Cahier des Charges : DoMoCap

I – Présentation générale du projet :

## Les acteurs du projet

Maitre d’ouvrage : Marc CHEVALDONNÉ, professeur

Maitre d’œuvre et chef de projet : Elliott LE GUEHENNEC, élève de 2nde année

Equipe : Mattéo BROQUET, élève de 2nde année

Yorick GEOFFRE, élève de 2nde année

Erwan THERON, élève de 2nde année

Jordan GADET, élève de 2nde année

## Nature de la prestation demandée

Aujourd’hui, construire ses propres équipements de réalité virtuelle est fastidieux, car ce processus implique de devoir coder un driver selon l’API OpenVR, dans le but de se connecter à la plateforme SteamVR. Le projet DoMoCap sera un outil facilitant cette procédure, en connectant automatiquement un équipement de réalité virtuelle à SteamVR. Pour que notre application soit la plus adaptée aux besoins de l’utilisateur, nous prévoyons que l’utilisateur puisse paramétrer lui-même ses contrôles grâce à une interface plus intuitive que l’API OpenVR. Toute la communication avec SteamVR sera effectuée par l’application.

## Estimation des grandes étapes et dates butoirs

Produit(s) attendu(s) :

Livrable(s) fin P2 DEMONSTRATION

Statique

-communication périphériques

-nettoyage des données

-conversion (angles -> quaternion)

-communication OpenVR

-simulation (Steam VR ou Unity)

Livrable(s) fin P3 FIN

Dynamique

-communication entrée externe (Kinect)

-mappage (application)

-outils de paramétrage mannette

Bonus :

-capteurs divers et variés

-simulation UNITY (démonstration plus poussée)

## Organisation de l’équipe et du travail

**II – Cahier des charges fonctionnelles :**

**Fonctionnalités attendues**

**(contexte)**

Lorsque l’utilisateur utilise une application en réalité virtuelle, ses équipements envoient des données dites « brutes ». Celles-ci sont « raffinées » à l’aide d’un filtre qui assure que les valeurs reçues sont justes, et définit les erreurs qui pourraient survenir. Puis ces données passent par un outil de mapping qui va définir et associer les différents boutons avec les contrôles dans le jeu. Puis elles sont transmises à un driver qui communique avec SteamVR. En parallèle à cela, nous avons un outil qui permet de paramétrer les différents boutons de notre équipement dans différentes applications. Afin de tester notre outil et de le montrer à de futurs clients, plusieurs démonstrations de diverses qualités seront créées au fur et à mesure du projet.

**Contraintes pesant sur la réalisation du projet**

Une des contraintes principales de ce projet est le temps passer sur la documentation pour comprendre comment SteamVR fonctionne et comment communiquer avec cette application.

«Deuxième version»

**II – Cahier des charges fonctionnels**

**Fonctionnalités attendues**

**(Qu’est ce que le commercial attend?)**

Pour commencer nous avons besoin d’un fichier qui va récupérer les données brutes du flux Arduino qui vas se trouver sous la forme d’une chaine de caractère « A 123 | … | X 235 » avec chaque lettre une touche du gant et en chiffre la valeur donnée par le gant. Nous allons ensuite, toujours grâce à ce fichier nous allons séparer cette chaine de caractère par le nombre de touche et de fonctionnalité que pocède le gant.

Ensuite, les données vont passer par un filtre qui vas raffiner les données pour les adaptés au langage voulus en fin de chaine. Ici nous allons raffiner les données pour les envoyer plus tard à Steam VR.

Le mapping va servir à configurer plus en profondeur Steam VR pour attribuer une action à chaque donnée que l’on possède. Il va donc recevoir des données raffinées pour les mettre en phase avec Steam VR avant de les envoyer à un driver.

Le driver sert à communiquer directement avec Steam VR, il est donc la juste en passerelle entre Steam VR et le mapping.

Nous voulons aussi intégrer une interface qui vas permettre à l’utilisateur de modifier les touches de leur gant a une action souhaitée dans leur jeu sous Steam VR. Pour faire cela, il y aura une modification dans le mapping qui vas changer l’endroit où il va envoyer ces données.

Enfin, pour finir, nous aimerions construire une démo sous Unity afin de montrer la capacité de notre application à s’adapter dans un jeu.

**Contraintes pesant sur la réalisation du projet**

Une des plus grandes contraintes de ce projet est la compréhension d’OpenVR et le fait de trouver des documents bien commentés pour nous aider à comprendre le fonctionnement de cette application. De plus, nous avons dû apprendre le C++ en autodidacte car nous n’avions pas encore eu de cours à l’IUT dessus.

Certaines ressources n’étant pas présent sur les ordinateurs de l’IUT, et n’ayant pas les droits de les installer nous-même, nous avons dû faire appel à un technicien qui a fait les mises à jour et les installations dont nous avions besoin pour notre espace de travail.

De plus, nous voulons mettre un point d’honneur sur la documentation, une source bien détaillée et précise sur le fonctionnement de notre application, pour aider d’autres informaticiens

à la comprendre sans qu’ils doivent, comme nous, chercher partout sur internet pour trouver une documentation propre et détaillée(Ceci est plus un resultat ou un attendus qu’une contrainte)

Une des autres contraintes principales de ce projet est de le rentre le plus performant possible pour ne pas causer des baisses de FPS et pour ne pas créer de latence dans leur session en jeu.

**Contraintes pesant sur l’utilisation du produit**

**(finances , OpenVR, formation,...)**

Pour commencer, nous voulons faire une application la plus simple possible à l’utilisation. Car cela permettra à des clients qui n’ont pas de connaissances poussées en informatique, d’avoir leur propre espace de réalité virtuelle chez eux. Pour ainsi dire nous visons plutôt des clients qui sont passionnés par les jeux-vidéos et qui s’intéressent à la VR.(NAZE HS)

De plus pour l’utilisation de notre produit, le client doit procéder une manette de VR fait par lui-même, ou qui ne passe pas par le système de SteamVR. Par exemple la Kinect n’a pas besoin d’une application supplémentaire car tout est intégré dans son code et celui de OpenVR, tandis que des gants fait maison ont besoin de notre application car ils ne possèdent pas les outils de communication nécessaires pour transmettre leurs données à SteamVR.

**Critères d’appréciation de la qualité du produit**

Une application qui est dynamique, permet le changement des actions que font les touches du gant, et qui, une fois en jeu, à une grande fluidité pour ne pas contraindre les mouvements des joueurs. Nous aimerions aussi faire une interface esthétique et facile à comprendre pour des (potentiels) changement de touche. Enfin, nous aimerions avoir une démonstration capable de montrer ce dont notre application est capable de réaliser. ( ?)

qui arriverait à démontrer toute la puissance de notre application. » »

(M’a l’air assez bancal (répétition de démontrer + puissance meh imo) mais je ne sais pas si j’ai fait bien mieux)

**III – Résultats Attendus**

Notre résultat le plus important serait une première démonstration du fonctionnement de notre logiciel, probablement sur SteamVR mais si possible sur Unity, afin de conclure notre projet comme mené à bien pour la partie visée. La démonstration sera concluante si les gants permettent bien d’envoyer des signaux dans le « jeu », si ces signaux permettent en effet d’interagir avec le jeu sans problèmes.

Nous aimerions aussi atteindre un résultat particulièrement compliqué à réaliser : celui de faire fonctionner notre logiciel sur n’importe quel autre matériel VR que des gants.

**Lexique :**

API : Application Programming Interface = Interface de Programmation d’Application

FPS : Frames Per Second = nombre d’Images Par Seconde

VR : Virtual Reality = Réalité Virtuelle

Quaternion : Nombre hypercomplexe (voir définition sur Wikipédia)

<https://fr.wikipedia.org/wiki/Quaternion>