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.

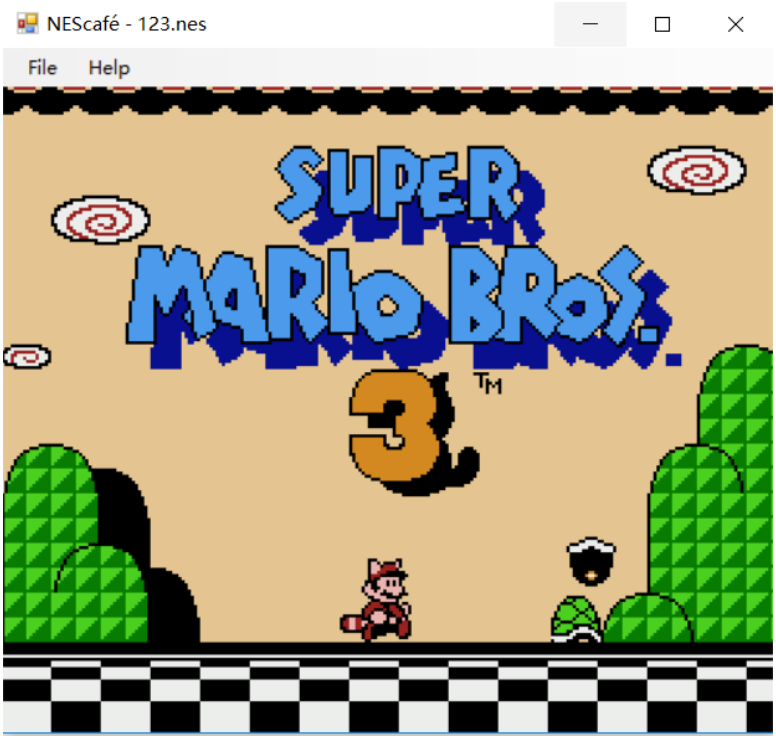
Je chois l’émulateur de NES – ‘NEScafé’[1], parce qu’il n’a pas besoin de beaucoup de ressource de CPU et RAM et je peux ajouter ou modifier des fonctions plus facilement.

En fait, au début, j’ai choit l’autre émulateur de NES : FCEUX[2], parce que FCEUX sont écrit par un grand groupe. Il a plus d’efficace et plus de fonctions, parce qu’il est écrit de la lange C++ et C#. Mais, j’ai eu un problème quand j’ai envoyé des messages de contrôle avec ‘Handle’ à l’émulateur. Le FCEUX a répondu rien quand il a récupéré tel message. Le détail de ‘Handle’ sont écrits à la suite. Mais NEScafé pouvais répondre bien tel message. Donc j’ai le choit à la fin.

l’émulateur d’Android et l’émulateur

[1] https://github.com/GunshipPenguin/nescafe
[2]

l'émulateur de FCEUX

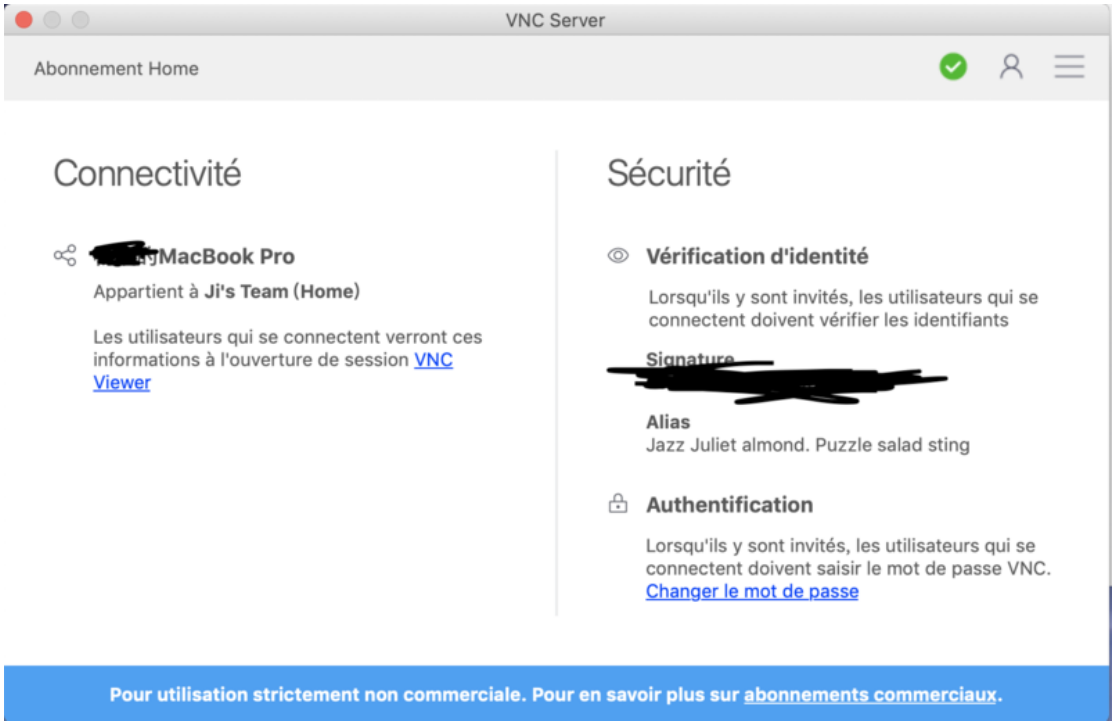


l'émulateur de NEScafé

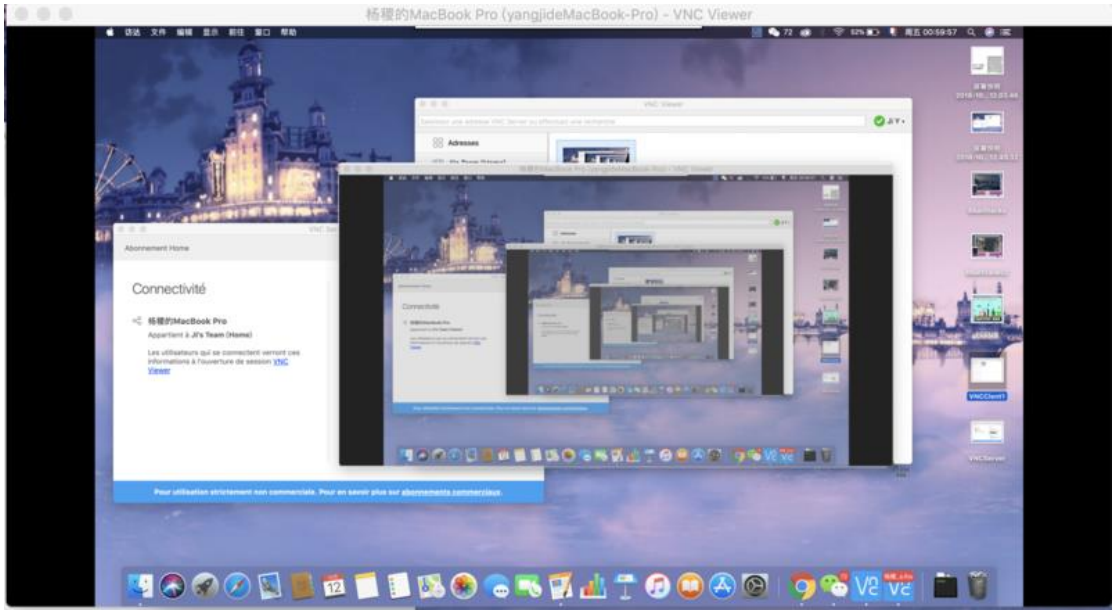
Cherche d'une configuration de VNC

Au début, je voulais trouver une configuration de VNC pour réaliser l'interaction entre le casque de réalité virtuelle. On peut contrôler un ordinateur à distance avec VNC, on peut aussi faire l'interaction avec deux ou plusieurs ordinateurs par VNC. Soit un casque et un émulateur sont deux ordinateurs, l'objet de mon projet est ce que le VNC fait.

Après, j'ai installé le VNCServer et le VNCViewer dans deux machines virtuelles et vérifié leurs fonctions. Ils sont marchés bien. Je pouvais contrôler une machine virtuelle d'autre machine virtuelle par VNC.



VNV Server



VNC Client

Après, j'ai cherché à implémenter VNCViewer à Unity. Parce qu'on doit utiliser l'environnement de Unity à envoyer l'image de jeux vidéo au casque virtuel. Mais, je n'ai pas le trouvé. Il n'existe pas telle technique.

Après, mon idée était que j'allais utiliser des programmathèques existées à créer un VNCClient propre qui pouvait être implémenté à Unity. L'Unity utilise la lange C#, donc, j'ai commencé à chercher les programmathèques de C# sur VNC.

J'ai trouvé une programmathèque qui s'appelle VNC# [3]. J'ai essayé un exemple de la programmathèque, mais, ça n'a pas marché.

Après, j'ai trouvé un projet de github qui s'appelle Unity-VNC-Client[4]. J'ai l'implémenté dans mon ordinateur, mais ça n'a pas marché aussi.

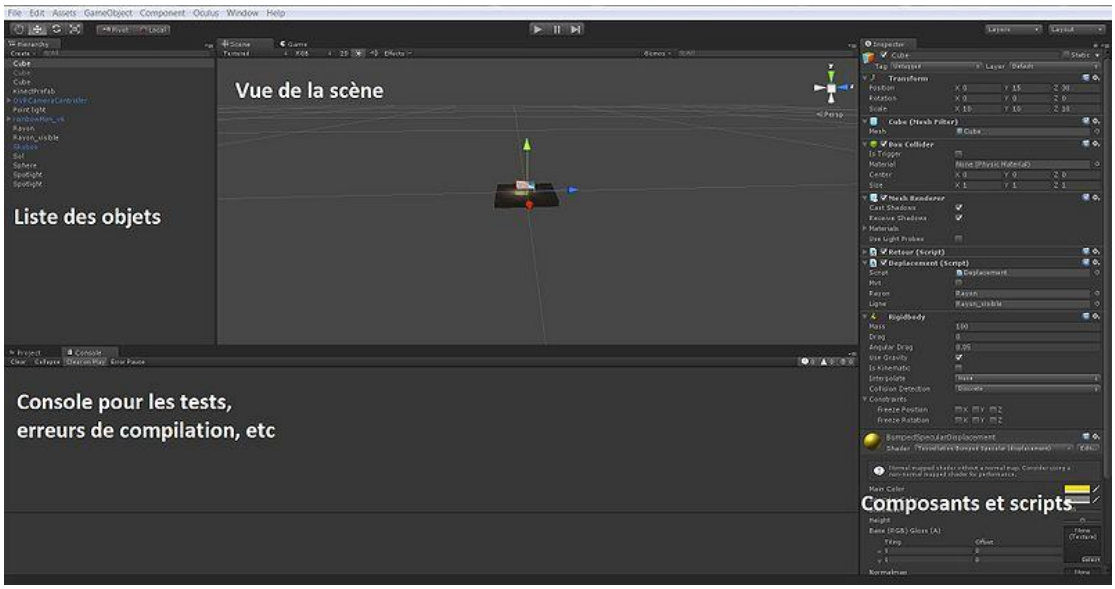
Après, j'ai réfléchi et j'ai pensé que ce que je veux faire était juste d'afficher l'image de jeux vidéo en casque de réalité virtuelle et contrôler le jeux vidéo par clavier et souris. Ce n'est pas obligatoire à utiliser le VNC. Je peux utiliser la technique TCP/IP à envoyer les donné entre L'Unity et l'émulateur de jeux vidéo.

[3] https://cdot.senecacollege.ca/projects/vncsharp/doc.html
[4] https://github.com/cfloutier/Unity-VNC-Client

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 devais faire deux choses : 1. Envoyer des donnés d'image d'émulateur à l'Unity ; 2. Unity laisse FOVE présenter l'image de jeux vidéo.

1. Envoyer des donnés d'image d'émulateur à l'Unity

Pour transférer les donné d'image, j'ai choisi à utiliser TCP/IP. J'ai créé un serveur de TCP/IP pour coordonner et ajouter deux clients de TCP/IP en UNITY et l'émulateur de jeux vidéo. L'émulateur envoie périodiquement par TCP/IP sa frame à serveur, et le serveur renouvelle l'image.

Unity demande périodiquement par TCP/IP au serveur l'image d'émulateur, après le serveur envoie l'image stocké à Unity et Unity laisse FOVE afficher l'image.

Dans Unity et l'émulateur, 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.

```
private void StartListening()
{
    server.Start();
    Byte[] bytes = new Byte[256];
    int Id;
    TcpClient client = new TcpClient();
    while (true)
    {
        try
        {
            client = server.AcceptTcpClient();
```

```
                Id = 0;
                NetworkStream stream = client.GetStream();
                BinaryFormatter binaryFormatter = new BinaryFormatter();
                ArrayList message = (ArrayList)binaryFormatter.Deserialize(stream);
                Id = (int)message[0];
                Console.WriteLine(Id);
                //if (!clients.ContainsKey(Id))
                //{
                    clients[Id] = client;
                    mainStreams[Id] = stream;
                    Console.WriteLine("client : " + Id + " connected");
                    getMessage[Id] = new Thread(() => ListeningClient(Id));
                    getMessage[Id].Start();

                    BinaryFormatter binaryFormattet = new BinaryFormatter();
                    ArrayList messageR = new ArrayList();
                    messageR.Add(true);
                    binaryFormattet.Serialize(stream, messageR);
            }
            catch
            {
                break;
            }
        }


    public void SendMessageTo(ArrayList message)
    {
        BinaryFormatter binaryFormattet = new BinaryFormatter();
        ArrayList messageR = new ArrayList();
        messageR.Add(message[0]);
        messageR.Add(message[2]);
        if (message[2] is bool) messageR.Add(message[3]);
        try
        {
            binaryFormattet.Serialize(mainStreams[(int)message[1]], messageR);
        }
        catch
        {
            CloseClient((int)message[1]);
        }
    }
```

Les codes dessus sont deux fonctions importantes qui établit un serveur de TCP/IP et permet à envoyer donné de message à une adresse IP (IP de Unity ou émulateur) ;

Dans Unity ey l'émulateur, il y a deux fonctions suivantes :

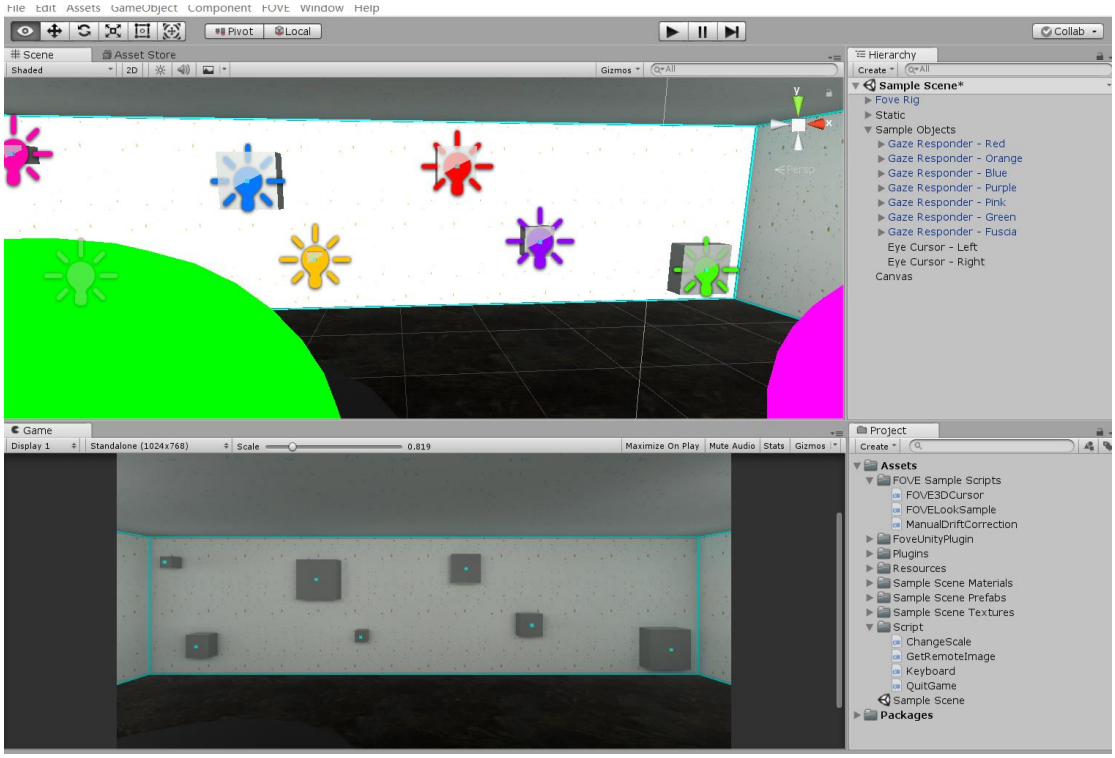
```
private void StartClient()
 {

     client = new TcpClient();
     try
     {
         client.Connect(_ipAddress, _myPort);
         //Byte[] data = System.Text.Encoding.ASCII.GetBytes(id);
         stream = client.GetStream();

         BinaryFormatter binaryFormattet = new BinaryFormatter();
         ArrayList message = new ArrayList();
         message.Add(_myId);
         binaryFormattet.Serialize(stream, message);
         ArrayList messageR = (ArrayList)binaryFormattet.Deserialize(stream);
         if ((bool)messageR[0])
         {
             getMessage = new Thread(ReceiveMessage);
             getMessage.Start();
         }
         else
         {
             //Console.WriteLine("Client avec cet id, déja pris");
             CloseClient();
         }
     }
     catch
     {
         //Console.WriteLine("pas de connection au serveur");
         client = null;
         MessageBox.Show("Fail connection to server");
     }

     //stream.Write(data, 0, data.Length);

 }
 public void SendMessage( string text)
 {
     if (client != null && client.Connected)
```

```
        {
            BinaryFormatter binaryFormattet = new BinaryFormatter();
            ArrayList message = new ArrayList();
            message.Add(_myId);
            message.Add(_idReceive);
            message.Add(text);
            binaryFormattet.Serialize(stream, message);
        }
        else
        {
            //Console.WriteLine("client non connécté");
        }
    }
```

Avec les fonctions, je peux envoyer des donnés d'image d'émulateur à l'Unity.

2. Unity laisse FOVE présenter l'image de jeux vidéo.

Pour laisse FOVE présenter l'image de jeux vidéo, j'ai utilisé un exemple standard de FOVE[5] qui nous permet à afficher les objets d'Unity en FOVE ;

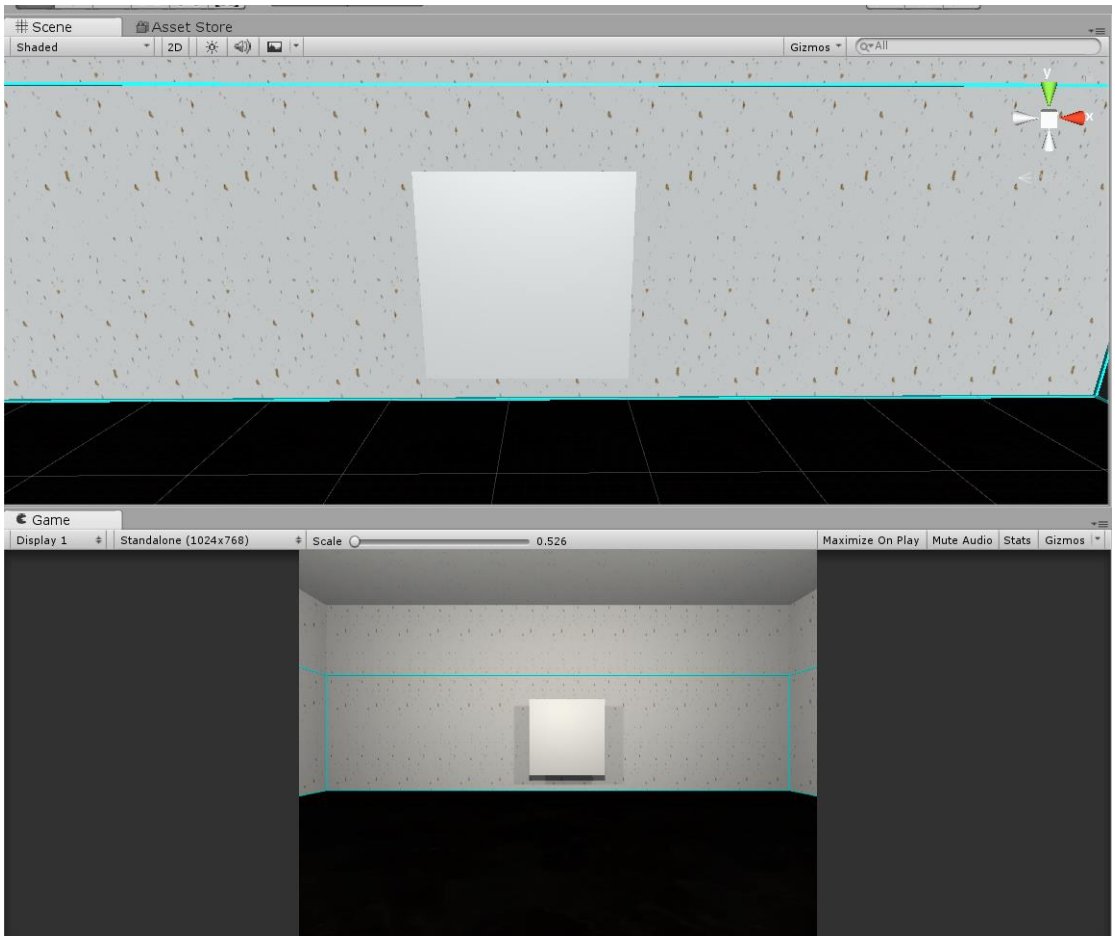


FoveUnitySample

L'image de la fenêtre basse est ce que FOVE affiche.

[5] https://github.com/FoveHMD/FoveUnitySample

Pour afficher une image à FOVE, j'ai choit l'objet de UNITY : Quad. Parce que juste les objet de 3D qui peuvent être affiché en FOVE, et Quad est un objet de 3D, et il peut contenir de renouveler les images.



Un Quad

Pour renouveler les images, j'ai écrit un script qui a une fonction de renouvellement :

```
    void Update () {
        Debug.Log("Update : " + isImageCanUse);
        if (isImageCanUse[0])
        {
            Debug.Log("writing image");
            Texture2D texture = new Texture2D(local_image[0].Width,
local_image[0].Height);
            System.Drawing.ImageConverter imageConverter = new
System.Drawing.ImageConverter();
            byte[] imageData = (byte[])imageConverter.ConvertTo(local_image[0],
typeof(byte[]));
            texture.LoadImage(imageData);
            //spriteRenderer.sprite = Sprite.Create(texture, new Rect(0, 0,
texture.width, texture.height), new Vector2(0.5f, 0.5f));
            //image.sprite = Sprite.Create(texture, new Rect(0, 0, texture.width,
```

```
texture.height), new Vector2(0.5f, 0.5f));
            //print(1);
            meshRenderer.material.mainTexture = texture;
        }
    }
```

La fonction permet à remplacer l'image de Quad par l'image envoyé du serveur de TCP/IP.

Après finir les deux parties, j'ai arrivé à afficher l'image de l'émulateur en FOVE.

Contrôler le jeux vidéo par clavier

La programmation principale est dans 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 messages 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 messages 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);
```

J'ai encapsulé la fonction :

```
        public void SendMessageTo(ArrayList message)
        {
            BinaryFormatter binaryFormattet = new BinaryFormatter();
            ArrayList messageR = new ArrayList();
            messageR.Add(message[0]);
            messageR.Add(message[2]);
            if (message[2] is bool) messageR.Add(message[3]);
            try
```

```
            {
                binaryFormattet.Serialize(mainStreams[(int)message[1]], messageR);
            }
            catch
            {
                CloseClient((int)message[1]);
            }
        }
```

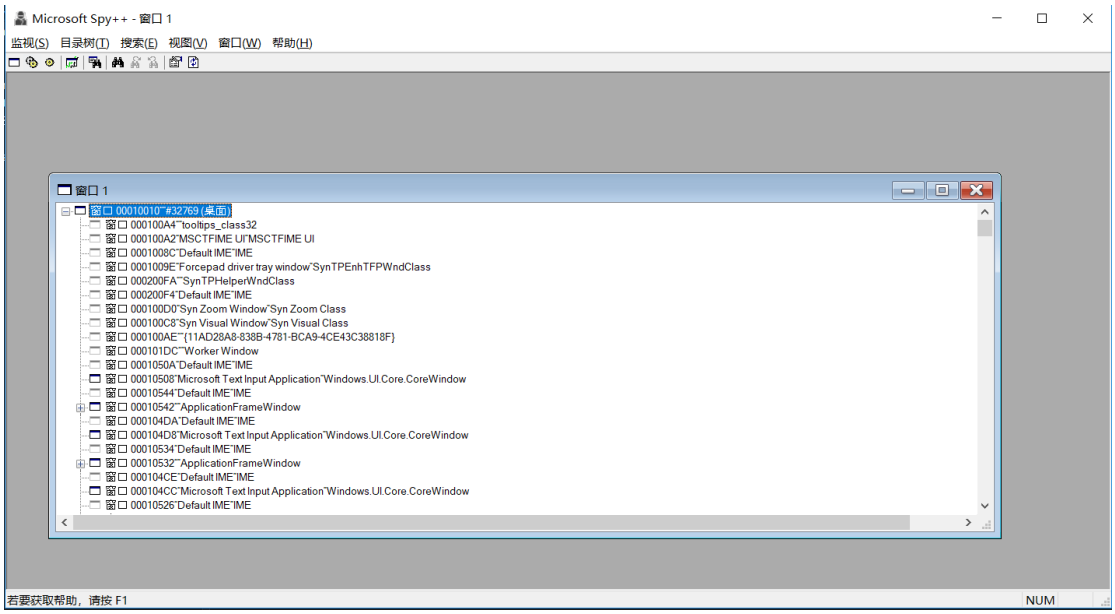
La paramètre de la fonction inclut les consigne du clavier.

Les codages de la clé de clavier est suivant :

```
    public enum VKKey
        {
            // mouse movements
            move = 0x0001,
            leftdown = 0x0002,
            leftup = 0x0004,
            rightdown = 0x0008,
            rightup = 0x0010,
            middledown = 0x0020,
            //keyboard stuff
            VK_LBUTTON = 1,
            VK_RBUTTON = 2,
            VK_CANCEL = 3,
            VK_MBUTTON = 4,
            VK_BACK = 8,
            VK_TAB = 9,
            VK_CLEAR = 12,
            VK_RETURN = 13,
            VK_SHIFT = 16,
            VK_CONTROL = 17,
            VK_MENU = 18,
            VK_PAUSE = 19,
            VK_CAPITAL = 20,
            VK_ESCAPE = 27,
            VK_SPACE = 32,
            VK_PRIOR = 33,
            VK_NEXT = 34,
            VK_END = 35,
            VK_HOME = 36,
            VK_LEFT = 37,
             ....
}
```

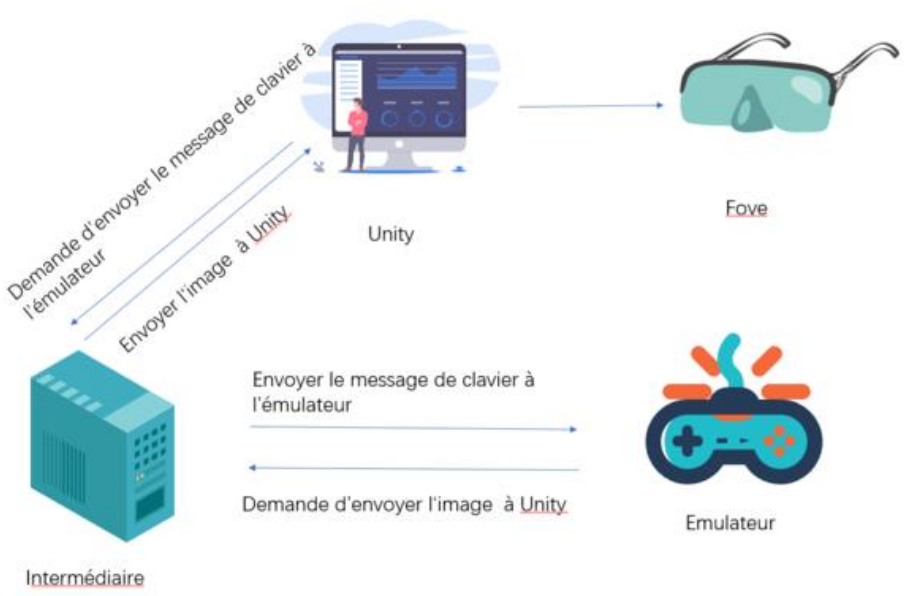
Pendant la configuration, j’ai utilisé un outil qui s’appelle spy++ pour observer les messages envoyé entre les programmations différentes, pour vérifier que la communication est correct.

Le spy++ nous permet à regarder tous les messages entre les programmations différentes.



SPY++

Architecture retenue



Poursuite du Projet

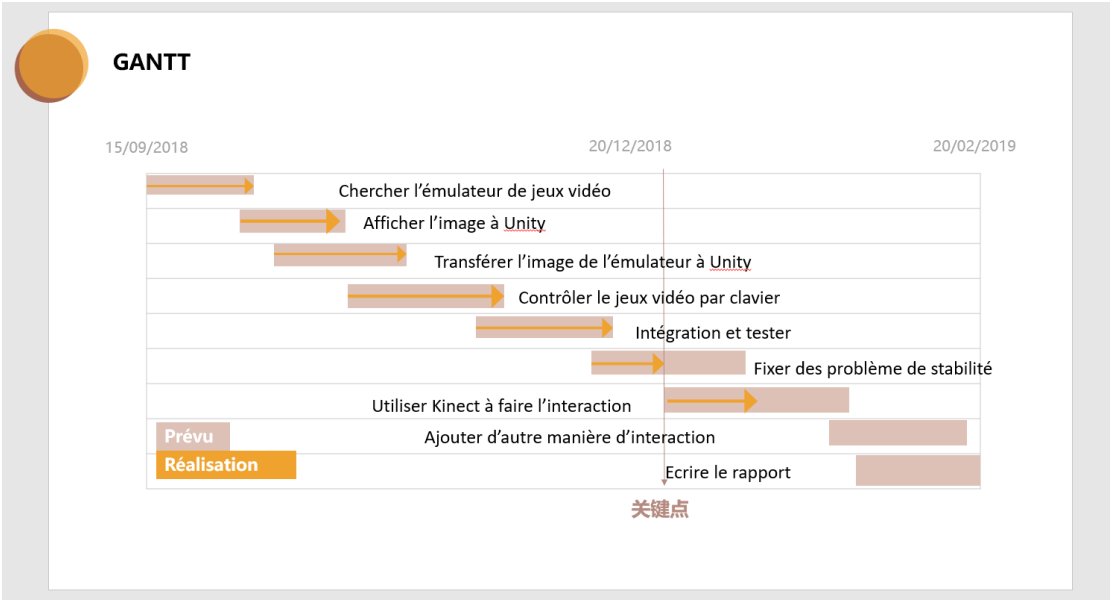
Maintenant, j'ai réalisé à afficher l'image de jeux vidéo à FOVE, et permettre le joueur à modifier la dimension d'image et contrôler le jeux vidéo, mais, il y a un problème que quand les programmations lance, le renouvellement d'image est très lent. Maintenant, je pense que le problème est que la FOVE 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 s'il va être normal.

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

En conclusion, j'ai des tâches suivantes :

20/12/2018 – 10/01/2018	Fixer le problème que le nouvellement de frame est trop lent
20/12/2018 – 13/02/2018	Ajouter la fonction que le joueur peut simuler le 'touch' à partir des mouvements ses mains.
05/02/2018 – fin du projet	Discuter avec les collaborateurs allemands pour ajouter d'autres manière d'interaction pour aider les enfants cérébrolésés à se recouvrer.
10/02/2018 – fin du projet	Ecrire le rapport

GANTT :


GANTT
15/09/2018
20/12/2018
20/02/2019
Chercher l'émulateur de jeux vidéo
Afficher l'image à Unity
Transférer l'image de l'émulateur à Unity
Contrôler le jeux vidéo par clavier
Intégration et tester
Fixer des problème de stabilité
Utiliser Kinect à faire l'interaction
Ajouter d'autre manière d'interaction
Ecrire le rapport
Prévu
Réalisation
关键点