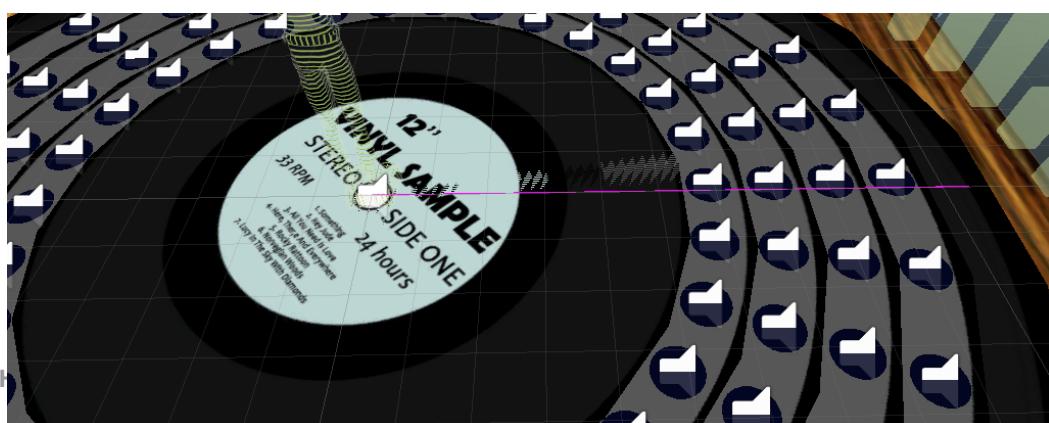
A0	A1	A2	A3
A4	A5	A6	A7
Ab1	Ab2	Ab3	Ab4
Ab5	Ab6	Ab7	B0
B1	B2	B3	B4

Ces notes sont affectées à leurs représentations graphiques: des petites sphères imbriquées dans leurs pistes parentes.

Un objet nommé Needle, est placé en travers d'une moitié du vinyl, et entre donc en collision avec les notes entraînées par la rotation du vinyl.

Au moment de la collision avec cet objet, chaque note est jouée pendant un court instant ce qui permet de créer une boucle musicale avec une révolution du vinyl.

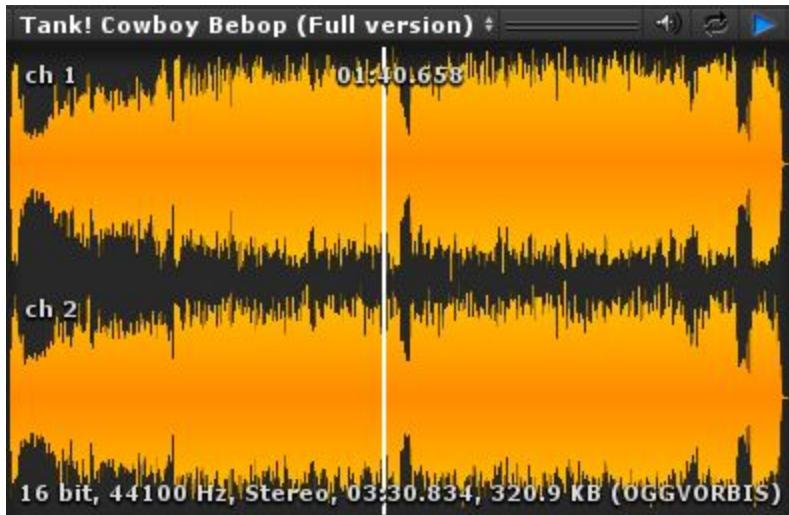
On peut observer Needle en violet ci-dessous, ainsi que sa collision avec quatre notes:



### II.II.II Réaction des objets au son

L'objet  `AudioSource`  dans Unity peut être manipulé de plusieurs manières, et particulièrement en tant que spectre de fréquence.

En effet, comme on peut le voir sur cette image, un morceau est une succession de pics en fréquence:



Grâce à l'instruction  `AudioSource.GetSpectrumData`  , il est possible de récupérer ces données de fréquences, il ne reste plus alors qu'à exécuter des tests sur l'ensemble de fréquences jouées à l'instant t et qu'à animer les objets en fonction de leurs résultats.

Parmi des exemples que nous avons implémentés on peut citer entre autres:

- le comportement du spectre cubique où chacun de ses composants réagit à une fréquence donnée, ou bien le cercle de cube où la hauteur de chaque cube est mise à l'échelle de la puissance de la fréquence qui lui est attribuée.
- les systèmes de particules qui changent de couleur d'émission en fonction du volume, et la mise à l'échelle des particules en fonction du battement.

## II.III Curseur

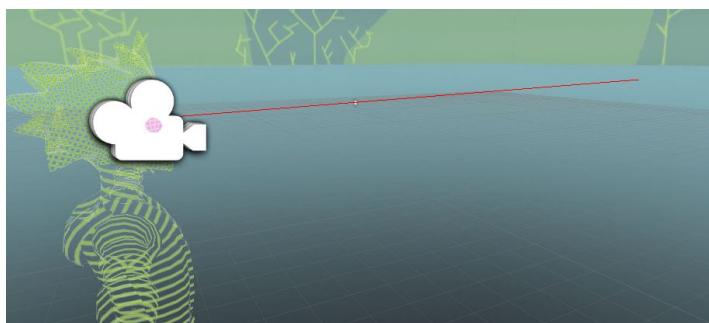
### I.III.I Réticule

Le système de visée est basée sur le Prefab Unity/Oculus: OVRPlayerController.

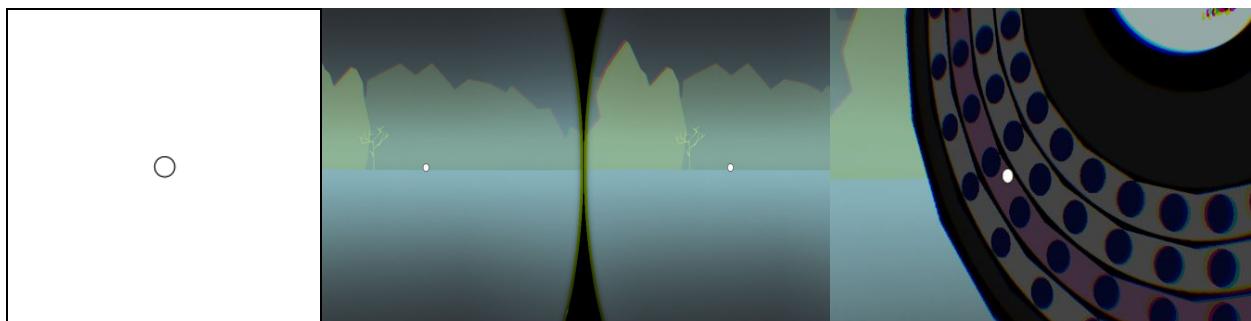
De ce Prefab on n'utilise que le système de caméra OVRCameraRig, composé de deux caméras qui représentent les deux yeux de l'utilisateur ainsi qu'une ancre représentant la position entre les deux yeux.

Notre réticule est visé par l'oeil gauche et représenté sur un plan qui se place de manière à être toujours en face du point d'ancrage entre les deux yeux.

On cast un rayon rouge entre la caméra de l'oeil gauche et le plan du réticule pour représenter la visée:



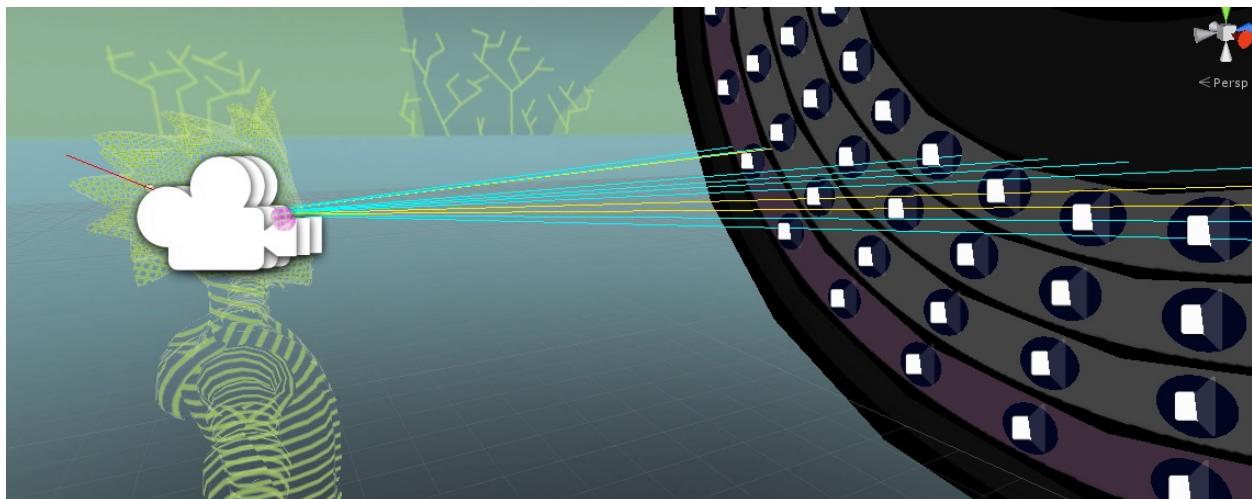
Ce réticule est composée d'un simple point blanc pour ne pas trop obstruer la vision et amener une certaine précision à la sélection par mouvement de tête:



### I.III.II Survol et clic

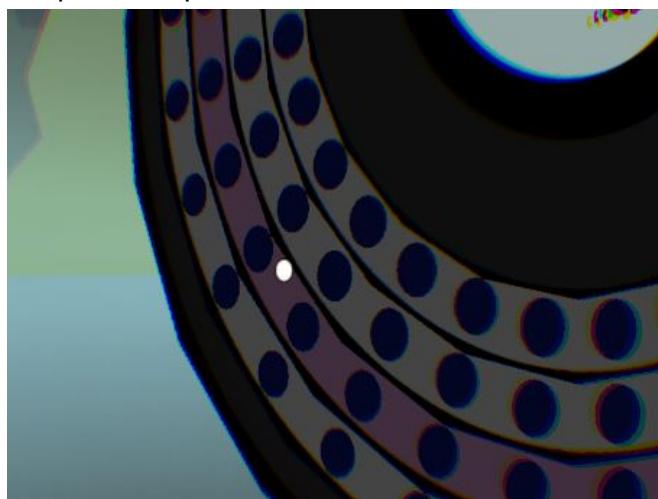
On implémente le survol des objets grâce à un Raycast, c'est à dire un rayon tracé dans la direction du regard qui s'arrête à la première collision avec un objet et retourne cet objet ainsi que les informations qui lui correspondent.

Sur cette image on affiche, après un balayage de l'écran de la droite vers la gauche, un rayon bleu au survol d'une piste, et jaune au survol d'une note. Quant au regard, il est toujours représenté par le rayon rouge orienté vers l'extrême gauche de l'image.



On aperçoit en couleur violette que le Track#4, la piste la plus extérieure, est celle qui a été survolée en dernier par le regard de l'utilisateur, le rayon s'effacera rapidement lors de la frame suivante, entraînant la fin du survol de cette piste, et donc un retour vers sa couleur d'origine. Bien sûr ces rayons sont créés dans un but de développement et d'aide visuelle et ne sont pas représentés dans l'application.

Voici d'ailleurs la vue à la première personne lors de la sélection:



Quant au clic, il est simplement utilisé pour sélectionner ou désélectionner des objets. Par exemple, il permet de choisir la piste, d'activer les notes etc.. Il est le seul élément actif extérieur avec la Kinect et l'Oculus Rift.

## II.IV Alternance des modes

### II.IV.I Détection de mouvement

La détection de mouvements est assurée par le script `KinectVinylGestureListener`. Il s'appuie sur le package `Kinect with MS-SDK` qui propose une détection de mouvements prédéfinis. Les mouvements utilisés sont respectivement `Swipe Up`, qui permet de lever le vinyle en position verticale de façon intuitive pour activer le mode écriture, et `Swipe Down` qui réciproquement abaisse le disque pour le faire revenir à sa position initiale et passer en mode lecture.

Dans un premier temps, un curseur avait été conçu à l'aide d'un petit objet sphérique flottant dans l'espace et suivant la main de l'utilisateur. Cependant quelques tests ont révélé une confusion entre les mouvements liés au changement de mode et les mouvements liés au curseur, causée par le fait que tous trois sont basés sur la main de l'utilisateur. La solution du réticule a alors été retenue et développée.

Ces opérations sont effectuées à partir de la carte de profondeur envoyée par la Kinect, et à un niveau supérieur la position de la main droite. Un mode gaucher peut être activé dynamiquement à l'aide d'une variable publique.

### II.IV.II Scripts de contrôle

Les scripts de contrôle sont l'ensemble des scripts internes et transparents qui assurent le bon fonctionnement du projet. On y trouve par exemple une transition correcte entre les modes avec blocage des fonctions propres à la lecture lors de l'écriture. Sont également classés comme tels tous les scripts de sélection/désélection des notes et des pistes, en incluant le changement d'apparence au survol.

## III. Critique

### II.I Atteinte des objectifs

Nous avons respecté nos objectifs dans le sens où nous avons réalisé tout ce que nous avions fixé sur papier lors du premier rendu.

L'application est agréable, originale et respecte toutes les conditions concernant le confort de son utilisateur.

Nous avons concrétisé les métaphores que nous avons présenté d'une manière qui nous satisfait tout en restant dans la ligne conductrice de notre projet: présenter une immersion satisfaisante.

Peut-être aurions-nous pu toutefois créer un séquenceur plus "professionnel" avec un peu plus de temps, car cela nous aurait demandé de nous impliquer dans une étude plus musicale de cet aspect du projet.

En conclusion, nous considérons nos objectifs atteint: une application originale, complète, accessible, la création d'une immersion agréable et d'une ambiance spéciale.

### II.II Difficultés rencontrées

Comme expliqué précédemment, la partie la plus difficile fût de créer le séquenceur audio puisqu'il a fallu opter pour une solution à la fois facile à développer et facile à utiliser:

En effet, un séquenceur trop complexe aurait demandé des utilisateurs des aptitudes musicales certaines, tandis qu'un trop simple perdrat vite de son intérêt.

D'évidente difficultés ont surgi lors de l'intégration de la Kinect et de l'Oculus Rift ou bien lors d'utilisations plus poussées de Unity, mais la présence d'une communauté de développeurs prêt à résoudre les problèmes de leurs collègues et une documentation fournie nous ont permis de passer outre assez rapidement.

### II.III Améliorations futures

Trouver un compromis plus performant concernant le côté musical pourrait être une amélioration de cette application.

Peut-être le choix de plusieurs environnements, c'est à dire ajouter des scènes aux visuels différents pour créer d'autres ambiances, ou bien une augmentation du nombre de vinyles pourrait aussi être un apport intéressant à notre expérience musicale interactive.

## CONCLUSION

## RESSOURCES UTILISÉES

Kinect with MS-SDK par Rumen Filkov  
<http://rfilkov.com/2013/12/16/kinect-with-ms-sdk/>

**Inspiration:** Installation DYSKOGRAF  
<http://www.avoka.fr/portfolio/dyskograf/>

## DESCRIPTION DES SCRIPTS

### ActivateSound

Permet l'activation ou la désactivation d'une note avec attribution d'un son.

### CubeWall

Permet la génération du mur d'objets animés cerclant le vinyl et réagissant au spectre audio à partir d'un objet initial *Cube*.

### GenerateNotes

Permet la génération d'un cercle de notes sur chaque piste à partir d'une *Note#0*, du rayon moyen de la piste et du nombre de notes souhaitées.

### HilightOnHover

Permet le surlignement de l'objet au passage du réticule ainsi que la sélection

### KinectHandCursor

Permet d'afficher un objet sphère qui suit la main préférentielle de l'utilisateur et fait office de curseur. Remarque : ce curseur n'est pas utilisé dans la version finale du projet.

### KinectVinylGestureListener

Permet de détecter un mouvement de levé de bras-main ou d'abaissement de bras-main en étendant *KinectGestures.GestureListenerInterface*.

### PlayVinyl

Permet de lancer ou de stopper la rotation du vinyl sur l'axe orthogonal passant par son centre.

### Reticle

Permet l'utilisation du réticule au centre de l'Oculus Rift pour créer un *RayCast* qui va entrer en contact avec les *Collider* des différents objets et appeler lorsque c'est le cas leur fonction *OnMouseExit/Down/Enter*. Il fait ainsi office de curseur complété par le clic de la commande à distance.

### Sequencer

## VI50 RAPPORT DE PROJET EXPÉRIENCE MUSICALE VISUELLE ET INTERACTIVE

Permet la détection de la collision entre l'aiguille *Needle* et les notes, causant le jeu du son attribué aux notes touchées et leur coloration au moment où elles sont jouées.

### SpectrumAnalyser

Permet la mise en mouvement du mur de *Cube* en fonction du spectre audio enregistré à chaque *Frame*.

### SwitchMode

Permet le passage physique et logique entre les deux modes à l'aide d'un *KinectVinylGestureListener*.