# PROJET DE FIN D'ETUDE

## Interaction 2D en réalité virtuelle

**Ji YANG**

**Tuteur : Laurant GRISONI**

# REMERCIEMENTS

Nous souhaitons dans un premier temps, remercier Laurent GRISONI, notre tuteur, de nous permettre la réalisation de ce projet. Nous aimerions aussi remercier Michel AMBERG ainsi que Frédéric GIRAUD qui nous ont aidés à faire avancer notre travail.

# Table des matières

# Contexte du projet

Un certain nombre de logiciels permettent à l'heure actuelle de redonner vie à de vieux logiciels de jeux (jeux devenus libres de droit car délaissés par leurs propriétaires, ces jeux sont dits "abandonware"). D'autres jeux existent, totalement libres de droits dès leur diffusion. Dans le cadre d'un projet européen auquel l'équipe de recherche MINT participe, nous souhaitons mettre en place un prototype permettant à une structure de rééducation spécialisé dans la rééducation de l'enfant cérébrolésé de disposer d'un système de réalité virtuelle via lequel l'enfant peut jouer à l'un ou l'autre de ces vieux jeux. Le fait de réaliser cet objectif en réalité virtuelle plutôt que via un dispositif d'interaction standard est le suivant : la réalité virtuelle permettra ici de se libérer des contraintes matérielles, permettra d'adapter l'écran virtuel à la réalité de l'enfant, pourquoi pas par la suite de mettre en place des distracteurs ou d'augmenter la difficulté de l'interaction, etc... et ce afin de permettre aux soignants de disposer d'une application qui leur permette de graduer la difficulté d'interaction (permettant ainsi à l'enfant d'entrainer ses capacités motrices et cognitives), tout en disposant d'une base de jeux importantes et donc la dimension ludique n'est plus à prouver. Le travail consistera tout d'abord en un examen des différents émulateurs existants, et en la proposition d'une configuration permettant de réaliser l'interaction d'un affichage 2D dans un environnement type HTC Vive.

On travaillera ensuite à l'intégration d'un système simulant le "touch" à partir des mouvements des mains de l'utilisateur.

En fonction de l'avancement, on pourra explorer quelques configurations d'interaction mettant en valeur plusieurs configurations virtuelles différentes, permettant par la suite d'explorer les modalités d'interaction (ce dernier point pourra se faire en collaboration avec une structure médicale basée à Meerbush, près de Düsseldorf en Allemagne).

# Choix de l’émulateur de jeux vidéo

Maintenant, il y a deux groupes principales d’émulateur : l’émulateur d’Android et l’émulateur de Nintendo Entertainment System (NES).
Par exemple, pour l’émulateur d’Android, on a BlueStacks, Nox, Koplayer. Dans les émulateur, nous pouvons faire les jeux vidéo qui sont implentés en système d’Android.
Pour l’émulateur de NES, on a FCEUX, JNES, NEStopia, NEScafé. Dans les émulateur, nous pouvons faire les jeux vidéo qui sont implentés en système de Nintendo Entertainment System.

Parmi les deux groupes d’émulateur, l’émulateur d’Android peut lancer des jeux vidéo complexe, mais il a besoin de plus de ressource de CPU et RAM, et en générale, ils ne sont pas ‘opensource’. Par contre, l’émulateur de NES ne peut que lancer les jeux vidéo anciens, mais il n’a pas besoin de beaucoup de ressource de CPU et RAM, et il y a beaucoup de tels émulateur de Opensource qui me permet à ajouter ou modifier des fonctions plus facilement.

Je chois l’émulateur de NES – ‘NEScafé’, parce qu’il n’a pas besoin de beaucoup de ressource de CPU et RAM et je peux ajouter ou modifier des fonctions plus facilement.

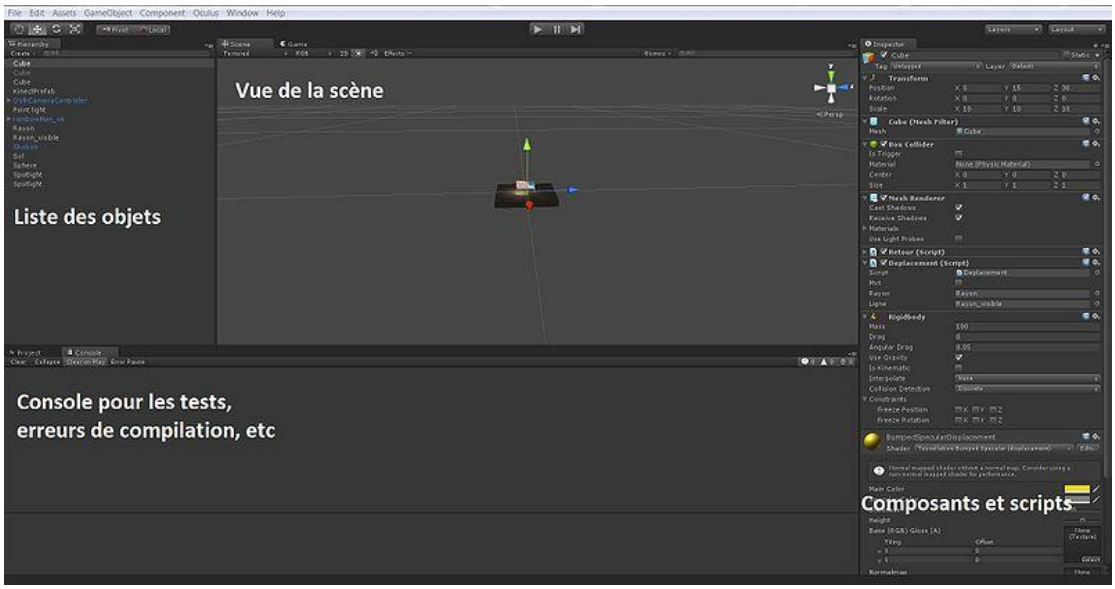

l’émulateur d’Android et l’émulateur

l’émulateur de NES

# Affichage d’image de jeux vidéo à casque de FOVE

**FOVE**

FOVE est un casque de réalité virtuelle qui peux simuler l’environnement de 3D. Nous pouvons utiliser l’environnement de UNITY à envoyer l’image ou le vidéo de l’ordinateur à FOVE.




**UNITY**

Pour afficher l'image de jeux vidéo en FOVE. Je crée un serveur de TCP/IP et ajouter deux client de TCP/IP en UNITY et l'émulateur de jeux vidéo. Chaque 20ms, l'émulateur envoie par TCP/IP sa frame à serveur, et le serveur renouvelle l'image. Chaque 20ms, Unity demande par TCP/IP au serveur l'image d'émulateur, après le serveur envoie l'image stocké à Unity et Unity renouvelle l'image de FOVE.

Dans Unity et l'émulateur choit, les programmations sont écrites de C#, et il y a des classes et des interfaces de TCP/IP qui nous permet à créer un serveur ou client et les faire communication facilement.

J'ai ajouté des boutons qui permets le joueur à modifier la dimension d'image dans la réalité virtuelle et un bouton qui permets à tourner l'image dans la réalité pour que le joueur doit tourner son corps à le suivre.

# Contrôler le jeux vidéo par clavier

La programmation principale est UNITY, donc nous ne pouvons pas contrôler l'émulateur directement. Pour faire l'interaction avec jeux vidéo, j'utilise un outil de Windows : 'handle'.

Handle est un pointeur de Windows avec qui nous pouvons envoyer les messages de contrôle entre programmation différentes. Par exemple, pour le projet, Quand le joueur appuie sur le clavier, l'UNITY va utiliser la handle à envoyer des message qui inclut la même commande de clavier à l'émulateur. C'est comme nous contrôlons le jeux vidéo directement.

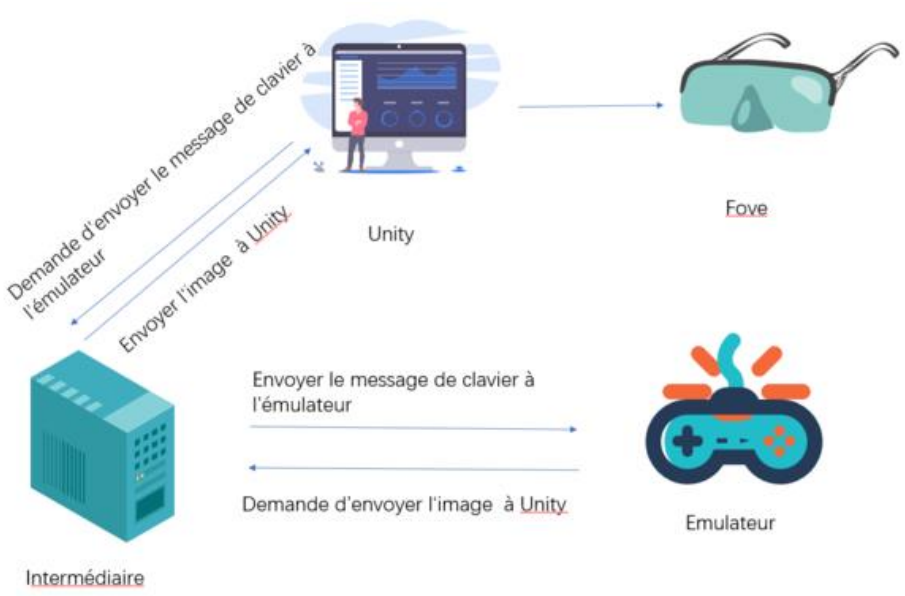
Pour obtenir les handle, nous devons importer les programmathèques de Windows :

```
      [DllImport("user32.dll", EntryPoint = "FindWindow")]
      public static extern IntPtr FindWindow(string IpClassName, string
IpWindowName);
      [DllImport("user32.dll")]
      public static extern IntPtr SetActiveWindow(IntPtr hWnd);
      [DllImport("user32.dll")]
      public static extern bool SetForegroundWindow(IntPtr hWnd);
```

Pour envoyer les message entre des programmation différentes, nous devons importer une programmathèque :

```
      [DllImport("User32.dll", EntryPoint = "SendMessageA")]
      private static extern int SendMessage(IntPtr hWnd, int Msg, int
wParam, int lParam);
```

# Architecture retenue



# Poursuite du Projet

Maintenant, j'ai réalisé à afficher l'image de jeux vidéo à FOVE, et permettre le joueur à modifier la dimension d'image et contrôler le jeux vidéo, mais, il y a un problème que quand les programmations lance, le renouvellement d'image est très lent. Maintenant, je pense que le problème est que la FOVE utilise beaucoup de ressource de CPU(85%), donc l'émulateur et l'intermédiaire n'ont pas assez de ressource de CPU, donc le renouvellement est lent. Après , je vais mettre l'émulateur et l'intermédiaire à l'autre ordinateur et faire un teste pour vérifier si il va être normal.

L'autre chose que je vais faire est de utiliser le Kinect à ajouter une manière d'interaction d'un système simulant le "touch" à partir des mouvements des mains de l'utilisateur.