Théo KOKEL
GI03 - TN09

GENIE INFORMATIQUE

Suiveur UTC : Dominique LENNE

Évaluation des séquelles des patients post-COVID-19 par la réalité augmentée et les jeux sérieux

Laboratoire UTC BMBI – UMR CNRS 7338

Marie-Christine HO BA THO & Tien Tuan DAO

Septembre 2020 – Février 2021

Remerciements

Bien entendu, je tiens à remercier ma famille, ma mère, mon père ainsi que mes deux frères qui m'ont toujours soutenu et encouragé dans mon isolement compiégnois pendant ce semestre de stage.

Je souhaite aussi remercier Mme Marie-Christine Ho Ba Tho et M. Tien Tuan Dao qui m'ont fait confiance et ont laissé libre cours à mon imagination quant à la réalisation de ce projet, et dont les retours et conseils ont toujours été agréables et bienveillants.

Je tiens également à témoigner ma reconnaissance aux professeurs M. Dominique Lenne et Mme Domitile Lourdeaux qui m'ont aidé et orienté lorsque j'avais des difficultés. En cela je remercie vivement Yann Kerkhof, étudiant de l'UTC, qui m'a grandement aidé et répondu à mes nombreuses questions sur l'HoloLens.

Enfin, je remercie chaleureusement UTC Recherche et le laboratoire BMBI pour le financement de ce stage.

Sommaire

Résumé technique

Ce stage est un stage de recherche dans le laboratoire de Biomécanique et de Bioingénierie (BMBI) de l'Université de technologie de Compiègne (UTC). Il a pour objectif la création d'un démonstrateur, d'un prototype d'une application en réalité augmentée de test et d'évaluation des capacités fonctionnelles de patients guéris de la covid-19. Cette plateforme d'étude est implémentée dans le casque de réalité mixte HoloLens 2 de Microsoft grâce au moteur de jeu unity3D. La caméra Kinect de Microsoft est également utilisée pour la détection du corps du patient. Ce stage a abouti à la réalisation de deux applications permettant de mettre en œuvre ce démonstrateur. La première fonctionne sur un ordinateur sur lequel est connectée une Kinect et qui envoie en temps réel à l'HoloLens les données du corps détecté par un transfert en réseau local. La seconde application fonctionne sur le casque et implémente différents tests fonctionnels en utilisant les données de la caméra. Ce rapport commence par présenter le laboratoire et l'équipe d'accueil pour après préciser plus en détails les contours et objectifs du stage. Il retrace ensuite les différentes étapes du projet, de la découverte des technologies et des travaux existants à la création des applications, sans oublier les difficultés rencontrées ainsi qu'une conclusion établissant un bilan et une prise de recul sur le stage et ses différents apports et perspectives. En fin de document se trouvent une bibliographie, une table des matières détaillée ainsi que des annexes recueillant des informations supplémentaires mais non essentielles à la compréhension du document.

Présentation du laboratoire BMBI et de l'équipe C2MUST

1) Laboratoire BMBI

Le laboratoire BioMécanique et BioIngénierie, situé à Compiègne, est actuellement une Unité Mixte de Recherche (UMR) de l'Université de technologie de Compiègne (UTC) et du Centre National de la Recherche Scientifique (CNRS). L'unité de recherche a vu le jour en 1982 au sein de l'UTC sous l'impulsion de Michel Jaffrin et Dominique Barthès-Biesel. L'UTC et le CNRS s'associeront ensuite dès 1982 pour former l'Unité de Recherche Associée (URA) 858. Il a par la suite plusieurs fois changé d'appellation, devenant l'UMR 6660 en 1996 puis l'UMR 6600 en 2008 pour enfin être renommé UMR 7338 en 2012, nom que le laboratoire porte toujours [1]. Son budget annuel est d'environ deux millions d'euros.

Les activités de recherche du laboratoire sont partagées entre trois équipes : l'équipe « Cellules, Biomatériaux, Bioréacteurs » (CBB), l'équipe « Interactions Fluides Structures Biologiques » (IFSB) et l'équipe « Caractérisation et Modélisation personnalisée du Système Musculosquelettique » (C2MUST) [2] dans laquelle j'ai travaillé. Ces trois équipes portent l'effectif total du laboratoire à une centaine de personnes, comprenant trente-cinq permanents (enseignants chercheurs, ingénieurs et personnels administratifs), quarante contractuels (doctorants, postdoctorants et ingénieurs) et une vingtaine de stagiaires et masters. Au travers de ces trois équipes, les domaines d'activités du laboratoire sont pluridisciplinaires et reposent sur une large palette de compétences dont le principal et fondamental contour est l'ingénierie de la santé et du vivant. Ainsi ces activités vont de la biomécanique aux matériaux, en passant par l'électronique et l'informatique, et bien-sûr la biologie et la physiopathologie. On trouve donc des applications diverses et variées, comme le développement de biomatériaux ou de prothèses, l'étude de la physiologie neuromusculaire, de la mécanique des fluides ou encore des structures biologiques.

À travers toutes ces études et applications, l'objectif principal du laboratoire est de comprendre le fonctionnement et les mécanismes des systèmes vivants à toutes les échelles, afin de contribuer à améliorer la qualité de la vie de l'Homme [3]. Ces systèmes vivants sont tous les systèmes présent dans le corps humain, comme les systèmes squelettique et/ou musculaire, les organes, les tissus ainsi que les cellules et molécules qui leur sont associées. Tout cela prend son sens quand on apprend que la

devise du laboratoire est « comprendre pour faire ». Comprendre comment fonctionne une articulation, un muscle, comment réagissent les tissus pour améliorer leurs modélisations, pour développer des prothèses afin de les suppléer ou bien des systèmes d'aide à la rééducation fonctionnelle.

La dimension éthique est également très importante pour le laboratoire, car à l'heure du patient connecté et à l'aube du transhumanisme la question de l'acceptabilité et de la recevabilité des technologies est primordiale dans notre société. Le laboratoire, fortement impliqué dans les formations des étudiants de l'UTC dans tous les cursus, appuie l'orientation des formations dans ce sens.

Le laboratoire BMBI, qui compte environ 50 publications dans des revues scientifiques, conférences et brevets chaque année, est un laboratoire très réputé à l'international et dans les cercles de biomécanique et de bioingénierie qui le considèrent comme une référence dans ces deux domaines. De plus, certains de ses membres sont également personnellement reconnus dans la communauté scientifique, en occupant des postes dans des sociétés savantes comme le Conseil Mondial de Biomécanique ou encore la Société Européenne pour les Organes Artificiels. Enfin, BMBI a joué un grand rôle lors de l'organisation et l'accueil par l'UTC de la 7ème conférence internationale en modélisation biomécanique en septembre 2017.

2) Présentation de l'équipe C2MUST

Comme expliqué plus tôt, le laboratoire BMBI est composé de trois équipes et j'ai évolué au sein de l'équipe C2MUST (« Caractérisation et Modélisation personnalisée du Système Musculosquelettique »). Cette équipe est née en 2018 de la fusion d'une équipe de biomécaniciens et d'une autre spécialisée dans le traitement d'informations. Elle est composée en 2020 d'environ trente chercheurs, enseignants-chercheurs, doctorants, masters et stagiaires, tous contribuant ainsi au rayonnement du laboratoire et à sa productivité.

Comme son nom l'indique, l'activité de cette équipe gravite autour des deux thèmes que sont la caractérisation et la modélisation du système musculo/tendineux/squelettique [4]. Les travaux de caractérisation, qui reposent sur des approches expérimentales, visent à mieux comprendre les différents tissus du corps humain, qu'ils soient osseux, tendineux ou musculaire, en interactions ou seuls, sains ou pathologiques. Ces recherches de caractérisation permettent ainsi de mieux connaître les réactions et propriétés chimiques et mécaniques des tissus dans le but

de proposer des modélisations les plus fidèles et fiables possibles. Ainsi viennent les travaux de modélisation, dont l'objectif est d'améliorer les modèles électriques et biomécaniques existants des tissus et d'en proposer des nouveaux [5], mais aussi de développer des outils d'analyse et de décision.

Cette équipe du laboratoire travaille en collaboration avec de nombreux partenaires. Il y a d'une part des partenaires cliniques comme des hôpitaux et des centres hospitaliers universitaires, qu'ils soient en France ou à l'étranger comme en Australie ou aux États-Unis. Et il y a d'autre part un grand nombre de partenaires scientifiques comme des Universités ou des Instituts de médecine, dispersés partout en Europe et même en Chine [4]. Tous ces partenariats traduisent des collaborations autour de projets nationaux ou internationaux. L'équipe a aussi de nombreux autres rôles, comme dans l'organisation de manifestations internationales ou nationales, la participation à de nombreuses expertises ou autres comités éditoriaux.

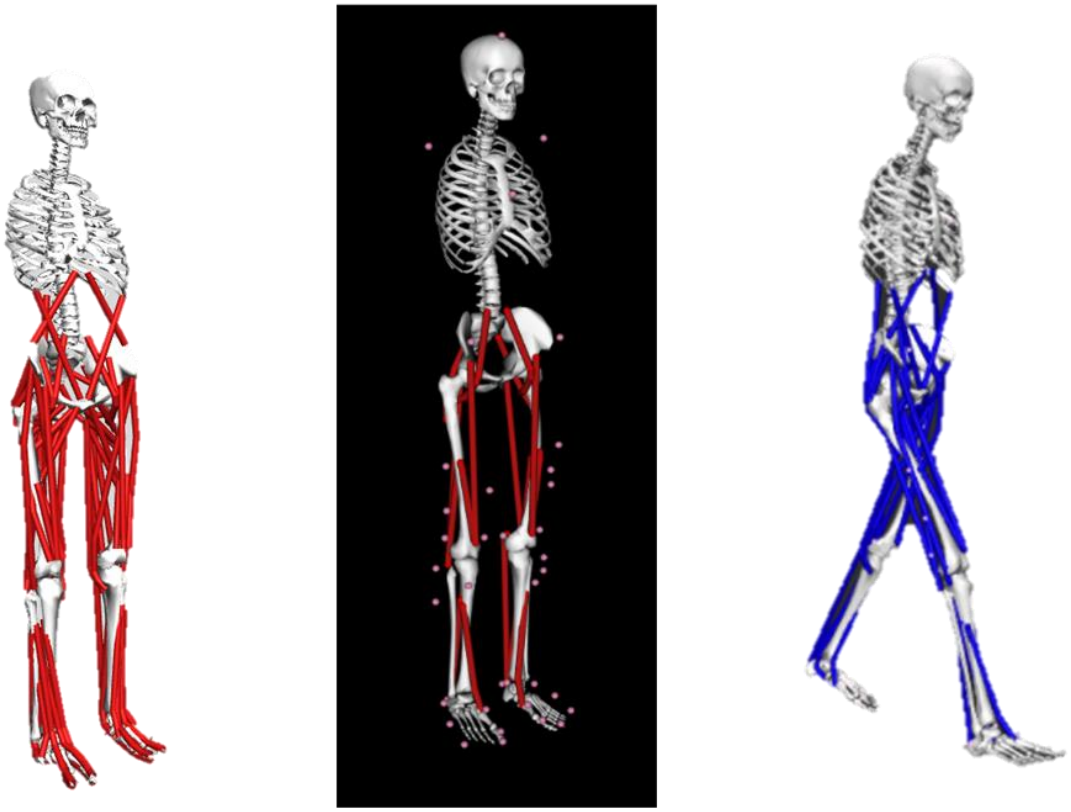

Figure 1: exemple de modélisation du système musculosquelettique développé à BMBI [5]

Mission et objectifs du stage

1) Sujet

L'intitulé du stage est « Évaluation des séquelles post-COVID-19 par la réalité augmentée et les jeux sérieux ». On retrouve donc dès ce titre la dimension de caractérisation, à travers le terme « évaluation », fondamentale dans les travaux de l'équipe C2MUST du laboratoire BMBI. L'idée de ce projet est née de l'appel d'offre de l'UTC AMI COVID-19 lancé en mai 2020 ayant pour but de proposer des solutions à des problématiques liées au virus du Covid-19 qui bouleverse le monde depuis décembre 2019. BMBI a été lauréat du projet POST COVID19 dont le postulat vient d'une constatation simple : le virus peut laisser au patient guéri ce qui semble être un éventail infini de séquelles. Problèmes neurologiques, fatigue extrême, difficultés respiratoires et douleurs articulaires et musculaires sont des exemples de ce qu'on peut garder malgré la guérison du virus.

L'objectif à long terme du projet est de développer une plateforme numérique en réalité augmentée de diagnostic des séquelles fonctionnelles et neurocognitives chez les patients ayant été touchés par la COVID-19. L'ambition du projet est de proposer cette plateforme sous deux formes. La première version, la plus aboutie et la plus conséquente est l'objectif final du projet, elle serait utilisée directement par les professionnels de santé dans le but de proposer des thérapies de rééducation spécifiques et stratégiques en fonction des séquelles des patients, pour ainsi optimiser et faciliter leur rétablissement. La seconde version, plus « légère », serait une application à destination du grand public souhaitant s'auto-évaluer.

Le but de ce stage est alors de proposer une plateforme de démonstration et de test de ce qu'il est possible de faire avec les technologies utilisées, servant ainsi de base et de fondation pour le futur du projet.

2) Cahier des charges

Le projet n'étant qu'à ses prémisses au début du stage, il n'y avait pas de cahier des charges précis et défini. Seuls l'objectif final cité avant ainsi que les technologies à utiliser étaient clairs : il fallait utiliser l'environnement de développement Unity3D pour développer et implémenter des jeux sérieux dans le casque de réalité mixte HoloLens 2 de Microsoft couplé à une caméra Kinect 2. Le cahier des charges est

apparu et s'est affiné au cours des jours et semaines de travail et des réunions de suivi du stage, où la présentation des avancées amenait des questions, des réponses et souvent d'autres idées pour compléter et étoffer ce qui avait déjà été fait. Au début du stage il a été fixé que l'application sur HoloLens devait être pensée et réalisée pour le médecin car c'est lui qui porterait le casque.

3) Planning

L'utilisation des lunettes de réalité mixte HoloLens 2 étant nouvelle pour le laboratoire et mes responsables, je n'avais pas de documents ou travaux internes sur lesquels m'appuyer pour guider mon apprentissage et la maîtrise de l'appareil. De plus, nous utilisons la deuxième itération et dernière en date du casque de Microsoft, sortie en novembre 2019, et la plupart des travaux et aides liés à la réalité mixte que l'on peut trouver concernent la première version de 2016 du casque. Les aides en lignes ne sont donc pour l'instant pas très riches et il y a encore beaucoup de choses à l'état expérimental. La technologie de la réalité mixte est elle-même encore beaucoup à l'état expérimental, avancée certes mais expérimentale. Enfin, il a fallu attendre la fin du mois d'octobre, soit deux mois après le début du stage, pour que le casquel soit livré au laboratoire, ce qui n'était bien évidemment pas prévu. Tout cela a fait que le planning constaté (cf. tableau 2) est très différent du planning prévisionnel (cf. tableau 1) imaginé lors de la soumission du projet dans le cadre de l'appel d'offre de BMBI. En effet, il était prévu de travailler tout le long du stage sur l'environnement virtuel en réalité augmenté, or cela n'a pu se faire qu'après la réception puis quelques temps de découverte et d'apprentissage du casque, soit autour du mois de décembre. De même, à l'heure actuelle il n'y a pas de réelle modélisation biomécanique implémentée comme cela était prévu et souhaité, et disponible au laboratoire.

On pourrait qualifier chaque début de tâche comme une date importante car elle sonne la fin de la précédente et donc le passage d'une étape du projet. Ainsi le début en novembre du travail sur HoloLens marque la fin de la longue attente du casque. J'ai ensuite dû découvrir l'appareil, le tester et sonder ses capacités, tant cette technologie et son fonctionnement m'étaient inconnus. C'est naturellement à la suite de ces essais que la création de l'application sur le casque pouvait débuter, à partir de décembre. Si je ne devais retenir qu'une seule date et étape clé dans le déroulement du stage ce serait celle où j'ai réussi à mettre en place la communication par réseau local entre l'ordinateur qui utilise la Kinect et l'HoloLens. Cela était assez compliqué à faire car je n'avais aucune connaissance en réseau, mais aussi et surtout car les deux applications du projet ne sont pas compilées de la même manière car elles ne visent pas les même plateformes, ce qui implique des non-compatibilités entre les deux applications. Cette étape est cruciale car sans ce transfert de données par le réseau, l'HoloLens ne peut avoir accès aux données de détection du corps de la Kinect donc on

ne peut implémenter aucun système d'analyse des données et par extension aucun exercice et jeu sérieux.

Tâches	Septembre	Octobre	Novembre	Décembre	Janvier	Février
Environnement en réalité augmentée						
Vision par ordinateur et IA						
Modélisation biomécanique						
Tests et déploiement						

Tableau 1: planning prévisionnel du déroulement du stage

Tâches	Septembre	Octobre	Novembre	Décembre	Janvier	Février
Prise en main de la RA et de la Kinect						
Prise en main de l'HoloLens						
Communication par réseau local						
Environnement en réalité augmentée						

Tableau 2: planning réel et constaté du déroulement du stage

4) Contributions

Pour ce qui est de mes contributions, comme ce stage est un stage de recherche et que le projet n'avait pas encore commencé avant le stage, je travaillais seul et étais en totale autonomie et tout ce qui existe actuellement du projet a été fait pendant le stage et donc par moi. J'ai ainsi développé les deux applications nécessaires au fonctionnement du démonstrateur, la première sur l'ordinateur qui utilise la Kinect et qui envoie ses données à la seconde application sur l'HoloLens qui implémente l'utilisation de ces données et les scénarios de tests.

5) Tests et vérifications

Comme je travaillais seul sur le projet, je me chargeais de tester et de vérifier que mes implémentations fonctionnaient bien. Cela est assez simple car l'éditeur d'Unity permet de simuler et tester le programme sans perdre de temps à le compiler pour la plateforme cible. Il fallait donc juste que je me place devant la Kinect pour que mon corps soit détecté et ensuite vérifier dans Unity que tout fonctionnait comme je le voulais. C'est seulement quand tout est fonctionnel sur Unity que les tests sur l'HoloLens sont pertinents, car les temps de compilation sont relativement longs et feraient perdre beaucoup de temps s'il fallait compiler et déployer l'application dans le casque à chaque ajustement ou nouvelle fonction.

Mes réalisations

1) Les technologies utilisées

Pendant le stage, j'ai utilisé majoritairement les trois technologies que sont la caméra Kinect v2, le casque de réalité mixte HoloLens 2 ainsi que le moteur de jeu Unity3D.

a) Kinect V2

Figure 2: photographie de la Kinect V2

La caméra Kinect v2 est la deuxième itération de la caméra Kinect produite par Microsoft pour la Xbox One en 2013. C'est donc un appareil créé pour les jeux-vidéo. Cependant, il est loin d'être le seul domaine d'activité du capteur. En effet, à cause du mauvais lancement de la Xbox One et du manque de jeux et de soutien de la part des éditeurs, ce n'est pas chez les joueurs que la Kinect sera la plus adoptée mais dans les laboratoires de recherches et chez les développeurs explorant le vaste champ de la vision par ordinateur.

En effet la Kinect est un appareil reconnu pour ses très bonnes performances obtenues à un prix très abordable (une centaine de dollars). Il s'agit d'une caméra permettant de filmer en HD en 1080p à 30 images par secondes mais elle est également dotée d'un capteur de profondeur infrarouge (cf. figure 3). De type « temps de vol », ce capteur calcule la profondeur des points en fonction du temps que met le signal infrarouge émis à revenir sur le récepteur de la caméra.

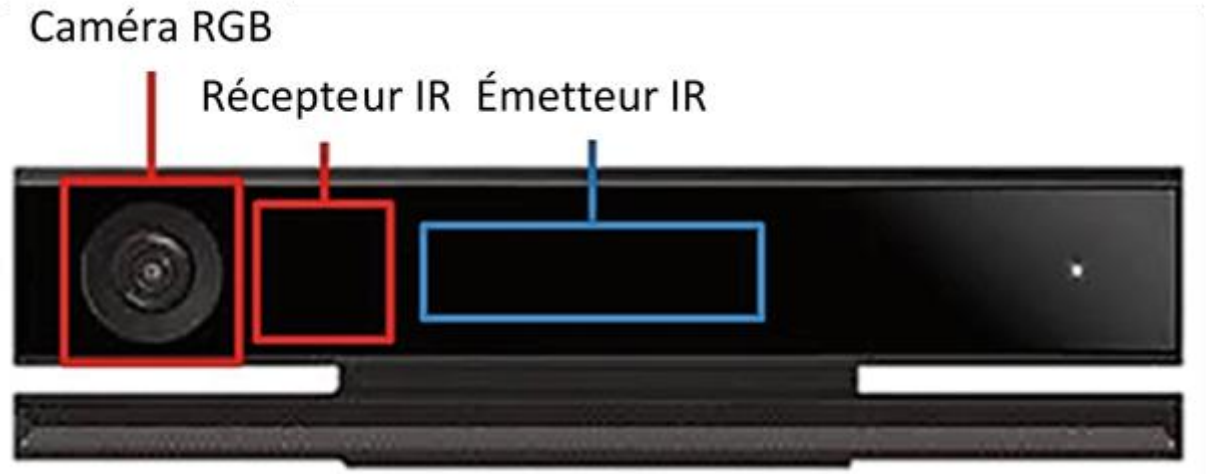

Figure 3: position des différents capteurs de la Kinect

C'est précisément ce capteur de profondeur qui fait la force et la réputation de l'appareil, car il permet de détecter jusqu'à six corps et d'extraire sur chacun vingt-cinq points d'articulations (cf. figure 4). Ces points d'articulations permettent de reconstituer virtuellement les mouvements des corps détectés, sous les traits d'un avatar par exemple, afin de les étudier et de les utiliser dans le cadre d'un jeu ou d'une application comme la nôtre.

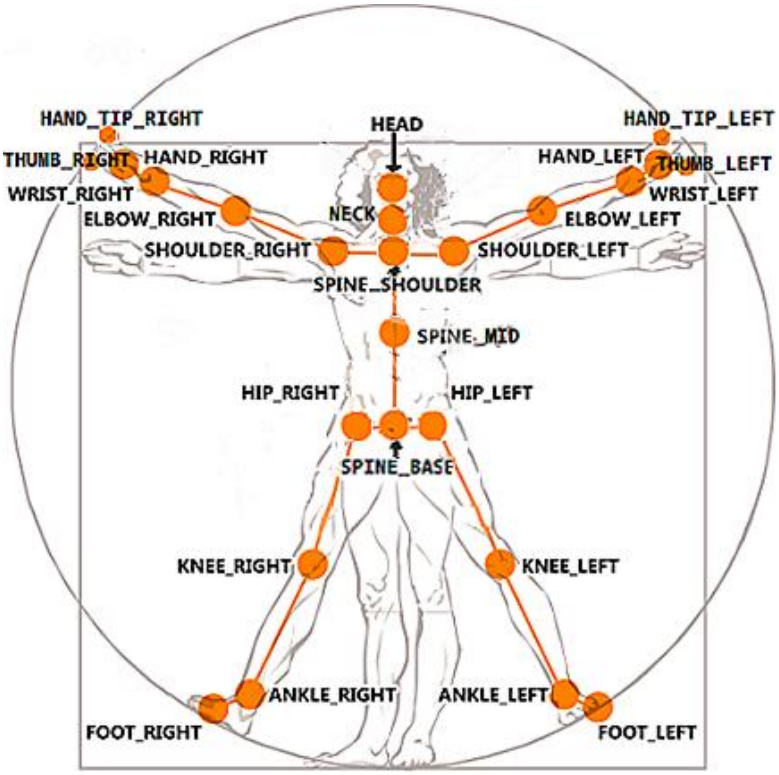

Figure 4: positions des vingt-cinq points d'articulations détectés par le capteur

Enfin, la force de cette caméra réside également en son kit de développement logiciel (SDK) proposé par Microsoft, qui facilite l'utilisation de ses fonctionnalités car il les implémente au travers d'une bibliothèque exploitable entre autres dans Unity.

Pour le projet, la Kinect est utilisée pour détecter les positions des membres du patient qui est idéalement placé face à elle. Ces positions sont ensuite traitées sur ordinateur et envoyées sur l'HoloLens. L'objectif est alors d'utiliser ces données pour donner vie à un avatar holographique visible dans notre espace grâce à l'HoloLens.

b) HoloLens 2

Figure 5: photographie de l'HoloLens 2

Le casque de réalité mixte HoloLens 2 de Microsoft, leader commercial et technique sur ce marché, est un appareil permettant de visualiser des hologrammes dans notre environnement et de les manipuler. Pour cela, le casque est un concentré de technologie, et tout particulièrement de caméras. En effet, il est doté de quatre caméras (deux sont visibles sur le côté supérieur droit du casque cf. figure 5) dédiées au suivi des deux mains en temps réel, de deux caméras infrarouges opérant le suivi des deux yeux en temps réel et d'un capteur de profondeur (au centre du casque cf. figure 5), semblable à celui de la Kinect. Ce dernier capteur sert notamment à détecter l'environnement de l'utilisateur pour en calculer un maillage (cf. figure 6) afin de reconnaître, par exemple, les obstacles et les murs, de sorte à les prendre en compte dans le rendu des hologrammes : si l'on vient à déplacer un objet holographique et qu'une partie de celui-ci se retrouve derrière un obstacle, alors cette partie ne sera pas visible par l'utilisateur et semblera réellement être derrière l'obstacle, comme ce serait le cas pour un objet réel. De même, ce maillage permet d'appliquer des lois physiques sur des objets virtuels : si un hologramme vient à être lâché et qu'on a préalablement

décidé de le soumettre à la gravité, alors sa chute s'arrêtera sur le sol ou tout autre objet présent sur sa trajectoire.

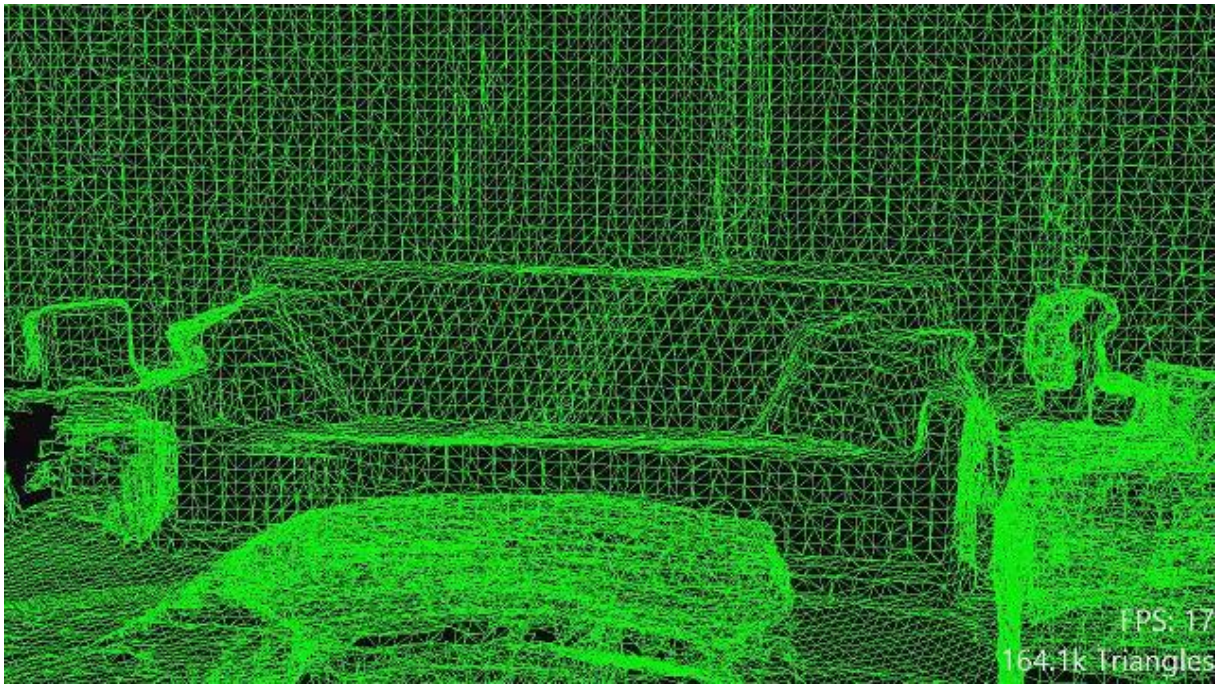

Figure 6: exemple de maillage obtenu grâce au capteur de profondeur de l'HoloLens

L'HoloLens est la clé de voûte de ce projet, car ce dernier repose entièrement sur le casque. C'est au travers des lunettes que nous voyons et apprécions les fruits du stage. C'est grâce au casque que le projet est innovant et unique. C'est parce qu'il n'empêche pas la relation et l'interaction entre le médecin et son patient qu'il a un réel intérêt thérapeutique. Nous utilisons ici le casque pour animer et visualiser un squelette holographique d'après les données reçues de la Kinect. Ce squelette nous sert alors de support pour calculer différents angles et distances afin de tester les capacités fonctionnelles du patient lors de séries de mouvements.

c) Unity3D

Figure 7: logo du moteur de jeu Unity

Les capacités et fonctionnalités de la Kinect et de l'HoloLens sont remarquables, mais si on ne les rassemble pas, si on ne les unit pas dans un environnement commun,

alors bien évidemment on ne peut pas les utiliser ensemble et bénéficier des atouts de chacun des appareils. Unity joue ce rôle de lieu de convergence de ces deux technologies. C'est un des moteurs de jeu les plus utilisés dans l'industrie vidéoludique, car il est relativement simple d'utilisation, car c'est un outil de création très puissant qui permet le développement d'applications sur de nombreuses plateformes et car il est supporté et enrichi par une communauté fidèle qui lui permet de constamment s'étoffer en fonctionnalités.

Pour expliquer simplement comment fonctionne Unity et comment programmer dans cet environnement, on pourrait simplement dire que le moteur regroupe dans des scènes des objets sur lesquels viennent être attachés des composants qui vont leur donner leur fonctionnalités, comportements ou valeurs clés. Ces composants sont par exemple des scripts, des textures ou des corps rigides (« *rigid body* », pour appliquer la gravité à un objet par exemple). Dans Unity, les objets sont appelés « *gameObjects* » et le langage de programmation utilisé est le C#. C'est un langage orienté objet créé par Microsoft en 2002, dérivé du C++ et proche du Java. Ainsi un jeu ou toute autre application sur Unity est un assemblage de gameObjects sur lesquels sont attachés des composants.

Malheureusement, plusieurs raisons font que les deux appareils ne sont pas utilisables dans la même application. La première est que l'HoloLens ne peut pas utiliser un périphérique externe auquel il est branché. Même si cela devenait possible, la Kinect et l'HoloLens sont deux appareils ayant deux architectures différentes, et donc deux méthodes de compilation sur Unity différentes, ce qui implique qu'on ne peut pas les utiliser en même temps dans la même application. Il est donc nécessaire d'avoir deux applications structurellement différentes, l'une utilisant la Kinect sur PC et l'autre sur HoloLens qui parvient à avoir accès aux données de la caméra tout en en étant totalement indépendante.

Ainsi dans ce projet, Unity est utilisé pour créer les deux applications nécessaires. Dans un premier temps, celle sur ordinateur est branchée à la Kinect et récupère les données des articulations puis les envois par le réseau local. Dans un second temps vient celle sur l'HoloLens qui va lire et récupérer ces données, pour les affecter aux points pertinents du modèle 3D du squelette qui sera affiché et vu sous la forme d'un hologramme grâce au casque. Enfin sur cette seconde application sont mis en place différents exemples de tests fonctionnels.

2) État des lieux des technologies utilisées

Au début du stage, l'HoloLens n'avait pas encore été livré au laboratoire et il ne l'a été qu'à la toute fin du mois d'octobre. J'ai donc eu deux mois pour apprendre à utiliser et m'initier à la Kinect qui était disponible au laboratoire, mais aussi pour faire des recherches sur l'HoloLens et surtout le couplage entre la caméra et le casque, car le stage repose entièrement sur cette association.

a) Utilisations de la Kinect v2

La Kinect est une caméra créée pour les jeux-vidéo. Elle est très intéressante d'un point de vue vidéoludique car elle permet une véritable implication du joueur dans le jeu, car il doit utiliser son corps pour jouer et déplacer son avatar. De ce fait, elle améliore également les interactions et la compétitivité entre les joueurs dans le cadre de jeux à plusieurs car ce sont désormais les performances physiques, et non plus celles manette en main, qui vont déterminer le résultat de la partie.

Figure 8: visuel promotionnel d'un jeu de sport utilisant la Kinect

Du point de vue de la recherche, la caméra est énormément utilisée grâce à son rapport performances/prix très correct. Elle permet un vaste champ d'applications reposant sur la détection et reconnaissance des corps, comme dans le domaine médical, le contrôle de robots humanoïdes ou de robots suiveurs ou encore la domotique.

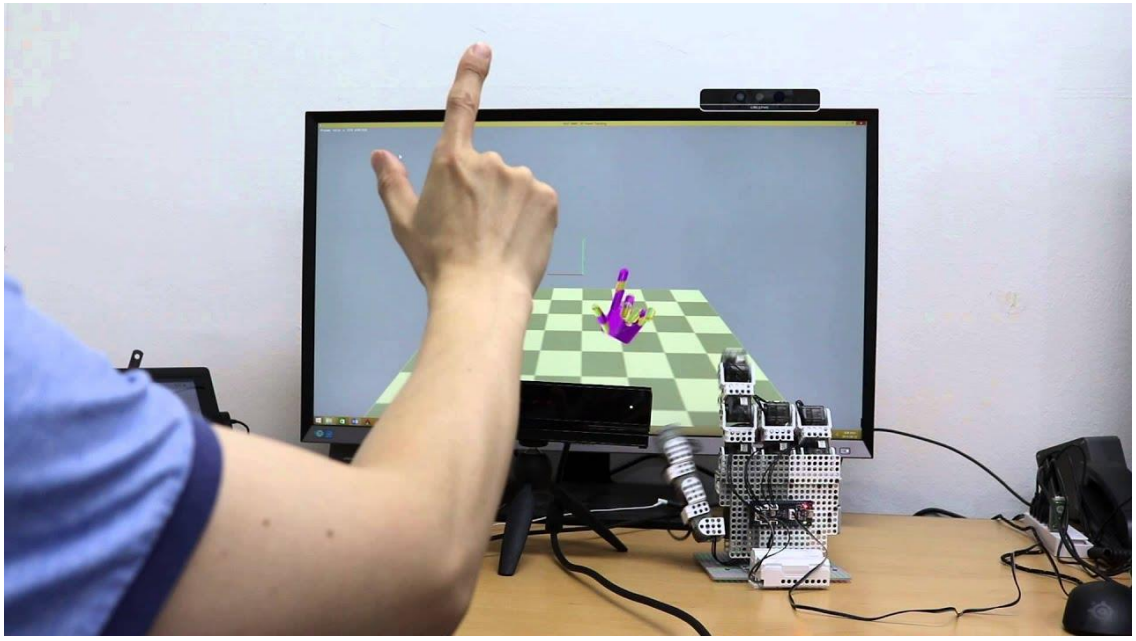

Figure 9: exemple d'utilisation de la Kinect dans le contrôle d'une main robotique

Pour exploiter la Kinect dans Unity, Microsoft propose un kit de développement logiciel (SDK) ainsi qu'un paquet destiné à l'utilisation du capteur dans le moteur [6]. Cependant, la caméra est utilisée en grande partie par des chercheurs ou des développeurs chevronnés et spécialisés dans la vision par ordinateur et qui n'ont donc pas forcément besoin d'un kit très avancé, car ils créent et développent leurs propres outils selon leurs objectifs. Mais ce n'est pas mon cas et mon manque de connaissances sur le fonctionnement du matériel et de la vision par ordinateur ne me permettait pas d'intégrer et utiliser la Kinect dans Unity. Il me fallait des outils offrant cette intégration ainsi qu'une utilisation plus simple de la caméra. J'ai trouvé ces outils dans les travaux de Rumen Filkov [7] qui propose une série de scènes faisant la démonstration de ce qui est faisable avec le capteur. Son paquet d'outils se nomme « *Kinect v2 Examples with MS-SDK* » et comme on peut le retrouver dans sa documentation [8], il propose près de cinquante scènes de démonstration. Ces dernières vont du contrôle d'avatar virtuel au retrait de l'arrière-plan, en passant par le suivi des mouvements faciaux ou encore la reconnaissance de gestes.

b) Utilisations de l'HoloLens 2

L'HoloLens est une technologie principalement utilisée dans l'industrie. En effet, comme elle permet de visualiser et interagir avec des hologrammes, les professionnels lui portent un réel intérêt pour faciliter leurs tâches de prototypage, de maquettage ou encore de maintenance. On retrouve par exemple des utilisations du casque sur des chaînes de production [9], pour visualiser des constructions en devenir [10] et dans la maintenance de bâtiments [11]. Mais cette technologie a aussi un énorme potentiel et pourrait révolutionner la médecine [12], en remplaçant par exemple les écrans des

salles d'opérations au profit de panneaux virtuels placés directement devant les yeux du chirurgien (cf. figure 10).

Figure 10: visuel promotionnel illustrant une utilisation possible du casque dans la médecine

Mais l'HoloLens 2 reste tout de même un appareil de niche avec pour cible principale l'industrie, et il est encore peu utilisé dans la recherche. Cependant il est à souligner que l'appareil est disponible depuis novembre 2019, il est donc jeune et cela peut aider à comprendre l'actuel manque de projets l'utilisant. On trouve cependant quelques projets remarquables notamment dans le domaine médical, qui se rapprochent un peu plus du cadre de ce stage. Par exemple, une équipe de Microsoft a mis au point Cardiolens, un programme permettant de mesurer le pouls et le rythme cardiaque d'un patient seulement en le regardant [13]. De même, un groupe universitaire utilise la technologie HoloLens pour enrichir et améliorer l'éducation médicale proposée à ses étudiants [14].

c) Couplage Kinect v2 - HoloLens 2

Ce qui nous intéresse le plus pour ce stage, ce sont les méthodes de couplage entre ces deux appareils, et plus précisément les moyens d'utiliser les données de la caméra sur le casque pour les exploiter et les étudier. Le mot-clé dans ces recherches d'inspiration et de projets existants s'est révélé être « avateering », le contrôle d'avatar, car c'est ce que nous voulions depuis le début : voir dans le casque un squelette que l'on contrôle par nos mouvements perçus par la caméra. On trouve en ligne plusieurs exemples en vidéo de cette réalisation, mais aucun des projets que j'ai

pu trouver n'était open-source et à disposition gratuitement. Je savais donc que cela était possible mais je n'avais pas d'informations sur la manière de faire.

J'ai par la suite trouvé le projet open-source « LiveScan3D – HoloLens » de Marek Kowalski [15] qui propose une solution de « téléprésence ». Cela consiste à utiliser plusieurs Kinect pour détecter le même corps et en produire un nuage de points, afin de le recréer virtuellement en tant qu'hologramme à un endroit différent, comme visible sur la figure 11. Cependant je n'ai pas utilisé ce projet car il ne correspondait pas réellement à nos besoins, mais je savais assurément qu'il était possible de faire fonctionner les deux appareils ensemble et j'étais sur la bonne voie.

Figure 11: capture d'écran d'une vidéo démonstrative du projet LiveScan3D

Ma plus grande inspiration, et le projet qui m'a le plus aidé dans la réalisation de ce couplage est le projet de Michelle Ma [16] dont les travaux sont relatés dans un article publié sur le Windows blog de Microsoft [17]. Cet article présente ce que permet de faire le projet, comment il le fait et explique les différents choix d'architecture ou de méthodes dans le code. Ce projet implémentait exactement ce que l'on voulait, à savoir la visualisation d'un avatar holographique contrôlé par la Kinect, à une différence près : il utilisait la première version de l'HoloLens. Les deux appareils étant différent (l'HoloLens 1 est construit sous une architecture ARM alors que l'HoloLens 2 utilise une architecture ARM 64), j'ai préféré mettre en place notre propre programme plutôt que risquer de trop m'attarder à adapter le projet de Michelle Ma. De plus, c'est avec cet article et ce projet que j'ai compris que j'allais devoir faire deux applications structurellement différentes. En effet, à cause des trop grandes différences entre la Kinect et le casque, il faut une application sur PC utilisant la caméra et une autre sur l'HoloLens qui doit être totalement indépendante de la caméra mais qui doit quand

même pouvoir utiliser ses données. Enfin, je me suis également inspiré de ces travaux pour l'animation et le contrôle de l'avatar, notamment dans le choix de la donnée pertinente à envoyer au casque pour pouvoir retranscrire au mieux les mouvements du squelette.

3) Application Kinect sur PC

L'application sur PC est la seule des deux qui est branchée à la Kinect et qui l'utilise entièrement. Son objectif est de détecter les vingt-cinq points correspondant aux articulations du patient pour diffuser leurs données sur le réseau local.

a) Intégration de la Kinect dans Unity

Pour faire fonctionner la Kinect et ses fonctionnalités dans Unity, j'ai utilisé le paquet de Rumen Filkov [7] vendu sur l'Asset Store, la plateforme de ventes d'outils, de modules et de modèles 3D d'Unity. C'est un des seuls et surtout le meilleur outil disponible pour utiliser la Kinect dans Unity. Il est normalement payant mais son auteur l'envoi gratuitement aux étudiants.

Figure 12: capture d'écran de la superposition des points d'articulations

Figure 13: capture d'écran du contrôle d'avatar

Ce paquet fourni tous les scripts nécessaires au bon fonctionnement de la Kinect dans l'environnement Unity. Parmi tout ce qu'il rend possible de faire, ce qui nous intéresse le plus dans le cadre de ce projet sont la superposition des points

d'articulations et des lignes représentant les membres sur le flux vidéo de la caméra (*« joint overlaying »* cf. figure 12*),* le contrôle d'un avatar par le mouvement du corps (*« skeleton avateering »* cf. figure 13*)* et « la somme » de ces deux fonctionnalités, à savoir la superposition sur la vidéo de la caméra de l'avatar (*« skeleton avateering overlay »* cf. figure 14).

Figure 14 : capture d'écran de la superposition du contrôle d'avatar

b) Récupération des données de la Kinect

J'ai pu tester les fonctionnalités précédentes les deux premiers mois du stage, en attendant l'arrivée de l'HoloLens, pour apprendre à utiliser Unity d'une part et les outils Kinect d'autre part. Mais pour le projet, je n'ai eu besoin que de la superposition des points (cf. figure 12).

J'ai réutilisé deux scripts du paquet permettant d'exploiter la caméra. Le premier s'appelle « KinectManager » et est long de presque cinq mille lignes de code. Il constitue la principale et indispensable passerelle entre la caméra et Unity, en codant les fonctions et données fondamentales à l'utilisation du capteur. Sans ce script, on ne pourrait rien faire. Je n'ai pas eu à le modifier. Le second script se nomme « SkeletonOverlayer » et utilise ce premier script pour implémenter l'affichage sur la

vidéo des points ainsi que des lignes correspondant aux membres, dessinant ainsi un squelette simplifié.

L'estimation et la reconstruction du squelette par la Kinect se fait en plusieurs étapes [18] [19]. Tout d'abord une représentation des profondeurs des objets de la scène faisant face à la caméra est effectuée grâce aux données calculées par le capteur infrarouge (cf. figure 15). De cette image, le logiciel de la Kinect va extraire et ne va garder que le premier plan correspondant aux potentiels corps détectés. Le capteur va ensuite appliquer un algorithme de partitionnement (« clustering ») qui va étudier chaque pixel dans le but de le classer et de le référencer comme appartenant à une partie précise du corps (cf. figure 16). Pour cela, l'algorithme va calculer pour chaque pixel la différence de profondeur avec un autre pixel décalé sur les axes des abscisses et/ou des ordonnées. Cette différence sera alors étudiée à travers trois arbres de décision qui attribueront la partie du corps à laquelle appartient le pixel. Les arbres de décision permettent une plus grande flexibilité dans l'étude des pixels. Cela est nécessaire à la précision de la reconnaissance du fait de la grande diversité des corps possibles (taille, forme, position etc…). Ils ont chacun une profondeur de vingt niveaux et sont issus d'un apprentissage du logiciel sur près d'un million de données de position et de capture du mouvement. Enfin, le programme utilise un algorithme de recherche de la valeur dominante pour déterminer le point de la zone qui sera renvoyé par la Kinect.

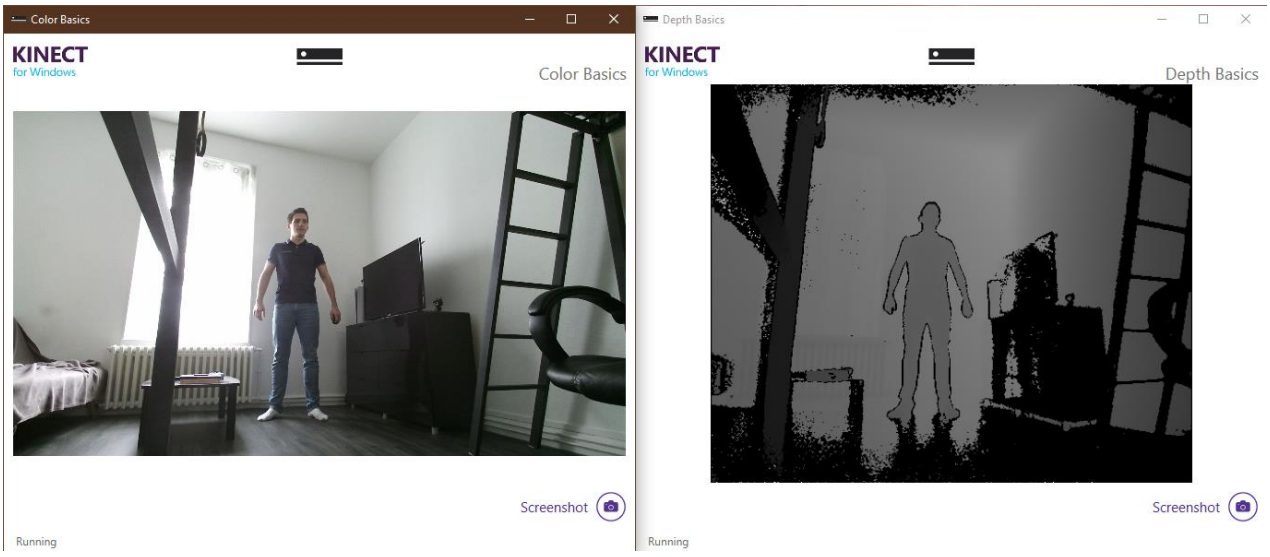

Figure 15: captures d'écran des flux de la caméra et du capteur de profondeur obtenues avec le SDK de la Kinect

Figure 16: capture d'écran du document [18] expliquant le procédé du calcul des points du corps, on voit ici la répartition des pixels en différentes zones du corps

La scène constituant l'application « Kinect App » est composée de quatre principaux objets qui permettent d'organiser les autres objets selon leur rôle et fonction sous la forme d'une hiérarchie (ces objets principaux sont appelés « parents » des objets qui les composent) . Ainsi on retrouve les objets :

- « Video Flow » qui contient les objets nécessaires à l'affichage du flux vidéo de la caméra
- « UI Manager » qui centralise les éléments gérant l'interface comme les boutons et les textes
- « Kinect Data Manager » qui regroupe les objets utilisant la Kinect
- « Server » qui implémente le coté serveur du transfert de données

Ce sont les deux derniers objets qui sont particulièrement utiles. Tout d'abord, intéressons-nous ici à l'objet « Kinect Data Manager », car l'aspect réseau de l'application aura sa propre partie par la suite. Il a un script nommé « KinectDataManager » qui lui est attaché et qui s'occupe de mettre à jour la chaine de caractères qui sera envoyée par le serveur. Il a ensuite trois objets « enfants » intitulés « Joints », « Lines » et « Kinect Controller ». Le premier contiendra les points (appelés « joints » en anglais) du corps détecté et le second les lignes symbolisant les os. Enfin sur le dernier sont attachés les deux scripts « KinectManager » et « SkeletonOverlayer » vus au début de cette partie et qui font le lien entre la caméra et la scène. Ainsi l'objet « Kinect Data Manager » utilise trois scripts qui permettent d'utiliser la caméra, de recueillir ses données puis de les mettre en forme pour les envoyer à l'HoloLens.

c) Formatage des données pour l'envoi sur le réseau local

Maintenant que nous avons accès aux données des articulations du corps, il faut les envoyer à l'HoloLens. Il faut donc décider quelle donnée permet de caractériser au

mieux le comportement des points. À la première réflexion, il m'est apparu évident d'envoyer la position des points : il parait logique de mettre à jour en temps réel la position des points pour faire bouger le squelette de la même manière que le patient. Mais cette solution n'est pas fonctionnelle pour toutes les utilisations. Si l'objectif était d'afficher et visualiser à travers le casque uniquement les points alors cela ne poserait pas de problème. En effet, les points sont tous indépendants les uns des autres et donc en utilisant leurs coordonnées on garderait les bonnes proportions et distances entre eux et on obtiendrait un ensemble de points correspondant au squelette détecté par la Kinect. Le résultat obtenu et sa forme seraient cohérents. Cependant l'affichage des points n'est pas notre objectif. Ce que nous voulons faire est le contrôle d'un avatar, d'un modèle 3D pour qu'il se meuve comme le patient devant la caméra, et cela implique une réelle contrainte sur les proportions. Le modèle 3D du squelette est un assemblage articulé de points qui sont tous reliés entre eux. C'est-à-dire que lorsque nous tournons par exemple un bras du squelette autour de l'épaule, alors l'avant-bras et la main tournent également. Par conséquent, les distances entre les points sont fixes et ne peuvent être modifiées (cela reviendrait à vouloir modifier la longueur d'un os, ce n'est évidemment pas possible). On ne peut donc pas réaffecter en temps réel la position des points du modèle 3D car cela reviendrait exactement à vouloir modifier les distances entre les points. D'une part car les affectations ne se font pas en même temps, et bien qu'elles soient séparées d'un temps infiniment petit, cela implique de déplacer un point avant l'autre donc la distance entre les deux est forcément modifiée. D'autre part, comme pour tout capteur, la détection de la Kinect n'est pas parfaite et absolue et il y a toujours des petites variations dans les points détectés ce qui implique également des distance entre les points non constantes. Il faut donc trouver une donnée autre que la position des points à utiliser. C'est ainsi qu'en m'inspirant du projet de Michelle Ma [16] j'ai utilisé les rotations des points, qui sont caractérisés dans Unity par l'outil mathématique que sont les quaternions. En utilisant les rotations des points, comme le modèle 3D est articulé, on peut retranscrire sans problèmes toute la position du corps.

Il faut alors mettre ces données sous une forme qu'il est possible de transférer, à savoir un nombre ou une chaine de caractères. Je me suis directement dirigé vers l'envoi d'une chaine de caractères contenant les données de tous les points plutôt qu'envoyer les valeurs une par une comme un nombre flottant. En effet, on pourrait très bien imaginer n'envoyer qu'une donnée à la fois en utilisant par exemple un compteur variant de zéro à vingt-quatre (car on a vingt-cinq points) et revenant à zéro après vingt-quatre, qui permettrait de savoir à quel point affecter la donnée nouvellement reçue. Cependant cette solution présente un désavantage notable : si une donnée venait à se perdre et n'arrivait pas à l'HoloLens, alors le compteur et l'index du point à réaffecter seraient faussés et le squelette aurait une position aberrante. Le fait de regrouper toutes les données dans une seule et même chaine de caractères a donc l'avantage d'empêcher ce potentiel désordre. Et même dans le cas où des données venaient à ne pas arriver à destination, ce ne serait pas dérangeant car nous

mettons à jour les rotations des points en temps réel, donc cette donnée perdue a peu de valeur puisqu'elle appartient déjà au passé. Cette perte de donnée n'est donc tolérable que dans le cas de l'envoi d'une chaine contenant les informations de tous les points et non seulement d'un seul.

La chaine de caractères est mise à jour dans le script « *KinectDataManager »* introduit plus tôt. Ce script est relativement court et ne présente que trois fonctions : la fonction Start() qui va s'exécuter à l'activation du script (donc à l'activation de l'objet sur lequel le script est attaché) et qui initialise les références sur les objets de la scène nécessaires, la fonction Update() qui va s'exécuter à chaque image (« *frame* ») calculée par l'application et qui va appeler la troisième fonction, UpdateQuaternions(GameObject[] jointsArray) qui a pour rôle de construire la chaine de caractères à partir du tableau des points. Cette dernière fonction effectue un parcours basique du tableau. Pour chaque point, elle va récupérer le quaternion et lui ajouter une rotation de 180° autour des axes y et z. Ces rotations sont nécessaires pour annuler l'effet miroir qui est appliqué pour la superposition des points à la vidéo. Ensuite la fonction va ajouter les quatre coordonnées du quaternion à la chaine de caractères (un quaternion est caractérisé par quatre données qui sont notées x, y, z et w).

À la fin de la boucle de parcours du tableau, on obtient donc une chaine de caractères qui contient vingt-cinq quaternions. Les éléments d'un même quaternion sont séparés par le signe « % » et les quaternions sont quant à eux isolés par un « X ». La chaine envoyée a donc finalement la forme suivante : x0%y0%z0%w0Xx1%y1%z1%w1X….Xx24%y24%z24%w24. La fonction UpdateQuaternions est appelée dans la fonction Update seulement si un corps est détecté par la Kinect, autrement la chaine de caractères est directement rendue vide. Cela nous permettra plus tard de redonner sa position de départ au modèle 3D du squelette lorsque le patient n'est plus devant la caméra.

d) Interface

L'interface finale de l'application « Kinect App » fonctionnant sur un ordinateur et exploitant la Kinect se décompose en deux phases. La première est l'écran d'accueil de l'application que l'on obtient au lancement. Il contient le nom du logiciel dans le coin supérieur gauche ainsi qu'un bouton au centre. Ce bouton sert à véritablement lancer l'application en activant la caméra et la détection du corps, amenant ainsi à la seconde phase. La figure 17 montre ce qu'on obtient après avoir appuyé sur ce bouton. On retrouve le nom de l'application ainsi que deux boutons, plus discrets, dans le coin supérieur gauche. Le premier bouton permet d'afficher l'adresse IPv4 locale de l'ordinateur, afin de vérifier les réglages de l'application sur l'HoloLens et permettre la bonne connexion entre les deux appareils. Le second bouton est un bouton d'arrêt de

la détection qui va faire revenir l'application à l'écran d'accueil. Enfin, on aperçoit dans la figure 17 les points des articulations ainsi que les lignes représentant les membres du corps détecté. Ils pourraient être calculés sans obligatoirement être rendus visibles, mais leur affichage permet à l'utilisateur d'être bien sûr que le corps est détecté, ce qui n'est pas une mauvaise chose.

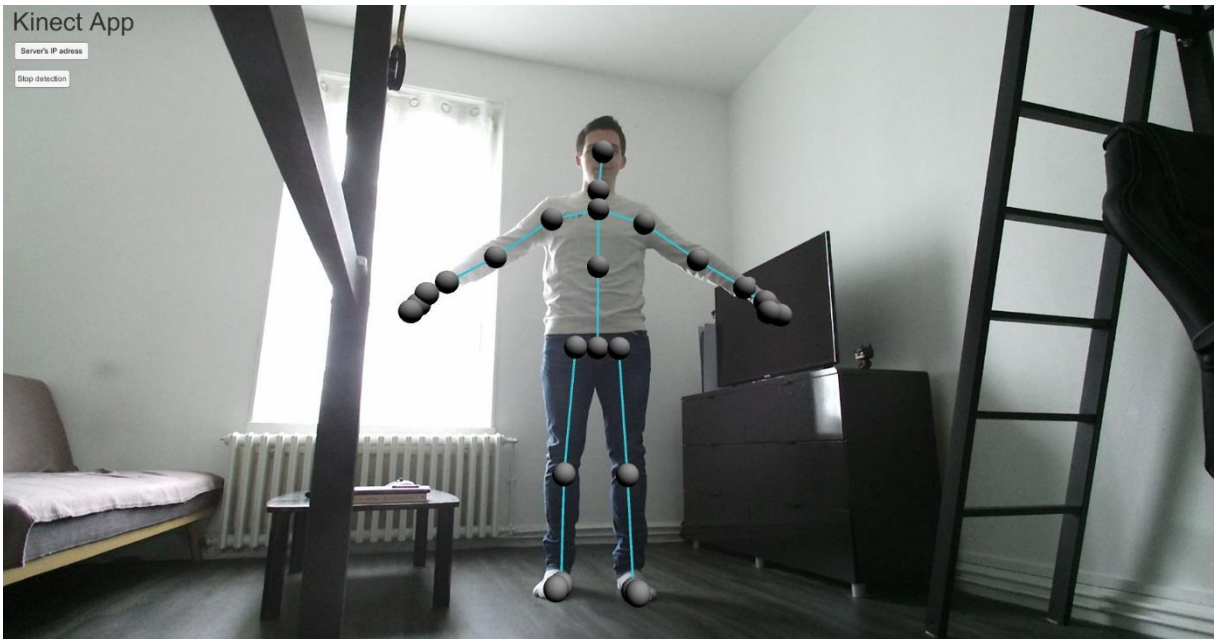

Figure 17: capture d'écran de la "Kinect App" en fonctionnement

4) Transfert des données par le réseau local

Avant de vouloir transférer les données des points par le réseau local, j'ai essayé de le faire par un lien USB entre l'ordinateur et l'HoloLens. L'idée était d'écrire continuellement avec l'ordinateur un fichier texte sur l'HoloLens contenant la chaine de caractères formée plus tôt et de la lire en temps réel sur le casque. Mais cette solution n'a pas abouti car je n'ai pas réussi à écrire directement sur l'HoloLens à partir de l'ordinateur. Lors des tests dans Unity, cela posait également des problèmes d'accès en lecture et écriture du fichier, ce n'était donc définitivement pas une bonne solution. Le transfert par le réseau permet aussi de ne pas se contraindre à être relié à l'ordinateur, cela est donc beaucoup plus pratique pour l'utilisateur.

a) Choix du protocole

Mes recherches sur l'implémentation d'un transfert de données par un réseau local dans Unity m'ont appris qu'il était possible de le faire selon deux protocoles de

transfert différents, qui sont le TCP et l'UDP, respectivement le « Transmission Control Protocol » et le « User Datagram Protocol ». Le choix est laissé aux développeurs afin qu'ils puissent utiliser la solution optimale pour leur jeu et leurs besoins. Une représentation schématique du fonctionnement de ces deux protocoles est proposée en la figure 18.

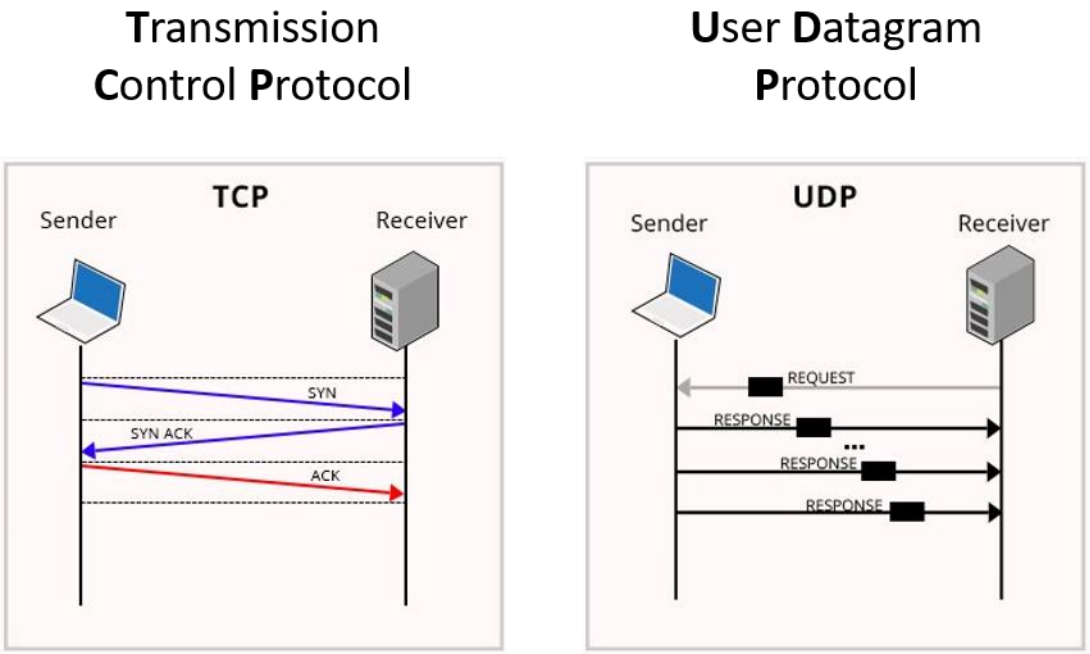

Figure 18: schéma représentant le fonctionnement des protocoles TCP et UDP

Avec le TCP, chaque donnée est envoyée quand l'expéditeur a reçu une confirmation de connexion avec le destinataire. Dans la partie gauche de la figure 18, les abréviations « SYN », « SYN ACK » et « ACK » désignent respectivement « synchronisation » pour une demande de connexion, « synchronisation acknowledgement » pour une confirmation de connexion et « acknowledgement » qui correspond à la donnée envoyée. Ainsi, avec le TCP, l'expéditeur demande un établissement de la connexion avec le destinataire, ce dernier lui confirme la connexion et la donnée est enfin envoyée. Le TCP est donc un protocole assez fiable car il garantit que les deux parties sont bien connectées avant le transfert de la donnée. Cette dernière a donc très peu de chance de ne pas arriver à destination. En contrepartie, comme il y a une vérification à chaque fois, ce protocole peut entraîner quelques lenteurs.

L'UDP ne fonctionne pas de la même manière : chaque donnée est envoyée indépendamment des autres. On peut alors transférer beaucoup d'informations à la suite mais le risque d'en perdre certaines est plus conséquent qu'avec le TCP. Cependant, comme expliqué plus tôt, dans notre cas une perte de données n'est pas

dérangeante car on met à jour en temps réel nos points, on reçoit donc constamment des nouvelles données et par conséquent cette donnée égarée du passé n'est plus intéressante. Le protocole UDP et donc plus rapide que le TCP et les risques qu'il implique ne sont pas néfastes dans notre cas. On retiendra donc ce protocole de transfert des données pour le projet.

b) Difficultés rencontrées

Le choix de la manière de procéder fait, il fallait alors implémenter et coder ce transfert de données, et c'est véritablement cette étape qui m'a posé le plus de difficultés pendant ce stage. Je n'avais aucune connaissance technique sur les réseaux et je me suis vite rendu compte que l'HoloLens allait me compliquer la tâche. En effet, du fait de l'architecture différente des deux appareils cibles, les applications pour le PC et pour l'HoloLens ne sont pas compilées de la même façon par Unity. L'application sur PC est une application de type « PC, Mac & Linux Standalone » alors que celle pour l'HoloLens est de type « Universal Windows Platform » (UWP). Cette différence apporte un problème majeur : ce qui fonctionne sur une plateforme ne fonctionne pas forcément sur l'autre et même, dans le cas de l'UWP, ce qui fonctionne lors des tests dans l'éditeur d'Unity ne fonctionne pas forcément dans l'application compilée et déployée dans le casque.

Vient s'ajouter à ces complications liées aux plateformes un autre problème : le système d'Unity permettant le multijoueur et donc les connexions par réseau, UNet, est voué à disparaitre et est déclaré obsolète par Unity. Il n'est alors pas logique d'utiliser un système qui ne sera plus supporté dans les années à venir et il fallait trouver une solution de substitution. Heureusement, Unity ne renie pas son système sans en proposer un autre. Mais ce successeur, Unity Transport, est toujours en développement donc il ne remplace pas encore toutes les fonctionnalités d'UNet. De plus, parce qu'il est un système récent, et que l'abandon d'UNet est progressif pour faciliter les transitions, Unity Transport n'est pour l'instant que très peu utilisé et on ne trouve pas beaucoup d'exemples sur son utilisation. Néanmoins, Unity met à disposition une série de codes [20] faisant la démonstration de quelques fonctionnalités de ce nouveau système.

c) Solution retenue

C'est un de ces exemples qui m'a finalement permis d'implémenter ce transfert de données entre l'ordinateur et l'HoloLens. L'un des six exemples proposés a pour fonction l'envoi d'un ping et la réception d'un pong : le client envoie un ping au serveur

qui répond à la réception avec un pong, et le temps entre l'envoi et la réception est calculé afin de déterminer la latence. Cet exemple se décompose en deux scripts, le premier, « ServerBehaviour », pour le serveur qui sera dans l'application sur PC et l'autre, « ClientBehaviour » qui sera sur l'HoloLens qui implémente le client. Cette connexion est établie assez simplement : le script sur l'ordinateur crée un serveur en le liant à un port spécifique (ici, c'est le port 9000 qui est arbitrairement choisi) et écoute les demandes de connexion. Ensuite lorsque la connexion est enregistrée le serveur va envoyer les données des points. De la même manière sur l'HoloLens, un client sera initialisé et lié au même port que le serveur, puis quand la connexion est établie il écoute le trafic sur le port et reçoit les données envoyées par le serveur.

L'exemple utilisait l'entier 1 pour représenter le ping et le pong. Il a donc juste fallu modifier la ligne qui déclarait la donnée à envoyer comme étant un entier en remplaçant la déclaration « int » par « string » pour pouvoir envoyer une chaine de caractères. Cette chaine est une variable ajoutée au script du serveur et est continuellement mise à jour par le script « *KinectDataManager* ». En effet, Unity permet la communication entre les scripts. Dans ce cas, le script « *KinectDataManager* » possède une référence sur l'objet « *serverBehaviour* » et a donc accès à la variable publique qu'est la chaine de caractères à envoyer.

Le temps calculé entre l'envoi et la réception de la donnée, généralement appelé « ping », pourrait paraître inintéressant désormais. Mais il permet néanmoins de vérifier que la connexion est bien opérationnelle : si le ping est nul alors le client et le serveur ne sont pas connectés alors que s'il est non nul, la connexion est bien présente. Cette donnée temporelle est donc réutilisée dans l'interface de l'application sur l'HoloLens pour informer l'utilisateur de l'état de la connexion.

5) Application sur HoloLens

L'application sur l'HoloLens est véritablement le cœur du projet. C'est avec elle que l'on va utiliser les données de la Kinect et proposer une série d'exercices dans le but d'étudier les capacités fonctionnelles du patient.

a) Débuts sur HoloLens

Le projet repose essentiellement sur l'HoloLens, mais avant de pouvoir commencer à travailler sur nos objectifs, il m'a fallu un temps d'adaptation et de découverte de l'appareil pour le comprendre et apprendre à le maitriser un minimum.

C'est après ces tests que j'ai pu réellement débuter l'implémentation de la restitution de l'avatar ainsi que des exercices. Pour apprendre et débuter la programmation d'applications sur HoloLens dans Unity, j'ai utilisé les tutoriels proposés par Microsoft [21]. Ils sont très clairs et permettent de faire nos premiers pas avec les outils mis à disposition pour exploiter le casque, à savoir le paquet « Mixed Reality ToolKit » (MRTK) [22]. Dans ces tutoriels, on nous apprend par exemple à configurer un projet Unity pour l'utilisation du casque, à positionner des objets holographiques dans la scène ou encore à utiliser les différents modèles de menus et panneaux pour établir notre interface utilisateur.

b) Architecture globale et interface de l'application

L'organisation des éléments composant la scène de l'application sur HoloLens est relativement similaire à celle utilisant la Kinect sur PC. On y retrouve donc :

- Deux objets nécessaires au bon fonctionnement du projet en réalité mixte sur HoloLens, ces objets ne sont jamais modifiés
- Un objet « UI Manager » qui est ici plus important que pour l'application Kinect car il y a beaucoup plus de boutons et de panneaux à contrôler
- Un objet « Joints Updater » qui a pour rôle de mettre à jour les rotations des points du squelette avec les informations reçues par le réseau
- Et enfin un objet « Client », qui implémente le côté client du transfert de données

L'UI Manager est l'objet central de l'architecture de l'application car c'est lui qui centralise toutes les interactions entre l'utilisateur et le logiciel. Il possède quatre objets enfants et a un script attaché. Ses objets enfants sont les quatre panneaux définissant l'interface, à savoir le panneau principal que l'on retrouve dès le lancement de l'application, le panneau sur lequel se trouve le squelette, le panneau des paramètres, ainsi que le panneau des exercices. Les deux premiers sont toujours affichés tandis que les deux autres peuvent être masqués et désactivés par l'utilisateur, ils ne sont pas visibles au lancement de l'application. Le panneau des paramètres est situé à droite du panneau principal et le panneau des exercices se trouve sous le panneau principal, leur position est calculée en fonction de ce menu principal . Ces quatre panneaux sont visibles sur la figure 19. Le panneau principal possède trois boutons permettant de se connecter à l'ordinateur, d'afficher le menu des exercices et d'accéder aux paramètres. Le menu des paramètres contient un bouton, un champ de texte ainsi que deux curseurs. Le bouton permet d'afficher un clavier virtuel permettant de modifier l'adresse IPv4 à laquelle le casque va tenter de se connecter, cette adresse est visible dans le champ de texte. Enfin, les deux curseurs permettent de changer l'échelle du squelette (le grandir ou le rétrécir, il n'est pas à taille humaine au démarrage pour faciliter son placement) et de régler un décalage du modèle 3D par

rapport à son panneau parent, le long de son axe vertical. Pour un meilleur visuel de cette interface et de ses différents éléments, le lecteur pourra se reporter à l'annexe 2 page 48.

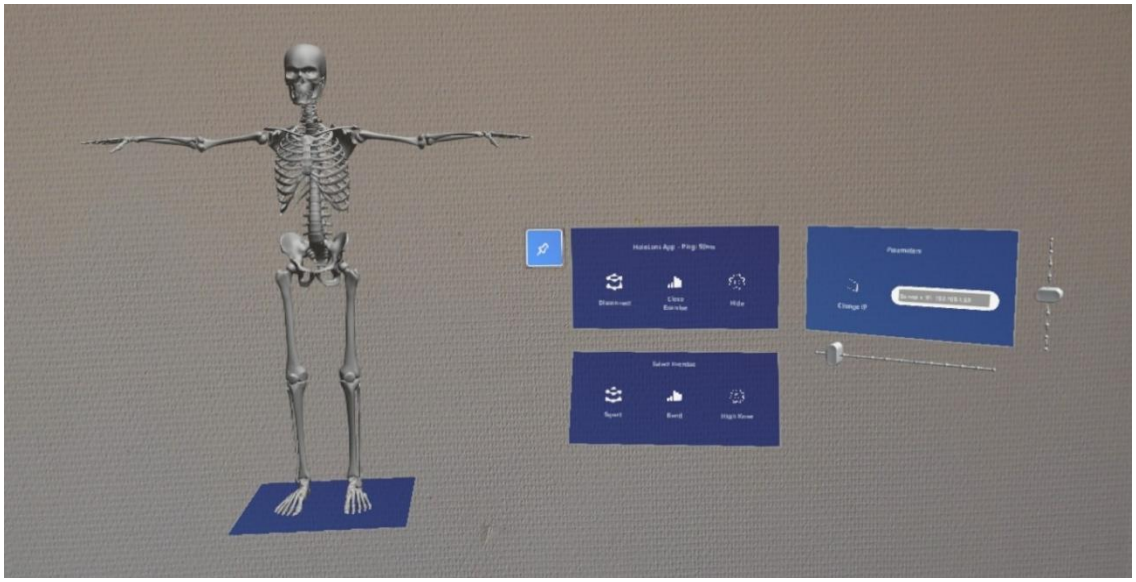

Figure 19: les quatre panneaux de l'interface de l'HoloLens App

Le script attaché à l'UI Manager, intitulé « UIManager » est le script qui régit toutes les interactions de l'utilisateur avec cette interface. Pour cela, il contient des fonctions publiques qui seront appelées par les boutons lorsqu'ils seront pressés. Par exemple, lorsque l'on appuie sur le bouton « Parameters » du panneau principal, la fonction « ShowHideParameters() » du script va être appelée. Cette fonction va tester si le panneau des paramètres est actuellement activé : s'il ne l'est pas alors elle va le rendre visible, sinon elle va le désactiver. Tous les boutons fonctionnent de cette même façon : à leur pression une fonction qui active ou désactive des éléments est automatiquement appelée. Ce script calcul aussi la position des panneaux en fonction du panneau principal. Cela aurait pu se faire de manière plus simple et automatique en rendant ces panneaux « enfants » du panneau principal mais cela complique et complexifie légèrement la lisibilité de la hiérarchie.

c) Restitution des données dans le casque : contrôle de l'avatar

L'UI Manager et les interactions qu'il définit sont très importants, mais sans le modèle 3D du squelette en mouvement comme le patient devant la caméra, l'application n'a pas de réel intérêt. C'est là qu'interviennent les objets « Client » et « Joints Updater ». Le premier récupère les données de rotation des points sur le réseau et il met à jour la chaine de caractères qui sera déchiffrée par le second. Le Client a deux scripts attachés, l'un met en place à proprement parler la connexion avec le serveur tandis que l'autre a pour fonction de convertir l'adresse IP renseignée par l'utilisateur sous la forme d'une chaine de caractères en une donnée acceptée et

reconnue par les fonctions et méthodes établissant la connexion réseau. Le script « Client Behaviour » qui implémente la connexion est issu du code exemple proposé par Unity [20]. J'y ai seulement ajouté une référence sur l'objet JointsUpdater de la scène afin d'avoir accès à la chaine de caractères « quaternionsString » dont sont extraites les données des points. Ainsi quand on reçoit une chaine par le serveur, on l'affecte directement à « quaternionsString » qui sera décomposée par l'objet Joints Updater. Pour cela, le script « JointsUpdater » utilise une boucle similaire à celle effectuée sur l'ordinateur pour créer la chaine de caractères. On commence par séparer la chaine reçue à chaque signe « X » pour avoir un tableau de vingt-cinq chaines de caractères, représentant les informations de tous les quaternions. Ensuite, on parcourt chacun de ces éléments et on décompose encore une fois la chaine à chaque « % » (on rappelle que la chaine reçue est sous la forme x0%y0%z0%w0Xx1%y1%z1%w1X….Xx24%y24%z24%w24). On affecte ensuite à la rotation du point correspondant à l'indice de la boucle les valeurs obtenues, converties au préalable en nombres flottants.

d) Règles des exercices et des notes

Les exercices et l'étude des capacités fonctionnelles du patients sont au cœur du projet. Le démonstrateur contient trois exercices : le squat, la montée de genou et la courbure du dos. Ces exercices ont été choisis car ils sont relativement simples à effectuer et à implémenter mais surtout car malgré leur simplicité d'exécution, ils peuvent permettre de déceler des douleurs et difficultés chez le patient car ils impliquent beaucoup de muscles. Pour chaque exercice, l'utilisateur est invité à choisir un nombre de répétitions à faire entre cinq et dix puis il doit faire les mouvements. À la fin et selon le meilleur résultat performé, une note (A, B ou C) est attribuée. C'est l'objet « Exercise Manager » qui implémente cela. Le script « ExerciseManager » qui lui est attaché regroupe les méthodes et valeurs clés qui seront utilisées par l'UI Manager pour afficher les résultats des exercices ainsi que la note attribuée. Chaque exercice fonctionne de la même manière : une condition sur la position de certains points ou sur des angles formés par le corps permet de définir le mouvement de l'exercice. La valeur définissant la condition est choisie arbitrairement. Pour le squat, c'est l'angle au genou entre la cuisse et le tibia qui définit le mouvement. Ce dernier doit être inférieur à 140° pour que l'on considère être en position de squat. Pour la courbure du dos, il s'agit en fait de rapprocher le plus possible les mains du sol sans plier les genoux, donc la distance entre le sol et les mains est un bon indicateur. On choisit d'accorder le mouvement à partir d'une distance entre les mains et le sol inférieure à 70 centimètres. Enfin pour la montée de genou, c'est l'angle formé entre la jambe non pliée et la cuisse de la jambe opposée et levée qui définit le mouvement. On considère le genou levé pour tous les angles supérieurs à 50°. À chaque instant du mouvement, on compare la valeur caractéristique à la meilleure déjà relevée et on la remplace si la performance est améliorée. À la fin de la série de répétitions de l'exercice, l'application propose une

note en fonction de la meilleure valeur de l'angle ou de la distance effectuée par le patient. Il y a trois notes possibles : A, B ou C. Comme pour déterminer les valeurs permettant de définir les mouvements, les valeurs « paliers » des notes ont été trouvées empiriquement lors des tests que j'ai réalisé en effectuant moi-même les mouvements. Pour chacun des exercices il fallait trouver deux paliers, celui définissant la limite entre les notes B et C et celui entre les notes A et B. Ces différents paliers sont représentés dans les figures 20, 21 et 22.

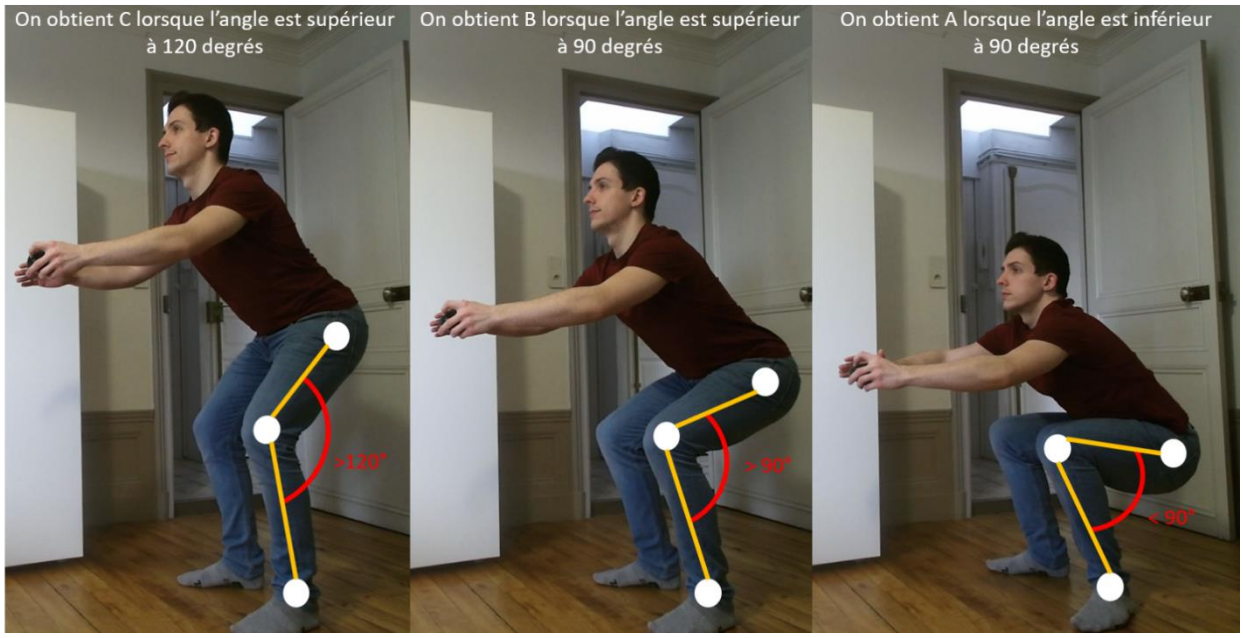

Figure 20: représentation des paliers d'attribution de la note pour le squat

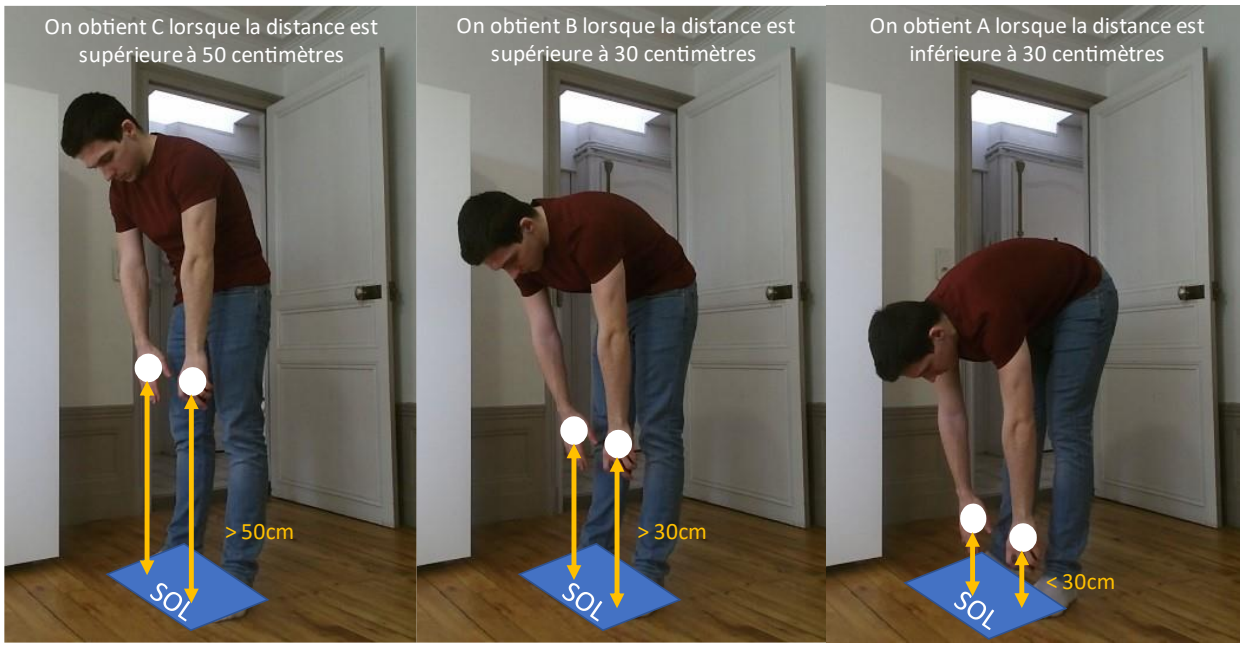

Figure 21: : représentation des paliers d'attribution de la note pour la courbure du dos

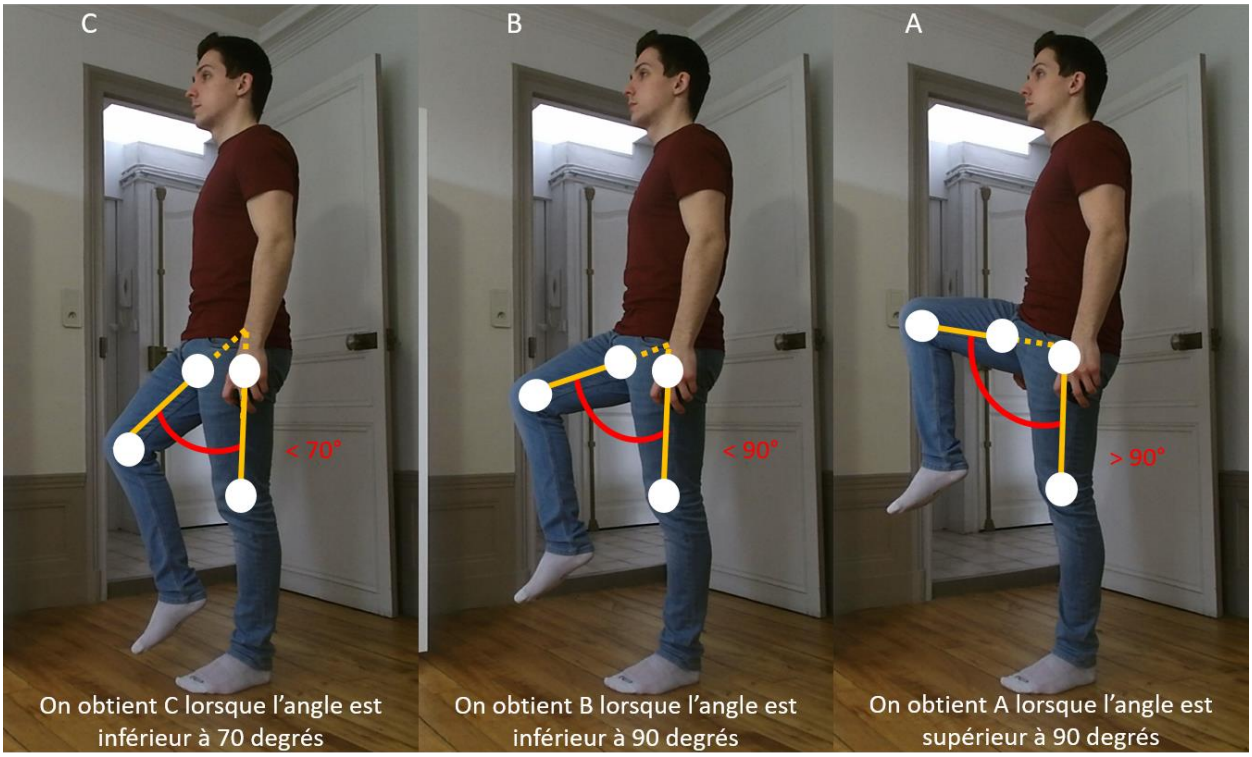

Figure 22: représentation des paliers d'attribution de la note pour la montée de genou

Pour attribuer ces notes, on étudie à la fin de la série la meilleure performance : si elle est inférieure à la note limite pour avoir B, alors le patient obtient C ; si elle est supérieure à la précédente limite mais inférieure à la limite pour avoir A, alors le patient obtient B ; enfin, si la meilleure performance du patient est supérieure aux deux paliers, alors il obtient A (cf. figure 23).

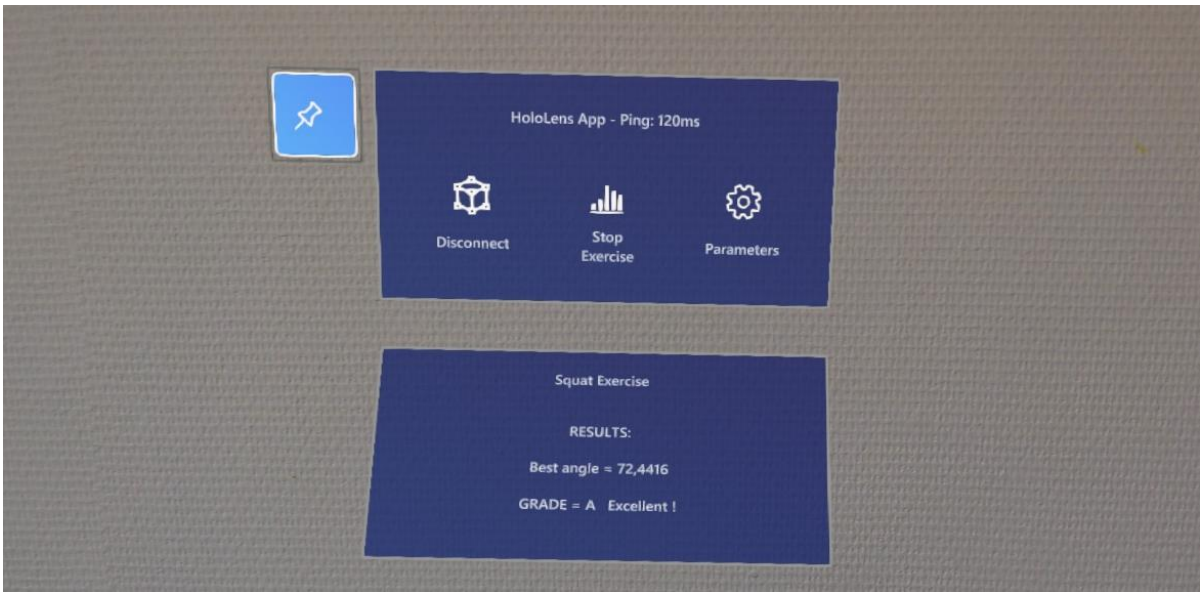

Figure 23: Exemple de l'affichage d'une note A pour l'exercice du squat

e) Fonctionnement et architecture des exercices

Pour implémenter les fonctions des exercices, j'ai utilisé la fonction Update() de Unity qui est automatiquement appelée à chaque frame, c'est-à-dire à chaque image calculée et produite par l'application. Ainsi, c'est dans Update() que s'effectue l'appel à l'origine de l'exercice. Un exercice est lancé lorsque l'utilisateur a choisi l'exercice qu'il souhaite faire ainsi que le nombre de répétitions à effectuer. Ainsi apparait la condition à placer dans Update : si le nombre de répétitions est nul, alors il ne sert à rien d'appeler la fonction de l'exercice. Il y a deux étapes amenant à la validation de cette condition : l'utilisateur clique sur le bouton correspondant à l'exercice qu'il souhaite faire puis il presse le bouton de choix du nombre de répétitions. La première étape va assigner une valeur précise à une variable appelée « exerciseId » (nulle jusqu'à ce point) : le bouton du premier exercice va affecter la valeur 1 à exerciseId, le second bouton la valeur 2 etc... Cela est réalisé par une fonction qui est appelée automatiquement quand le bouton est pressé. Il y en a une par bouton et chacune d'entre elles procède à cette affectation ainsi qu'à d'autres en lien avec les paliers des notes de l'exercice sélectionné. Le premier choix fait, il faut désormais choisir le nombre de répétitions à faire. Sur le même principe que pour les exercices, les boutons du choix des répétitions vont affecter une valeur entière à une variable « exerciseReps », qui était nulle jusqu'à cette étape. Une fois que le nombre de répétitions est choisi, la condition dans la fonction Update citée plus haut est vérifiée et la fonction Exercise() est appelée. Cette fonction va elle-même exécuter une autre méthode en fonction de l'identifiant de l'exercice : l'identifiant 1 entraîne le squat, le 2 appelle la courbure du dos et le 3 la montée de genou. Ce sont enfin ces fonctions qui implémentent réellement l'affichage des données des exercices ainsi que la gestion du compteur des répétitions et du meilleur résultat performé. La fonction Update() s'exécutant continuellement, il fallait trouver un moyen de connaitre pour chaque frame l'état du mouvement à la frame précédente pour ne pas incrémenter à chaque fois le compteur des répétitions. Pour cela j'ai introduit une variable booléenne nommée « lastFrameSituation ». Ainsi, ce booléen sera faux lorsque les conditions du mouvement ne seront pas respectées et il sera vrai dès lors que ces conditions seront vérifiées. Pour chaque exercice, le raisonnement est le même : s'il nous reste des répétitions à faire et si le mouvement est exécuté et qu'il ne l'était pas à la frame précédente (c'est-à-dire que le booléen est faux), alors on incrémente le compteur et le booléen devient vrai. À l'inverse, si le mouvement n'est pas exécuté, alors le booléen redevient faux. Enfin, s'il ne reste plus de répétitions à faire, alors la série est terminée et on affiche la note. Un schéma récapitulant ces multiples appels et conditions pour l'exemple du squat pour cinq répétitions est appréciable en la figure 24. De plus, l'annexe 3 page 52 propose un aperçu des codes correspondant à ces fonctions.

Quand un exercice est terminé, le seul choix possible pour en démarrer un nouveau est d'appuyer sur un bouton qui ramène l'utilisateur sur le panneau du choix

de l'exercice. La pression de ce bouton, en plus de procéder à ce retour à l'interface, réinitialise toutes les variables utilisées précédemment : exerciseId, exerciseReps et counter redeviennent nulles tandis que le booléen lastFrameSituation revient à faux. Ainsi, la condition sur la non-nullité de exerciseReps dans la fonction Update n'est à nouveau plus vérifiée et le processus de lancement d'un exercice est en attente que l'utilisateur presse à nouveau les deux boutons de choix.

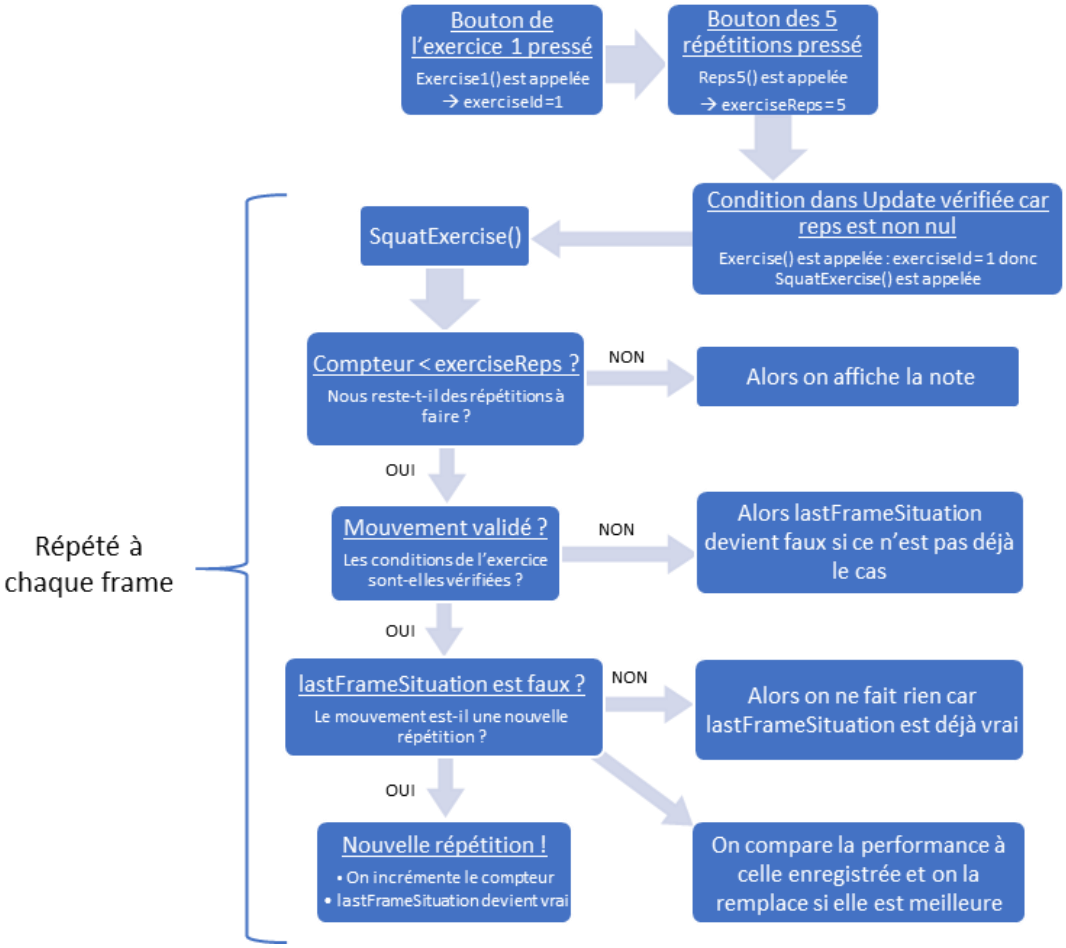

Figure 24: cheminement logique des exercices, ici pour celui du squat

Cette façon d'implémenter les exercices peut paraître assez complexe mais j'ai préféré procéder de cette manière pour diviser les tâches et ne pas faire une seule et énorme fonction par exercice. Cela permet aussi de factoriser une partie du code, comme l'affichage ou la désactivation des boutons et le calcul de la note. Cette architecture est également principalement nécessaire du fait de l'utilisation de la fonction Update qui nous oblige à ne pas utiliser des variables locales aux fonctions qui seraient réinitialisées à chaque frame. Il fallait donc trouver un moyen de garder une continuité là où on recommence perpétuellement. Ici, toutes les variables utilisées sont des objets du script, peuvent être utilisés par toutes les méthodes de ce dernier et ne sont donc pas réinitialisées à chaque frame.

Les fonctions Exercise1(), Exercise2()… Exercisei() appelées par le bouton correspondant et qui affectent les bonnes valeurs aux variables sont préférables à une seule fonction qui ferait les mêmes opérations en fonction du bouton qui l'appelle car elles ne nécessitent aucun test pour déterminer le bouton qui les a appelé. D'autant plus que ce test devrait certainement se faire sur le nom du bouton, ce qui n'est pas très optimal car un nom peut vite être amené à être modifié. Cela permet donc d'avoir trois courtes fonctions plutôt qu'une longue qui devrait tester l'identité du bouton pour effectuer les bonnes opérations. Il en est exactement de même pour les deux fonctions reps5() et reps10() qui affectent le nombre de répétitions à faire à exerciseReps. La fonction Exercise() qui va appeler la fonction correspondant à l'exercice sélectionné est également préférable à une fonction regroupant les trois exercices pour un soucis de lisibilité et de clarté du code. Pour les mêmes raisons, il est préférable d'utiliser cette fonction intermédiaire plutôt que tester dans Update la valeur de l'identifiant de l'exercice pour exécuter la bonne fonction. Enfin, je ne pense pas que cette implémentation contraint l'ajout de nouveaux exercices. Pour en ajouter, il suffit de créer une fonction pour le nouveau bouton qui va affecter l'identifiant à exerciseId et les valeurs des paliers des notes, de faire un nouveau cas sur exerciseId dans la fonction Exercise() puis d'écrire une fonction qui va s'occuper du cheminement présenté précédemment et de l'affichage des données de l'exercice.

Conclusion

J'ai vraiment trouvé tout très intéressant dans le projet : le sujet qui tend à de nobles objectifs car il vise à améliorer la vie des patients ainsi que l'utilisation de la réalité mixte à travers l'HoloLens qui est une technologie fascinante. Si le projet vient à se concrétiser et à produire la plateforme dont il a l'ambition, ce sera une fierté d'y avoir contribué. Cependant, comme j'ai participé au tout début du projet il est évident qu'à l'avenir une grande partie de ce que j'ai fait sera modifiée, remplacée, améliorée. Mais cela n'est en rien un défaut ou un point négatif du stage, car la découverte et l'exploration de cette technologie ainsi que la pose de fondations qui, je l'espère, serviront aux futurs développeurs du projet ont été très excitantes et m'ont énormément appris.

J'ai été intéressé et intrigué par ce stage dès que je l'ai aperçu dans l'espace de proposition de sujets que met l'UTC à disposition pour les étudiants en recherche de stage. Souhaitant travailler après mes études dans l'industrie des jeux-vidéo mais n'ayant aucune expérience dans l'utilisation des outils requis, ce stage me paraissait être une opportunité fantastique d'apprendre à utiliser Unity, un moteur de jeu qui est l'un des plus utilisés dans ce domaine. De plus, le fait de travailler avec une technologie incroyable comme l'HoloLens m'a aussi beaucoup attiré vers le projet. Je connaissais le produit et avais vu les vidéos de présentation de Microsoft mais je ne l'avais jamais testé. C'est une réelle chance d'avoir pu l'utiliser car c'est un appareil difficile à obtenir, réservé aux professionnels et laboratoires et distribué très précisément par son fabricant. Finalement, les promesses que je percevais de cette opportunité qu'était ce stage se sont révélées être bien réelles. Je suis à la sortie du stage relativement à l'aise avec Unity et je peux enfin considérer le fait de me lancer à la réalisation de projets personnels de jeux.

Cela n'a pas toujours été très simple de travailler seul sur le projet, car je n'avais que très peu de connaissances sur Unity et le langage C# qu'il utilise, sur la création d'interfaces utilisateur et que mes connaissances en programmation objet étaient basiques. Par conséquent, l'architecture globale des codes que j'ai produit ne correspond très certainement pas à un standard de production et limite aussi probablement la capacité d'expansion du code et l'ajout de nouvelles fonctionnalités. Cependant, bien qu'au final mon travail n'a pas réellement porté sur la création de jeux sérieux, je suis très satisfait de ce que j'ai pu faire et produire en seulement trois mois en possession de l'HoloLens.

Les deux premiers mois sans HoloLens ont également été assez durs et démotivants. Certes je faisais des choses en lien avec le projet comme découvrir la Kinect et me documenter sur la réalité augmentée, mais sans le casque je ne trouvais pas de réel but ni de valeur dans ce que je faisais. Deux mois est une période quand

même très longue, d'autant plus quand on attend l'objet qui est censé être la pierre angulaire de son travail. Heureusement, l'arrivée du casque fut un grand soulagement et j'ai pris énormément de plaisir à partir de là. Finalement, la majeure partie de ce qui est présenté ici a été produite en trois mois et je suis très content de ces résultats.

Pour l'avenir du projet, le champ des possibles est immense et ce qui a été créé pendant le stage pose des bases, en particulier quant à la communication entre les deux appareils, et constitue un point d'entrée obligatoire vers les objectifs finaux et à long terme du projet. Il est, à la fin du stage, à l'état de démonstrateur et de prototype. Les tests et exercices sont implémentés de façon très simple et les plafonds définissant les mouvements et les notes ne sont qu'empiriques et arbitraires et donc par conséquent pas très précis et fidèles à la réalité. Le projet prendra tout son sens lors de l'implémentation de réelles modélisations biomécaniques permettant d'étudier beaucoup plus précisément les mouvements du patients. On peut par exemple penser à une superposition du squelette sur le patient et une mise à l'échelle automatique en fonction de la taille et la corpulence du sujet, à la prise en compte du temps d'exécution du mouvement dans la notation ou encore à l'utilisation de modèles précis des muscles ou des articulations.

Mais je pense que le domaine dans lequel le projet est le plus prometteur est celui de l'évaluation des séquelles cognitives. Cela est certes plus compliqué à étudier et implémenter, car ne reposant pas sur des mouvements mécaniques plus facilement percevables, mais ce que permet de faire la technologie de l'HoloLens est incroyable et ouvre un nombre incalculable de portes. On parle tout de même de l'affichage, ainsi que l'interaction avec des hologrammes, l'une des choses les plus fantasmées par les œuvres de science-fiction. Pour le futur du projet, on peut imaginer une infinité d'applications. Des exercices sur la respiration dans le cas de troubles respiratoires en passant par des tests sur l'oreille interne et l'équilibre sans oublier l'étude de la mémoire, du goût, de l'ouïe ou de l'odorat ne sont que des exemples de domaines cognitifs que pourrait explorer l'application. Couplé à des capteurs de pouls, de tension, de sudation ou de chaleur posés sur le patient, ce projet pourrait, je pense, réellement révolutionner l'établissement de diagnostics et la rééducation.

La réalité virtuelle est en pleine expansion dans les jeux-vidéo depuis quelques années et quand les technologies des casques seront encore plus perfectionnées, notamment en termes de résolution d'affichage, elle deviendra un véritable standard et porte-étendard de l'industrie des jeux-vidéo. J'en suis convaincu. À l'aube des lunettes de réalité augmentée et mixte comme l'HoloLens, je suis également convaincu que dans le futur la réalité augmentée ne se contentera plus des « petites » applications sur smartphone et s'imposera également comme un « must » vidéoludique, mais également médical et technique. Et cela sera forcément très intéressant d'avoir travaillé et contribué aux balbutiements et à l'essor de cette technologie fascinante.

Bibliographie

[1] BMBI, «Historique du laboratoire BMBI,» [En ligne]. Available: https://bmbi.utc.fr/presentation/historique.html.

[2] BMBI, «Equipes de recherche,» [En ligne]. Available: https://bmbi.utc.fr/recherche/equipes-de-recherche.html.

[3] BMBI, «Axes de recherche,» [En ligne]. Available: https://bmbi.utc.fr/presentation/axes-de-recherche.html.

[4] BMBI, «C2MUST,» [En ligne]. Available: https://bmbi.utc.fr/recherche/equipes-de-recherche/caracterisation-et-modelisation-personnalisee-du-systeme-musculo-squelettique-c2must.html.

[5] M.-C. Ho Ba Tho et T. T. Dao, «Knowledge Extraction From Medical Imaging for Advanced Patient-Specific Musculoskeletal Models,» 2019. [En ligne]. Available: https://www.sciencedirect.com/science/article/pii/B9780128012383999355?via%3Dihub.

[6] Microsoft, «Kinect pour Windows,» [En ligne]. Available: https://developer.microsoft.com/fr-fr/windows/kinect/.

[7] R. Filkov, «Kinect v2 Examples with MS-SDK,» 1 Août 2014. [En ligne]. Available: https://rfilkov.com/2014/08/01/kinect-v2-with-ms-sdk/.

[8] R. Filkov, «K2-asset documentation,» [En ligne]. Available: https://ratemt.com/k2docs/Introduction.html.

[9] Microsoft, «Microsoft HoloLens 2: Partner Spotlight with VisualLive and NOX Innovations,» 21 Septembre 2020. [En ligne]. Available: https://www.youtube.com/watch?v=ItGmB2tQyY8.

[10] Microsoft, «Microsoft HoloLens 2: Partner Spotlight with Bentley,» 24 Février 2019. [En ligne]. Available: https://www.youtube.com/watch?v=pHjjqKLFMXM&t=17s.

[11] NCC AB, «Microsoft Hololens - examples of use in building maintenance,» 13 Janvier 2017. [En ligne]. Available: https://www.youtube.com/watch?v=XlkSV50bYyU.

[12] Microsoft, «Microsoft HoloLens 2: Partner Spotlight with Philips,» 24 Février 2019. [En ligne]. Available: https://www.youtube.com/watch?v=loGxO3L7rFE.

[13] Microsoft, «Cardiolens. "Pulse and Vital Sign Measurement in Mixed Reality using a HoloLens",» 15 Février 2018. [En ligne]. Available: https://www.youtube.com/watch?v=OyheMrkUiIY&feature=emb_title.

[14] Case Western Reserve University, «Transforming Medical Education with Microsoft HoloLens,» 8 Juin 2016. [En ligne]. Available: https://www.youtube.com/watch?v=h4M6BTYRlKQ.

[15] M. Kowalski, «LiveScan3D-HoloLens,» [En ligne]. Available: https://github.com/MarekKowalski/LiveScan3D-Hololens.

[16] M. Ma, «HoloLens-Kinect project,» [En ligne]. Available: https://github.com/michell3/Hololens-Kinect.

[17] Microsoft, "Building a Telepresence App with HoloLens and Kinect," 17 Avril 2017. [Online]. Available: https://blogs.windows.com/windowsdeveloper/2017/04/18/building-telepresence-app-hololens-kinect/.

[18] Microsoft, «Body Part Recognition and the Development of Kinect,» 21 Juin 2016. [En ligne]. Available: https://www.youtube.com/watch?v=QPYf6pXe_4Q.

[19] Microsoft, «Real-Time Human Pose Recognition in Parts from Single Depth Images,» [En ligne]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf.

[20] Unity Technologies, «Multiplayer samples,» [En ligne]. Available: https://github.com/Unity-Technologies/multiplayer.

[21] Microsoft, «Présentation des tutoriels MRTK,» 1 Juillet 2020. [En ligne]. Available: https://docs.microsoft.com/fr-fr/windows/mixed-reality/develop/unity/tutorials/mr-learning-base-01.

[22] Microsoft, «What is the Mixed Reality Toolkit,» [En ligne]. Available: https://microsoft.github.io/MixedRealityToolkit-Unity/README.html.

Table des matières

Table des illustrations

Annexes

Annexe 1 – Captures d'écran des hiérarchies des deux applications

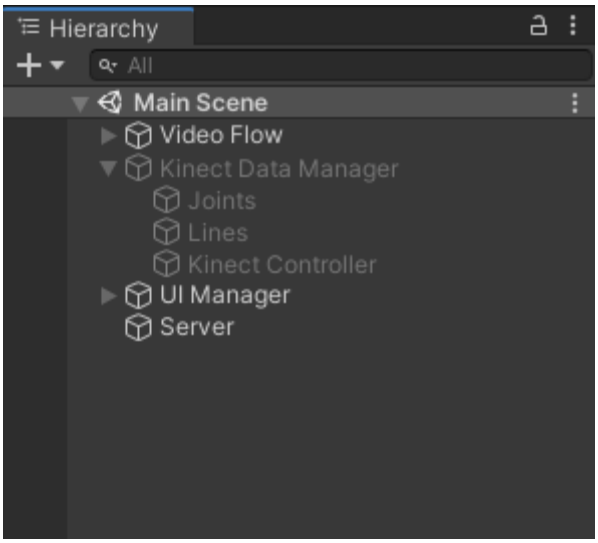

Figure 25: hiérarchie de la Kinect App

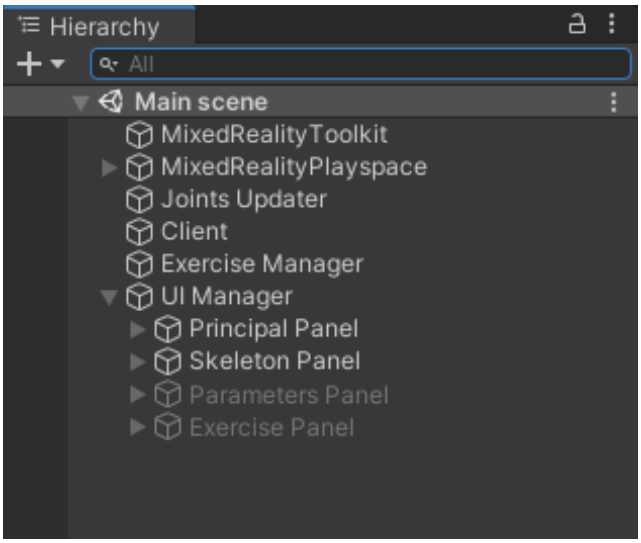

Figure 26: hiérarchie de l'HoloLens App

Les éléments qui sont situés sous un autre objet et légèrement décalés sont appelés « enfants » de cet objet. Par exemple, Joints est un enfant de Kinect Data Manager, tout comme Principal Panel est un enfant de UI Manager.

Annexe 2 – Photos de l'interface de l'HoloLens App

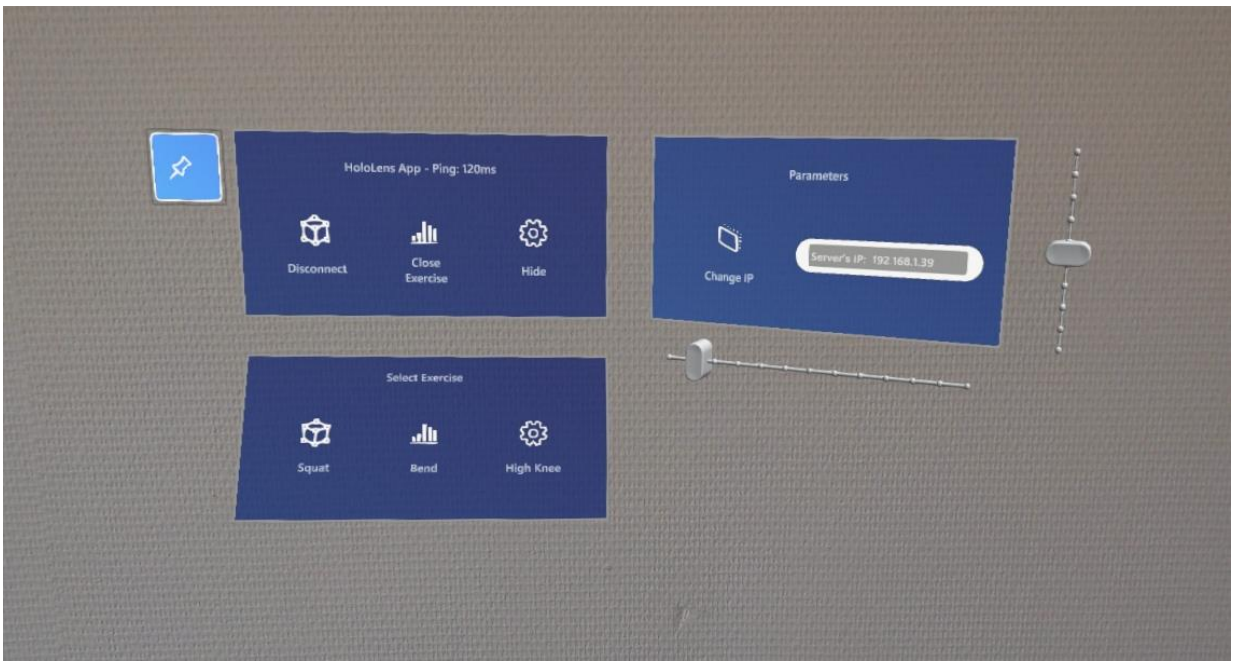

Figure 27: photo plus précise des trois panneaux textuels de l'interface

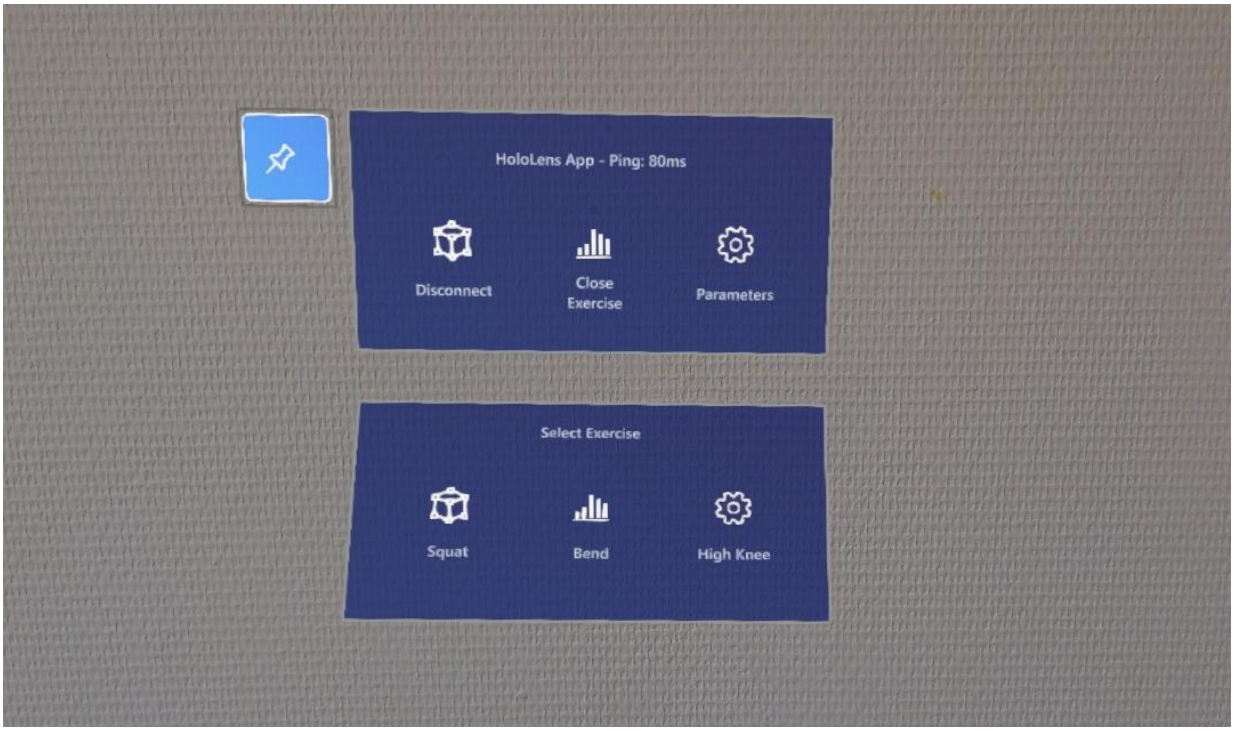

Figure 28: photo du panneau proposant le choix des exercices

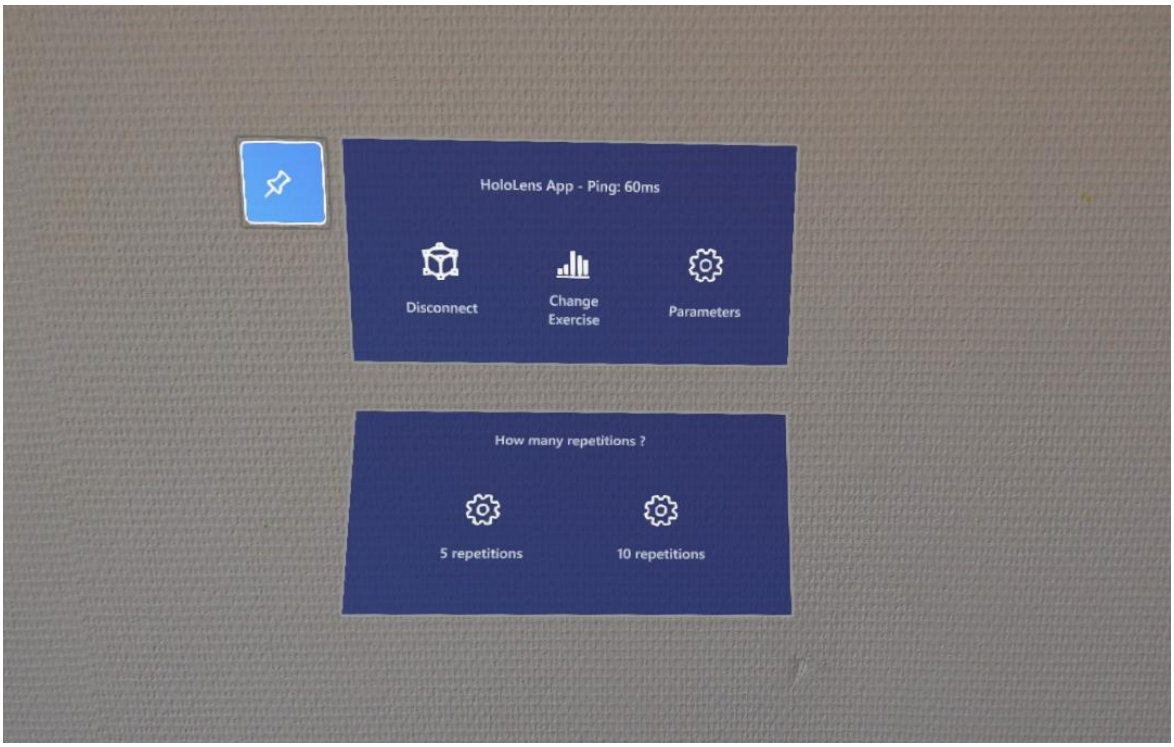

Figure 29: photo du panneau proposant le choix du nombre de répétitions de l'exercice choisi

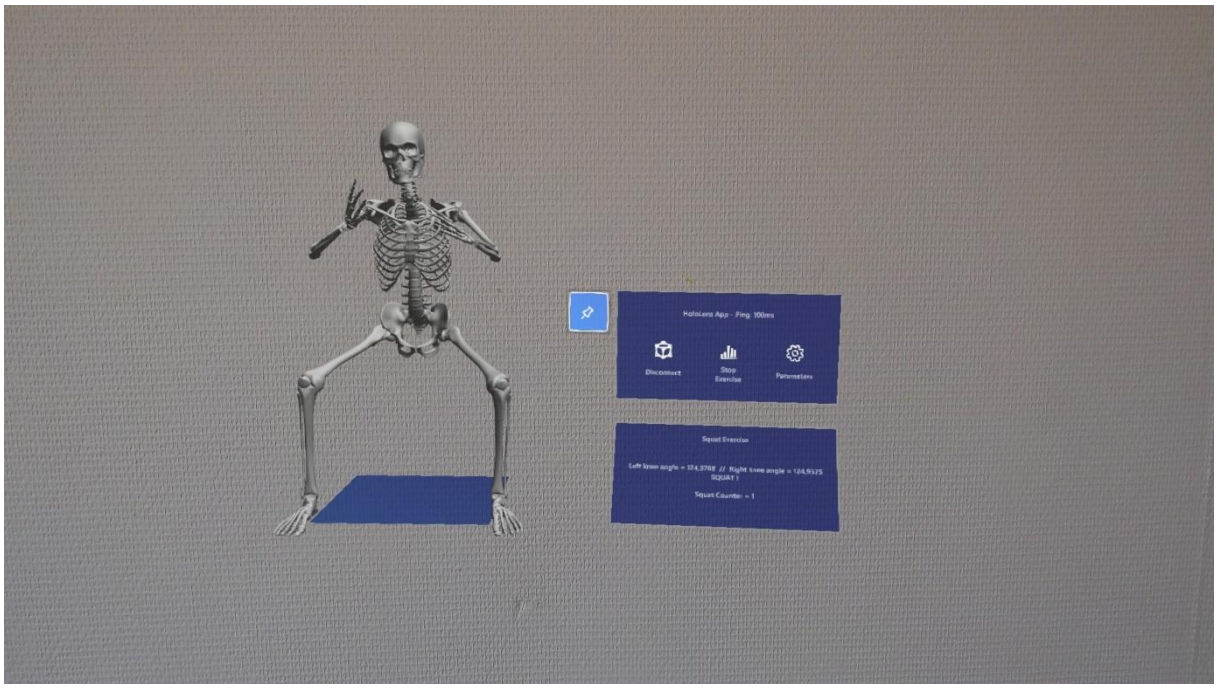

Figure 30: photo de l'interface lors de l'exercice du squat

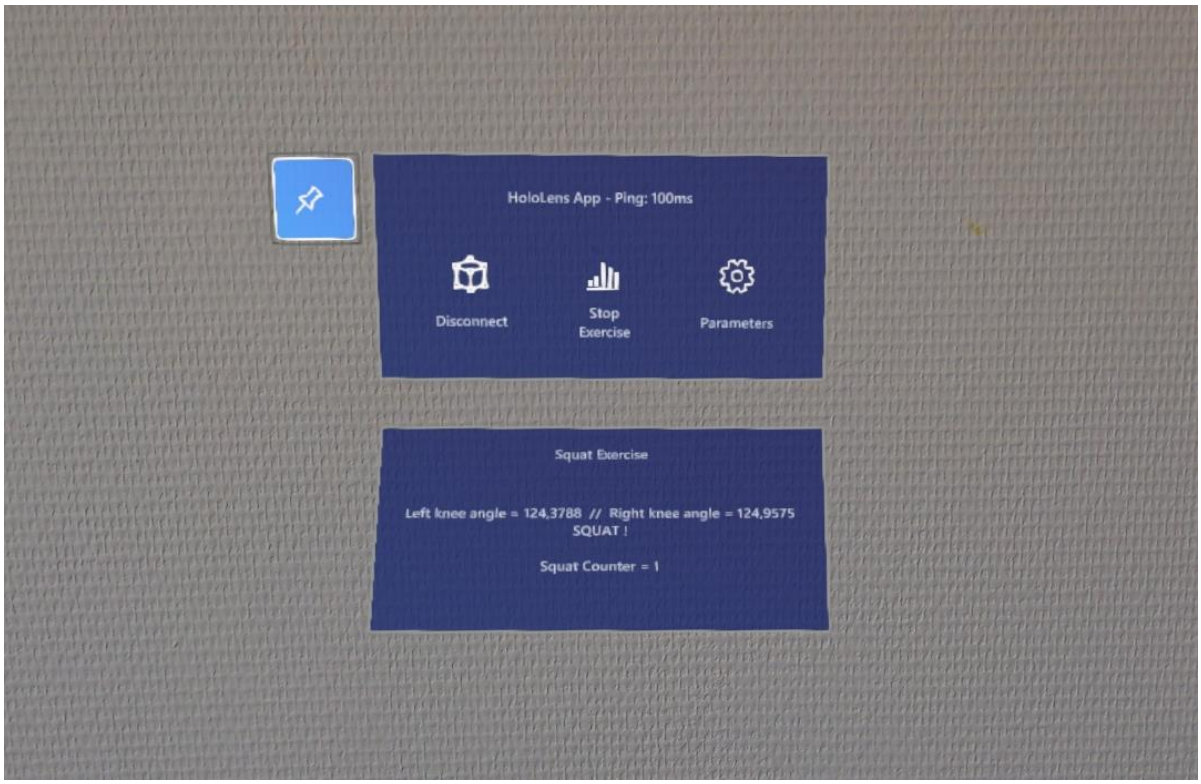

Figure 31: photo plus précise du panneau d'exercice lors du squat

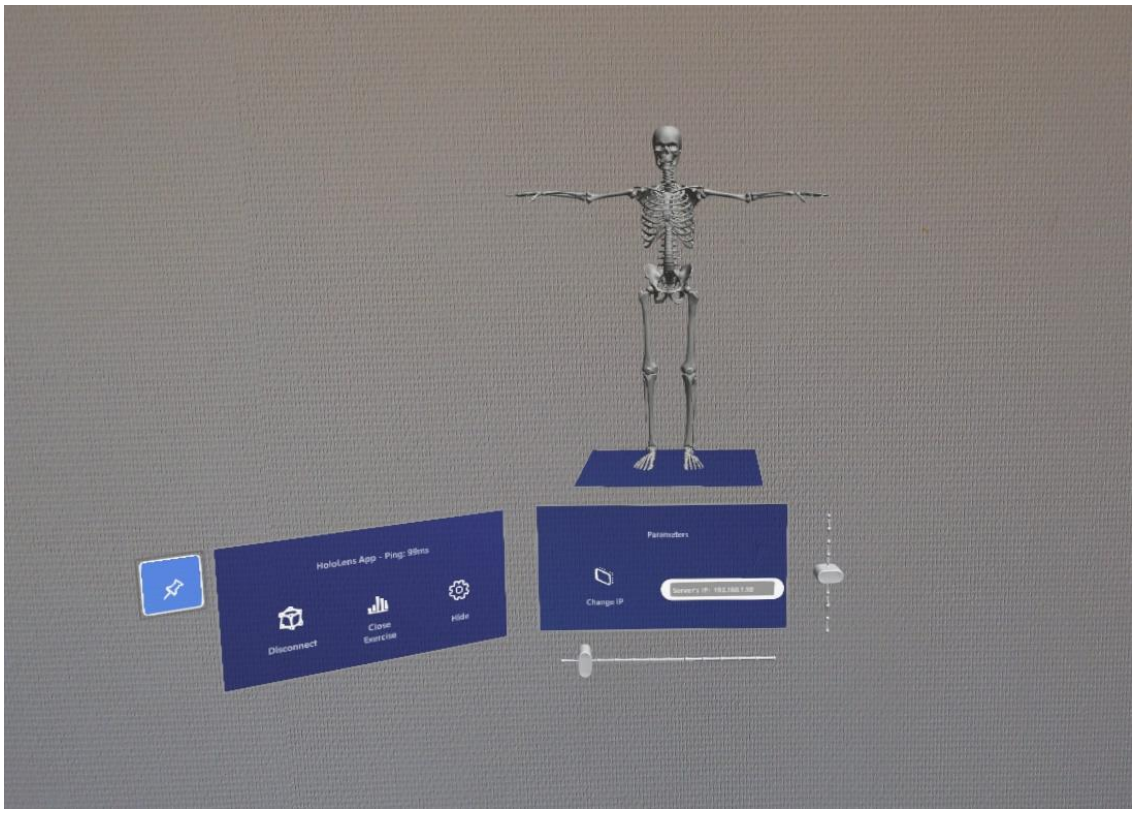

Figure 32: photo montrant la taille du squelette lorsque la barre réglable inférieure est proche de son origine

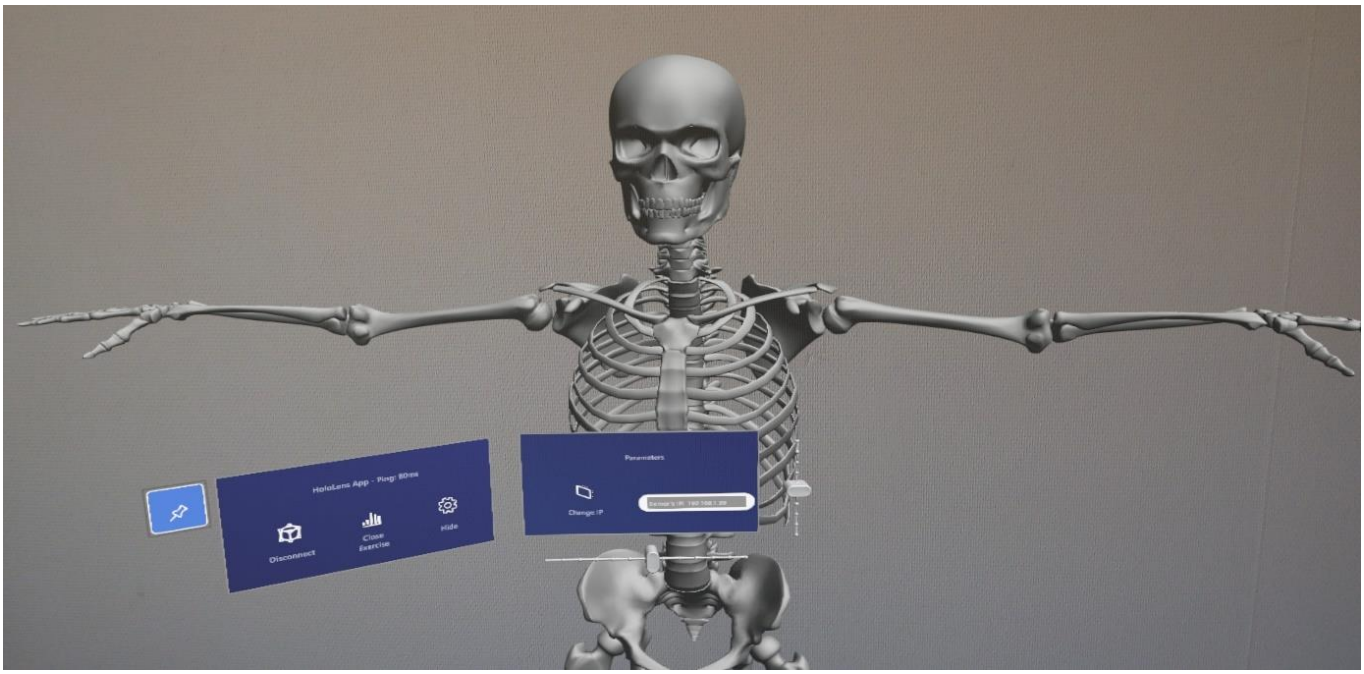

Figure 33: photo montrant la taille du squelette lorsque la barre réglable inférieure est au centre. Il est fait en sorte d'avoir un squelette à échelle humaine quand le curseur est placé à la moitié de sa course.

Annexe 3 – Captures d'écran du code responsable des exercices

```
// Update est appelée à chaque frame
void Update()
{
    // Si l'utilisateur n'est pas allé au bout de la démarche de lancement d'un exercice (s'il n'a pas sélectionnée un nombre de répétitions), on ne fait rien
    if (exerciseReps != 0) Exercise();
}

/*
   CHOIX D'UN EXERCICE
         ||
         \/
*/

// Appelée par le bouton 1, exercice du squat
public void Exercise1()
{
    exerciseId = 1;
    bestResult = 180; // On veut l'angle le plus petit possible donc on initialise le résultat à l'angle correspondant à la postition debout
    miniAGrade = exerciseManager.squatA; // On récupère les paliers des notes
    miniBGrade = exerciseManager.squatB;
    ShowHideReps(); // Cette fonction nous sert juste à afficher ou masquer les boutons de choix des répétitions
}

// Appelée par le bouton 2, exercice de la courbure du dos
public void Exercise2()
{
    exerciseId = 2;
    bestResult = 180; // Distance arbitraire, initialisiée suffisamment élevée pour pouvoir sauvegarder par la suite nos résultats
    miniAGrade = exerciseManager.bendingA;
    miniBGrade = exerciseManager.bendingB;
    ShowHideReps();
}

// Appelée par le bouton 3, exercice de la montée de genou
public void Exercise3()
{
    exerciseId = 3;
    bestResult = 0; // On veut l'angle le plus grand possible donc on initialise à 0
    miniAGrade = exerciseManager.highKneeA;
    miniBGrade = exerciseManager.highKneeB;
    ShowHideReps();
}
```

Figure 34: première partie du code, on y voit la fonction Update ainsi que les fonctions appelées par les boutons

```
/*
    CHOIX DU NOMBRE DE REPETITIONS
              ||
              \/
*/

// Ces fonctions sont appelées respectivement par le bouton du choix de 5 répétitions et celui de 10
public void reps5()
{
    exerciseReps = 5;
    ShowHideReps();
}

public void reps10()
{
    exerciseReps = 10;
    ShowHideReps();
}

/*
    exerciseId non nul donc Exercise() est appelé dans Update()
                          ||
                          \/
*/

private void Exercise()
{
    // En fonction de l'identifiant de l'exercice, on va appeler la fonction correspondante
    switch (exerciseId)
    {
        case 1:
            SquatExercise(exerciseReps);
            break;

        case 2:
            BendingExercise(exerciseReps);
            break;

        case 3:
            HighKneeExercise(exerciseReps);
            break;
    }
}
```

Figure 35: seconde partie du code, on y aperçoit les fonctions de choix des répétitions ainsi que celle responsable de l'appel de l'exercice

```
/*
    Affichage de l'exercice, comptage des répétitions et stockage du meilleur résultat
                                            ||
                                            \/
*/

private void SquatExercise(int reps)
{
    // On récupère les données essentiels de l'exercice
    float leftKnee = exerciseManager.GetKneeAngle("left");
    float rightKnee = exerciseManager.GetKneeAngle("right");
    float squatMiniAngle = exerciseManager.squatMiniAngle;

    // On affiche sur le panneau des exercices les angles des deux genoux
    exerciseDataTMPText.text = "Left knee angle = " + leftKnee + "  //  Right knee angle = " + rightKnee;

    // S'il nous reste des répétitions à faire
    if (counter < reps)
    {
        // Si le mouvement est validé : si les deux angles aux genoux sont inférieurs à notre valeur palier
        if (leftKnee < squatMiniAngle && rightKnee < squatMiniAngle)
        {
            // Si le mouvement n'était pas validé à la frame précédente : il s'agit d'une nouvelle répétitions
            if (lastFrameSituation == false)
            {
                counter++;
                lastFrameSituation = true;
            }

            // On affiche sur le panneau qu'il y a un squat d'effectué ainsi que le compteur
            exerciseDataTMPText.text += "\nSQUAT !\n\nSquat Counter = " + counter;

            // On sauvegarde le meilleur résultat
            float min = Mathf.Min(leftKnee, rightKnee);
            if (min < bestResult) bestResult = min;
        }
        // Si le mouvement n'est pas validé
        else
        {
            // On affiche seulement le compteur
            exerciseDataTMPText.text += "\n\n\nSquat Counter = " + counter;
            // Le booléen devient faux s'il ne l'est pas déjà
            if (lastFrameSituation != false) lastFrameSituation = false;
        }
    }
    // Ici, il ne reste plus de répétitions à faire donc on affiche le meilleur résultat ainsi que la note
    else
    {
        exerciseDataTMPText.text = "RESULTS: \n\nBest angle = " + bestResult;
        DisplayGrade();
    }

}
```

Figure 36: troisième partie du code, on y voit la fonction du squat appliquant le cheminement logique expliqué par la figure 23

```
// Ces deux fonctions fonctionnent de la même manière que la précédente et appliquent le même cheminement logique
private void BendingExercise(int reps)...

private void HighKneeExercise(int reps)...

/*
    Affichage de la note
            ||
            \/
*/

private void DisplayGrade()
{
    if (bestResult < miniBGrade)
    {
        if (bestResult < miniAGrade)
        {
            exerciseDataTMPText.text += "\n\nGRADE = A   Excellent !";
        }
        else
        {
            exerciseDataTMPText.text += "\n\nGRADE = B   Good !";
        }
    }
    else
    {
        exerciseDataTMPText.text += "\n\nGRADE = C   You can do better !";
    }
}
```

Figure 37: dernière partie du code, présentant la fonction affichant la note