Rapport de projet de quatrième année

Navigation 3D en environnement visuellement immersif

Mathieu BOSSENNEC

Encadrant : M.Laurent GRISONI

Table des matières

1 Le projet **4**

1.1 Présentation 4

1.2 Cahier des charges 5

1.3 Les outils utilisés 5

1.3.1 L'oculus Rift 5

1.3.2 Kinect 6

1.3.3 Unity 7

2 Travail effectué **9**

2.1 Découverte d'Unity 9

2.2 Intégration de l'Oculus Rift 9

2.3 Intégration de Kinect 10

2.4 Le script de positionnement 11

2.5 Le script de sélection des objets 11

2.6 Le script de déplacement des objets 12

2.7 Conclusion 14

Remerciements

Je tiens à remercier mon encadrant M.Laurent GRISONI ainsi que M.Damien MARCHAL pour leur soutient et leurs conseils durant toute la durée de ce projet, ainsi que pour m'avoir fourni tous les outils nécessaires a la réalisation de ce projet.

Chapitre 1

Le projet

1.1 Présentation

Pour ce projet de 4e année à Polytech Lille, j'ai décidé de faire la Navigation 3D en environnement visuellement immersif. Ce projet consiste à combiner plusieurs équipements récents tels que l'Oculus Rift et Kinect dans le but de proposer une expérience de réalité virtuelle la plus immersive possible. Cela passera donc par les aspects de la vision et de l'interaction avec l'environnement virtuel. A terme, ces technologie pourront êtres utilisées dans plusieurs domaines comme la visite de projets architecturaux, de musées, ou de lieux inaccessibles autrement, ou dans le milieu médical pour guérir des phobies sans risque si le cerveau interprète cette réalité virtuelle comme suffisamment réaliste par exemple.

1.2 Cahier des charges

Le but final du projet est de permettre l'interaction avec un univers virtuel par la vue grâce à l'Oculus Rift et par les mouvements grâce à Kinect. L'idée est donc de rendre l'expérience la plus immersive possible. Voici donc le cahier des charges :

- Permettre la visualisation de la scène à l'aide de l'Oculus Rift

- Ajouter la gestion du Kinect pour avoir la possibilité de voir son propre corps dans l'espace virtuel, rajoutant à l'immersion

- Permettre l'interaction, avec l'univers, comme pouvoir sélectionner ou déplacer des objets simplement en les pointant du doigt.

1.3 Les outils utilisés

1.3.1 L'oculus Rift

L'Oculus Rift est un casque de réalité virtuelle, permettant une vision en 3D stéréoscopique et couvrant tout le champ de vision contrairement à un écran classique, ce qui permet une plus grande immersion d'où son utilisation dans ce projet, l'immersion en étant la problématique principale. Il n'existe pour le moment qu'en version pour développeurs, une version plus avancée sortira au mois de Juin 2014, avec un écran Full HD et un système de Head Tracking. La version grand public sortira quand à elle début 2015.

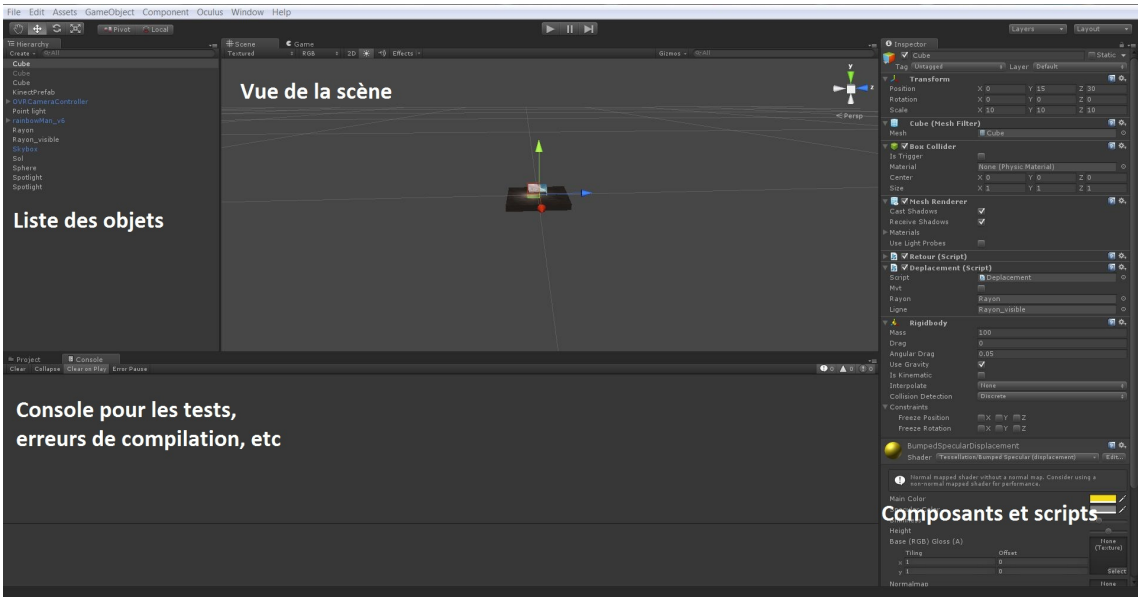
1.3.2 Kinect

Kinect est un périphérique initialement créé pour la Xbox 360 mais qui a fini par arriver sur pc. C'est une caméra permettant de suivre les mouvements d'une ou plusieurs personnes. Kinect est idéal pour ce projet car grâce à son capteur de profondeur, elle peut localiser les différentes parties du corps en 3 dimensions.

1.3.3 Unity

Unity est un logiciel 3D temps réel. Il est plus grandement utilisé pour créer des jeux mais ses fonctionnalités permettent de l'utiliser dans tout ce qui concerne la 3D. Ce logiciel a été choisis car l'Oculus Rift et Kinect étant avant tout des appareils destinés aux jeux vidéos, le logiciel profite de son statut de logiciel de création de jeux vidéos pour avoir un bon support de ces appareils et des librairies simples à utiliser.

Le logiciel se prend en main assez facilement, on peut créer une scène avec divers objets sans avoir besoin de compétences en programmation. Ceci dit, il n'est par pour autant limité, il dispose d'une gestion de scripts : les scripts, qui peuvent êtres écrits en JavaScript, C# ou en Boo, s'attachent aux objets pour modifier leurs comportements, comme par exemple les faire se déplacer suivant un schéma précis ou intéragir entre eux.



On peut voir ici que dans les composants, on peut rentrer des valeurs : Ce sont les variables publiques des scripts, et elles peuvent êtres utilisées pour paramétrer les scripts. Par exemple pour un script plaçant un objet par rapport à un autre, il faut avoir accès aux positions des deux objets : pour cela il est possible de désigner un objet en particulier par l'intermédiaire de ces variables, un objet étant de type gameObject.

Chapitre 2

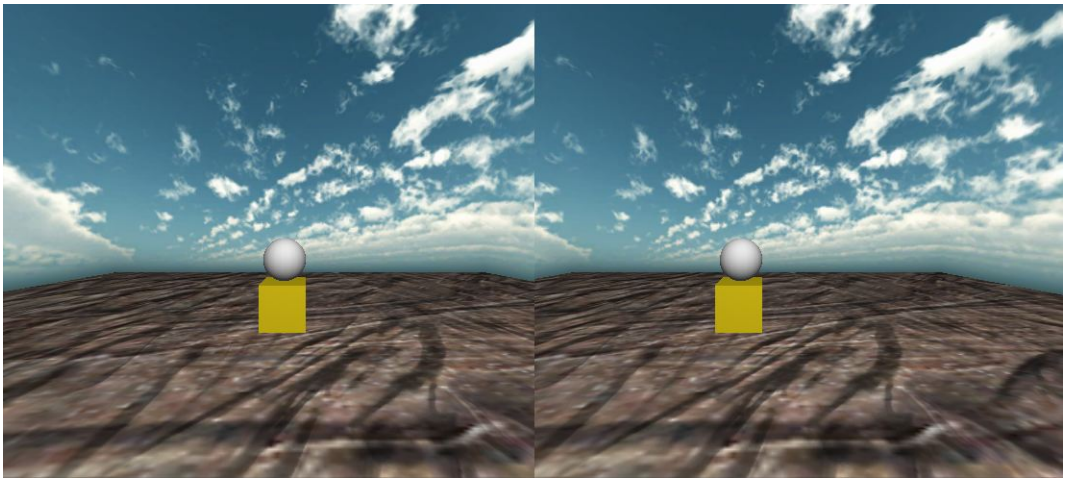
Travail effectué

2.1 Découverte d'Unity

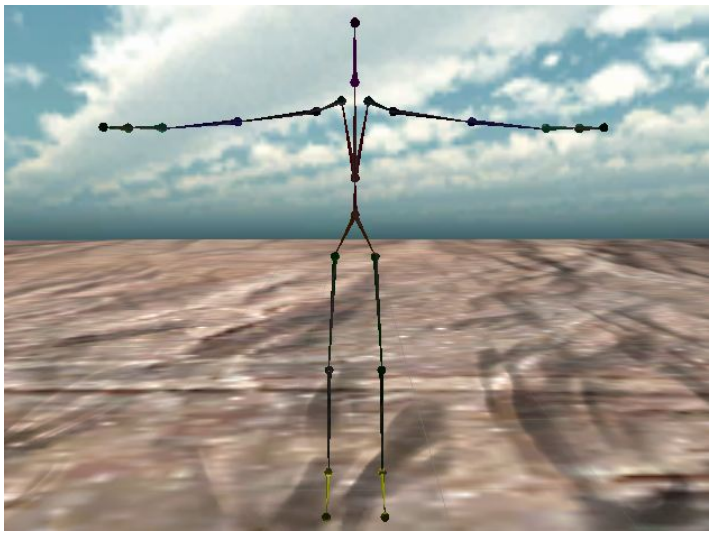
La première étape a été de se familiariser avec Unity. Pour cela des tutoriels étaient disponibles sur internet. J'ai donc vu comment créer des scènes, y placer des objets, les divers composants des objets, et , bien sur, les scripts. J'ai écrit tous les scripts en JavaScript, qui était le premier pour lequel j'ai trouvé des tutoriels.

2.2 Intégration de l'Oculus Rift

L'intégration de l'Oculus Rift dans Unity est très simple, il n'y a que deux différences avec une caméra classique d'Unity : - Les capteurs de L'Oculus Rift remplacent la souris de la caméra classique : au lancement du programme, pour ce qui est de l'axe des ordonnées le casque se calibrera tout seul. Cependant la direction du casque sera la direction initiale de la caméra pour ce qui est de l'axe des abscisses, ce qui fait que l'on peut se retrouver dans la mauvaise direction si l'on met le casque après le lancement du programme. Ce souci va être réglé dans la prochaine version du casque, qui pourra se calibrer aussi sur l'axe horizontal grâce à des diodes IR sur le casque et à une caméra. - L'image résultant de la caméra est différente, pour s'adapter au format de l'écran de l'Oculus Rift. En voici un exemple :



2.3 Intégration de Kinect



Le fonctionnment de Kinect est le suivant : - Au lancement, la caméra va commencer à rechercher des personnes - Une fois quelqu'un détecté, le logiciel associe les articulations de la personne aux points visibles sur le squelette ci dessus. - Le squelette va donc à présent suivre les mouvements de la personne.

Pour cela, un objet est nécéssaire dans la scène, le KinectPrefab, celui-ci est fournit par le SDK du Kinect et sert a paramétrer le Kinect avec des paramètres comme la hauteur du Kinect, le nombre de frames, etc

2.4 Le script de positionnement

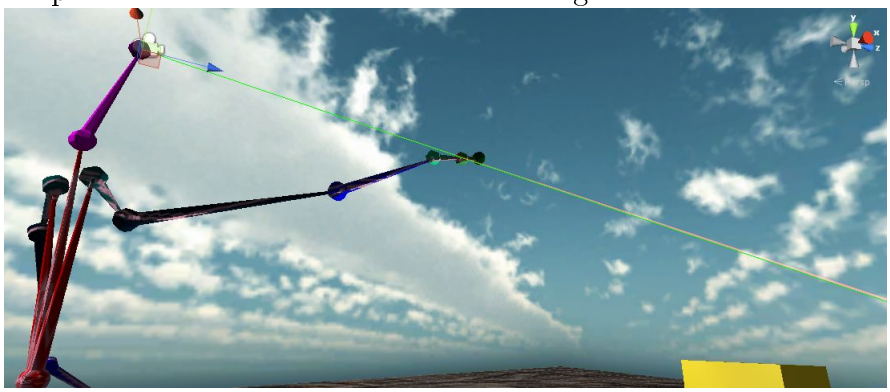
```
#pragma strict

public var tete : GameObject;

function Update () {
transform.position = tete.transform.position;
}
```

C'est un script très simple et aussi le premier que j'ai écrit. Il sert à placer un objet à la même position qu'un autre objet, à tout instant. Il est ici principalement utilisé pour positionner la caméra de l'Oculus Rift au point représentant la tête qui lui dépend du Kinect. Ainsi la caméra suis si l'on se baisse ou penche dans une direction sans problèmes.

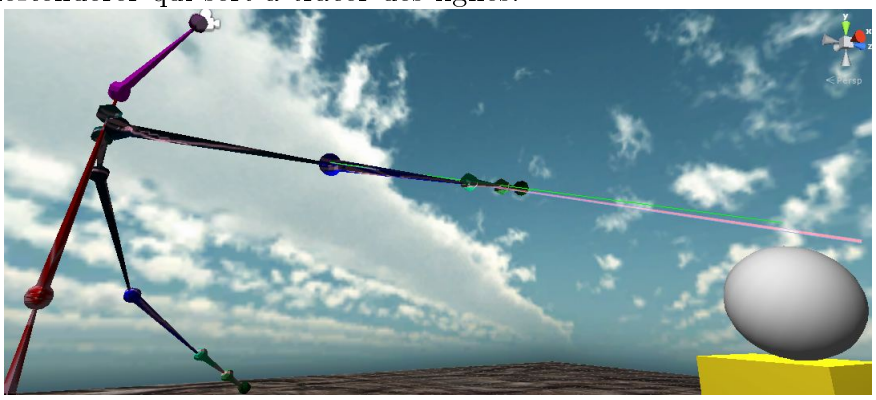
2.5 Le script de sélection des objets

```
#pragma strict

public var main : GameObject;
public var coude : GameObject;
public var tete : GameObject;
public var r_visible : GameObject;
var hit : RaycastHit;
public var test : boolean;
public var avt_bras : boolean;

function Update () {
       Debug.DrawRay(transform.position,
       (main.transform.position - transform.position).normalized * 50, Color.green);
       test = Physics.Raycast(transform.position,
       (main.transform.position - transform.position).normalized, hit, 300);
       if(test == true){
        if(hit.collider.gameObject.name == "Cube"){
            Debug.Log("Sol");
          hit.collider.gameObject.renderer.material.color = Color.red;}
       }
       if(avt_bras == true){
            gameObject.GetComponent(Positionnement).tete = coude;
            r_visible.GetComponent(Ligne).tete = coude;
            }
        else{
            gameObject.GetComponent(Positionnement).tete = tete;
            r_visible.GetComponent(Ligne).tete = tete;
            }
}
```

Ce script utilise la fonction Raycast de Unity, qui prend en paramètre un vecteur et renvoie le premier objet touché suivant ce vecteur. Ici le premier objet touché voit sa couleur changée au rouge, ce qui permet de voir quel objet est sélectionné. Une variable avt_bras permet de choisir si l'on veut que le rayon parte de la tête pour passer par la main ou parte du coude pour passer par la main. Voici la différence en images :



Le rayon pars ici de la tête pour passer par la main. A noter que l'affichage du rayon se fait par un autre script modifiant un composant d'Unity nommé LineRenderer qui sert à tracer des lignes.



Dans ce cas le rayon pars du coude pour passer par la main.

La première solution pour le rayon fonctionnait mais ne paraissait pas naturelle car ne correspondant pas à la façon habituelle de pointer un objet du doigt. La deuxième solution a été bien plus convaincante d'un point de vue immersion malgré une perte de précision due au capteur du Kinect, car dans la première méthode, la tête ne bougeant presque pas, la precision venait de la main seule, alors que l'on suis dans la deuxième méthode deux points

bougeant beaucoup, d'où la perte de précision.

2.6 Le script de déplacement des objets

```
#pragma strict

public var mvt : boolean;
private var dist : float;
private var delta : float;
public var tete : GameObject;
public var main : GameObject;
public var rayon : GameObject;
public var ligne : GameObject;
public var coude : GameObject;

function Update () {
    if(rayon.GetComponent(Rayon).test == true){
            if(rayon.GetComponent(Rayon).hit.collider.gameObject == gameObject){
                if(Input.GetMouseButton(0)){
                    mvt = true;
                    Debug.Log("Hit");
                    dist = rayon.GetComponent(Rayon).hit.distance;
                }
            }
    }
    if(Input.GetMouseButton(1) && mvt == true){
                    mvt = false;
                    gameObject.rigidbody.velocity = Vector3(0,0,0);
                    }
    if(Input.GetMouseButton(2) && mvt == true){
                    mvt = false;
                    if(rayon.GetComponent(Rayon).avt_bras == true)
                    gameObject.rigidbody.velocity = ((main.transform.position - coude.transform.position).normalized*50);
                    else
                    gameObject.rigidbody.velocity = ((main.transform.position - tete.transform.position).normalized*50);
                    }
    delta = Input.GetAxis("Mouse ScrollWheel");
    dist = dist + 5*delta;
    if(mvt == true){
        gameObject.rigidbody.useGravity = false;
        if(rayon.GetComponent(Rayon).avt_bras == true)
        transform.position = (coude.transform.position + (main.transform.position - coude.transform.position).normalized*(dist+4));
        else
        transform.position = (tete.transform.position + (main.transform.position - tete.transform.position).normalized*(dist+4));
        }
    else
        gameObject.rigidbody.useGravity = true;
}
```

Ce script permet à l'aide d'une souris sans fil d'intéragir avec les objets présents dans la scène :

- Un clic gauche permet "d'attraper" un objet, celui-ci va alors suivre le rayon en restant à une distance constante.

- Un clic droit permet de lâcher l'objet, qui tombera alors au sol.

- La roulette permet de rapprocher ou d'éloigner l'objet.

- Enfin un clic sur la roulette permet de lancer l'objet.

C'est dans ce script que le fait de faire passer le rayon par le coude prend toute son ampleur : contrairement à l'autre méthode, on a vraiment l'impression d'avoir l'objet en main.

Le fonctionnement du script est le suivant : lorsqu'un objet est "attrapé", sa position va être actualisée à chaque frame en accord avec le rayon ; en effet, le script récupère le vecteur directeur du rayon ainsi que son origine pour placer l'objet suivant une distance constante de l'origine du rayon, initialisée à la valeur de la distance lors du clic gauche. Les valeurs du rayon sont d'ailleurs récupérées du script précédent pour savoir si l'on est dans le cas du rayon tête-main ou du rayon coude-main.

2.7 Conclusion

Le projet fonctionne correctement au final, cependant un point aurait pu être amélioré : le Kinect manque parfois cruellement de précision ainsi que de réactivité. Ainsi il n'a pas été rare de voir le bras de l'avatar faire des torsions surréalistes avec un rayon qui part dans une direction incorrecte. Une solution pour palier à cela aurait pu être d'utiliser le système Art Track au lieu de Kinect. L'Art Track est un système composé de plusieurs caméras infrarouges et d'un gantelet à diodes infrarouges. Ce système aurait été plus précis bien que plus difficile à mettre en place, car la présence de plusieurs caméras aurait permis une meilleure localisation dans l'espace.