Applications de streaming vidéo sur Hololens

Ing. G. VANVINCKENROYE
Dr F. SENNY
GRAMME - Liège

Ce papier constitue le résumé d'un travail fruit de 13 semaines de stage au sein de l'entreprise EVS Broadcast Equipment. Il traite du développement d'application pour les lunettes de réalité augmentée Hololens. Différents points sont abordés, comme la présentation de la lunette, la communication réseau ou encore les performances de la lunette.

Mots-clefs : Hololens, EVS Broadcast Equipment, C\#, Socket, 3D, Hologrammes, Serveur, Réseau, Parsing, POO

This paper is the summary of a thesis which is the result of a 13-weeks internship at EVS Broadcast Equipment. It discusses the development of applications for the holographic glasses Hololens. Different aspects are approached, among other a presentation of the Hololens device, network communication or the performances of the Hololens device.

Keywords : Hololens, EVS Broadcast Equipment, C\#, Socket, 3D, Holograms, Server, Network, Parsing, OOP

1. Introduction

« La technologie de l'information a changé la façon dont les gens créent de la valeur économique », comme l'a dit Alan Greenspan, docteur en sciences économiques et ancien président de la Réserve fédérale de la Banque Centrale des Etats-Unis. Cet article présente le développement d'une nouvelle façon de vivre des évènements mondiaux, retransmis en direct. Il est le fruit d'un travail de fin d'études découlant de 13 semaines de stage au sein de l'entreprise EVS Broadcast Equipment située à Seraing, en Belgique.

Le produit utilisé lors de la réalisation de ce travail est la lunette de réalité augmentée Microsoft Hololens. Cette lunette permet d'intégrer des éléments virtuels dans le champ de vision de la personne les portant, et de faire interagir ces éléments virtuels avec l'environnement autour de celui-ci.

Le but final du projet était de proposer une manière innovante au milieu du broadcast, bien que la version du produit soit invendable au grand public, étant donné qu'il s'agit d'une version développeur, et non un produit fini.

Ce travail s'est déroulé en plusieurs parties : la première consistait en une prise de connaissance avec le milieu du broadcast, un enseignement des bonnes pratiques de celui-ci, et de l'écosystème propre à l'Hololens.

Ensuite, un premier projet a été proposé, celui de mettre en place un mur d'écran virtuel remplaçant les écrans physiques actuellement utilisés. Les différents écrans seraient alimentés par différents flux vidéo provenant, par exemple, des caméras situées dans une salle de concert. Cependant, ce projet a été abandonné suite à des contraintes de deadline et technologiques.

Enfin, le projet ayant abouti consiste à visualiser en 3D le déplacement des différents acteurs d'un match de football, ayant été soumis à un dispositif de tracking lors d'un match s'étant déjà déroulé. A cette visualisation du match s'ajoutent différentes fonctionnalités comme la navigation dans le match, ou des outils d'analyse tactique.

Cet article présente un condensé du travail, et nous conseillons une lecture du travail complet pour une compréhension globale du sujet.

2. Notions préliminaires

Afin de comprendre le travail réalisé, il est nécessaire de connaître certaines informations tant sur le milieu du broadcast que sur la lunette de réalité augmentée Hololens.

2.1. Présentation du milieu du broadcast

Le milieu du broadcast est un milieu mobile. En effet, des évènements se déroulent aux 4 coins du monde, et les infrastructures permettant la retransmission de ces évènements est donc concentrée dans des camions, où les réalisateurs proposent aux téléspectateurs le meilleur angle et le meilleur ralenti. Cet aspect mobile du milieu entraine que toute l'infrastructure est concentré dans des espaces exigus.

Cette infrastructure, composée de serveurs, écrans ou encore câbles, est actuellement indispensable et laisse donc très peu d'espace aux opérateurs, et rend leurs conditions de travail difficile. La figure 1 illustre l'intérieur d'un camion de réalisation.

Figure 1: Intérieur d'un camion de réalisation [1]

Un des enjeux du progrès technologique est, pour le monde du broadcast, de réussir à augmenter le confort de travail des opérateurs, et donc de réduire l'encombrement lié au matériel nécessaire à la retransmission d'évènements télévisés.

2.2. Présentation de la lunette Hololens

La lunette de réalité augmentée Hololens est un produit développé par Microsoft. Elle permet la visualisation d'hologrammes en 3D, et de les intégrer dans l'environnement autour de l'utilisateur. Cette immersion des hologrammes se fait grâce à différentes fonctionnalités, qui sont ici brièvement présentées, mais de manière bien plus approfondie dans le travail complet.

Gaze :
Le *Gaze* (Regard en anglais), permet à l'utilisateur d'avoir un curseur suivant la direction de sa tête. Ainsi il pourra interagir avec certains éléments virtuels en les pointant avec ce curseur.

SpatialMapping :
Le *SpatialMapping* permet à la lunette de réaliser un scan de son environnement, et d'en réaliser un maillage triangulaire en 3D. Ce maillage est placé dans un repère propre à la lunette, et va donc permettre de situer les différents hologrammes d'une part entre eux, grâce au repère spécifique à la lunette, mais également par rapport aux éléments réels grâce au maillage permettant de situer l'origine du repère de la lunette dans le monde réel. Ce *SpatialMapping* peut être poussé un cran plus loin, grâce au *SpatialUnderstanding*. Ce module supplémentaire, permet de réaliser un maillage plus fin de la pièce, et donc de réaliser certaines détections automatique, comme une détection de plan par exemple le sol ou encore le plafond. Le scan, tant pour le *SpatialMapping* que pour le *SpatialUnderstanding* est réalisé sur des zones spécifiques. Ainsi, l'utilisateur pourra en pointant le *Gaze* sur certains endroits de la pièce, scanner la zone. La figure 2 met en évidence la différence entre le *SpatialMapping* et le *SpatialUnderstanding.* On constate que le maillage sur la partie gauche de l'image, issu du *SpatialUnderstanding*, est beaucoup plus précis et plus fin, notamment au sol, que le maillage blanc, issu du *SpatialMapping.*

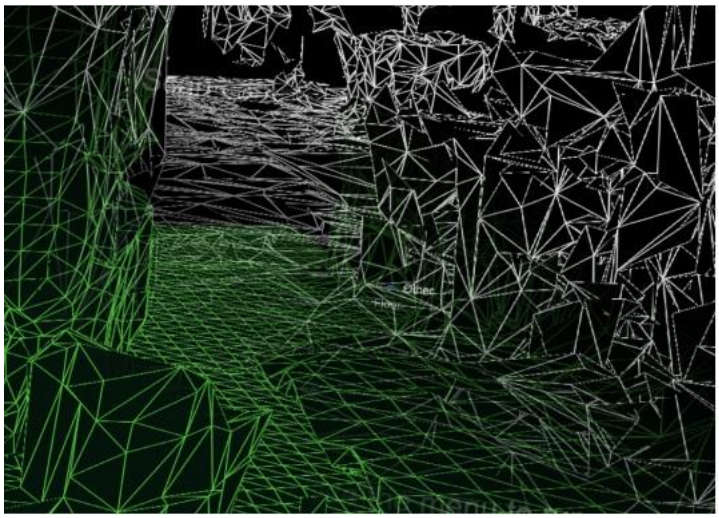

Figure 2: Différences entre le SpatialMapping et -Understanding

GestureUnderstanding :
Cette fonctionnalité permet à l'utilisateur de communiquer avec la lunette, comme en effectuant l'équivalent d'un click de souris, grâce à l'Air-Tap. Différents gestes,

prévu dans le système d'exploitation de la lunette[1], permettant la navigation dans la lunette peuvent être détectés par la lunette, et sont développés dans le travail complet concernant ce projet.

3. Développement d'un mur d'écrans virtuels

Comme mentionné précédemment, un des enjeux du milieu du broadcast est de réduire l'encombrement lié aux infrastructures nécessaires à la retransmission d'un événement. Ainsi, un des sujets abordés est de réaliser un mur d'écrans virtuels, permettant à chacun des utilisateurs de visualiser les écrans désirés dans ses propres lunettes. De cette façon, deux utilisateurs différents peuvent superposer leurs écrans respectifs, étant donné qu'ils ne sont visibles qu'à travers les lunettes.

3.1. Principe général de l'application

Le schéma de principe de l'application est celui illustré à la figure 3. Il est important de noter qu'en plus des étapes illustrées sur le schéma, il faut mettre en place une communication entre un serveur et la lunette, ainsi qu'entre les caméras et le serveur.

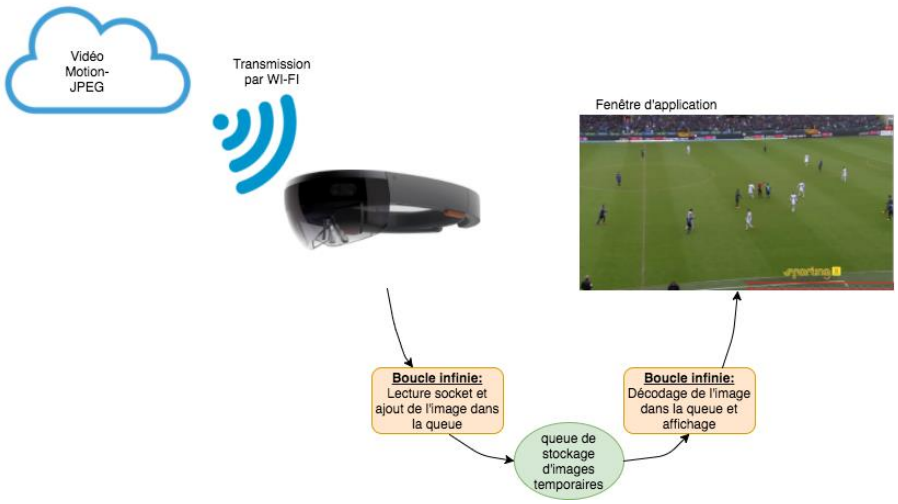

Figure 3: Schéma général de l'application

[1] Voir référence [1] : Microsoft Dev Center - Gestures

Le problème rencontré dans ce cas-ci est lié au fait que la lunette est extrêmement récente, et que les capacités de celle-ci sont donc toujours fort floues. Afin de définir ce qu'il était possible de réaliser avec la lunette, nous avons réalisé une application de type Universal Windows Platform (UWP). Comme son nom l'indique, une application UWP est une application qui peut tourner sur l'ensemble des appareils équipés d'un système d'exploitation Windows 10. Cela permet donc de faire tourner la même application sur la lunette, et sur un PC et donc de comparer les performances des deux applications.

Comme illustré à la figure 3, l'envoi et la lecture d'une image se déroulent en plusieurs étapes :

1. La première étape consiste à instaurer une liaison par socket entre le serveur et la lunette. Celle-ci est ouverte sur un port choisi directement dans le code. On transmet au serveur l'adresse IP de la lunette, ainsi que le port, et celui-ci « pousse » ensuite les images l'une après l'autre sur le socket. Afin de se débarasser de toute une série de header liés aux protocoles, cette communication se fait dans une couche de bas niveau, la couche TCP.
2. Le serveur commence ensuite à envoyer, toutes les 40 ms (pour du 25 fps), une image. Cette image est envoyée dans une séquence d'octets, où les 4 premiers contiennent la taille de l'image, tandis que tous ceux qui suivent contiennent l'image en elle-même.
3. La lunette, lorsqu'elle reçoit une image, lit les 4 premiers octets, afin de connaître la taille de l'image à afficher. Une fois sa taille connue, elle boucle sur la séquence d'octets, jusqu'à avoir séparé les octets contenant l'image du reste de la chaîne. Ainsi, on peut stocker dans un objet l'image, ainsi que sa taille séparément.
4. Une fois que cette image est stockée dans un objet, cet objet est poussé dans une queue. Une queue permet de retirer des objets d'une file d'attente dans le même ordre que celui dans lequel ils ont été ajoutés.
5. Parallèlement à la thread de lecture sur le socket, une seconde thread retire les objets au fur et à mesure de la queue, décode l'image contenue dans l'objet, et actualise une photo dans la fenêtre d'application.

Ces étapes se déroulent pour chacun des flux vidéo que l'on désire afficher dans la lunette.

3.2. Performances de l'application

Comme mentionné précédemment, les performances de la lunette sont à l'heure actuelle encore fort floues. C'est pour cette raison que nous avons décidé non seulement d'observer les performances de l'application sur Hololens, mais également de les comparer à la même application sur PC.

Analyse des performances sur l'Hololens

Différents problèmes ont été constatés lors de l'exécution de l'application sur la lunette, mais un seul sera discuté dans ce résumé : la présence d'artefacts sur la vidéo.

Ce phénomène d'artefacts consiste à observer à l'œil nu une pollution des images affichées dans l'application. La pollution observée est une pixellisation dans le coin inférieur droit de l'image, comme illustré à la figure 4.

Figure 4: Artefacts observés sur des images de 282,3 kB (à droite) et 290,5 kB (à gauche)

Le paradoxe de cette pollution des images est que la pollution sur l'image de droite est supérieure à celle de gauche, alors qu'elle est de taille plus petite. Il n'y a donc pas de lien direct entre la taille de l'image à décoder, et la qualité du décodage de celle-ci. Cette pollution est observée dès que la fréquence d'envoi d'une nouvelle image atteint 25 fps, tandis qu'il est nécessaire d'augmenter énormément cette fréquence (jusqu'à 60-70 fps) lors de l'exécution sur PC. On peut donc penser que la lunette ne parvient pas à décoder assez rapidement les images. Cette hypothèse est confirmée par un problème d'accumulation des images dans une queue, discuté en détail dans l'ouvrage complet.

3.3. Conclusion intermédiaire

Ce premier projet nous a permis de faire connaissance avec l'entièreté de l'écosystème de la lunette, tant au niveau du langage de programmation (C#) que des logiciels permettant le développement des applications (Unity, Visual Studio,etc.). Il a également permis, via les limites rencontrées par la lunette, de prendre plus conscience du potentiel, et du cahier des charges auquel la lunette est capable de répondre.

4. Visualisation d'un match de football en 3D

Les conclusions amenées par le premier projet nous ont poussés à développer une application plus en phase avec le cahier des charges de la lunette. Cette application

consiste à visualiser, en 3D, le déplacement des différents acteurs d'un match (balle, joueurs, arbitre). Afin de connaître les différentes données sur les joueurs au cours d'un match, ceux-ci ont été soumis à un tracking GPS pendant un match.

Le but de l'application finale étant de permettre à un utilisateur de communiquer avec l'application, à l'aide d'une commande, différentes structures ont été mise en place :

1. La première étape consiste à transformer les données GPS, stockées dans un fichier .txt (que nous appelerons fichier TRACAB, du nom de l'entreprise ayant récolté les différentes données au cours d'un match), en un ensemble de données exploitable pour l'application, c'est à dire des objets.
2. Ensuite, un serveur Http a été mis en place pour permettre la communication entre la commande, et la lunette.
3. Enfin, différentes fonctionnalités (des analyses tactiques par exemple) ont été mise en place, afin de rendre l'application plus dynamique et interactive.

Un schéma global de l'application est illustré à la figure 5. On peut observer sur cette figure la commande reliée via un câble USB à un serveur. Ensuite, le serveur envoie des données à la lunette, qui les analyse et effectue des actions en fonction de ces données.

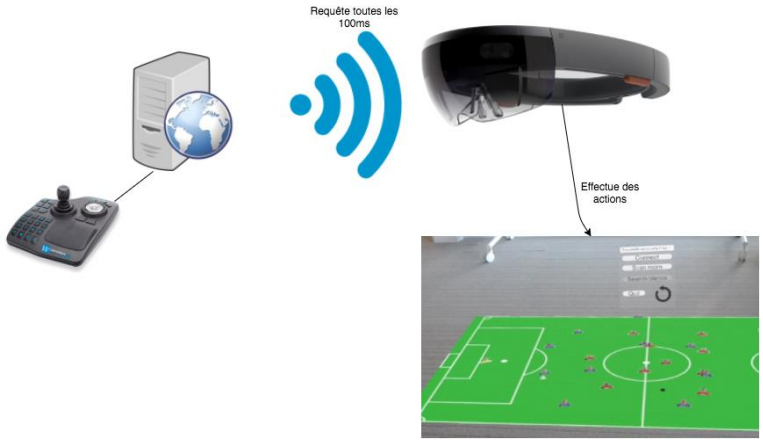

Figure 5: Schéma de principe de l'application de visualisation d'un match en 3D

4.1. Parsing du fichier TRACAB

Comme mentionné ci-dessus, les différentes données GPS (position X, Y, Z, vitesse) sont stockées dans un fichier texte, où chaque ligne représente un échantillon des

positions de toutes les positions des acteurs. Cet échantillonnage s'est déroulé à du 25 Hz.

Pour pouvoir exploiter ces données, nous avons décidé de créer différentes classes, dont la structure est illustrée à la figure 6 :

1. La première classe, appelée Snapshot, contient l'échantillonage de toutes les positions à un instant t, et correspond donc à une ligne du fichier texte. Ces Snapshots sont caractérisés par un frameID, qui est l'index du Snapshot dans la liste de tous les Snapshots, et un timeCode, qui contient le repère temporel du Snapshot dans le match. Ils contiennent également un objet de type SoccerAgents.
2. La seconde classe, la classe SoccerAgents, contient les positions de tous les acteurs, c'est à dire les joueurs, l'arbitre et la balle. Ainsi, elle contient donc 2 listes de SportAgent, pour l'équipe à domicile et à l'extérieur, et deux objets SportAgent, la balle et l'arbitre.
3. La dernière classe créée est la classe SportAgent, qui contient les différentes données sur les joueurs. Ces données sont, comme mentionné précédemment, les données GPS, ainsi que des données comme le numéro de maillot, l'équipe, ou encore un SystemID permettant au système de garder des numéros de joueurs entre 1 et 22.

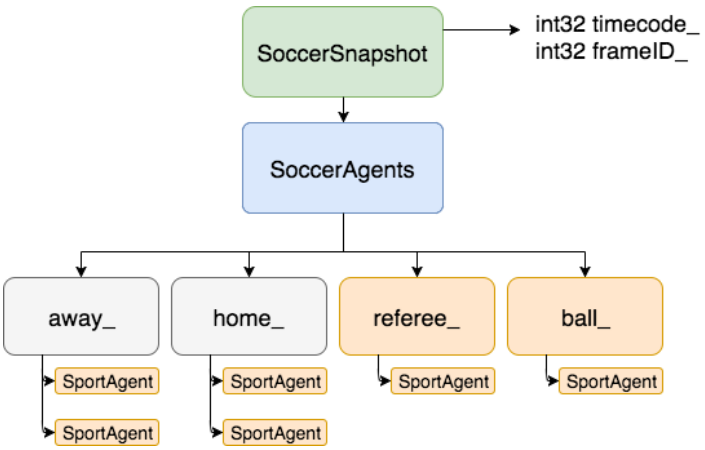

Figure 6: Relations entre les différentes classes de l'application GameVisualisation

Cette hiérarchie entre classes permet, en passant un objet de type SoccerSnapshot au programme, d'actualiser les positions de tous les joueurs, de l'arbitre ou encore de la balle (les objets de type SportAgent). On donnera donc, à vitesse réelle, toutes les 40 ms (la fréquence d'échantillonage étant de 25Hz) un nouveau SoccerSnapshot

afin d'actualiser les positions des différents éléments du terrain. L'entièreté des données sera ensuite lue au sein de cet objet de type SoccerSnapshot.

4.2. Communication entre la commande et la lunette

Afin de permettre à l'utilisateur de communiquer avec la lunette, il a été nécessaire d'ajouter une commande. Cette commande, illustrée à la figure 7, permet de réaliser un grand nombre d'actions.

Figure 7: Illustration de la commande permettant de communiquer avec l'application

Le port USB de la lunette étant exclusivement réservé à la recharge de la batterie de la lunette, une communication réseau a été mise en place entre la lunette, et un serveur, connecté en filaire à la commande. Toutes les 100 ms, une requête de type GET est effectuée par la lunette sur le serveur, qui renvoie une réponse adaptée aux évènements qui se sont déroulés sur la commande.

Cette réponse est structurée de la manière suivante :

deltaX, deltaY, deltaZ, compteurBrowe, keyPressed

Les termes deltaX, deltaY, deltaZ correspondent à l'écart entre la position du joystick et sa position d'équilibre, selon les 3 axes. Le terme compteurBrowse représente l'incrémentation (ou décrémentation) d'un compteur lié à la rotation de la roulette de la commande permettant de naviguer dans le match. Enfin, la variable keyPressed représente la dernière touche sur laquelle l'utilisateur a appuyé.

Une fois cette chaîne de caractères reçue par la lunette, elle est traitée par l'application, et l'action correspondant aux évènements est effectuée.

4.3. Fonctionnement de l'application

Dans son déroulement normal, l'application actualise la position des différents éléments à une fréquence de 25 Hz, qui est aussi la fréquence d'échantillonnage du fichier tracking GPS. Cependant, différentes actions peuvent être effectuées lors de l'utilisation de la commande. Certaines sont assez classiques, par exemple le Zoom-In/-Out ou encore la navigation dans le match, mais certaines sont plus complexes et sont donc développées ici.

Sélection et déplacement d'un joueur

Afin de permettre une analyse tactique, il est possible pour l'utilisateur de sélectionner un joueur, et d'ajuster sa position. Pour ce faire, l'utilisateur déplace le curseur lié au joystick, et sélectionne un joueur en appuyant sur une touche. Un « fantôme » est alors placé à la position initiale du joueur, et une droite relie le joueur, qui suit le curseur. L'utilisateur peut ensuite « déposer » le joueur à un autre endroit sur le terrain et répéter l'opération. Une illustration de cette fonctionnalité se trouve à la figure 8.

Figure 8: Illustration de la fonctionnalité permettant de modifier la position d'un joueur

Détection de zones

Etant dans un cas d'analyse de match de football, il est intéressant de repérer les différents espaces sur un terrain. Ainsi, il est possible de déplacer le curseur dans une zone du terrain, et en appuyant sur un bouton, la zone libre entre les 5 défenseurs les plus proches est tracée. Une illustration de cette fonctionnalité se trouve à la figure 9.

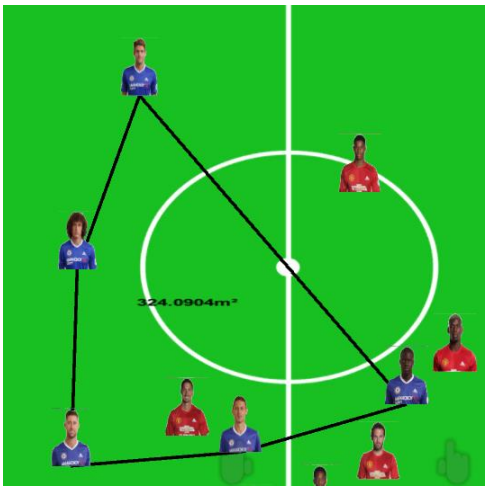

Figure 9: Illustration de la détection de zones

A noter qu'il est possible de combiner les deux fonctionnalités, et donc d'observer l'impact de la position d'un joueur sur la zone laissée libre par les défenseurs.

5. Conclusions

Bien que la lunette soit encore extrêmement récente, il est possible de la prendre en main assez rapidement, et de développer facilement des applications. Cependant, sortir du cahier des charges défini pour la lunette reste compliqué, étant donné le peu d'expériences vécues par les utilisateurs en général.

Ainsi, l'ensemble des ressources fréquemment utilisées pour le développement d'application, tels que les forums de développeurs, reste d'une aide assez limitée.

Bien que la seconde application ait donné des résultats très satisfaisants, il reste encore un grand nombre de pistes d'améliorations, notamment dans les connaissances du potentiel de la lunette. Les principales pistes d'améliorations se situent cependant

au niveau de la première application, en exploitant plus le Graphical Processor Unit (GPU) que le CPU.

Enfin, l'enseignement principal de ces 13 semaines de stage réalisées au sein de l'entreprise EVS Broadcast Equipment, est qu'il est important, lors de l'utilisation d'une technologie encore inconnue, de valider une à une les différentes capacités de celle-ci, avant de projeter de développer des applications plus complexes.

6. Sources

[1] Microsoft Dev Center, (consulté en août 2017), *Gestures*.
Adresse URL : https://developer.microsoft.com/en-us/windows/mixed-reality/gestures

[2] ESPN, (consulté en septembre 2017), *How ESPN plans to save millions on broadcasts without you ever knowing*.
Adresse URL : http://ftw.usatoday.com/2015/06/espn-broadcasts-remote-integration-save-millions

[3] VANVINCKENROYE, G., *Applications de streaming pour Hololens*, Travail de fin d'études, Liège, Belgique : Institut HELMo Gramme, juin 2017.