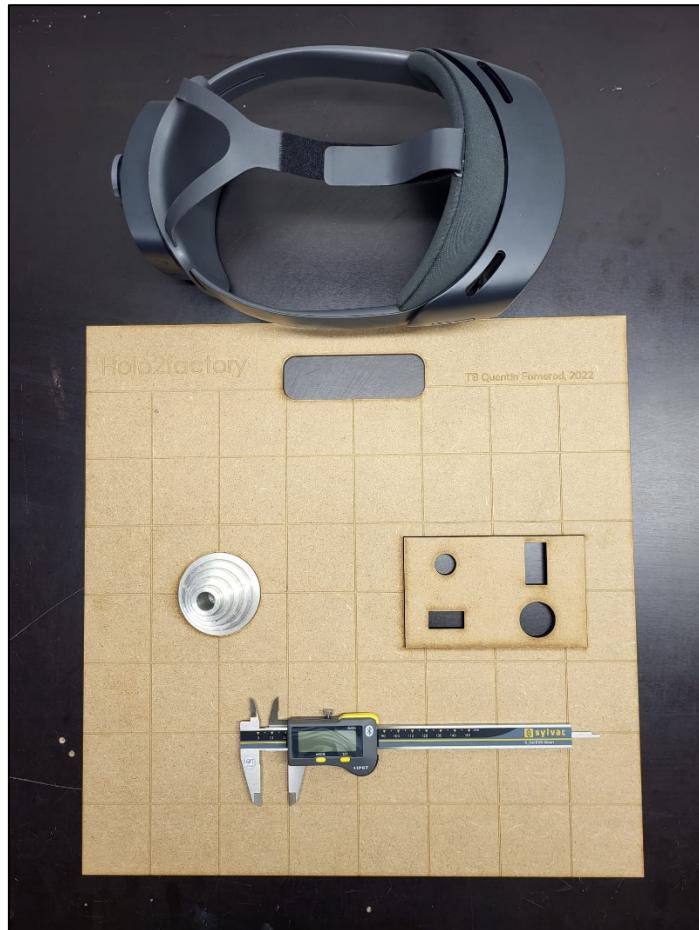


Travail de Bachelor

Holo2factory

Non confidentiel



Étudiant :

Quentin Fornerod

Travail proposé par :

François Birling, iAi

Enseignant responsable :

François Birling

Année académique :

2021-2022

Département TIN
Filière Génie électrique
Orientation Electronique et Automatisation industrielle
Étudiant Quentin Fornerod
Enseignant responsable François Birling

Travail de Bachelor 2019-2020
Holo2factory

Résumé publiable

Il s'agit de développer une application permettant de guider un opérateur dans l'exécution d'une séquence de mesures avec un outil connecté type pied à coulisse ou multimètre. Grâce à l'utilisation de la réalité mixte et des hololens, l'utilisateur sera guidé pour effectuer les mesures dans l'ordre attendu sur la pièce à contrôler.

- Recherche d'un cas concret permettant de démontrer l'intérêt de la réalité mixte dans le contexte de l'inspection qualité ou la calibration d'appareils.
- Analyse de cas d'utilisation sur la base du cas choisi.
- Prise en main de la technologie Hololens 2 de Microsoft.
- Définition des fonctionnalités applicatives, prototypage d'interface utilisateur.
- Conception de l'architecture logicielle
- Développement de l'application incluant le contrôle qualité et le réglage d'un appareil.
- Réalisation d'une démonstration probante et vidéo d'une utilisation typique.
- Rapport et présentation des résultats, vidéo de démonstration

Étudiant :

Fornerod Quentin

Date et lieu :

22.07.2022 Yverdon

Signature :



Enseignant responsable :

Birling François

Date et lieu :

22.07.2022 Yverdon

Signature :



Département	Technologies industrielles
Filière	Génie électrique
Orientation	Electronique et Automatisation industrielle
Candidat	Quentin Fornerod
Responsable	François Birling

Holo2Factory : Guidage opérateur en production flexible avec Hololens**Institut** iAi**Enoncé**

Il s'agit de développer une application permettant de guider un opérateur dans l'exécution d'une séquence de mesures avec un outil connecté type pied à coulisse ou multimètre. Grâce à l'utilisation de la réalité mixte et des hololens, l'utilisateur sera guidé pour effectuer les mesures dans l'ordre attendu sur la pièce à contrôler.

Cahier des charges

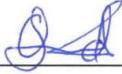
Ce projet vise les objectifs suivants :

- Prendre en main la technologie Hololens 2 de Microsoft pour la réalité mixte.
- Mener l'analyse fonctionnelle, rédiger la spécification et concevoir l'architecture logicielle d'une application de réalité mixte pour réaliser le guidage d'un opérateur de contrôle qualité dans un contexte de production flexible.
- Développer l'application de guidage opérateur comprenant un outil de mesure connecté, un affichage géolocalisé des points de mesure sur la pièce à contrôler et un serveur d'enregistrement des mesures.
- Choisir et exploiter une technologie de localisation 3D permettant d'avoir un positionnement précis des hologrammes sur la pièce à contrôler.
- Elaborer des principes de représentation 3D parlantes et ergonomiques dans le casque de réalité mixte.
- Réaliser une vidéo de démonstration de qualité professionnelle pour valoriser les potentialités de la réalité mixte dans le domaine de la production flexible.

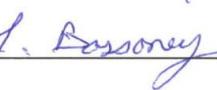
Bibliographie

- Recherche d'un cas concret permettant de démontrer l'intérêt de la réalité mixte dans le contexte de l'inspection qualité ou la calibration d'appareils.
- Analyse de cas d'utilisation sur la base du cas choisi.
- Prise en main de la technologie Hololens 2 de Microsoft.
- Définition des fonctionnalités applicatives, prototypage d'interface utilisateur.
- Conception de l'architecture logicielle

- Développement de l'application incluant le contrôle qualité et le réglage d'un appareil.
- Réalisation d'une démonstration probante et vidéo d'une utilisation typique.
- Rapport et présentation des résultats, vidéo de démonstration

Candidat Quentin FornerodDate : 22.07.2022Signature : **Responsable** François BirlingDate : 22.07.2022Signature : **Responsable de la filière Génie électrique**Date : 22.07.2022

Luc Bossoney

Signature : 

Préambule

Ce travail de Bachelor (ci-après TB) est réalisé en fin de cursus d'études, en vue de l'obtention du titre de Bachelor of Science HES-SO en Ingénierie.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'Ecole.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d'auteur.

HEIG-VD

Le Chef du Département

Yverdon-les-Bains, le 8 août 2022

Diplômant : Fornerod Quentin

Titre du travail de Bachelor : Holo2Factory : Guidage opérateur en production flexible avec Hololens

Enseignant responsable du TB : Birling François

Tous les TB sont déposés à la Bibliothèque de la HEIG-VD qui en gère l'archivage et la consultation. Quel que soit le niveau de confidentialité du TB, le nom du diplômant, le nom de l'enseignant responsable, le titre du TB et le résumé publiable figurent dans tous les documents de présentation des TB ainsi que sur la plateforme de consultation des TB (<http://tb.heig-vd.ch>). L'enseignant responsable veille à ce que le titre du TB et le résumé publiable soient rédigés conformément au niveau de confidentialité voulu.
Les TB peuvent être soumis à un logiciel anti-plagiat. Dans ce cas, leur contenu sera traité de manière confidentielle.

Le TB n'est pas confidentiel

Outre les informations mentionnées ci-dessus, les documents de présentation du TB contiennent également les noms des entreprises partenaires, le résumé publiable et une affiche. Le TB peut être consulté sur la plateforme des TB.

Le TB est confidentiel.

Les conditions suivantes de diffusion des informations sont appliquées :

*Aucune consultation ou emprunt du TB n'est permis hormis par l'enseignant responsable du TB et le diplômant qui s'engagent à ne pas faire usage des informations mises à leur disposition. Le TB porte la mention « **confidentiel** ».*

Oui Non *Nous acceptons que les noms des entreprises partenaires figurent dans les documents publiés (titre, résumé, affiche, etc.), ainsi que dans la plateforme de consultation des TB.*

Oui Non *Nous acceptons que l'affiche du TB figure sur la plateforme de consultation des TB (l'affiche est au préalable validée par les entreprises partenaires).*

Dans tous les cas, un accord de confidentialité doit être signé par le diplômant, l'expert et toutes les personnes participant à l'évaluation du TB.

Nous déclarons accepter les conditions de diffusion du Travail de Bachelor indiquées.

Diplômant

Date

22.07.2022

Nom et signature

Fornerod Quentin

1

Enseignant responsable

Date

22.07.2022

Nom et signature

Birling François

2

N.B.:

Ce document fait partie intégrante du cahier des charges du TB.

La forme masculine est utilisée comme genre neutre et désigne à la fois les hommes et les femmes.

Authentification

Le soussigné, Quentin Fornerod, atteste par la présente avoir réalisé seul ce travail et n'avoir utilisé aucune autre source que celles expressément mentionnées.

Forel, le 8 août 2022

Quentin Fornerod



Abstract

Dans le cadre de mon Bachelor of Science HES-SO en Génie Electrique orientation Electronique et Automatisation Industrielle à la HEIG-VD, le projet Holo2factory¹ a pour but de démontrer l'intérêt de la réalité augmentée dans l'industrie. Un cas d'étude a pour cela été choisi et étudié dans le cadre de ce projet. Il consiste à guider un opérateur de production au contrôle des cotations d'une pièce quelconque.

Une machine de production conventionnelle débitait un seul modèle de pièce par production. L'arrivée des machines d'usinage CNC a amplifié la tendance à réaliser des pièces uniques, ce qui rend la tâche de l'opérateur plus complexe. Ceci augmente inexorablement les erreurs de mesures ainsi que la complexité de la tâche de l'employé. La réalité augmentée est une technologie intéressante pour mener au bon fonctionnement cette opération.

Ce rapport se décompose en différentes parties. Il comportera une analyse des différentes technologies étudiées ainsi que leur intérêt pour ce projet. Des prototypes seront réalisés afin de déterminer les capacités de ces outils. Dès lors, il sera possible de sélectionner les technologies les plus intéressantes pour la suite de ce dernier. Un développement approfondi sera exposé et expliqué.

¹ Le 2 de Holo2factory (en anglais *two*) doit être interprété par son homonyme *to* signifiant : à, vers → le titre sous-entendu est HoloLens 2 dans l'industrie

Remerciements

Je tiens sincèrement à remercier M. François Birling, enseignant à la Haute Ecole d'Ingénierie et de Gestion du canton de Vaud, qui m'aura encadré avec grande attention durant ce semestre. Ses précieux conseils et ses brillantes idées m'auront permis de m'orienter convenablement dans le cadre de ce travail de Bachelor.

Je remercie Dylan Morocutti, collaborateur à la Haute Ecole d'Ingénierie et de Gestion du canton de Vaud, qui m'a initié à la configuration du logiciel de développement Unity ainsi qu'au fonctionnement de Woopsa.

J'exprime également ma gratitude aux experts ainsi qu'à toutes les personnes qui prendraient le temps de lire ce document.

Table des matières

Table des matières

1	Introduction	17
1.1	Inspection de pièces	17
1.2	Calibration d'un appareil assistée.....	17
1.3	Cas concret industriel	17
1.4	Choix du cas d'étude.....	18
1.5	La réalité augmentée	19
1.5.1	Matériel utilisé.....	19
2	Pré-étude	20
2.1	Définition de l'application.....	20
2.1.1	Mission de l'opérateur.....	20
2.1.2	Mission du superviseur.....	22
2.1.3	Mission du configIBUTEUR.....	22
2.2	Cahier des charges.....	23
2.2.1	Analyse des fonctionnalités.....	23
2.2.2	Étape 1	23
2.2.2.1	Description.....	23
2.2.2.2	Marche à suivre	23
2.2.2.3	Prototype visuel.....	24
2.2.2.4	Diagramme UML.....	25
2.2.2.5	Spécifications fonctionnelles	26
2.2.3	Étape 2	26
2.2.3.1	Description.....	26
2.2.3.2	Marche à suivre	27
2.2.3.3	Prototype visuel.....	27
2.2.3.4	Diagramme UML.....	28
2.2.3.5	Spécifications fonctionnelles	29
2.2.4	Étape 3	29
2.2.4.1	Description.....	29
2.2.4.2	Marche à suivre	29
2.2.4.3	Prototype visuel.....	29
2.2.4.4	Diagramme UML.....	30
2.2.4.5	Spécifications fonctionnelles	31

2.2.5	Spécifications du rendu	31
2.2.6	Synthèse.....	32
2.3	Architecture logicielle	33
3	Pré-étude	36
3.1	Pied à coulisse connecté.....	36
3.1.1	Mode simple	37
3.1.2	Mode pair	37
3.1.3	Mode Hid	37
3.1.4	Communication avec le PC	37
3.2	Application WPF	38
3.2.1	Prototype de l'application	38
3.2.2	Description.....	38
3.3	Développement de la réalité augmentée.....	39
3.3.1	Configuration du logiciel Unity	40
3.3.2	Chargement sur le casque	40
3.3.3	Remoting.....	41
3.3.4	Prototype visuel.....	41
3.4	Tracking.....	42
3.5	Posage fixe	42
3.6	Serveur ASP.....	43
3.7	Requêtes API.....	43
3.7.1	NewtonSoft.....	44
3.7.2	ReqBin.....	44
3.7.3	Woopsa	44
3.8	Helix	44
3.9	Blazor	45
3.10	Github	46
3.11	Conclusion de la pré-étude.....	47
4	Etude du projet	48
4.1	Introduction	48
4.2	Restructuration de l'architecture logicielle	48
4.3	Développement ASP	49
4.3.1	Diagramme UML.....	49
4.3.2	Start	53
4.3.3	Stop	54
4.3.4	Next.....	54

4.3.5	Previous	54
4.3.6	Chargement des pièces enregistrées.....	54
4.3.7	Sauvegarde des pièces mesurées	55
4.3.8	Chargement d'une recette.....	56
4.4	Développement Blazor	56
4.4.1	Récupération des données de mesure	56
4.4.2	Requête GET	57
4.4.3	Virtualize	57
4.4.4	Rendu visuel.....	58
4.5	Application Unity	59
4.5.1	Diagramme UML.....	59
4.5.2	Fonctionnalités	59
4.5.3	Choix de l'utilisateur	60
4.5.4	Choix de la pièce	61
4.5.5	Séquence de mesure	61
4.5.6	Aide dynamique (handmenu)	62
4.5.7	Hiérarchie du programme	63
4.5.8	Description du code	64
4.6	Tracking.....	67
4.6.1	Déroulement de l'implémentation	67
4.6.2	Création du dataset par algorithme de Deep Learning	68
4.7	Application WPF	69
4.7.1	Diagramme UML.....	69
4.7.2	Description de l'application	69
4.7.3	Analyse du code	70
4.7.4	Aperçus visuels	72
4.8	Holo2factory configurator	73
4.8.1	Diagramme UML.....	73
4.8.2	Analyse du code	73
4.8.3	MainWindow	75
4.8.4	Détails des fonctionnalités MainWindow.....	76
4.8.5	ProductionWindow.....	77
4.8.6	Détails des fonctionnalités ProductionWindow	78
4.8.7	Résultat au format JSON.....	79
4.9	Convention de codage	80
4.9.1	Analyse du code	80

4.9.2	Incohérences de compatibilité Unity.....	81
4.10	Tests unitaires/protocole de tests.....	81
4.10.1	Tests unitaires dans l'application WPF	81
4.10.2	Couverture du code	82
4.11	Performance du repérage spatial	82
4.11.1	Posage fixe	83
4.11.1.1	Code QR	83
4.11.1.2	Microsoft Azure Anchor.....	83
4.11.2	Tracking Vuforia.....	83
4.11.3	Résultats	84
4.12	Alternative intéressante (Dynamics 365 Guides)	85
4.12.1	Connexion à Guides	85
4.12.2	Démonstration des fonctionnalités	85
4.12.2.1	Témoignage d'une entreprise.....	86
5	Synthèse.....	87
5.1	Problèmes rencontrés	87
5.1.1	Mise à jour Windows	87
5.1.2	Version expérimentale de MRTK.....	87
5.1.3	Problèmes quelconques	87
5.2	Planification	88
5.3	Statistiques du projet	89
5.4	Accomplissement des objectifs (à développer).....	92
5.4.1	Vérification des spécifications de l'application Desktop	92
5.4.2	Vérification des spécifications de l'application Unity	93
5.4.3	Vérification des spécifications de l'application Blazor	94
5.5	Perspectives de développement	95
5.6	Conclusion	96
6	Bibliographie	97
6.1	Images.....	97
6.2	Documentation	97
7	Abréviations/Définitions.....	98
8	Liste des figures	100
9	Liste des tableaux.....	101
10	Liste des diagrammes UML.....	101
11	Liste des témoignages.....	102
12	Liste des graphiques	102

13	Liste des extraits de code.....	102
14	Annexes.....	104
14.1	Planning	105
14.2	Manuel d'utilisateur Holo2FactoryConfigurateur	106
14.3	Manuel d'utilisateur du pied à coulisse.....	108
14.4	Protocole de test Holo2factoryConfigurator	110

1 Introduction

Dès la nuit des temps, l'homme n'a jamais cessé d'améliorer ses compétences et ses performances afin d'augmenter sa productivité. La révolution industrielle a obtenu ce nom en imaginant un nouveau concept de travail : se spécialiser dans une tâche et la répéter de manière à l'effectuer de façon optimale. La révolution qui s'instaure dès les années 2010 n'est autre que l'industrie 4.0, alias la 4^{ème} révolution industrielle. Celle-ci s'affirme comme la convergence du monde virtuel, de la conception numérique, de la gestion (opérations, finance et marketing) avec les produits et objets du monde réel.² Le contenu virtuel entoure l'homme et est devenu omniprésent dans notre quotidien jusqu'aux lignes de productions. La réalité augmentée n'est que peu présente dans ces domaines et l'application illustrée dans ce projet aura pour but d'enluminer cette technologie.

Plusieurs cas d'études étaient disponibles :

1. Inspection de pièces (vérification des cotations après usinage)
2. Assistance pour la calibration d'un appareil
3. Cas concret issu de l'industrie

1.1 Inspection de pièces

Dans l'industrie de production, le contrôle qualité est un élément essentiel faisant partie intégrante de la chaîne de production. Un fabricant ne peut pas se permettre de remettre une pièce usinée non conforme à un client. Ceci arrive parfois, mais il cherchera à faire tendre cette erreur vers zéro. Pour cela, un opérateur de production aura pour mission de mesurer toutes les cotations d'une pièce afin d'assurer la conformité de cette dernière. La production flexible ou communément appelée *Ateliers flexibles* sont des ateliers rapidement reconfigurables qui offrent une diversification des produits contrairement à une production en ligne standard. Ceci incrémenterait inconditionnellement les risques d'erreurs de supervision. La réalité augmentée aurait pour mission d'améliorer significativement les performances et ajouterait un suivi du contrôle qualité.

1.2 Calibration d'un appareil assistée

Tout appareil paramétrable nécessite une calibration ou un ajustement de variables. Les artefacts électroniques sont réputés pour leur complexité de configuration et leur nombre incalculable de paramètres. Enseigner la calibration d'une panoplie d'appareils à un nouveau collaborateur est une action chronophage. Si cette action allait être répétée régulièrement, ceci impacterait la mission principale de l'employé chevronné qui n'est certainement pas la démonstration d'une marche à suivre. Réaliser une application de réalité augmentée effectuant le guidage de la configuration d'un appareil pour un nouvel employé assure une autonomie de la personne.

1.3 Cas concret industriel

La dernière proposition était d'établir le contact avec une entreprise de la région et d'y intégrer la réalité augmentée dans une ligne de production quelconque. L'entreprise en question était intéressée, mais cette dernière s'est manifestée tardivement, ce qui est contraignant pour un projet ayant une durée déterminée restreinte. Malgré cela, un témoignage de leur part concernant une application sur leur ligne de production pourrait figurer dans ce rapport.

² Wikipédia, Industrie 4.0 [en ligne], date inconnue, consulté [10.06.2022], https://fr.wikipedia.org/wiki/Industrie_4.0

1.4 Choix du cas d'étude

Les deux cas proposés par l'école sont similaires et le développement d'un de ces projets a été retenu : l'inspection de pièces issues de la production flexible. Les deux cas pourraient être survolés, mais il a été décidé de se concentrer sur un cas pratique et de le développer au maximum.

1.5 La réalité augmentée

La réalité augmentée plus communément nommée par son acronyme AR est l'action d'intégrer des éléments holographiques spatiaux dynamiques dans le champ de vision d'une personne. Ceci a pour but d'ajouter des informations telles que des consignes, des éléments 3D. Ces éléments spatiaux sont uniquement visibles pour la personne disposant du casque/lunettes de réalité augmentée ou tout autre appareil prédisposé à cela. Voici un exemple issu du projet avec les HoloLens 2 de Microsoft.



Figure 1: Aperçu de la réalité augmentée

Cette technologie est parfois confondue avec la réalité virtuelle qui consiste à plonger intégralement la personne dans un monde entièrement numérique.

1.5.1 Matériel utilisé

L'appareil mis à disposition par l'école pour ce travail de Bachelor est le casque HoloLens 2 de Microsoft. Plusieurs modèles de ce casque sont disponibles :



Figure 2: Modèle du casque HoloLens 2 disponibles [1]

Ce TB ayant pour but de démontrer l'intérêt de la réalité augmentée dans l'industrie, il est donc légitime que le modèle mis à disposition soit le *HoloLens 2 Industrial Edition*. Il a l'avantage d'être certifié *UL Classe I Division 2*³ de respecter les normes *ISO 14644-14*⁴ et classe ISO désignée 5.0. Les prix de ces différents casques oscillent entre 3500 et 5000 CHF.

³ Conçu pour des environnements contenant des gaz, vapeurs et liquides inflammables, mais sous condition anormale.

⁴ Salles propres et environnements maîtrisés apparentés — Partie 14: Évaluation de l'aptitude à l'emploi des équipements par la détermination de la concentration de particules en suspension dans l'air

2 Pré-étude

2.1 Définition de l'application

L'application distingue trois cas d'utilisations distincts effectués par trois personnes différentes :

1. L'opérateur (Operator)
2. Le superviseur (Supervisor)
3. Le configurateur (Configurator)

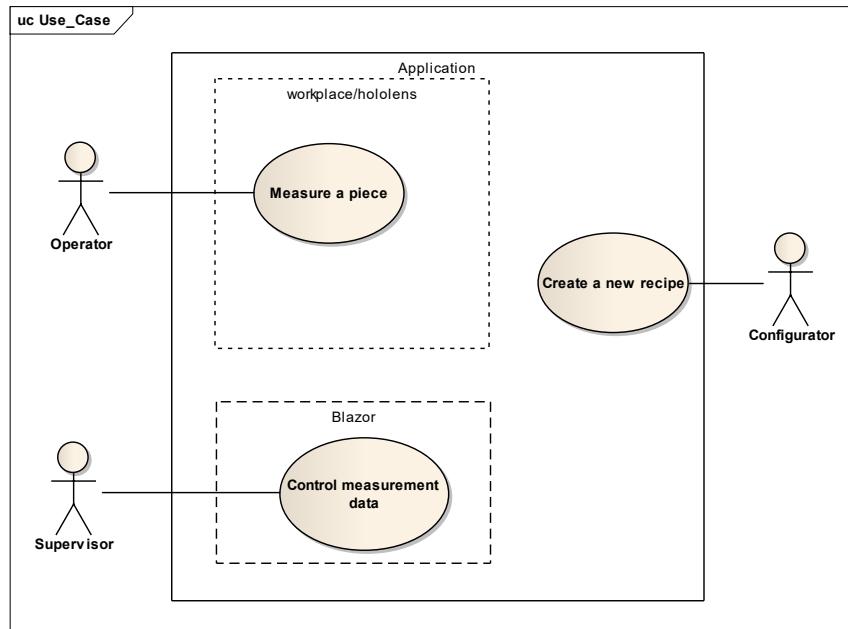


Diagramme UML 1: Cas d'utilisations

Remarque : L'application est développée pour un utilisateur anglophone

2.1.1 Mission de l'opérateur

La tâche de l'opérateur a été mentionnée à plusieurs reprises, elle consiste à réaliser une séquence de mesure afin de vérifier la conformité d'une pièce.

Il a à disposition un pied à coulisse connecté pour la mesure des pièces. Ce dernier sera connecté à un poste de travail.

La séquence peut être illustrée à l'aide d'un diagramme d'activité ainsi qu'un diagramme de séquence :

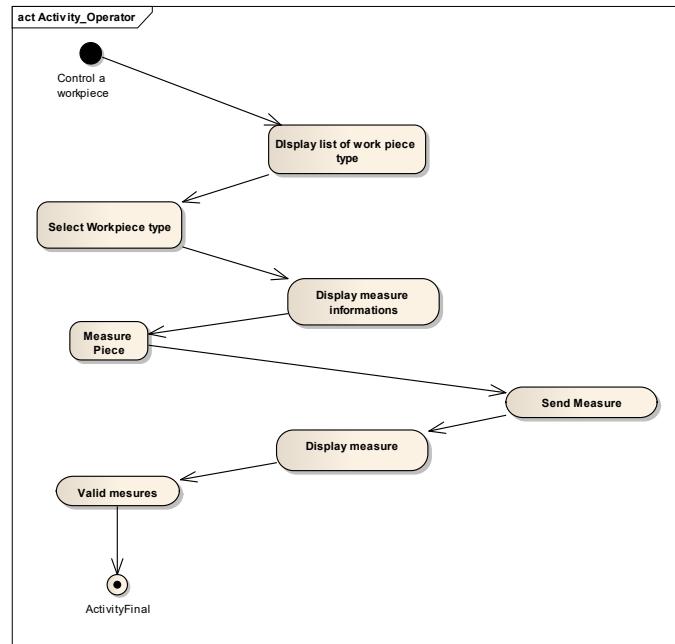


Diagramme UML 2: Activités de l'opérateur

Les méthodes principales sont représentées dans ce diagramme, il se peut que ces dernières soient modifiées d'ici la version finale.

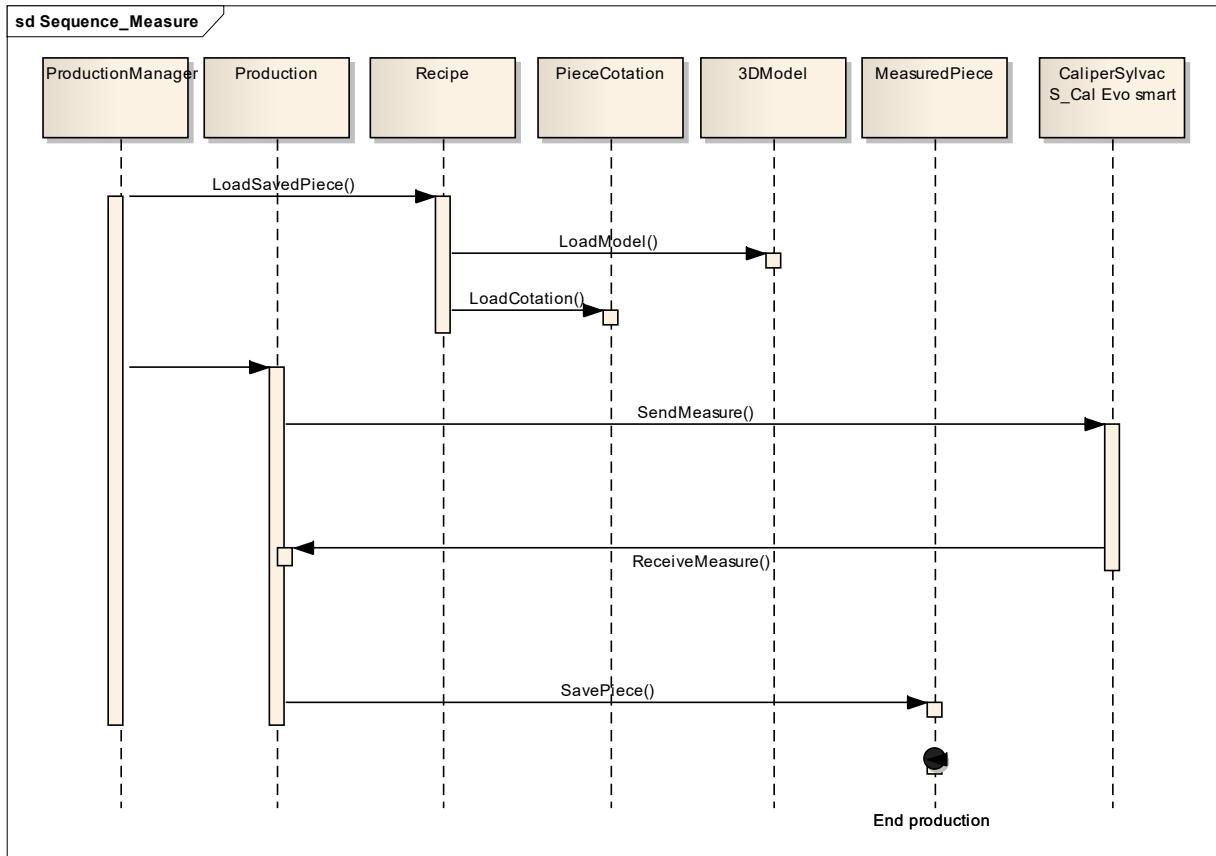


Diagramme UML 3: Séquence de mesure

2.1.2 Mission du superviseur

La tâche du superviseur est de contrôler en tout temps le bon déroulement de la séquence de vérification de pièces. Il aura donc accès via un poste de travail les différentes mesures réalisées par l'opérateur. Il pourra dès lors extraire les statistiques de production et faire ordonner des réglages sur la machine si nécessaire.

2.1.3 Mission du configurateur

Chaque pièce pouvant être unique, la tâche du configurateur sera de créer des recettes et d'adapter l'application pour l'opérateur.

2.2 Cahier des charges

Nous pouvons résumer ces éléments sous la forme de spécifications :

Spécification	Description
0.0.10	Le système doit guider l'opérateur pour une séquence de mesure d'une pièce placée sur un plan de travail
0.0.20	Il doit être possible pour configurateur de générer une recette de production
0.0.30	Les données mesurées par l'opérateur doivent être accessibles pour le superviseur

Tableau 1: Cas d'utilisations

Le développement de l'application de réalité augmentée sera réalisé sur le logiciel Unity. Il est recommandé par le constructeur, car il intègre des outils de développement pour le HoloLens 2. Une démonstration de ce logiciel sera présentée dans la suite de ce document.

2.2.1 Analyse des fonctionnalités

Les principales tâches ayant été définies, nous pouvons à présent les développer. Dans le but d'assurer un prototype fonctionnel, il a été décidé de séparer le projet en plusieurs étapes.

1. L'étape 1 consiste à réaliser un prototype fonctionnel de l'application
2. L'étape 2 intègre la base de données pour le superviseur
3. L'étape 3 implémente des mécanismes sophistiqués améliorant le confort de l'opérateur

Chaque étape sera illustrée par des exemples de prototypes. *Note : les images suivantes sont utiles pour se donner une idée du résultat de l'étape, mais elles ne représentent pas le rendu final.*

2.2.2 Étape 1

2.2.2.1 Description

L'opérateur place la pièce à mesurer au centre d'un référentiel sur la table. Ce référentiel est repéré par le casque HoloLens à l'aide de QR code ou d'autres repères. Le pied à coulisse est connecté en Bluetooth avec le PC et transmet les mesures lors d'un appui sur la touche de transfert présente sur le pied à coulisse. L'application prend en charge une seule pièce qui est en l'occurrence un porte-stylo cylindrique à plusieurs niveaux en aluminium (voir photo). Toutefois, cette pièce peut être mesurée à plusieurs reprises. Le PC communique avec le casque HoloLens via un serveur ASP hébergé par le PC. Ils échangent des données en s'envoyant des télégrammes de type API REST.

L'application Unity a pour but de guider l'opérateur dans les mesures à effectuer. Un hologramme visuel devra indiquer la côte à mesurer.

2.2.2.2 Marche à suivre

L'opérateur enclenche les différents périphériques (casque HoloLens, pied à coulisse, PC). Dès lors, une synchronisation est automatiquement effectuée (connexion au PC, connexion au serveur ASP). L'opérateur démarre la séquence sur le casque, mesure la cote indiquée, transfert la mesure en appuyant sur le bouton du pied à coulisse. La mesure sera à présent indiquée sur le casque, d'autres informations seront disponibles telles que la valeur attendue ainsi que la tolérance. Un élément visuel indiquera si la mesure est valide ou non. L'opérateur peut reprendre la mesure s'il estime qu'il est l'auteur de cette mesure non conforme. Il procède de la même manière pour toutes les cotes (il sera guidé par l'application Unity). Lorsqu'il a terminé, les données seront enregistrées sur le PC.

2.2.2.3 Prototype visuel

- Placement de la pièce fixe
- Repérage spatial à l'aide de repères



Figure 3: Prototype visuel étape 1

2.2.2.4 Diagramme UML

Diagramme de l'application PC

La gestion de la séquence est gérée par le *ProductionManager*, il contient une recette de production contenant elle-même une liste de cotations. C'est également lui qui interagira avec les acteurs extérieurs.

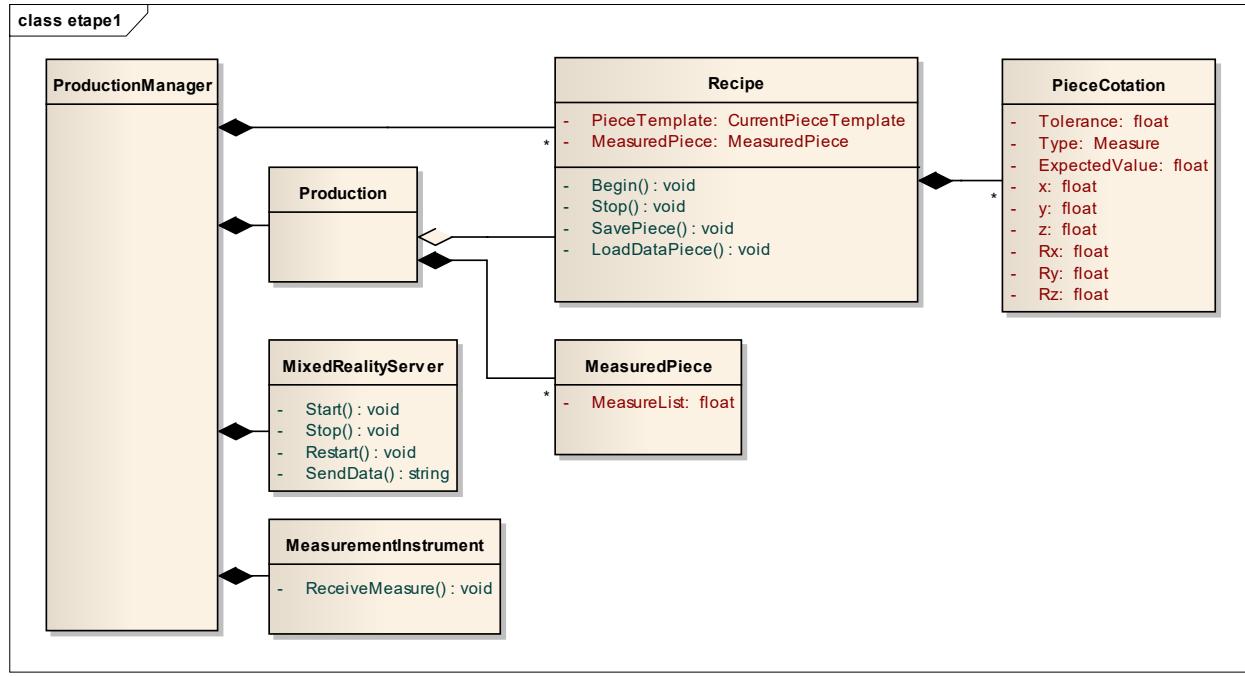


Diagramme UML 4: Application PC étape 1

Diagramme de l'application Unity

L'application contiendra une scène principale. Cette dernière sera composée d'une bannière contenant des boutons et des champs de texte 3D.

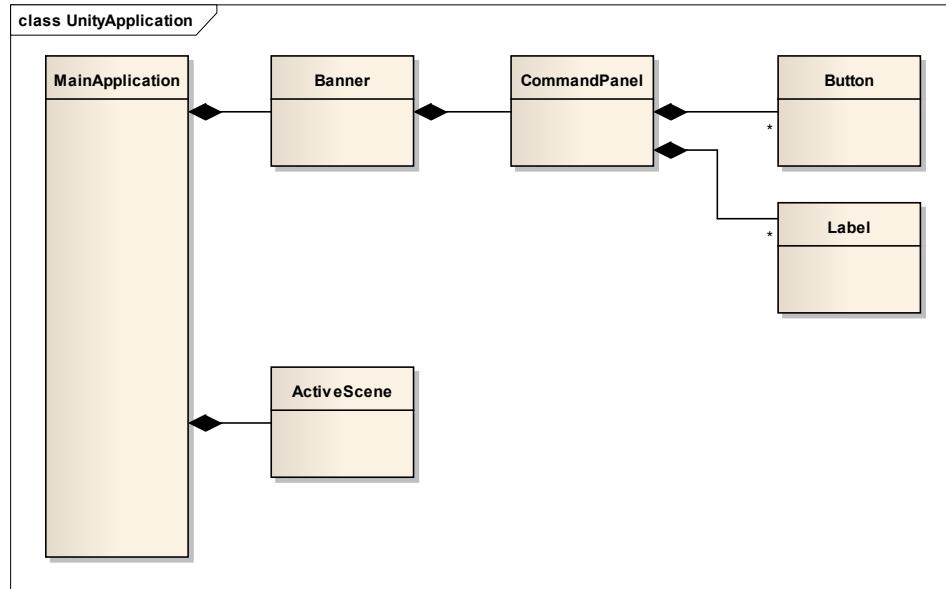


Diagramme UML 5: Application Unity étape 1

2.2.2.5 Spécifications fonctionnelles

Nous pouvons ressortir de ces différents diagrammes et définitions les spécifications fonctionnelles suivantes :

Spécification	Application Desktop
10.10.10	Le système doit être capable de lire les mesures d'un périphérique Bluetooth
10.10.20	Un système de communication doit être établi entre le casque HoloLens et le PC de l'opérateur
10.10.30	La mesure émise par le pied à coulisse doit être affichée sur le casque AR
10.10.40	Le système doit être capable de fonctionner pour au moins un type de pièce
10.10.50	Les données mesurées par l'opérateur doivent être sauvegardées dans un fichier

Tableau 2: Spécifications application Desktop étape 1

Spécification	Application Unity
10.20.10	L'application Unity doit indiquer la cote à mesurer à l'opérateur
10.20.20	L'application Unity doit implémenter une communication pour échanger des données entre le casque HoloLens et le PC de l'opérateur
10.20.30	L'application Unity doit indiquer la plage de valeur attendue de la mesure
10.20.40	L'application Unity doit indiquer si la mesure est conforme ou non
10.20.50	L'application Unity doit afficher le statut de la connexion avec le PC
10.20.60	L'application Unity doit afficher le statut de la connexion avec le pied à coulisse
10.20.70	L'application Unity devrait contenir une aide contextuelle pour guider l'opérateur en cas de besoin
10.20.80	L'application Unity doit être capable de se référencer spatialement à l'aide des QR code

Tableau 3: Spécifications Unity étape 1

2.2.3 Étape 2

L'étape 2 contient l'intégralité de l'étape 1, mais de nouvelles fonctionnalités y sont ajoutées ou améliorées.

2.2.3.1 Description

L'opérateur peut à présent mesurer différents types de pièces, il devra donc choisir au début de la production la pièce qu'il souhaite mesurer. La pièce sera toujours placée dans un référentiel fixe. Les modèles de la pièce (indiquent où doivent se trouver les cotations) sont chargés manuellement sur le casque HoloLens. Les hologrammes modélisés dans l'application Unity seront désormais plus aboutis et perfectionnés.

Le principal ajout de cette étape est la présentation des résultats. Désormais, un serveur Blazor affiche les résultats des mesures. Ces résultats sont accessibles depuis un autre poste de travail pour autant qu'il se trouve sur le même réseau local.

L'application Unity fonctionne de la même manière, mais intègre un système de sélection de pièces en début de production.

2.2.3.2 Marche à suivre

Le fonctionnement est identique à l'étape 1, toutefois, il est adapté en fonction de la description de l'étape 2.

2.2.3.3 Prototype visuel

- Placement de la pièce fixe
- Repérage spatial à l'aide de QR code ou autre repère spatial
- Hologrammes retravaillés
- Affichage des mesures (Interface web)

The screenshot shows a Blazor web application interface. On the left is a sidebar with navigation links: Home, Counter, and Fetch data (which is highlighted). The main area contains a table titled 'About' with the following data:

Date	Type de pièce	Valeur min attendue	Valeur max attendue	Valeur mesurée	Conforme
1.1.2022	Porte-stylo	Diam [mm] 29.90 39.90 49.90 59.90	Diam [mm] 30.10 40.10 50.10 60.10	Diam [mm] 30.00 40.00 50.00 60.00	Oui
1.1.2022	Porte-stylo	Diam [mm] 29.90 39.90 49.90 59.90	Diam [mm] 30.10 41.10 50.10 60.10	Diam [mm] 30.00 40.00 50.00 60.00	Non

Figure 4: Prototype visuel étape 2



Figure 5: Prototype visuel Blazor

2.2.3.4 Diagramme UML

Diagramme de l'application PC

L'intégration d'un serveur Blazor est ajoutée dans le projet. Cela signifie qu'il est nécessaire de transmettre les données des mesures et d'établir une connexion avec le serveur ASP.

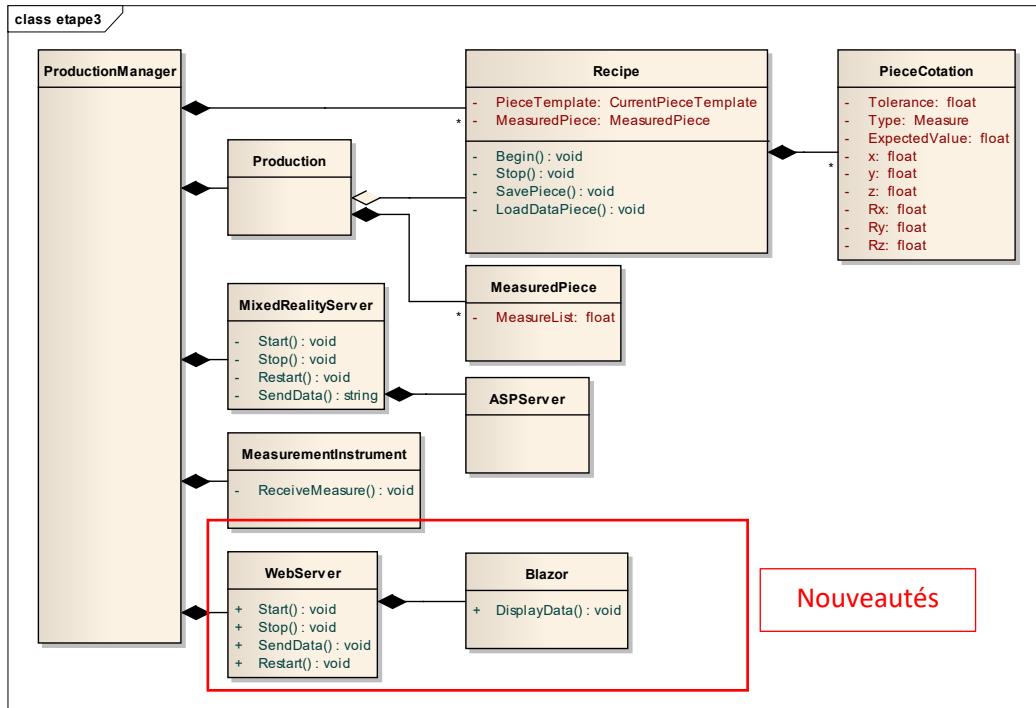


Diagramme UML 6: Application PC étape 2

Diagramme de l'application Unity

L'échange des données entre le casque et l'application ASP se fera à l'aide du protocole Woopsa. L'ajout d'un bouton supplémentaire (PinButton) permettra de placer la bannière à l'emplacement visuel de notre choix.

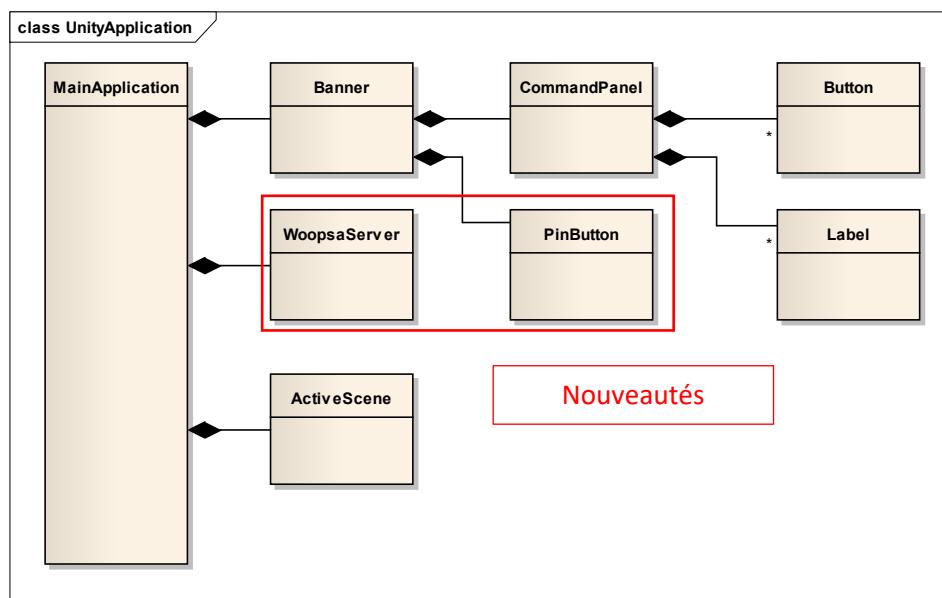


Diagramme UML 7: Application Unity étape 2

2.2.3.5 Spécifications fonctionnelles

Les spécifications suivantes sont un ajout des spécifications déjà réalisées à l'étape précédente.

Spécification	Application Desktop
20.10.10	Il doit être possible de mesurer plusieurs types de pièces
20.10.20	Les résultats des mesures doivent être transmis à l'application Blazor

Tableau 4: Spécifications application Desktop étape 2

Spécification	Application Unity
20.20.10	Les hologrammes changent de dimension en fonction de la cote à mesurer
20.20.20	L'application doit intégrer un système de sélection de types de pièce

Tableau 5: Spécifications Unity étape 2

Spécification	Application Blazor
20.30.10	L'application PC doit désormais héberger un serveur Blazor
20.30.20	Les données des mesures doivent être affichées sur un site web
20.30.30	L'application devrait intégrer un système de tri des mesures
20.30.40	Ces mesures doivent être accessibles pour une personne se trouvant sur le même réseau local

Tableau 6: Spécifications application Blazor étape 2

2.2.4 Étape 3

L'étape 3 contient l'intégralité des étapes précédentes, mais de nouvelles fonctionnalités y sont ajoutées ou améliorées.

2.2.4.1 Description

Il sera désormais possible de créer dynamiquement de nouvelles pièces à l'aide d'une application C# WPF.

L'application Unity fonctionne de la même manière, mais intègre cette fois-ci un système de *tracking* de pièce. Cela veut dire que la pièce ne sera pas nécessairement figée dans un référentiel fixe, mais l'opérateur pourra librement déplacer la pièce et les hologrammes d'indications sont déplacés dynamiquement.

2.2.4.2 Marche à suivre

Le fonctionnement est identique aux étapes 1 et 2, toutefois, il est adapté en fonction de la description de l'étape 3.

2.2.4.3 Prototype visuel

- Placement de la pièce dynamique (*tracking*)
- Repérage spatial à l'aide de QR code
- Hologrammes retravaillés
- Affichage des mesures
- Création de nouvelles pièces possible



Figure 6: Prototype visuel étape 3

2.2.4.4 Diagramme UML

Diagramme UML de l'application PC

Un modèle 3D sera désormais lié à la recette de production sera désormais.

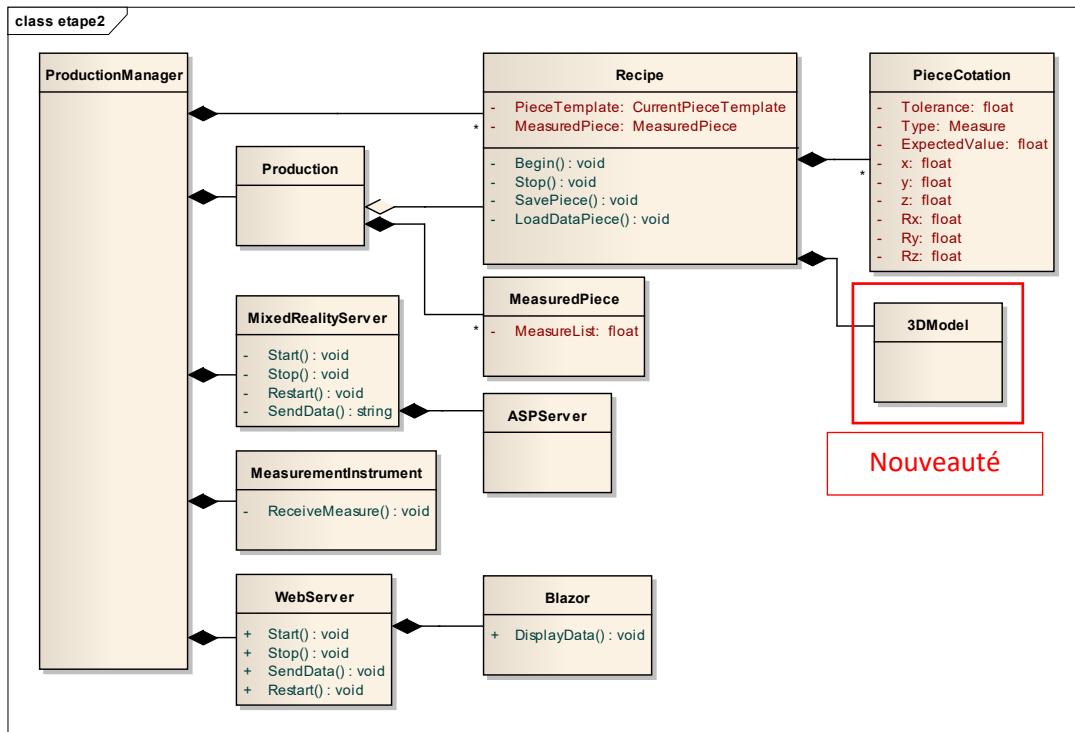


Diagramme UML 8: Application PC étape 3

Diagramme de l'application Unity

Un système de tracking sera intégré à la *MainApplication*.

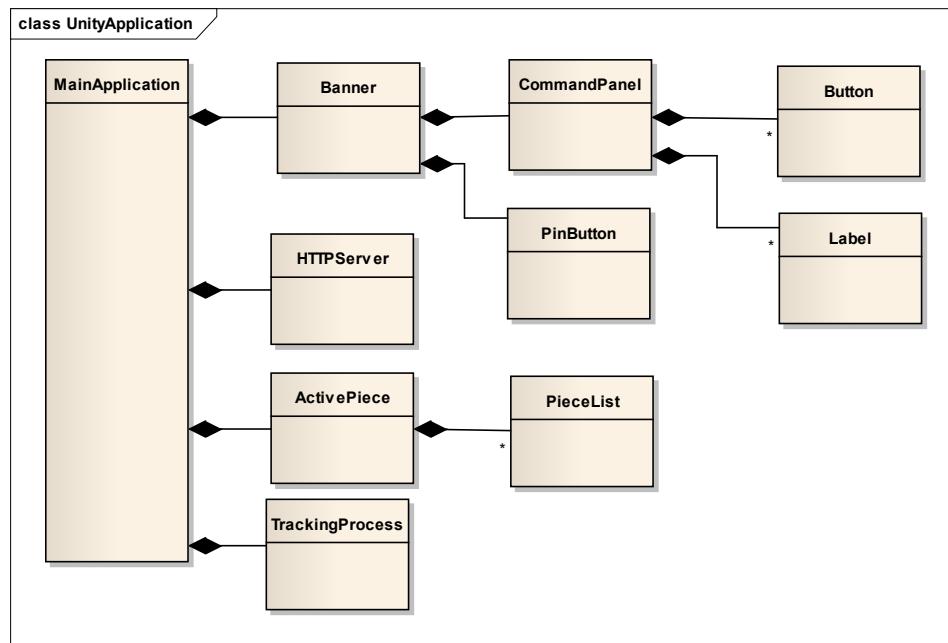


Diagramme UML 9: Application Unity étape 3

2.2.4.5 Spécifications fonctionnelles

Les spécifications suivantes sont un ajout des spécifications déjà réalisées aux étapes précédentes.

Spécification	Application Desktop
30.10.10	Une nouvelle application devrait permettre de créer de nouveaux types de pièces

Tableau 7: Spécifications application Desktop étape 3

Spécification	Application Unity
30.20.10	Un système de tracking de pièce devrait être implémenté
30.20.20	Les hologrammes d'indications de mesure devraient être placés en fonction du tracking

Tableau 8: Spécifications Unity étape 3

Spécification	Application Blazor
30.30.10	L'application devrait intégrer un affichage de mesures pour tout type de pièce

Tableau 9: Spécifications application Blazor étape 3

Spécification	Vérification - Description
40.10.10	Une analyse des exigences sera menée à la fin du projet afin de déterminer si elles ont été réalisées ou non
40.10.20	Le rapport devrait contenir un témoignage de l'intérêt de la réalité augmentée dans l'industrie

Tableau 10: Spécifications de rendu

2.2.6 Synthèse

Nous pouvons à présent récapituler ce contenu en regroupant les spécifications selon l'application tout en mentionnant l'étape.

Spécification	Application Desktop	Étape
10.10.10	Le système doit être capable de lire les mesures d'un périphérique Bluetooth	1
10.10.20	Un système de communication doit être établi entre le casque HoloLens et le PC de l'opérateur	1
10.10.30	La mesure émise par le pied à coulisse doit être affichée sur le casque AR	1
10.10.40	Le système doit être capable de fonctionner pour au moins un type de pièce	1
10.10.50	Les données mesurées par l'opérateur doivent être sauvegardées dans un fichier	1
20.10.10	Il doit être possible de mesurer plusieurs types de pièces	2
20.10.20	Les résultats des mesures doivent être transmis à l'application Blazor	2
30.10.10	Une nouvelle application devrait permettre de créer de nouveaux types de pièces	3

Tableau 11: Synthèse des spécifications application Desktop

Spécification	Application Unity	Étape
10.20.10	L'application Unity doit indiquer la cote à mesurer à l'opérateur	1
10.20.20	L'application Unity doit implémenter une communication pour échanger des données entre le casque HoloLens et le PC de l'opérateur	1
10.20.30	L'application Unity doit indiquer la plage de valeur attendue de la mesure	1
10.20.40	L'application Unity doit indiquer si la mesure est conforme ou non	1
10.20.50	L'application Unity doit afficher le statut de la connexion avec le PC	1
10.20.60	L'application Unity doit afficher le statut de la connexion avec le pied à coulisse	1
10.20.70	L'application Unity devrait contenir une aide contextuelle pour guider l'opérateur en cas de besoin	1
10.20.80	L'application Unity doit être capable de se référencer spatialement à l'aide des QR code	1
20.20.10	Les hologrammes changent de dimension en fonction de la cote à mesurer	2
20.20.20	L'application doit intégrer un système de sélection de type de pièce	2
30.20.10	Un système de tracking de pièce devrait être implanté	3
30.20.20	Les hologrammes d'indications de mesure devraient être placés en fonction du tracking	3

Tableau 12: Synthèse des spécifications Unity

Spécification	Application Blazor	Étape
20.30.10	L'application PC doit désormais héberger un serveur Blazor	2
20.30.20	Les données des mesures doivent être affichées sur un site web	2
20.30.30	L'application devrait intégrer un système de tri des mesures	2
20.30.40	Ces mesures doivent être accessible pour une personne se trouvant sur le même réseau local	2
30.30.10	L'application devrait intégrer un affichage de mesures pour tout type de pièce	3

Tableau 13: Synthèse des spécifications application Blazor

2.3 Architecture logicielle

Le casque HoloLens 2 étant un périphérique distant, il est nécessaire de mettre en place un protocole de communication. Le choix s'est décidé pour l'implémentation d'un serveur ASP.NET⁵ en C#. Etant étudiant EAI, j'ai suivi un cours qui m'a permis de m'initier aux bases du C#. Ce langage développé par Microsoft est utilisable pour créer toute sorte d'application, que ce soient des interfaces graphiques, du développement web ou autre. Le logiciel Unity requiert lui-même le C# pour créer une application.

Les composants utilisés pour ce projet sont les suivants :

- Un ordinateur
- Le casque HoloLens 2
- Le pied à coulisse

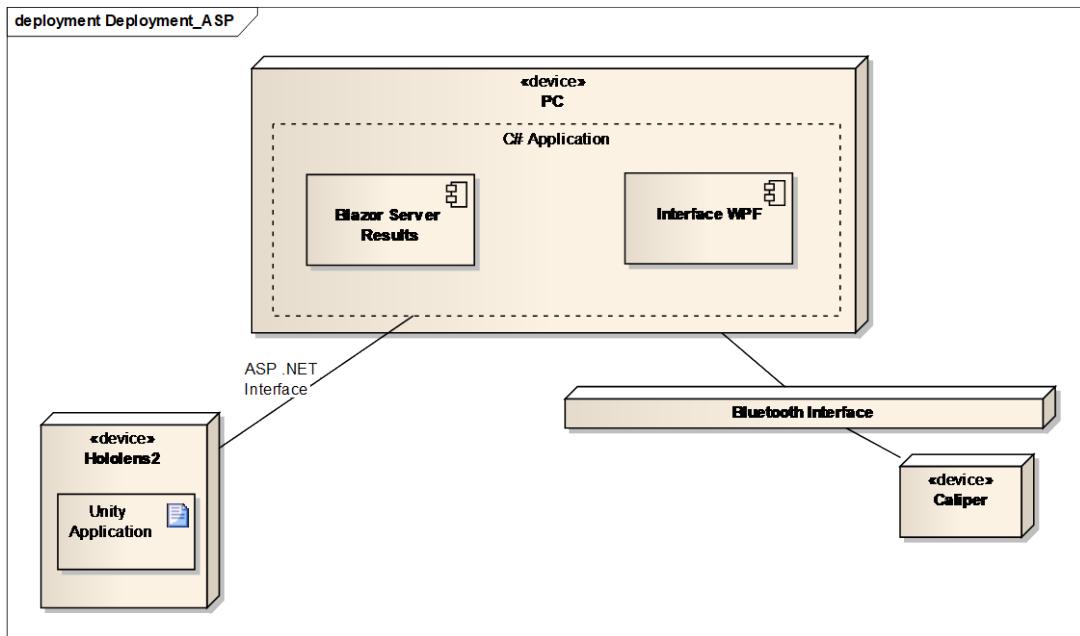


Diagramme UML 10: déploiement

⁵ Est développé dans l'étude du projet

La figure précédente indique que la transmission de la mesure du pied à coulisse vers le PC se fait par le biais d'un protocole Bluetooth. Il est toutefois nécessaire de documenter davantage la communication entre le PC et le casque.

L'application WPF transmet la mesure précédemment acquise au *controller C#* en effectuant une requête PUT, celle-ci sera définie dans la technologie des requêtes API. Les interactions entre le casque et le *controller* utilisent le même procédé. Une requête PUT (*BooleanCommand* en l'occurrence) et une requête GET (*GetMeasure*) pour recevoir la mesure et l'afficher dans le casque.

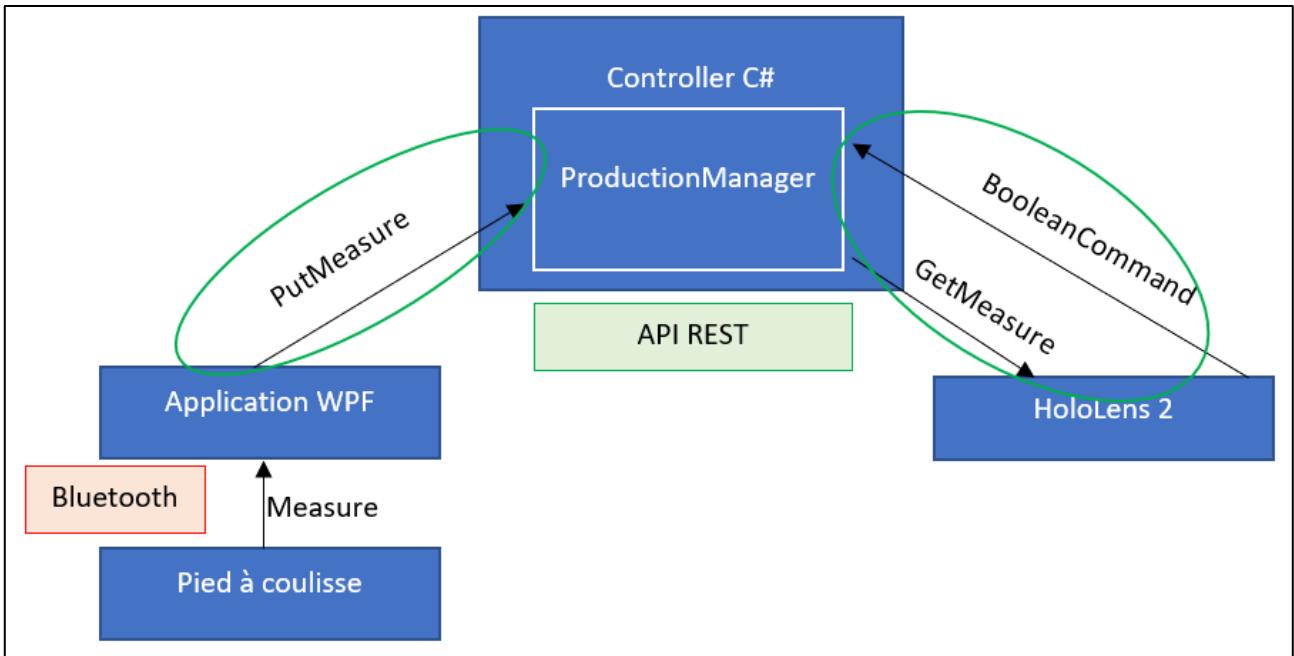


Figure 7: Schéma d'interaction entre les acteurs

Le production manager fonctionne telle une machine d'état et actualise ses états en fonction des commandes reçues par le casque. Il répondra par le contenu à afficher dans l'HoloLens 2.

L'opérateur choisit l'utilisateur connecté, le type de pièce qu'il souhaite mesurer puis suit les différentes instructions. Le *tracking* de la pièce et la récupération des mesures sont réalisés en parallèle pendant la séquence.

Note : Ce schéma est légèrement simplifié et peut différer du fonctionnement dans le code source

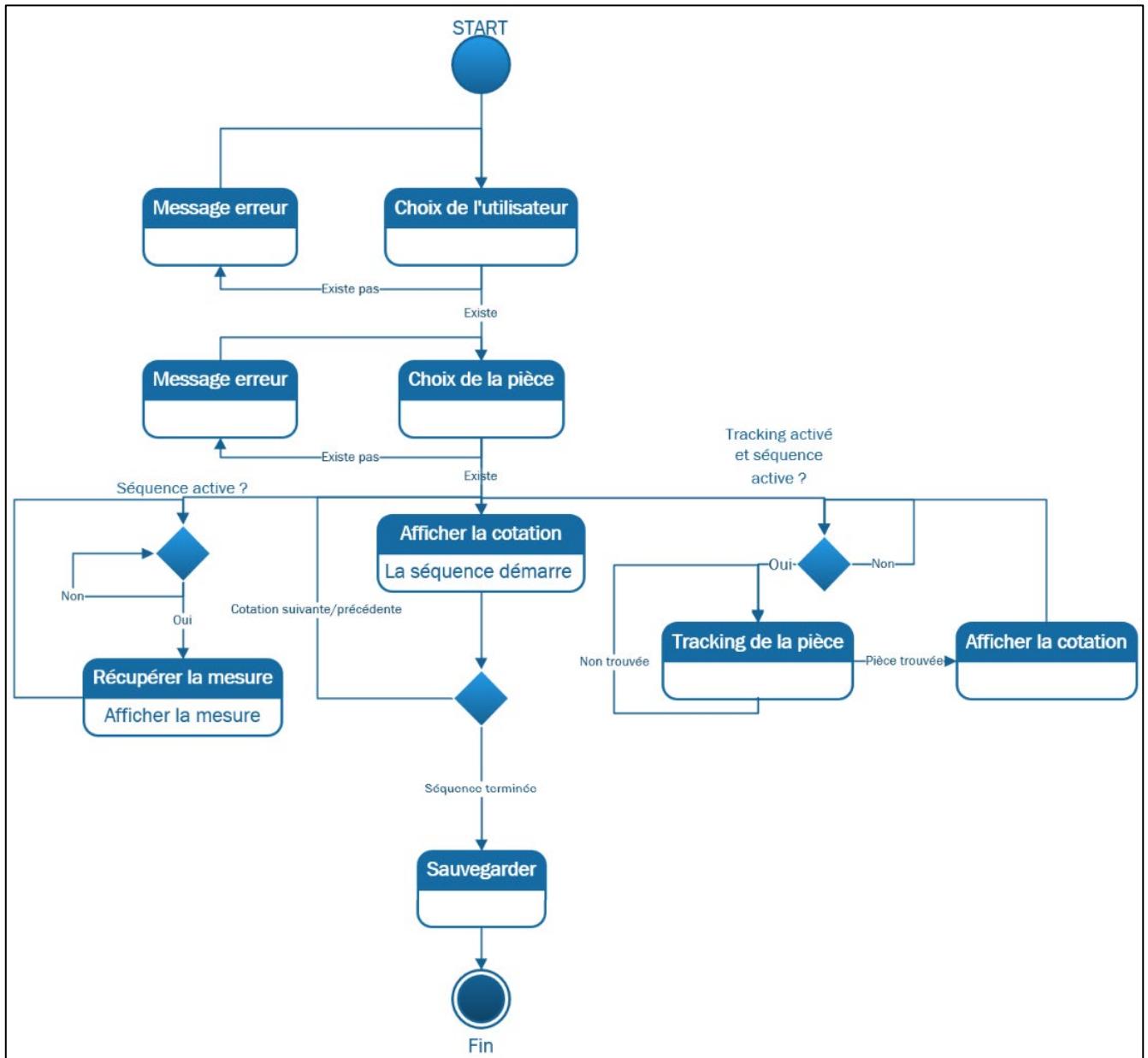


Figure 8: Machine d'état de la séquence

3 Pré-étude

La première étape de ce travail de Bachelor n'est autre que la planification, concernant la pré-étude, voici les tâches à effectuer ainsi que les technologies à étudier.

Pré étude	
Prise en main de la technologie	
Installation Unity+ MRTK et Affichage d'un hologramme dans l'espace	
Communication pied à coulisse/PC (Bluetooth)	
Transfert de données entre PC/Hololens (Woopsa/ASP)	
Implémentation des interactions possibles avec l'utilisateur (Affichage, boutons)	
Tracking d'une pièce basique avec Visionlib/Vuforia SDK	
"Hello World" avec Blazor	
Définition de l'application	
Identifier des types d'applications (Choisir la plus intéressante)	
Analyse de la fonctionnalité	
Diagrammes d'activités	
Diagrammes de cas d'utilisation	
Modèle de domaine (diag. De classes)	
Rédaction des spécifications	
Conception de l'architecture logicielle	
Planification détaillée de la réalisation	

Figure 9: Planning de la pré-étude (tâches à effectuer)

3.1 Pied à coulisse connecté

L'outil de mesure est un pied à coulisse *s_cal evo smart* de marque Sylvac exploitant le protocole Bluetooth LE (BLE⁶).

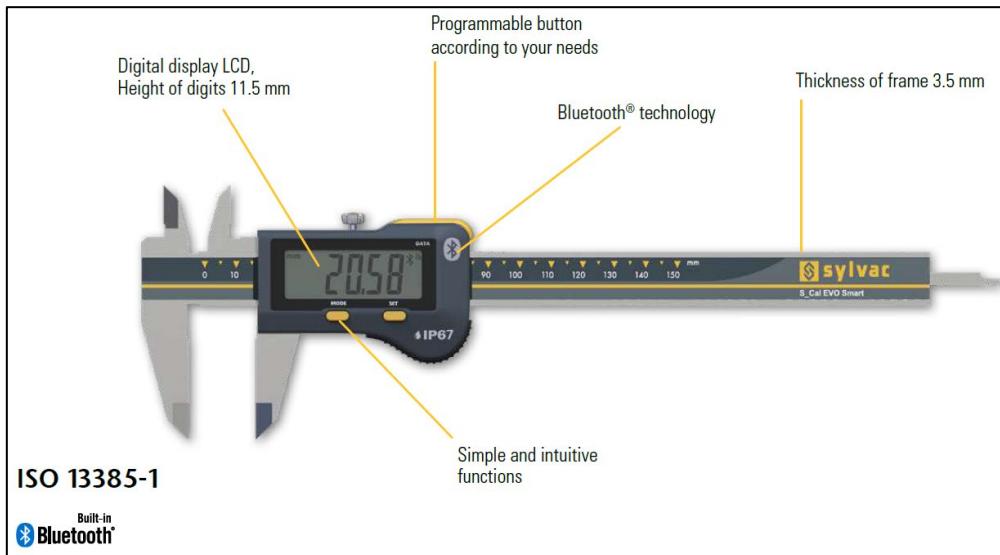


Figure 10: Pied à coulisse Sylvac [2]

⁶ BLE : Bluetooth Low Energy

Les caractéristiques de cet appareil sont les suivantes :

	810.1506	810.1516	810.1526	810.1536	810.9506	810.9516
Modèle	mm	Smart	Smart	Smart	Smart Micron	Smart Micron
Étendue de mesure	mm	150	150	200	300	150
Jauge de prof.	mm	4x1.4	Ø1.5	4x1.4	-	4x1.4
Résolution	mm	0.01	0.01	0.01	0.01	0.001
Protection		IP67	IP67	IP67	IP67	IP67

Figure 11: Modèles de pieds à coulisse Sylvac [3]

Il dispose de plusieurs profils de connexion

1. Mode *simple*
2. Mode *Pair*
3. Mode *Hid*

Ces trois modes ont été imaginés pour travailler avec l'application pour smartphone *Sylvac Anywhere*.



Figure 12: Application *Sylvac Anywhere* [4]

3.1.1 Mode simple

Ce mode *simple* permet de connecter les instruments sur plusieurs appareils, mais avec une seule connexion active, basée sur le principe du "premier arrivé, premier servi".

3.1.2 Mode pair

Ce mode permet de connecter et verrouiller (appairer) plusieurs instruments à un appareil spécifique, ce qui permet d'avoir de nombreuses stations de contrôle côté à côté dans un petit espace.

3.1.3 Mode Hid

Cet usage simule une action de clavier et permet donc d'envoyer des données sans l'intermédiaire d'un logiciel supplémentaire. L'instrument peut être connecté à un seul appareil spécifique, mais l'appareil peut accepter différentes entrées (clavier, lecteurs de code bar, etc.)

3.1.4 Communication avec le PC

Ces approches sont intéressantes, mais le pied à coulisse devra fonctionner avec une application C# déployée sur un ordinateur. Il est donc nécessaire d'implémenter une communication avec le PC. L'entrée clavier est une méthode qui d'apprête bien, mais elle requiert d'avoir le focus sur un champ de texte spécifique. Lors d'un appui sur le bouton de transfert de la mesure du pied à coulisse, la valeur sera introduite dans le champ de texte sélectionné. C'est le moyen utilisé actuellement et l'idéal serait de développer une application C# qui intégrerait directement le protocole Bluetooth. Ceci épargnerait la contrainte de devoir sélectionner le champ de texte.

Malheureusement, les documentations et les exemples de Microsoft sont pauvres concernant ce domaine. Une analyse sera réalisée si le temps à disposition le permet.

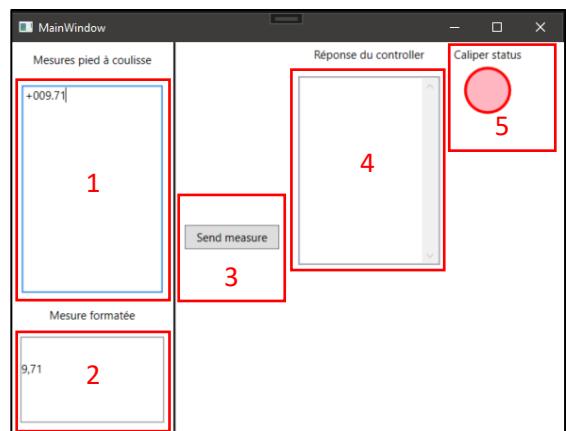
Il serait tout à fait possible d'utiliser un nano-ordinateur tel qu'un Raspberry Pi ayant pour unique but de recevoir la mesure Bluetooth et de la transmettre au serveur. Ceci restreindrait le matériel nécessaire à l'opérateur pour le bon fonctionnement de l'installation.

3.2 Application WPF

Le pied à coulisse transmet sa mesure via le protocole Bluetooth au logiciel développé en C# présenté ci-dessous.

3.2.1 Prototype de l'application

1. Mesure reçue par l'application sans formatage de texte
2. Mesure formatée au format attendu pour l'envoi
3. Bouton d'envoi
4. Réponse au télégramme reçu par le *controller*
5. Statut de connexion entre le *caliper* et le PC



3.2.2 Description

Cette fenêtre contient plusieurs contrôles utilisateurs superficiels. Le premier champ de texte a le focus sur lui (le contour du rectangle est bleu) et il est nécessaire au bon fonctionnement. Le 2^{ème} champ est indicatif et démontre le résultat de la conversion. Le 3^{ème} champ sera supprimé et l'envoi de la mesure se fera automatiquement lors de la réception d'un télégramme Bluetooth. La réponse du *controller* (4) pourrait être modifiée sous forme d'alerte en cas d'erreur. Le statut avec le pied à coulisse (5) n'est pas encore implémenté, ceci sera potentiellement réalisé par la suite.

En voici une version fonctionnelle simplifiée :

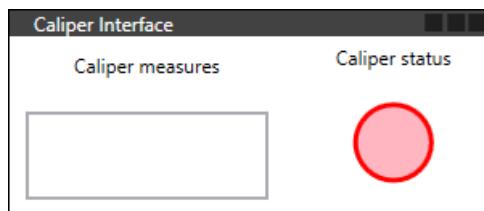


Figure 14: WPFApp épurée

3.3 Développement de la réalité augmentée

Microsoft recommande de travailler avec le logiciel Unity. Unity est un moteur de jeu multiplateforme développé par Unity Technologies. Cet *engine*⁷ a la particularité de fonctionner sur une variété de plateformes : Application PC, mobile, console ou encore de réalité virtuelle/augmentée.

Les HoloLens 2 ont été révélés en novembre 2019, mais sont encore maintenant disponible uniquement pour une utilisation commerciale. Ceci a pour conséquence d'impacter considérablement les ressources disponibles comme le démontre cette figure :

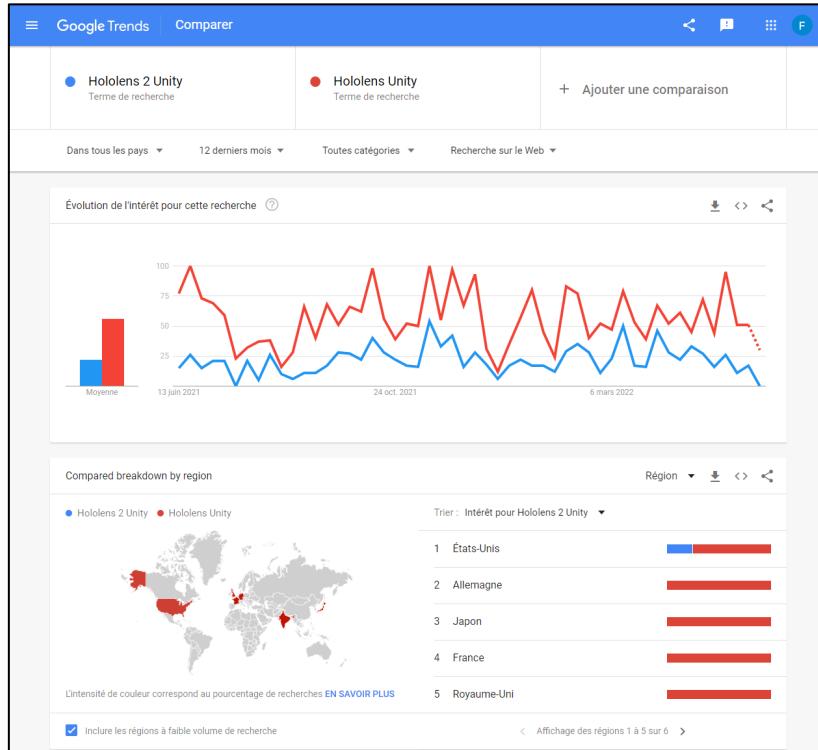


Figure 15: Recherches HoloLens Google Trends [5]

[Google Trends](#) est un outil statistique indiquant le nombre de recherches pour un sujet quelconque. De nombreuses librairies sont développées par des entreprises privées et le contenu n'est pas toujours disponible ni référencé.

Toutefois, la commercialisation pour le grand public des HoloLens 2 est attendue pour un futur proche en ce jour du 11.06.2022.

⁷ Framework software conçu pour le développement de jeux vidéo

3.3.1 Configuration du logiciel Unity

L'environnement Unity seul ne permet malheureusement pas de développer des applications de réalité augmentée. Il est nécessaire d'ajouter des plugins adaptés. Microsoft a déployé le logiciel *Mixed Reality Feature Tool*, alias MRTK. Elle a pour but de préconfigurer l'environnement Unity et d'y ajouter des packages spécifiques pour le développement d'applications sur le casque HoloLens 2. Ceci inclus par exemple des boutons, des interactions avec des objets etc. Un bouton est affiché ci-dessous.

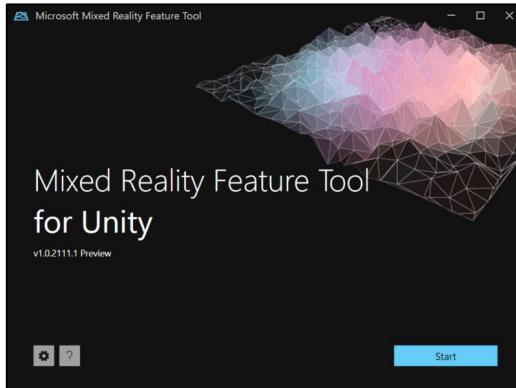


Figure 16: Toolbox MRTK

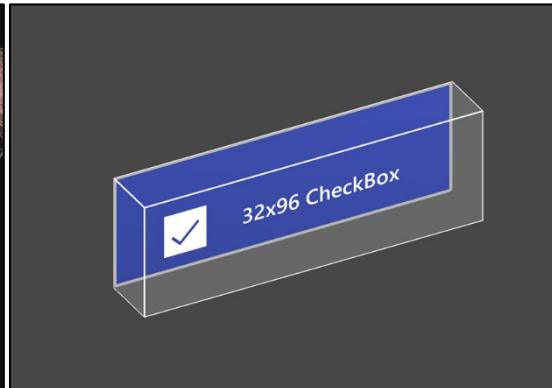


Figure 17: Exemple UI MRTK

A ceci s'ajoute OpenXR, une norme ouverte et libre de droits permettant l'accès aux plates-formes et dispositifs de réalité virtuelle et de réalité augmentée. Cette norme vise à fournir à terme deux composants : une API destinée aux développeurs d'applications et une couche de périphérique destinée au matériel de réalité virtuelle ou de réalité augmentée, présentant une interface d'abstraction avec le périphérique lui-même. Ceci Permet une homogénéisation de développement entre tous les appareils de réalité augmentée/virtuelle.

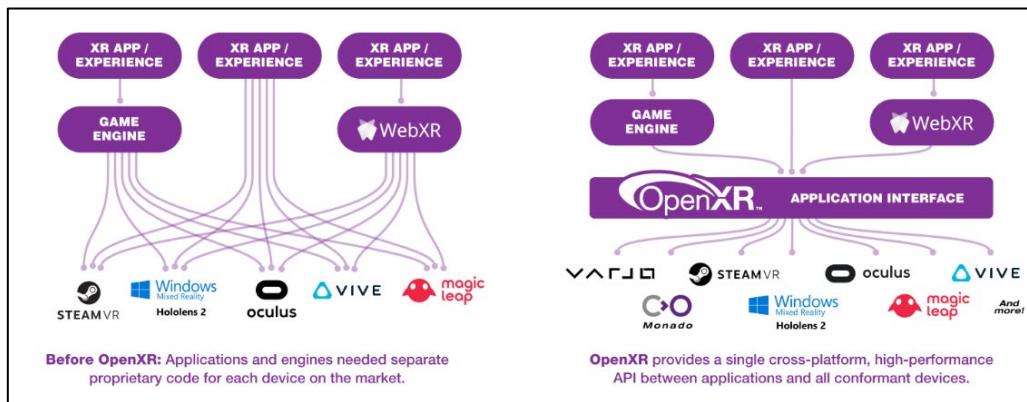


Figure 18: Overview OpenXR [6]

3.3.2 Chargement sur le casque

Le téléchargement de l'application sur l'HoloLens 2 est une opération chronophage. En effet, celle-ci se décompose en plusieurs processus :

- La création d'une solution Visual Studio
- Le *build* de cette solution
- Le déploiement dans le casque

En moyenne, la somme de ces opérations dure une dizaine de minutes. Ce temps est considérable et il est nécessaire de vérifier le code en le simulant dans Unity au préalable. Cette durée est notamment expliquée par le nombre de packages importés via MRTK, nombreux de ceux-ci ne sont pas utilisés, mais sont tout de même compilés.

```

Générer Déboguer Test Analyser Outils Extensions
Générer la solution Ctrl+ Maj+B
Regénérer la solution
Déployer la sélection
Nettoyer la solution
Exécuter l'analyse du code sur la solution Alt+F11
Générer la sélection Ctrl+B
Regénérer la sélection
Déployer la sélection
Nettoyer la sélection
Générer en tâche de fond...
Gestionnaire de configurations...

```

```

1>ObjectFiles: 872 of which compiled: 872
1> Time Compile: 76859 milliseconds I12CppCQNs90.cpp
1> Time Compile: 76607 milliseconds I12CppCQNs6.cpp
1> Time Compile: 53479 milliseconds Generics27.cpp
1> Time Compile: 45060 milliseconds System3.cpp
1> Time Compile: 44859 milliseconds I12CppCQNs42.cpp
1> Time Compile: 27653 milliseconds mscorlib6.cpp
1> Time Compile: 25843 milliseconds Unity.TextMeshPro3.cpp
1> Time Compile: 23720 milliseconds Unity.TextMeshPro4.cpp
1> Time Compile: 23627 milliseconds Unity.TextMeshPro2.cpp
1> Time Compile: 21995 milliseconds I12CppCQNs104.cpp
1>Total compilation time: 241361 milliseconds.

```

Figure 19: Génération de la solution Unity

Ce contenu n'est pas sans conséquences pour le casque, il sera nécessaire d'étudier les ressources requises minimales afin d'optimiser les performances de l'appareil.

3.3.3 Remoting

Une alternative au téléversement standard se nomme le *Remoting*, elle est implémentée et compatible entre Unity et le casque HoloLens 2, mais son comportement est totalement instable. L'image s'agit dans tous les sens, se fige avant que la communication ne *crashe*.

3.3.4 Prototype visuel

L'application Unity est encore au stade de développement, mais elle est en ce moment capable d'effectuer des requêtes GET et PUT. La langue de l'interface sera possiblement francisée ou paramétrable.

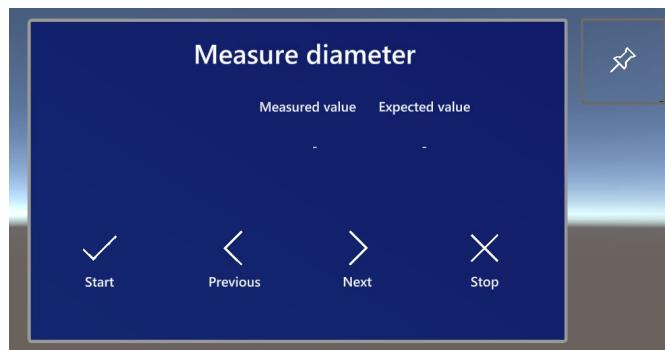


Figure 20: Aperçu visuel Unity

- **Start** effectue une requête PUT et reçoit en retour la première cotation de la liste.
- **Previous** effectue une requête PUT et accède à l'élément précédent de la liste
- **Next** effectue une requête PUT et accède à l'élément suivant de la liste
- **Stop** effectue une requête PUT termine la séquence de fonctionnement

Ce contrôle visuel est importé des packages MRTK et offre la possibilité à l'utilisateur *d'épingler* spatialement cette fenêtre ou bon lui semble.

3.4 Tracking

Le *tracking* est l'action de suivre un élément dynamiquement dans l'espace (position, orientation). Ceci permettrait à l'opérateur de saisir la pièce à mesurer dans sa main. Cela apporte un confort supplémentaire pour l'utilisateur et permet de s'abstraire du posage fixe.

C'est un concept difficile à développer par soi-même, car il nécessite du traitement d'image en temps réel et un repérage spatial dynamique. Ces algorithmes emploient couramment du Machine Learning.

Plusieurs entreprises mettent à disposition des SDK implémentant cette technologie, mais ces services sont payants. Deux offres ont été réalisées auprès des entreprises Visometry (VisionLib) et PTC (Vuforia). Toutes les deux proposent un essai gratuit, mais ils se sont résulté en un échec. Des erreurs de lancement apparaissent et le débogage est inextricable. Une seconde tentative sera effectuée si le temps restant le permet.



Figure 21: Logo VisionLib [7]

Figure 22: Logo Vuforia [8]

3.5 Posage fixe

L'alternative au système de *tracking* est un référentiel fixe. Lorsque le casque est référencé sur le posage, il sera permis d'ajouter des éléments holographiques à l'emplacement souhaité. Pour cela, un posage a été réalisé au FabLab de la HEIG-VD. Sa conception paraissant à première vue basique a été étudiée rigoureusement.

Les deux repères rouges placés sur la même colonne mesurent la même longueur. Nous constatons que la flèche supérieure ne coïncide pas avec le quadrillage. Ces distorsions peuvent être corrigées à l'aide d'une transformation homographique. L'homographie 2D est une transformation linéaire entre deux plans projectifs. Ceci signifie qu'un point quelconque peut-être projeté dans le plan par une transformation. L'homographie cherche à minimiser cet écart pour "redresser" l'image.

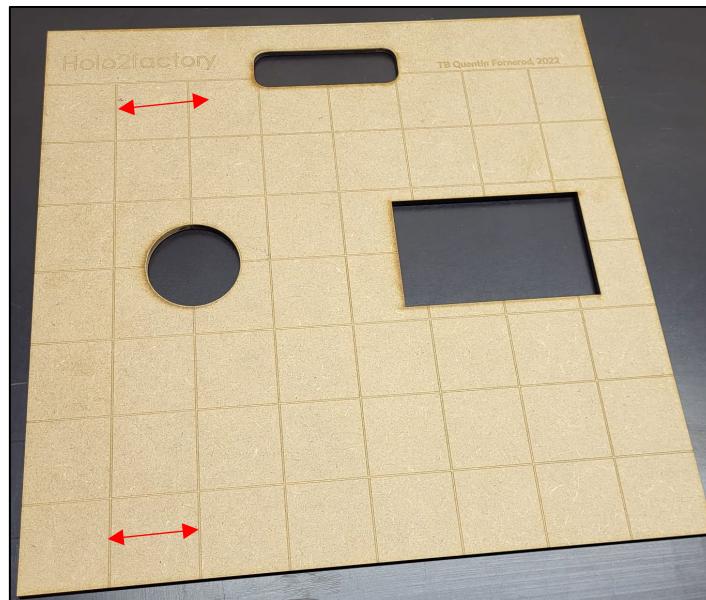


Figure 23: Posage fixe (plateau)

3.6 Serveur ASP

L'application Desktop contient à sa racine un Controller C#. C'est un framework d'application web ayant pour but de recevoir et d'envoyer des requêtes. Il appartient au modèle MVC (Model – View – Controller). Son fonctionnement sera détaillé dans le développement du projet.

3.7 Requêtes API

Les différentes requêtes sont à transmettre au *controller* présent dans l'application C#. Le mot REQUETE sera remplacé par un des termes proposés dans le tableau des API.

Url	Description	Type
https://localhost:7114/api/AR/REQUETE	Requête locale	Sécurisé
http://localhost:5114/api/AR/REQUETE	Requête locale	Non sécurisé
http://192.168.137.1/api/AR/REQUETE	Requête distante	Non sécurisé

Tableau 14: Liste des URL du serveur ASP

Voici la liste des requêtes actuellement disponibles, cette liste sera complétée lors du développement du projet.

Requêtes disponibles	API	Description	Objet sérialisé envoyé/reçu	Type
GetMeasure		Reçoit la mesure formatée	Classe SingleMeasure	GET
PutMeasure		Envoie la mesure formatée	Classe SingleMeasure	PUT
BooleanCommand		Envoie une commande (appui sur un bouton dans l'application du casque)	Classe BooleanCommand	PUT
GetCotation		Reçoit les coordonnées complètes d'une cote	Classe PieceCotation	GET
CaliperStatus		Envoie l'état de connexion du pied à coulisse (non implémenté)	Classe CaliperStatus	PUT

Tableau 15: Liste des requêtes API disponibles

Aperçu du résultat après une requête effectuée depuis un navigateur quelconque. Il est nécessaire d'être sur le même réseau que le serveur ASP.

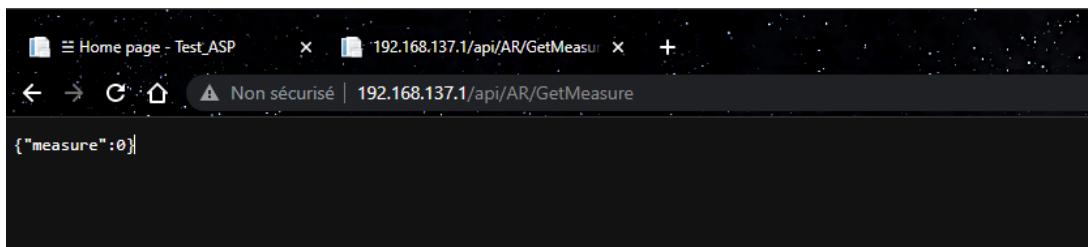


Figure 24: Résultat requête API dans le navigateur

3.7.1 NewtonSoft

C'est une librairie très intéressante qui permet de sérialiser des objets au format JSON. Elle peut également déserialiser un objet, c'est-à-dire instancier un objet et l'initialiser à partir de données contenues dans un fichier. Son utilisation dans les différents programmes de ce TB permet une homogénéisation des requêtes.

3.7.2 ReqBin

Le temps de déploiement de l'application sur le casque étant fastidieux, il est souhaitable de tester les API préalablement. [ReqBin](#) est un outil de test d'API en ligne et gratuit.

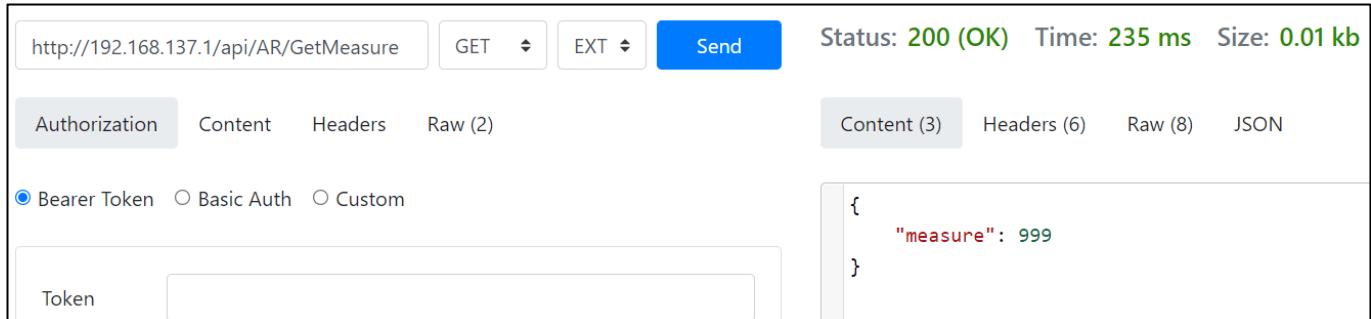


Figure 25: Test des API avec ReqBin

L'adresse utilisée pour ce test est <http://192.168.137.1/api/AR/GetMeasure>, le serveur répond par un objet sérialisé au format JSON. La valeur 999 n'est autre que la valeur par défaut d'une mesure. Elle a été choisie stratégiquement hors de la plage de mesure de l'appareil.

3.7.3 Woopsa

Woopsa est un protocole gratuit open-source développé par la société [Objectis](#). Il a pour but de transmettre des objets sérialisés au format JSON en se basant sur les fondements HTTP. Il est implantable dans Unity et a fonctionné en simulation. L'implémentation dans le casque s'est résulté en échec, il n'était pas possible de communiquer avec le serveur. Cet ajout nécessite une *dll* générée par une solution Visual Studio. Cependant, la génération de ce fichier était impossible, on m'a donc transmis "une dll qui devrait marcher" ainsi que d'autres dll telle que nancy.dll. Ceci fait beaucoup d'ajouts que je ne maîtrise pas, que je ne pourrai pas débuguer et c'est la raison pour laquelle j'ai décidé de me passer de cette surcouche performante.

3.8 Helix

Helix est un framework open source permettant d'inclure des éléments graphiques 3D dans une application WPF. Cette technologie pourrait être utilisée par le configurateur dans le but de visualiser la génération de nouvelles pièces. Malheureusement, Helix n'est pas à jour et ses dernières releases sont développées pour Visual Studio 2019 avec une version .NET 3.

Aperçu d'un référentiel spatial avec *Helix*

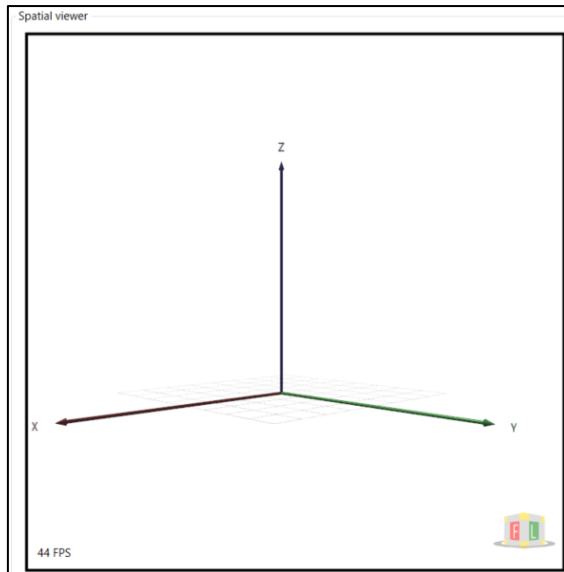


Figure 26: Référentiel spatial avec *Helix*

3.9 Blazor

Blazor est un framework web open source développé par Microsoft et permettant de créer des applications web en C#. Il pourrait être utilisé par le superviseur. Ceci lui permettra d'accéder à toutes les données de mesure réalisées par l'opérateur. Un prototype a été développé lors de la pré-étude :

Date	Temp. (C)	Temp. (F)	Summary
9/5/2020	-4	25	Freezing
9/6/2020	31	87	Mild
9/7/2020	52	125	Hot
9/8/2020	19	66	Mild
9/9/2020	-9	16	Freezing

Figure 27: Premier test avec Blazor

3.10 Github

Ce TB n'étant pas confidentiel, les codes sources rédigés seront présents sur mon GitHub à l'adresse suivante :

<https://github.com/fornerod-quentin/holo2factory>



Figure 28: QR code GitHub Holo2Factory

Le *build* de la solution Unity n'y sera pas déposé car la taille de son dossier est d'environ 8 Gb. Pour des raisons de confidentialités, les licences Vuforia seront également supprimées.

3.11 Conclusion de la pré-étude

Unity est, contrairement à certains articles, très compliqué à prendre en main. Ce logiciel étant multiplateforme, sa configuration en devient alambiquée, raison pour laquelle Microsoft a développé un logiciel effectuant une pré-configuration de l'éditeur. L'HoloLens 2 étant récent, sa documentation est modique et la communauté de développeur chimérique. N'ayant (plus) personne pour m'épauler dans cette recherche, je passe un temps considérable dans le débogage.

Le langage C# étant polyvalent et couvrant toute la gamme d'activité (applications console, web, interface graphiques), il est nécessaire de rendre les informations cross-plateforme compatibles. Le .NET 6 est sorti fin 2021 et n'est de loin pas implémenté dans tous les *frameworks*. C'est le cas du WPF, il est possible de faire un *upgrade* mais celui-ci s'est résulté en échec. Ceci souligne d'avantage l'intérêt d'ajouter un *controller C#* permettant de faire la transition entre les applications via des requêtes http sans se soucier des versions. Unity utilise sa propre librairie JSON, mais son comportement est différent d'une librairie standard *Newtonsoft*. Une syntaxe identique fonctionne avec *Newtonsoft*, mais résulte un comportement non concluant avec la librairie native de Unity. L'intégration de *Newtonsoft* dans Unity permet une homogénéisation des applications, car elle est également utilisée dans l'application WPF.

Cette pré-étude a principalement permis de cibler les besoins utilisateurs, de définir l'architecture de l'application et de survoler les technologies suscitant un intérêt pour le développement du projet.

4 Etude du projet

4.1 Introduction

L'étude du projet est décomposée en différentes étapes. Elle contient des explications concernant la modification de l'architecture logicielle. Chaque application sera détaillée, les syntaxes de code pertinentes seront explicitées et un rendu visuel sera présenté. S'en suivra une synthèse développée de ce travail.

4.2 Restructuration de l'architecture logicielle

Par suite de la pré-étude du projet, quatre applications distinctes se sont dessinées. Il était initialement prévu de développer trois applications :

- Application Desktop (Communication avec le casque, lecture des mesures du pied à coulisse)
- Application Unity (Développement d'une solution en réalité augmentée)
- Application Blazor (Affichage des mesures sur une page web)

Le changement s'explique de la nécessité de réaliser une application WPF pour la lecture des mesures du pied à coulisse. Ce n'est pas la seule modification d'architecture logicielle présente dans ce projet. En effet, l'avancée du projet a permis de développer l'application *optionnelle* Holo2factoryconfigurator. Cette application permet au configurateur de générer une liste de cotation et de les placer spatialement. Le configurateur n'ayant pas le même rôle et emplacement de travail que l'opérateur, il est donc légitime que cette application soit distincte de l'application WPF du pied à coulisse. Les échanges de données entre les différents acteurs logiciels se faisant par requête API, ce serveur peut être distant. C'est également le cas du serveur Blazor. Ceci a donc impliqué une fusion des deux applications.

- WPF (Reçoit une mesure du pied à coulisse et la transmet au Controller)
- Unity (Solution en réalité augmentée, échanges des informations avec le Controller)
- Holo2factory configurator (Permet la génération de recettes)
- ASP+Blazor (Héberge le Controller C# ainsi que le serveur web Blazor, gère la séquence de mesure)

Ces différentes applications seront documentées à l'aide d'un diagramme UML, une description, un rendu visuel, des extraits de codes importants ou des syntaxes complexes ainsi que des éléments complémentaires selon les besoins.

Note : concernant les diagrammes UML, la visibilité des attributs et des méthodes peut légèrement différer par rapport à celle des codes sources remis.

4.3 Développement ASP

4.3.1 Diagramme UML

L'objet *ARController* est l'élément qui interagit avec l'extérieur. Il reçoit des requêtes GET ou PUT. En fonction de cela, il met à jour la machine d'état présente dans *ProductionManager*. Ce dernier gère les cotations à envoyer en fonction de la recette chargée.

Ce diagramme est décomposé en deux parties, voici la première :

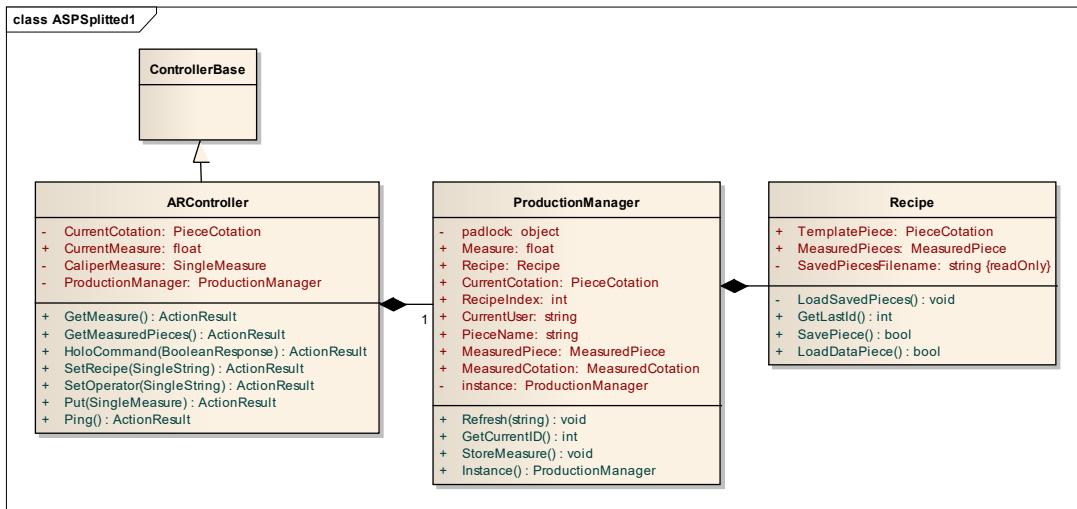


Diagramme UML 11: ASP (partie 1)

On distingue deux types de cotations :

- *MeasuredCotation* : les cotations mesurées (elles contiennent des informations utiles pour le **superviseur et l'opérateur**).
- *PieceCotation* : elle contient des informations pour l'affichage dans l'application Unity. Elle est générée par le **configurateur**.

Il aurait été possible de généraliser ces deux cotations car elles contiennent chacune la tolérance, le type et la valeur attendue. Ceci a volontairement été écarté car les classes sont (dé)sérialisées et ceci complexifie ce processus. Ce comportement était instable dans Unity.

Voici la seconde partie du diagramme :

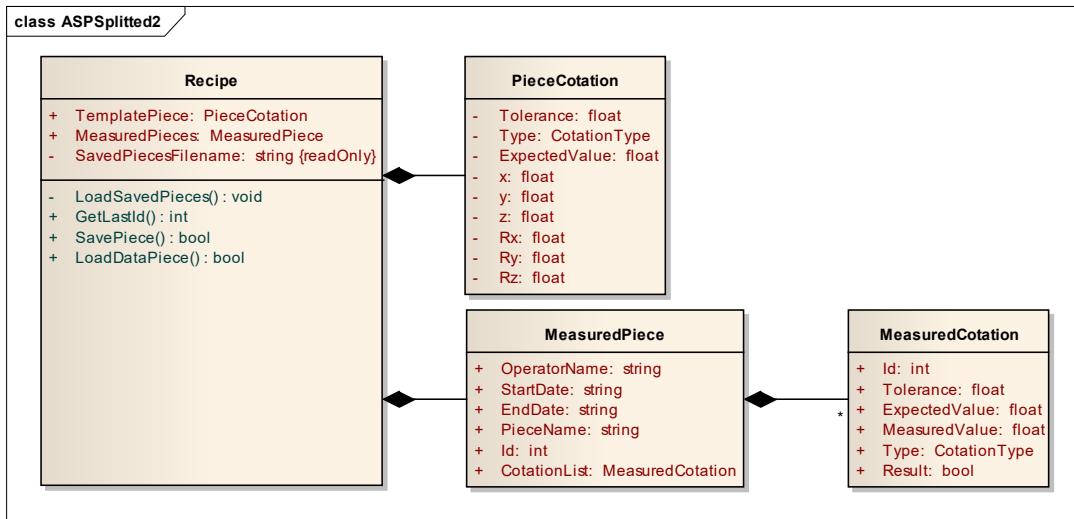


Diagramme UML 12: ASP (partie 2)

La création des serveurs ASP et Blazor se fait en instantiant une *WebApplication*.

```

var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddRazorPages();
builder.Services.AddServerSideBlazor();
var app = builder.Build();
  
```

Extrait de code 1: Création d'une WebApplication

Il est dès lors possible de configurer le Controller ainsi que de générer le mapping des pages Razor/Blazor.

```

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=ARController}/{action=Index}/{id?}");
app.MapBlazorHub();
app.MapRazorPages();
  
```

Extrait de code 2: Routing du Controller

La configuration des serveurs étant faite, il est désormais nécessaire de s'attarder à la séquence principale du programme. Le *Controller* instancie dans son constructeur un objet *ProductionManager* en suivant le design pattern Singleton.

La syntaxe Singleton C# diffère légèrement du langage C++. Le concept est la création d'une classe *sealed* (dont on ne peut pas hériter), puis d'y déclarer une instance avec un verrou.

```
public sealed class ProductionManager
{
    private static ProductionManager? instance = null;
    private static readonly object padlock = new();
```

Extrait de code 3: Création de la classe ProductionManager

Lors de linstanciation dans le Controller ASP, nous appelons la méthode de classe statique *instance* qui retournera une nouvelle instance si cette dernière est *null*. Dans le cas contraire, une nouvelle et unique instance sera générée.

```
//Singleton instance
ProductionManager ProductionManager = ProductionManager.Instance;
```

Extrait de code 4: Récupération de linstance ProductionManager

L'objet *lock* protège d'un accès concurrent d'un autre thread à linstance.

```
public static ProductionManager Instance
{
    get
    {
        lock (padlock)
        {
            if (instance == null)
            {
                instance = new ProductionManager();
            }
            return instance;
        }
    }
}
```

Extrait de code 5: Structure d'un singleton en C#

Le Controller effectuera désormais des requêtes PUT et GET en fonction des instructions du *ProductionManager*.

Chaque *Action* du Controller est précédée par des attributs de métadonnées spécifiant le type de requête ainsi que le nom de l'action. Concernant l'exemple ci-dessous, cela signifie qu'il sera nécessaire d'effectuer une requête de type GET et que l'URL de la requête sera (dépend du réseau sur lequel le serveur est lancé) <http://127.0.0.1:5114/api/AR/GetMeasure>. Lors de la réception de la requête, le Controller va récupérer la dernière valeur émise par l'application WPF puis la stocker dans le *ProductionManager*, ceci garantit une concordance des données entre le casque HoloLens et les valeurs prochainement enregistrées. La mesure est transmise via la fonction *Ok()* qui va sérialiser l'objet, en l'occurrence une variable de type *float*. La valeur de la mesure étant **toujours** vérifiée au préalable, seule une requête *Ok()* suffit.

```
[HttpGet]
[ActionName("GetMeasure")]
0 références
public ActionResult GetMeasure()
{
    CaliperMeasure.Measure = CurrentMeasure;
    ProductionManager.StoreMeasure(CurrentMeasure);
    return Ok(CaliperMeasure);
}
```

Extrait de code 6: Exemple d'une requête GET

Le schéma d'une requête PUT est très similaire, elle est précédée par le type de requête ainsi que son nom d'*Action* respectif. Dans le cas suivant, nous recevons un objet de type *SingleString* qui comme son nom l'indique, contient une variable de type *string*. Se suit le chargement de la recette correspondant au nom reçu. Si cette opération s'effectue avec succès (le fichier existe et le formatage des données est correct), la recette sera transmise au *ProductionManager* et la réponse au casque HoloLens sera positive. Dans le cas contraire, on indiquera que la requête n'a pas abouti et que ce nom n'existe pas. Une boîte de dialogue sera affichée sur le casque pour avertir l'opérateur.

```
//Choose correct recipe upon piece name, NotFound if unknown
[HttpPost]
[ActionName("PieceSelection")]
0 références
public ActionResult SetRecipe(SingleString pieceName)
{
    if (ProductionManager.Recipe.LoadDataPiece(pieceName.Name!))
    {
        ProductionManager.PieceName = pieceName.Name!;
        return Ok();
    }
    return NotFound();
}
```

Extrait de code 7: Exemple d'une requête PUT

D'avantage de requêtes existent, mais elles ne seront pas explicitées

Requêtes API disponibles	Description	Type
GetMeasure	Reçoit la mesure formatée	GET
PutMeasure	Envoie la mesure formatée	PUT
BooleanCommand	Envoie une commande (appui sur un bouton dans l'application du casque)	PUT
GetCotation	Reçoit les coordonnées complètes d'une cote	GET
PieceSelection	Charge les données de la pièce choisie	PUT
SelectUser	Sélectionne l'utilisateur	PUT
Ping	Envoie une requête <i>Ok()</i>	PUT
CaliperStatus	Envoie l'état de connexion du pied à coulisse (non implémenté)	PUT

Tableau 16: Liste des requêtes API disponibles (complétée)

Le déroulement de la séquence est principalement dépendant de la requête *BooleanCommand* qui transmettra les consignes :

- Start
- Stop
- Next
- Previous

4.3.2 Start

Un objet de type *MeasuredPiece* est instancié et les paramètres tels que le nom d'utilisateur, le nom de la pièce etc. sont affectés. La cotation actuelle est sélectionnée (la première de la liste).

```
if (command == "Start")
{
    RecipeIndex = 0;
    //create a new measured piece
    MeasuredPiece = new MeasuredPiece
    {
        CotationList = new List<MeasuredCotation>(),
        PieceName = PieceName,
        OperatorName = CurrentUser,
        Id = Recipe.GetLastId() + 1,
        StartDate = DateTime.Now.ToString(System.Globalization.CultureInfo.CreateSpecificCulture("fr-FR"))
    };
    CurrentCotation = Recipe.TemplatePiece.ElementAt(0);
}
```

Extrait de code 8: Démarrage de la séquence

Il est dès lors possible de remplir la liste avec informations connues. L'intérêt de générer cette liste dès le début et non en *runtime* est la simplification des interactions avec les boutons *next* et *previous*. En effet, lors de la navigation, il serait fâcheux de dépasser d'accéder à un élément hors de la liste (bien qu'impossible) et cela réduit les opérations pendant la séquence.

```
//fill cotation list with all known data
for (int i = 0; i < Recipe.TemplatePiece.Count; i++)
{
    MeasuredCotation newCot = new()
    {
        ExpectedValue = Recipe.TemplatePiece.ElementAt(i).ExpectedValue,
        Type = (MeasuredCotation.CotationType)Recipe.TemplatePiece.ElementAt(i).Type,
        Id = i,
        Tolerance = Recipe.TemplatePiece.ElementAt(i).Tolerance,
        //default value
        MeasuredValue = 999F
    };
    MeasuredPiece.CotationList.Add(newCot);
}
```

Extrait de code 9: Initialisation des cotations mesurées

4.3.3 Stop

Le principe de cette opération est très basic, la date de fin de séquence est inscrite dans la pièce mesurée puis la pièce est enregistrée dans un fichier JSON.

4.3.4 Next

Sauve la mesure du pied à coulisse ainsi que le résultat (Ok/Nok) dans la pièce mesurée puis incrémente l'index de recette pour autant que ce ne soit pas la dernière mesure. Dans le cas contraire, la cotation envoyée contiendra un ID de -1 signifiant que la séquence est terminée. Le bouton *next* deviendra indisponible dans l'interface holographique.

4.3.5 Previous

Par analogie à sa commande antagonique, le bouton *previous* n'est disponible uniquement lorsque l'index de recette est supérieur à zéro. Si tel est le cas, la cotation précédente ainsi que la valeur mesurée seront transmises au casque HoloLens.

4.3.6 Chargement des pièces enregistrées

Lors du lancement du programme, plus précisément lors de l'instanciation de la recette et lors de l'enregistrement d'une pièce mesurée, les pièces précédemment enregistrées sont chargées. Si le fichier JSON existe, les pièces seront chargées. Dans le cas contraire, rien ne se passe et un nouveau fichier sera créé lors de l'enregistrement de la première pièce.

La désérialisation est un processus complexe qui nécessite un fichier contenant une syntaxe rigoureuse. Dans ce cas précis, le but est d'extraire directement une liste de *MeasuredPiece* contenant elle-même une liste de *MeasuredCotation*.

```

private void LoadSavedPieces()
{
    if (File.Exists(SavedPiecesFilename))
    {
        try
        {
            string jsonString = File.ReadAllText(SavedPiecesFilename);
            MeasuredPieces = JsonConvert.DeserializeObject<List<MeasuredPiece>>(jsonString);

        }
        catch (Newtonsoft.Json.JsonException e)
        {
            Console.WriteLine(SavedPiecesFilename + " is corrupted");
            Console.WriteLine(e.Message);
        }
        catch (Exception e)
        {
            Console.WriteLine("Unable to load the file");
            Console.WriteLine(e.Message);
        }
    }
}

```

Extrait de code 10: Chargement des pièces mesurées

4.3.7 Sauvegarde des pièces mesurées

Lors du processus inverse, la pièce courante est ajoutée à la liste précédemment chargée, puis l'ensemble est sérialisé puis enregistré dans un fichier JSON.

```

public void SavePiece(MeasuredPiece measuredPiece)
{
    MeasuredPieces.Add(measuredPiece);
    try
    {
        string jsonString = System.Text.Json.JsonSerializer.Serialize(MeasuredPieces);
        File.WriteAllText(SavedPiecesFilename, jsonString);
    }
    catch (Newtonsoft.Json.JsonException e)
    {
        Console.WriteLine("MeasuredPieces has incorrect format");
        Console.WriteLine(e.Message);
    }
    catch (Exception e)
    {
        Console.WriteLine("Unable to write in the file");
        Console.WriteLine(e.Message);
    }
    LoadSavedPieces();
}

```

Extrait de code 11: Sauvegarde des pièces mesurées

4.3.8 Chargement d'une recette

Lors de la réception du nom de la pièce, le fichier contenant les cotations est chargé. La syntaxe est similaire au chargement de pièces mesurées.

```
public bool LoadDataPiece(string pieceName)
{
    string fileName = pieceName + ".json";
    if (File.Exists(fileName))
    {
        try
        {
            string jsonString = File.ReadAllText(fileName);
            if (new FileInfo(fileName).Length == 0)
            {
                return false;
            }
            TemplatePiece = System.Text.Json.JsonSerializer.Deserialize<List<PieceCotation>>(jsonString)!";
            return true;
        }
        catch (Newtonsoft.Json.JsonException e)
        {
            Console.WriteLine(fileName + " is corrupted");
            Console.WriteLine(e.Message);
        }
        catch (Exception e)
        {
            Console.WriteLine("Unable to load the file");
            Console.WriteLine(e.Message);
        }
    }
    return false;
}
```

Extrait de code 12: Chargement d'une recette selon le nom de la pièce

4.4 Développement Blazor

4.4.1 Récupération des données de mesure

Le serveur Blazor se trouve dans la même application que le serveur ASP, mais ces deux éléments restent distincts. L'unique interaction entre les deux services se fait lors de la requête API `GetMesuredPieces`. A ce moment, les pièces sont *dumpées* en JSON lors de la fonction `Ok()`.

```
//Get the list of all measured pieces (Request "normally" done by the Blazor server)
[HttpGet]
[ActionName("GetMesuredPieces")]
0 références
public ActionResult GetMesuredPieces()
{
    return Ok(ProductionManager.Recipe.MeasuredPieces);
}
```

Extrait de code 13: Récupération des pièces mesurées (requête API)

La requête côté mesure est réalisée dans un fichier *razor*. Le code C# est encapsulé dans des balises @code{...}. Elle s'exécute lors d'un appui sur le bouton "*Fetch measures*".

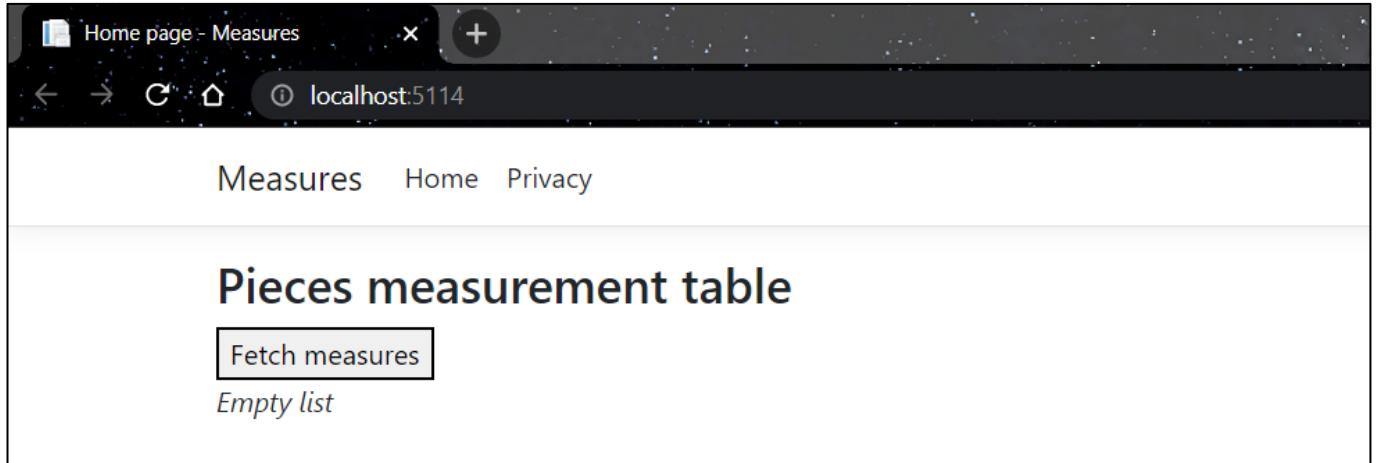


Figure 29: Header application Blazor

4.4.2 Requête GET

Le code exécuté soumet une requête de type GET au Controller ASP.

```
@code {
    public string? JsonString { get; set; }
    private List<MeasuredPiece>? MeasuredPieces;
    protected async Task Toggle()
    {
        var url = "http://127.0.0.1:5114/api/AR/GetMesuredPieces";
        //Has to be "System.Net.WebRequest..." or it won't compile even though Intellisense says it's ok
        var httpRequest = (System.Net.HttpWebRequest)System.Net.WebRequest.Create(url);
        using (System.NetWebResponse myResponse = await httpRequest.GetResponseAsync())
        {
            using (StreamReader sr = new StreamReader(myResponse.GetResponseStream(), System.Text.Encoding.UTF8))
            {
                JsonString = sr.ReadToEnd();
                MeasuredPieces = DeserializeObject<List<MeasuredPiece>>(JsonString)!;
            }
        }
    }
}
```

Extrait de code 14: Requête HTTP pour la récupération des données de mesure

4.4.3 Virtualize

Le code HTML est complété de syntaxes Blazor très performantes. L'affichage d'une collection d'objet peut s'avérer fastidieux en Javascript, il est exigible de réaliser une boucle et de parcourir toutes les données. Blazor admet un composant nommé "Virtualize" qui substitue une syntaxe C# *foreach*.

```
<Virtualize Items="MeasuredPieces" Context="Piece">
```

Extrait de code 15: Syntaxe "Virtualize" Blazor

Une référence est liée à la liste de pièces *MeasuredPieces* et nous pouvons accéder simplement, à chaque objet ainsi qu'à ses membres.

```
<td>@Piece.PieceName</td>
<td>@Piece.Id</td>
<td>@Piece.OperatorName</td>
<td>@Piece.StartDate</td>
<td>@Piece.EndDate</td>
```

Extrait de code 16: Syntaxe "Virtualize" Blazor (référence d'objet)

La liste des résultats contient une liste de pièces contenant elle-même une liste de cotations mesurées. Un embriquement de ces syntaxes s'applique correctement et satisfait les besoins.

4.4.4 Rendu visuel

Voici le rendu visuel des mesures, nous retrouvons tous les éléments nécessaires pour la supervision du travail de mesure. Chaque pièce est séparée par une bordure horizontale. On y retrouve pour chaque pièce :

- Son nom
- Son Id
- L'opérateur
- L'horodatage de début de séquence
- L'horodatage de fin de séquence
- Une liste de cotation (contenant elle-même un Id, un type, une valeur attendue, une valeur mesurée et son résultat (Ok si dans la tolérance, sinon Nok).

Pieces measurement table							
<input type="button" value="Fetch measures"/>							
Piece name	ID	Operator	Start date	End date	Cotation	Measured [mm]	Result
PenHolder	0	Laurent	05/07/2022 12:52:27	05/07/2022 12:53:16	ID	Type	Expected [mm]
					0	Diameter	30 ± 0.2
PenHolder	1	Laurent	05/07/2022 12:53:41	05/07/2022 12:55:18	ID	Type	Measured [mm]
					0	Diameter	30 ± 0.2
PenHolder	2	Didier	01/08/2022 17:53:55	01/08/2022 17:57:10	ID	Type	Measured [mm]
					0	Diameter	20 ± 0.3
PenHolder	3	Raoul	02/08/2022 14:52:46	02/08/2022 14:54:47	ID	Type	Measured [mm]
					0	Diameter	20 ± 0.3

Figure 30: Résultats des mesures Blazor

4.5 Application Unity

4.5.1 Diagramme UML

L'application contient un objet spécifique *ScriptObject*, c'est dans ce dernier que se trouve l'intégralité du code C#. Les fonctions de ce script suscitant un intérêt spécifique seront explicitées dans ce chapitre.

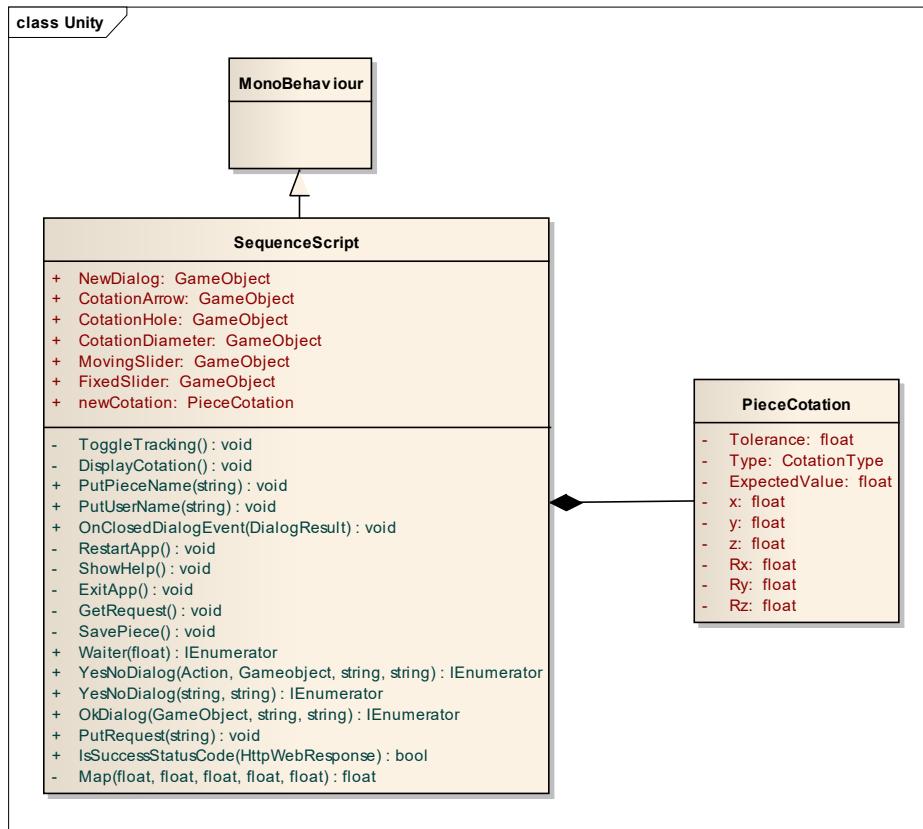


Diagramme UML 13: Unity

4.5.2 Fonctionnalités

Lors du lancement de l'application Unity dans le casque HoloLens, un premier menu apparaît. C'est la sélection de l'opérateur. Les rendus visuels suivants sont issus de la simulation dans Unity, ceci permet d'avoir un décor neutre. Le contenu dans le casque est toutefois identique. Des rendus "réels" sont présents plus tard (voir 4.11.3) dans le chapitre ainsi que dans la vidéo de présentation.

Chaque bouton émet un son lors de son activation et lors d'un focus visuel (*eye tracking*) maintenu, la commande vocale "select" active le bouton. Cet ajout permet à l'opérateur de garder le pied à coulisso proche de la pièce et lui évite de lever le bras. Les contrôles utilisateurs ont une taille ajustable, leur position est modifiable et sont évidemment configurables. D'autres commandes vocales telles que "Close", "Follow me" sont implémentées.

4.5.3 Choix de l'utilisateur

Lors de l'appui sur l'un des noms, une requête PUT est effectuée et elle contient le nom d'un bouton présent ci-dessous. En cas d'échec, une boîte de dialogue se présente devant l'opérateur, dans le cas contraire, la prochaine fenêtre se manifeste.

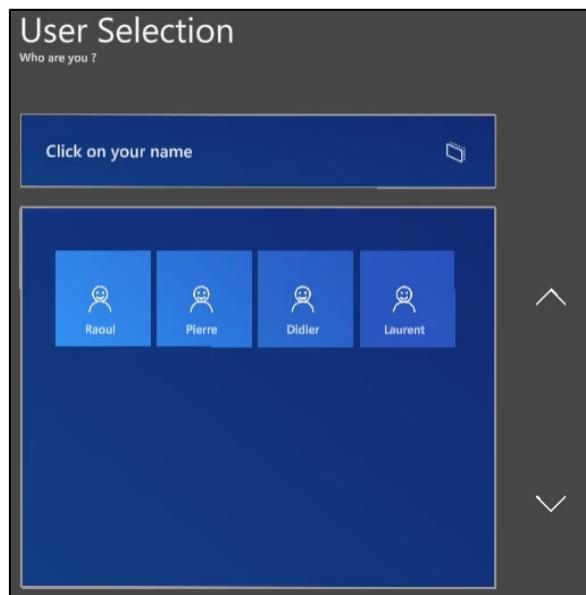


Figure 31: UI (sélection de l'utilisateur)

En cas d'erreur, voici le message de dialogue qui s'afficherait :

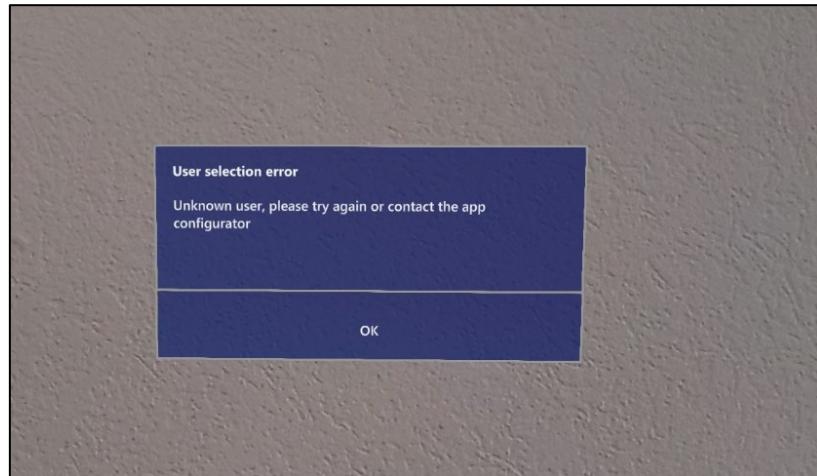


Figure 32: UI (Message d'erreur)

4.5.4 Choix de la pièce

Il est désormais possible de choisir la pièce à contrôler. Chaque pièce est représentée par un hologramme en 3D. Même principe qu'auparavant, lors d'un échec, un message d'erreur apparaît.

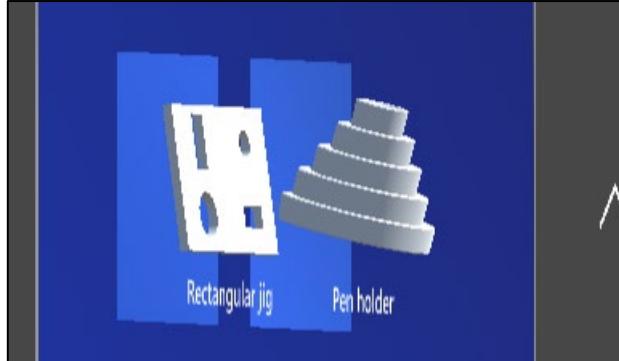


Figure 33: UI (focus choix de la pièce)

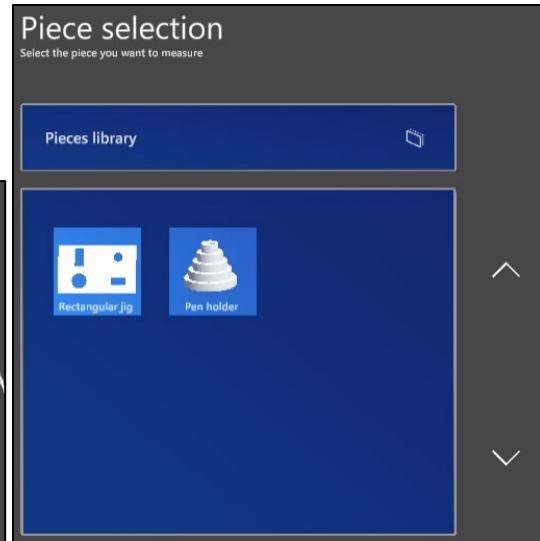


Figure 34: UI (choix de la pièce)

4.5.5 Séquence de mesure

La séquence de contrôle peut enfin démarrer, voici l'interface principal qui suivra l'opérateur :

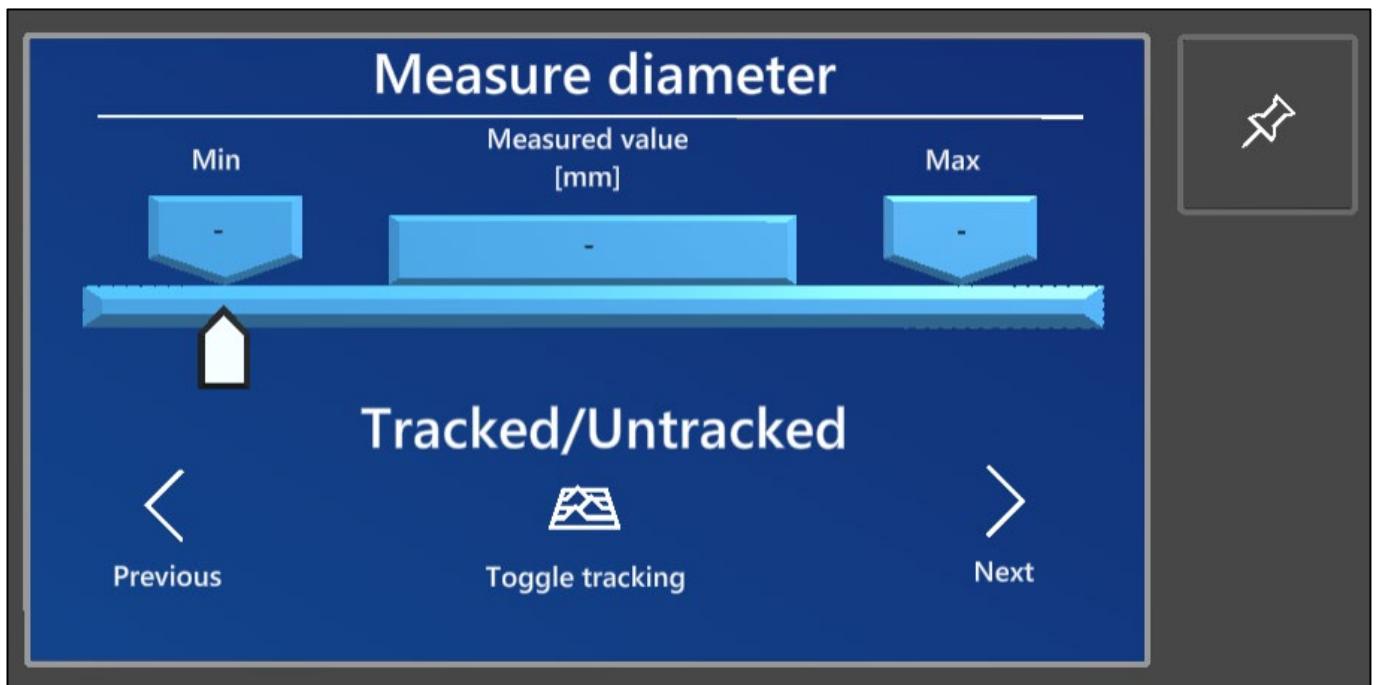


Figure 35: UI (bannière de la séquence de mesure)

Nous retrouvons la valeur minimale et maximale acceptée. Ces deux valeurs ne sont autres que la valeur attendue ajoutée ou soustraite de la tolérance de mesure. Lors de la transmission de la mesure via l'application *WPF*, cette dernière sera affichée en-dessous de "Measured value" et de couleur verte en cas de conformité, rouge sinon. Il y a également un *slider* qui indique l'emplacement de la mesure dans l'intervalle acceptable. En cas de valeur au-delà, le *slider* reste borné aux extrémités de la barre bleue. Trois boutons de commande sont disponibles ici, *Previous*, *Toggle tracking*, *Next*. *Toggle tracking* permet de désactiver la recherche de pièce dans l'espace si besoin. Les deux autres boutons ont déjà été explicités précédemment. Ce contrôle utilisateur étant importé de la *Toolbox* MRTK, il inclut des fonctionnalités intéressantes tel que l'ajustement de taille (*resize*), le déplacement selon le regard de l'opérateur ou alors une position fixe ajustable.

4.5.6 Aide dynamique (handmenu)

En tout temps, il est possible pour l'opérateur d'avoir de l'aide concernant la configuration du pied à coulisse ou le déroulement de la séquence. Il lui suffit de montrer la paume de main et un menu (*handmenu*) apparaîtra.

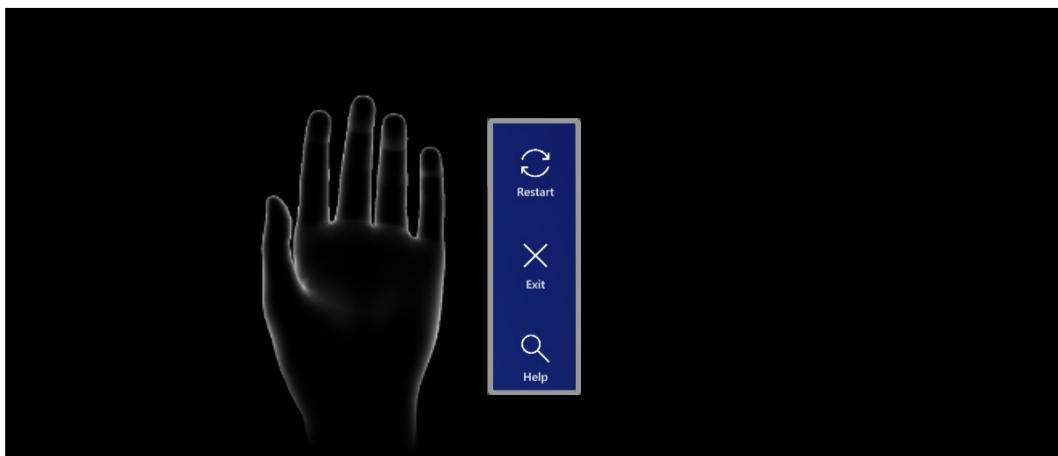


Figure 36: UI (hand menu)

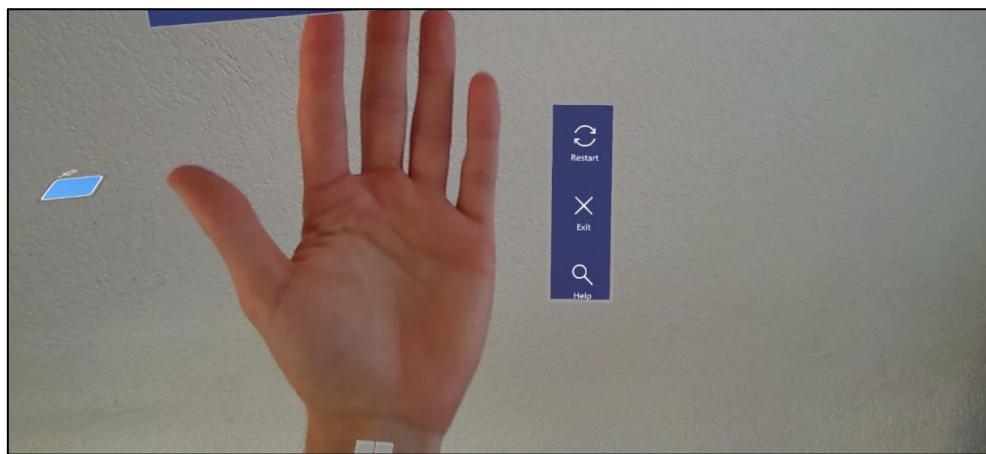


Figure 37: UI (hand menu) implémenté

Trois fonctions sont disponibles :

- Redémarrer la séquence de mesure (à confirmer sur la boîte de dialogue)
- Quitter la séquence et retourner à la sélection de l'opérateur
- Aide pour l'opérateur

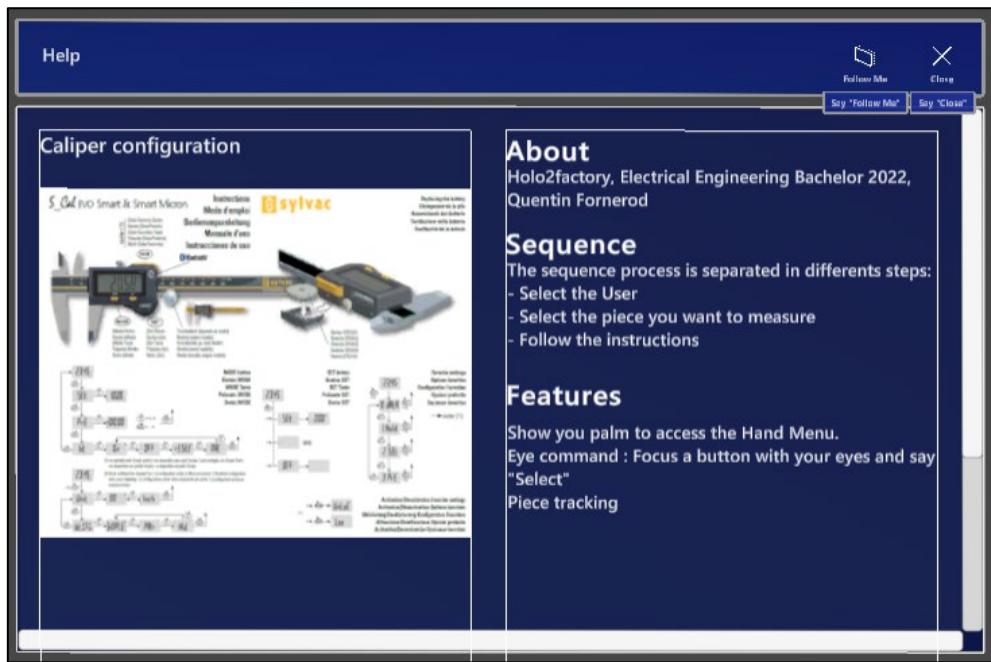


Figure 38: UI (guide utilisateur)

Note : la qualité de l'image est supérieure dans le casque HoloLens

4.5.7 Hiérarchie du programme

Tout élément présent dans la hiérarchie Unity est de type *GameObject*.

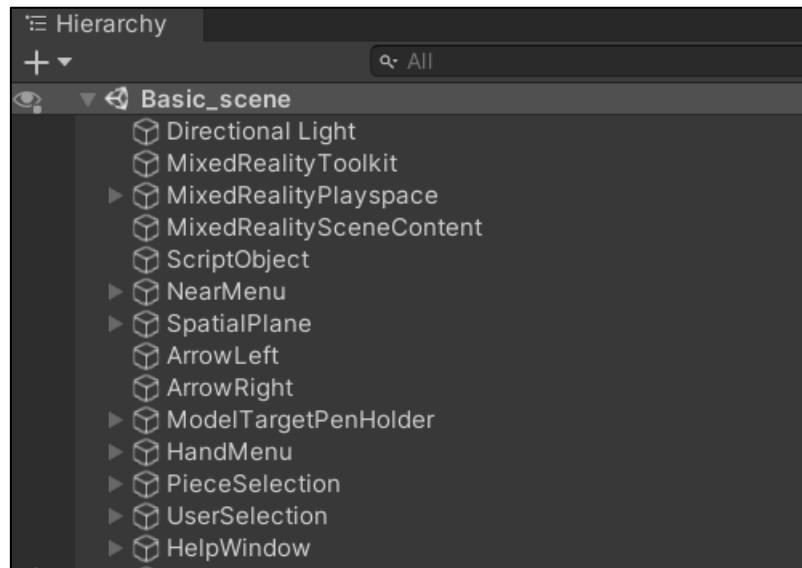


Figure 39: Hiérarchie logicielle Unity

L'un d'eux se nomme *ScriptObject*. Comme son nom l'indique, c'est lui qui contient l'intégralité du code C#. Chaque élément pourrait avoir un script dédié, mais ceci impliquerait des échanges de données entre chaque objet ainsi qu'une potentielle concurrence et des performances réduites.

4.5.8 Description du code

Un *GameObject* hérite (dans la majeure partie des cas) de la classe *MonoBehaviour*. Cette classe inclue des fonctionnalités fondamentales. Les plus utilisées sont :

- Start (S'exécute au lancement du programme)
- Update (S'exécute à chaque *frame*)

L'application étant parfois testée en simulation locale et parfois dans le casque, l'adresse IP diffère. Un moyen élégant de sélectionner dynamiquement la bonne adresse est la compilation conditionnelle. UNITY_WSA_10_0 définit la *Windows Store Apps* contenant notamment les application UWP (*Universal Window Platform*) incluant le HoloLens 2.

```
#if UNITY_WSA_10_0
    const string IP = "http://127.0.0.1:5114/api/AR/";
#else
    const string url = "http://192.168.137.1:80/api/AR/";
#endif
```

Extrait de code 17: Adresse IP dynamique selon l'appareil

Lors du démarrage (*Start*), chaque *GameObject* est récupéré et on y attache une fonction à certains de ces derniers. La variable *allInteractables* contient tous les objets de type *Interactable*, ceci correspond notamment aux boutons. L'ajout d'un *Listener* signifie un appel de la fonction *PutRequest* via le pointeur de méthode *delegate* dès l'appui sur le bouton. Le *delegate* ajoute la possibilité de passer un paramètre à la fonction.

```
void Start()
{
    var allInteractables = GameObject.FindObjectsOfType<Interactable>();
    foreach (var i in allInteractables)
    {
        if (i.name == "StartButton")
        {
            i.OnClick.AddListener(delegate { PutRequest("Start"); });
        }
    }
}
```

Extrait de code 18: Récupération des GameObject et affectation d'une fonction

Les champs de texte 3D sont récupérés en recherchant le nom du *GameObject* dans la hiérarchie

```
// Get TMP (TextMeshPro)
expectedMin = GameObject.Find("TMPExpectedMin").GetComponent<TextMeshPro>();
```

Extrait de code 19: Récupération des TextMeshPro

Une méthode issue de la classe *MonoBehaviour* est très intéressante, c'est la fonction *InvokeRepeating*. Ceci signifie que la fonction *GetRequest* sera appelée après une attente de 2s, chaque 300ms.

InvokeRepeating("GetRequest", 2.0f, 0.3f);

Extrait de code 20: Requête cyclique

Une autre fonction précédemment mentionnée (*Update*) est appelée à chaque frame. L'intervalle d'appel peut différer, mais cela ne pose pas de problème pour cette application. La moyenne oscille aux alentours des 60 fps.

Lors d'un *flag* de nouvelle cotation, le contenu visuel est ajusté (mesure attendue, valeur minimale, maximale etc.). Si la cotation reçue possède l'ID -1 (séquence terminée), alors une boîte de dialogue permettant la sauvegarde de la pièce apparaîtra. Si le *tracking* est actif, il sera réalisé à chaque frame et les cotations en dépendant seront spatialement ajustées.

L'envoi de requête est similaire aux requêtes effectuées depuis les autres applications. L'objet est premièrement sérialisé au format JSON puis converti en *string*.

```
async void PutPieceName(string buttonName)
{
    string url = IP + "PieceSelection";
    SingleString singleString = new SingleString()
    {
        Name = buttonName
    };
    string jsonString = Newtonsoft.Json.JsonConvert.SerializeObject(singleString);
```

Extrait de code 21: Création de la requête PUT

Envoi de la requête.

```
using (var writer = new StreamWriter(httpRequest.GetRequestStream()))
{
    writer.Write(jsonString);
}
```

Extrait de code 22: Envoi de la requête PUT

Obtention de la réponse puis récupération du statut. Génère une exception en cas d'échec.

```
using (WebResponse httpResponse = await httpRequest.GetResponseAsync())
{
    if (IsSuccessStatusCode((HttpWebResponse)httpResponse))
    {
        PieceSelection.SetActive(false);
        NearMenu.SetActive(true);
        PutRequest("Start");
    }
    using (StreamReader streamReader = new StreamReader(httpResponse.GetResponseStream()))
    {
        var result = streamReader.ReadToEnd();
        if (result == null)
        {
            throw new ArgumentNullException("");
        }
    }
}
```

Extrait de code 23: Réponse à la requête PUT

Dans ce cas-là, une boîte de dialogue surgit. Elle est issue d'une *StartCoroutine* permettant d'attendre la réponse de l'utilisateur sans bloquer le processus principal.

```
catch (ArgumentNullException e)
{
    if (DialogIsOpen == false)
    {
        DialogResponse = DialogAnswer.None;
        DialogIsOpen = true;
        StartCoroutine(OkDialog("Piece selection error",
        "Unknown piece, please try again or contact the app configurator\nDetails: " + e.Message, PieceSelection));
    }
}
```

Extrait de code 24: Affichage d'un message en cas d'erreur

Le *OkDialog* est de type *IEnumerator*, car il contient un *yield return*. L'action attend maintenant sur *DialogResponse* qui sera de type *DialogAnswer.Ok* pour une boîte de dialogue Ok et *DialogAnswer.Yes/No* pour une boîte de dialogue *YesNoDialog*. Cette syntaxe est assez complexe et m'a donné du fil à retordre, mais elle permet une attente passive sans perturber le bon fonctionnement du programme.

```
IEnumerator OkDialog(string dialogTitle, string subtitle, GameObject CurrentGO)
{
    myDialog = Dialog.Open(MyNewDialog, DialogButtonType.OK, dialogTitle, subtitle, true);
    CurrentGO.SetActive(false);
    myDialog.OnClosed = OnClosedDialogEvent;
    yield return new WaitUntil(() => DialogResponse != DialogAnswer.None);
    CurrentGO.SetActive(true);
}
```

Extrait de code 25: OkDialog (aperçu du code)

4.6 Tracking

4.6.1 Déroulement de l'implémentation

Le *tracking* s'est résulté en échec de la pré-étude mais il aurait été dommage de laisser tomber cette fonctionnalité. La génération d'un modèle de *tracking* se réalise en plusieurs étapes. C'est le système de l'entreprise PTC (Vuforia) qui a été implémenté.

Il faut premièrement créer le modèle 3D de la pièce au format *fbx*. Il est ensuite possible de l'importer dans l'application Vuforia nommée "*Model Target Generator*".

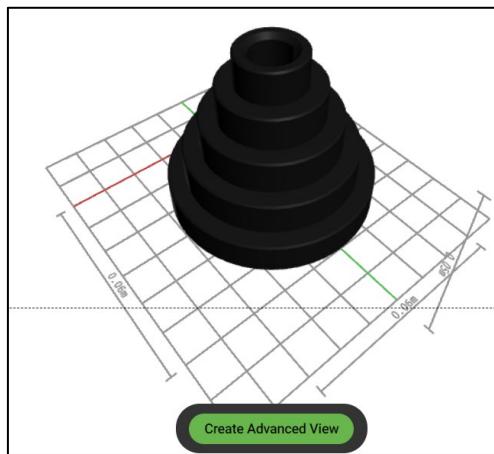


Figure 40: Model Target Generator (visuel)

La création d'un modèle se résume de la manière suivante :

- Importation du modèle
- Orientation du modèle dans le repère
- Dimensionnement de la pièce (choix des unités)
- Texture du modèle
- Analyse de complexité (plus la pièce est **complexe, meilleur** le *tracking* sera)
- Sélection du type d'objet (Objet de taille moyenne ou plus grand qu'une voiture)
- Détermination des mouvements de la pièce (Statique, peut bouger, en déplacement constant)
- Spécification des angles de vues de la pièce ainsi que son orientation initiale

Nous obtenons ensuite un *viewpoint* qui n'est autre que l'élément recherché par le processus de *tracking*.



Figure 41: Model Target Generator (viewpoint)

Il est maintenant possible d'exporter un *package Unity* puis de l'implémenter dans l'application. Le fonctionnement est acceptable, mais il nécessite de placer la pièce très précisément pour *matcher* avec l'hologramme. Le but était ici de placer la sphère rouge sur la pièce trackée. L'état du *tracking* (tracké ou non) n'étant pas indiqué, je fus contraint de modifier le code source des librairies Vuforia pour y ajouter des compléments.

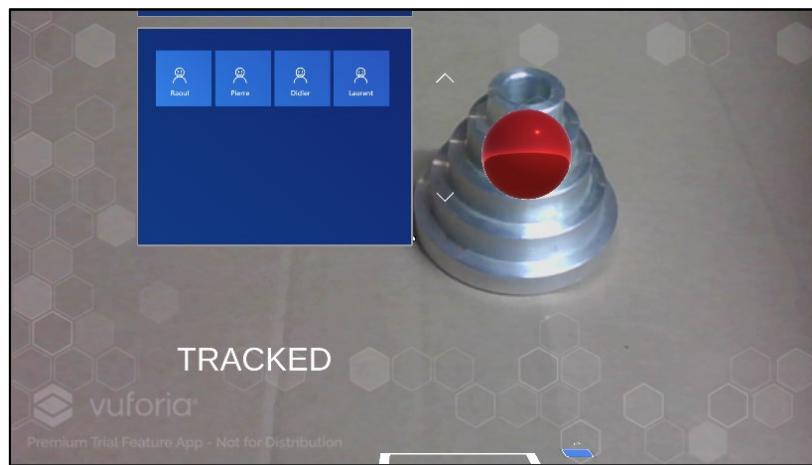


Figure 42: Tracking de la pièce

Le *tracking* fonctionne mieux avec la caméra de mon ordinateur plutôt qu'avec les multiples caméras présentes sur le HoloLens 2 pour une raison inconnue. Vuforia développe principalement ses solutions sur des appareils Android. Le public utilisant les HoloLens étant très restreint, ceci **pourrait** expliquer la raison d'un développement moins abouti.

4.6.2 Création du dataset par algorithme de Deep Learning

Il existe toutefois une méthode qui devrait améliorer significativement les performances. Vuforia inclus dans son logiciel un apprentissage du modèle avec des technologies de *Deep Learning*. Ces algorithmes sont malheureusement masqués, car c'est un service Cloud qui est proposé. Il faut exporter le modèle sur le serveur et le *dataset* est généré après neuf heures de temps (pour le porte-stylo).

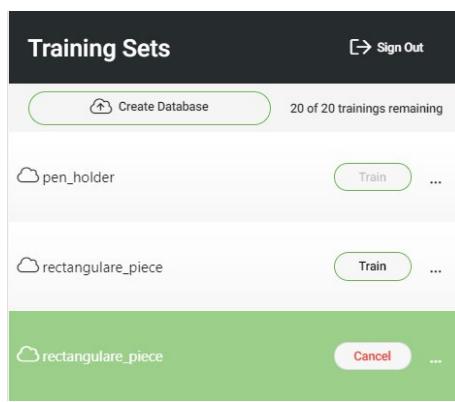


Figure 43: Crédit du dataset par DL en Cloud

Le résultat attendu est atteint, il est désormais possible de localiser la pièce dans un périmètre plus large et sans emplacement défini. Toutefois, les performances sont limitées, une rotation de la pièce ne place pas toujours la sphère rouge à l'origine du porte-stylo. Ceci est plutôt compromettant pour l'application choisie. Le même travail a été effectué pour la seconde pièce, désignée spécialement pour cette application.

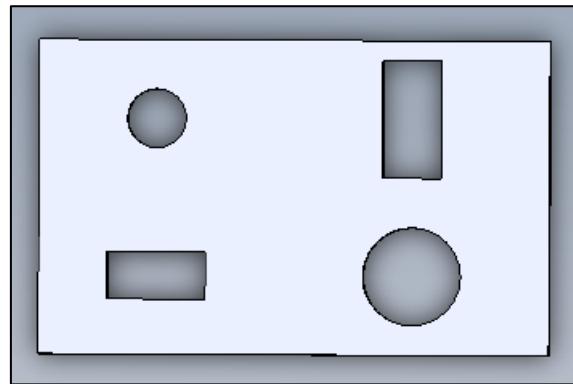


Figure 44: Pièce de test (modélisation)

Malgré un apprentissage basique puis à l'aide du *Deep Learning*, impossible de tracker la pièce que ce soit avec la caméra de l'ordinateur ou du casque. Ceci pourrait s'expliquer du fait de sa complexité qui est pour algorithme de reconnaissance, beaucoup trop basique. La pièce ayant des arêtes droites, elle peut se confondre très facilement avec le décor.

4.7 Application WPF

4.7.1 Diagramme UML

L'application WPF a légèrement évolué depuis la pré-étude, elle comporte les mêmes fonctionnalités mais sa robustesse a été travaillée.

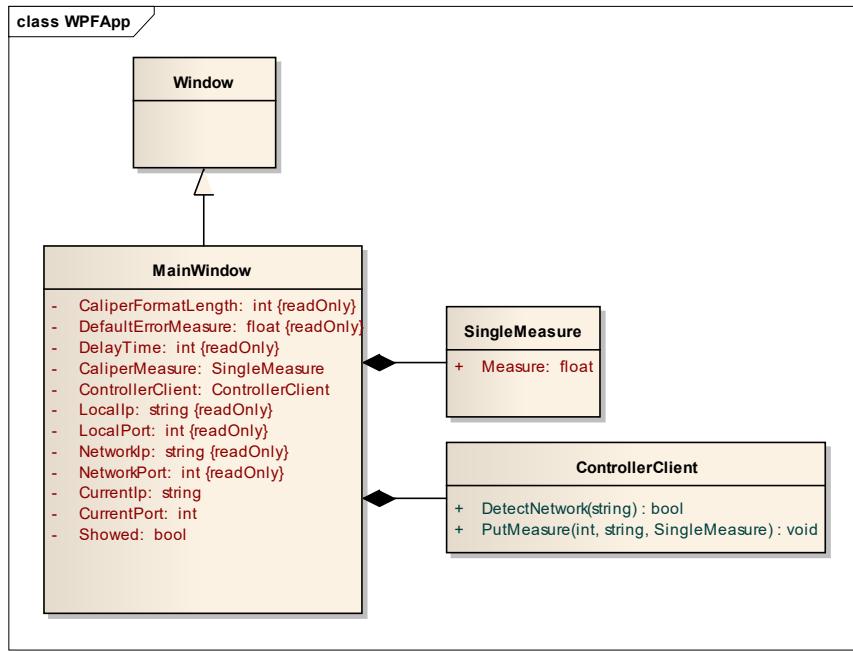


Diagramme UML 14: WPFApp

4.7.2 Description de l'application

Pour rappel : l'application est de type WPF, car le pied à coulisse est utilisé en tant que périphérique d'entrée de texte, la lecture immédiate via le protocole Bluetooth n'est donc pas, ou difficilement, réalisable. Il faudrait accéder au buffer d'entrée de l'ordinateur et distinguer les entrées provenant du pied à coulisse ou d'un clavier quelconque.

Il serait nécessaire d'intercepter le message entre la réception Bluetooth et l'écriture dans le buffer. Connecter le pied à coulisse directement au casque ne serait pas une solution. En effet, l'application Unity est indépendante et ne reçoit pas d'informations de périphériques externes.

4.7.3 Analyse du code

Le statut de connexion avec le serveur est désormais analysé cycliquement. A chaque *tick*, la méthode *CheckNetwork* est appelée.

```
DispatcherTimer timer = new DispatcherTimer
{
    Interval = TimeSpan.FromSeconds(TimeInterval)
};
timer.Tick += CheckNetwork;
timer.Start();
```

Extrait de code 26: Dispatchcher timer WPF

C'est une méthode issue de *MainWindow* qui actualise l'interface graphique selon l'état du réseau. Elle possède la capacité d'afficher un message d'erreur si besoin.

```
public void CheckNetwork(object sender, EventArgs e)
{
    if (ControllerClient.DetectNetwork(LocalIp))
    {
        BorderStatus.BorderBrush = Brushes.LawnGreen;
        TextBoxIp.Text = LocalIp + ":" + LocalPort;
        CurrentIp = LocalIp;
        CurrentPort = LocalPort;
        return;
    }
    else if (ControllerClient.DetectNetwork(NetworkIp))
    {
        BorderStatus.BorderBrush = Brushes.LawnGreen;
        TextBoxIp.Text = NetworkIp + ":" + NetworkPort;
        CurrentIp = NetworkIp;
        CurrentPort = NetworkPort;
        return;
    }
}
```

Extrait de code 27: Contrôle de la connexion au serveur

La méthode en question *DetectNetwork* est issue de la classe *ControllerClient*. Elle réalise un *ping* à l'adresse IP locale (pour la simulation) et celle du Hotspot (lors de l'utilisation du casque). La première approche était la création d'un client *tcp* ce qui permet de spécifier le port utilisé, mais cette action, bien qu'elle fût réalisée de manière asynchrone et dans une *Task* séparée influençait le programme principal. Cette fonction n'est pas sans intérêts pour autant, elle détermine fidèlement si le réseau du casque est disponible. La connexion au port est dans tous les cas testés lors de la requête PUT, un message d'erreur est affiché en cas d'exception.

```
public bool DetectNetwork(string hostUri)
{
    //doesn't specify the port but it takes way less time than a request
    Ping ping = new Ping();
    try
    {
        PingReply pingReply = ping.Send(hostUri);
        return pingReply.Status == IPStatus.Success;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Unable to ping the address\nDetails: "
            + ex.Message, "Ping error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    return false;
}
```

Extrait de code 28: Ping vers le serveur ASP

Contrôle du résultat de la requête

```
var Status = (HttpWebResponse)httpResponse;
if (Status.StatusCode != HttpStatusCode.OK)
{
    throw new WebException();
```

Extrait de code 29: Récupération du statut de la requête

4.7.4 Aperçus visuels

Aperçus visuels de l'application (Initialisation, en attente, en fonctionnement).

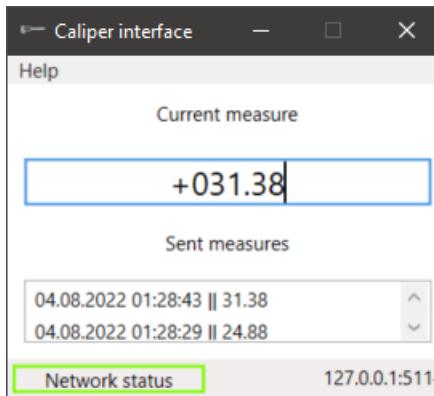


Figure 45: WPFAApp (réception d'une mesure)

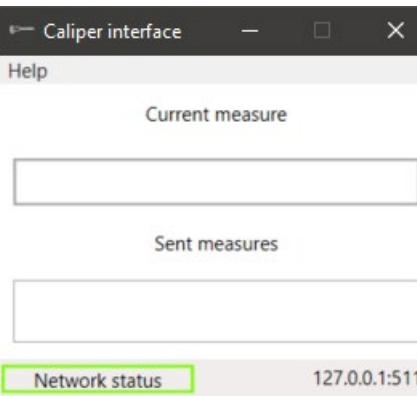


Figure 46: WPFAApp (en attente)

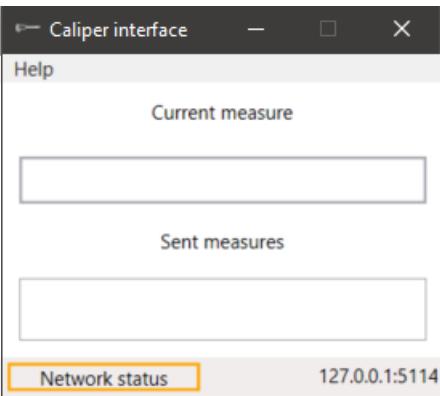


Figure 47: WPFAApp (initialisation)

Une aide est également accessible. La documentation est le fichier de configuration du pied à coulisse présent en annexe.

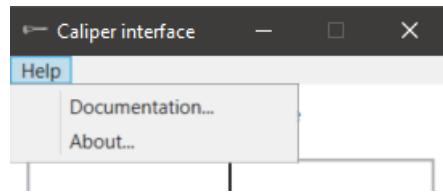


Figure 48: WPFAApp (aide utilisateur)

Rappel : il est nécessaire d'avoir le focus sur le champ de texte avant de transmettre la mesure **dépuis** le pied à coulisse (bouton jaune). Dans le cas contraire, la mesure sera écrite sur l'application de l'ordinateur ayant le focus de la souris.

Note : le statut de connexion du pied à coulisse a été supprimé car l'outil indique en permanence son état. Afficher deux fois la même information n'apporte rien à l'utilisateur et cela ajoute du contenu superflu.

4.8 Holo2factory configurator

4.8.1 Diagramme UML

Cette solution WPF a comme objectif de réaliser des recettes de production. Le configurateur peut générer une liste de cotations pour l'opérateur. Elle est composée d'une fenêtre principale (*MainWindow*) ainsi qu'une fenêtre secondaire (*ProductionWindow*).

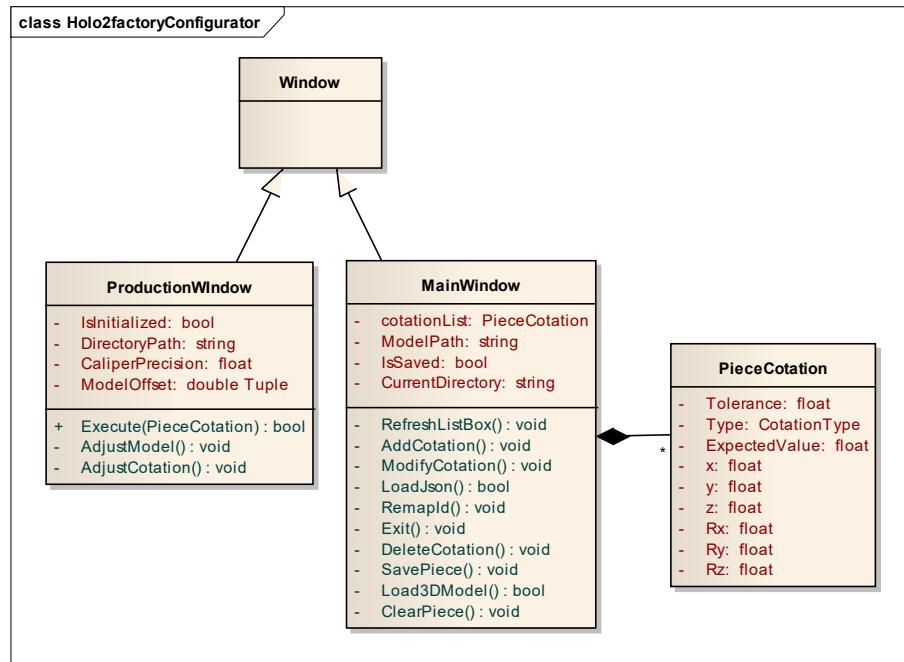


Diagramme UML 15: Holo2factoryConfigurator

4.8.2 Analyse du code

Le chargement d'un fichier quelconque s'effectue de cette manière. Ceci ouvre un explorateur de fichier standard Windows et propose uniquement les types de fichiers mentionnés. C'est-à-dire le format JSON dans ce cas.

```

private bool LoadJson()
{
    Microsoft.Win32.OpenFileDialog dlg = new Microsoft.Win32.OpenFileDialog
    {
        FileName = "Load json piece file",
        DefaultExt = ".json",
        Filter = "Text documents (.json)|*.json"
    };
  
```

Une fois la sélection réalisée, il est possible de désérialiser la liste de cotations. En cas d'erreur, une *MessageBox* est affichée avec un descriptif contenant le code d'erreur respectif.

```

try
{
    string jsonString = File.ReadAllText(dlg.FileName);
    CotationList = JsonConvert.DeserializeObject<List<PieceCotation>>(jsonString);
}
catch (Exception e)
{
    _ = MessageBox.Show("Unable to load Json file\nDetails: " + e.Message,
        "Load Json", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return false;
}

```

La syntaxe *Helix* est assez complexe et se décompose de la manière suivante :

1. Création d'un *ObjReader*
2. Lecture du fichier obj
3. Création d'un groupe de transformations
4. Instanciation des transformations de type rotations avec vecteurs unitaires du référentiel
5. Ajout des rotations au groupe de transformation
6. Création d'une translation avec les paramètres reçus
7. Ajout de la translation au groupe de transformation
8. Application de la transformation
9. Affectation de l'objet (modèle 3D) à l'élément graphique

```

ObjReader CurrentHelixObjReader = new ObjReader();
try
{
    //load the 3D Model with offset if set
    Model3DGroup Model3DGroup = CurrentHelixObjReader.Read(ModelPath);
    Transform3DGroup transform3DGroup = new Transform3DGroup();
    RotateTransform3D rotateTransform3DX =
        new RotateTransform3D(new AxisAngleRotation3D(new Vector3D(1, 0, 0), ModelOffset.Item4));
    RotateTransform3D rotateTransform3DY =
        new RotateTransform3D(new AxisAngleRotation3D(new Vector3D(0, 1, 0), ModelOffset.Item5));
    RotateTransform3D rotateTransform3DZ =
        new RotateTransform3D(new AxisAngleRotation3D(new Vector3D(0, 0, 1), ModelOffset.Item6));
    transform3DGroup.Children.Add(rotateTransform3DX);
    transform3DGroup.Children.Add(rotateTransform3DY);
    transform3DGroup.Children.Add(rotateTransform3DZ);
    transform3DGroup.Children.Add(new TranslateTransform3D(new Vector3D(
        ModelOffset.Item1, ModelOffset.Item2, ModelOffset.Item3)));
    myModel.Transform = transform3DGroup;
    myModel.Content = Model3DGroup;
}

```

Note : *ModelOffset* est un *tuple* contenant x, y, z, Rx, Ry, Rz.

4.8.3 MainWindow

Petit aperçu de la fenêtre principal :

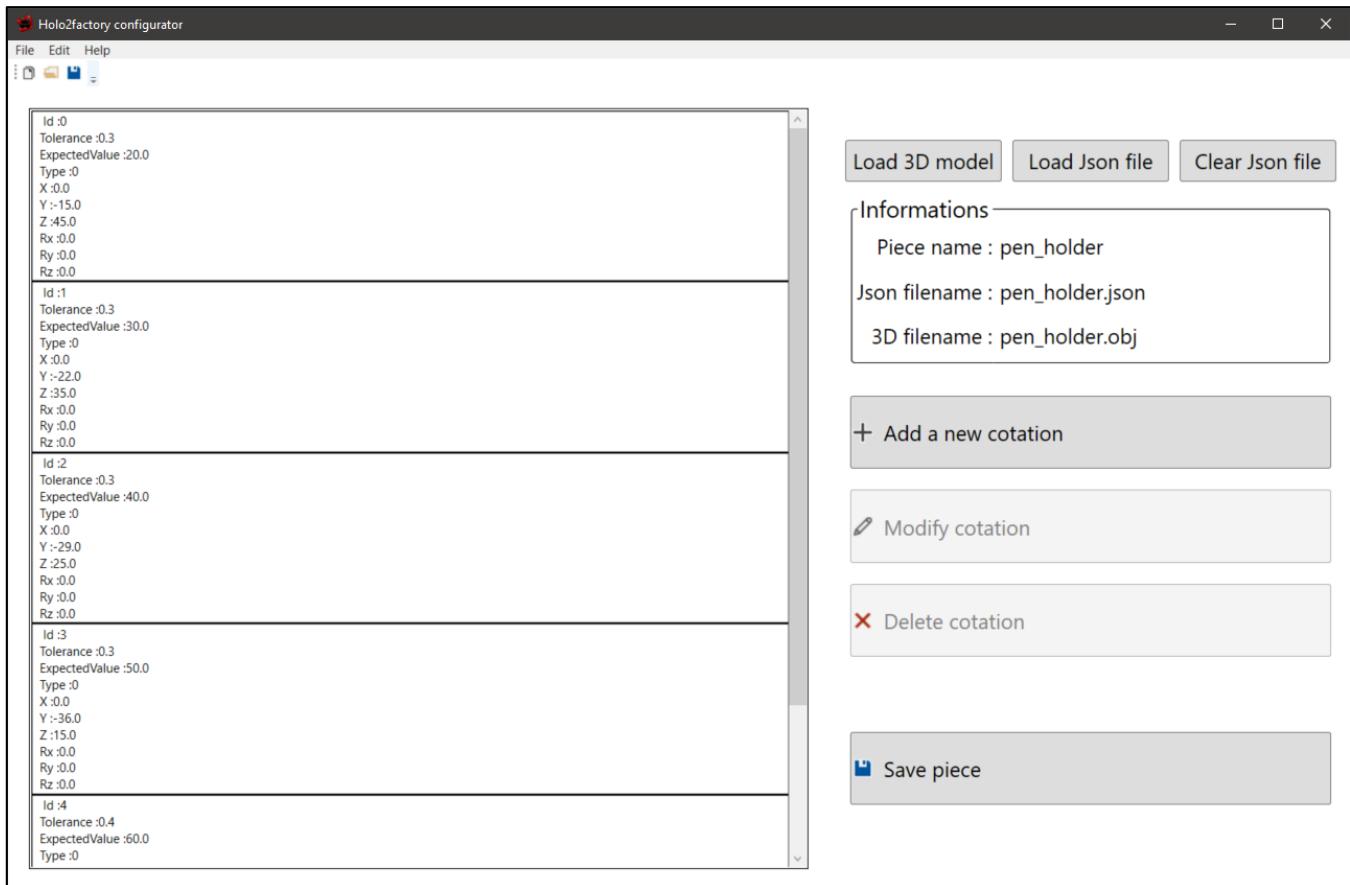


Figure 49: Holo2factoryConfigurator (MainWindow)

4.8.4 Détails des fonctionnalités MainWindow

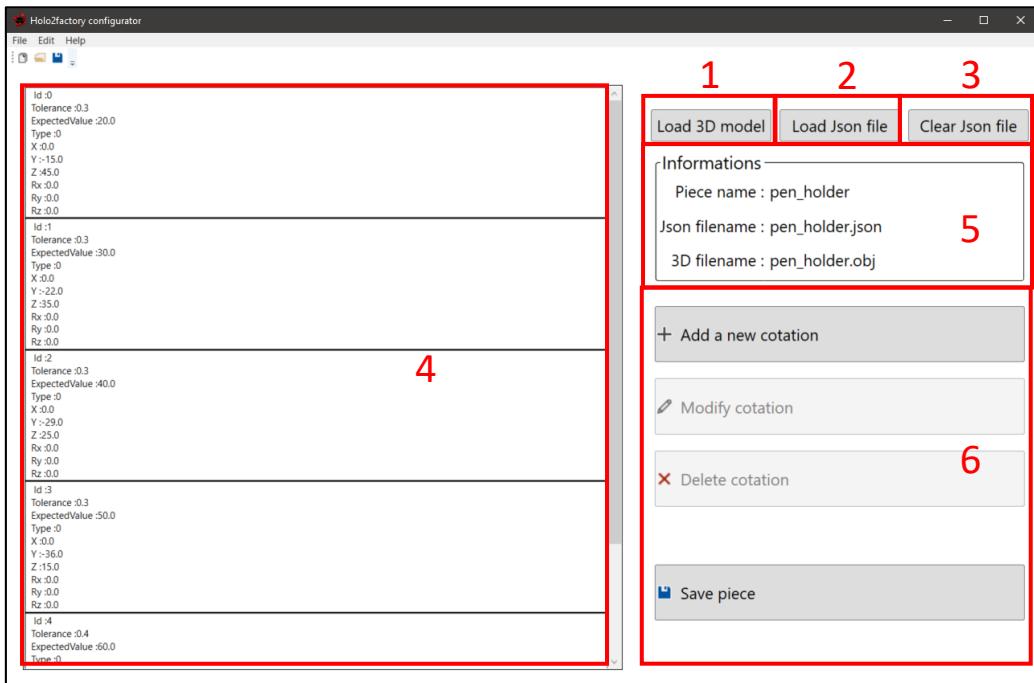


Figure 50: Holo2FactoryConfigurator (détails MainWindow)

1. Importe un modèle 3D au format obj
2. Charge une liste de cotations
3. Supprime la recette courante, n'impacte pas le fichier enregistré
4. Liste des cotations
5. Informations sur les fichiers chargés
6. Actions disponibles pour l'utilisateur

4.8.5 ProductionWindow

Il est difficile de placer spatialement des cotations sans avoir un aperçu visuel. C'est la raison pour laquelle la librairie *Helix* a été étudiée lors de la pré-étude. *Helix* est un *Toolkit* qui complète le *namespace* de *Microsoft Media3D* en y ajoutant des fonctionnalités. C'est une librairie communautaire qui est malheureusement mise à jour sporadiquement. Les versions proposées sont testées sur des solutions de Visual Studio 2019 et les exemples disponibles proviennent d'une édition de Visual Studio 2013. Travaillant sur Visual Studio 2022, il a été nécessaire de faire des ajustements et de générer un *upgrade* des outils. Il permet de générer un visualisateur 3D. Lors de l'appui sur le bouton "Add a new cotation" ou "Modify cotation", la fenêtre "New Cotation" s'ouvre.

Dans l'exemple présenté ci-dessous, le porte-stylo est visible. Une cotation de diamètre de 30mm a été placée sur la pièce par le configurateur. Le *spatial viewer* possède les commandes standards des 3D viewer, rotation de la pièce, zoom, positionnement immédiat à l'aide du cube dans le coin inférieur droit etc.

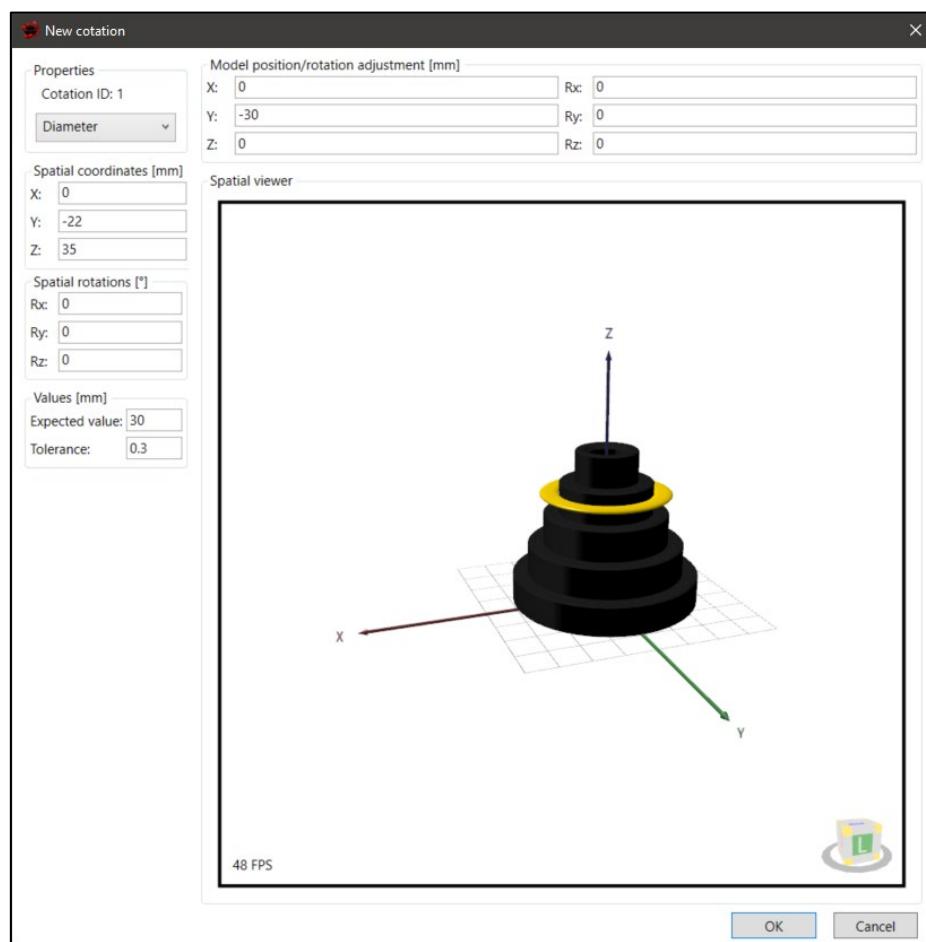


Figure 51: Holo2FactoryConfigurator (ProductionWindow)

4.8.6 Détails des fonctionnalités ProductionWindow

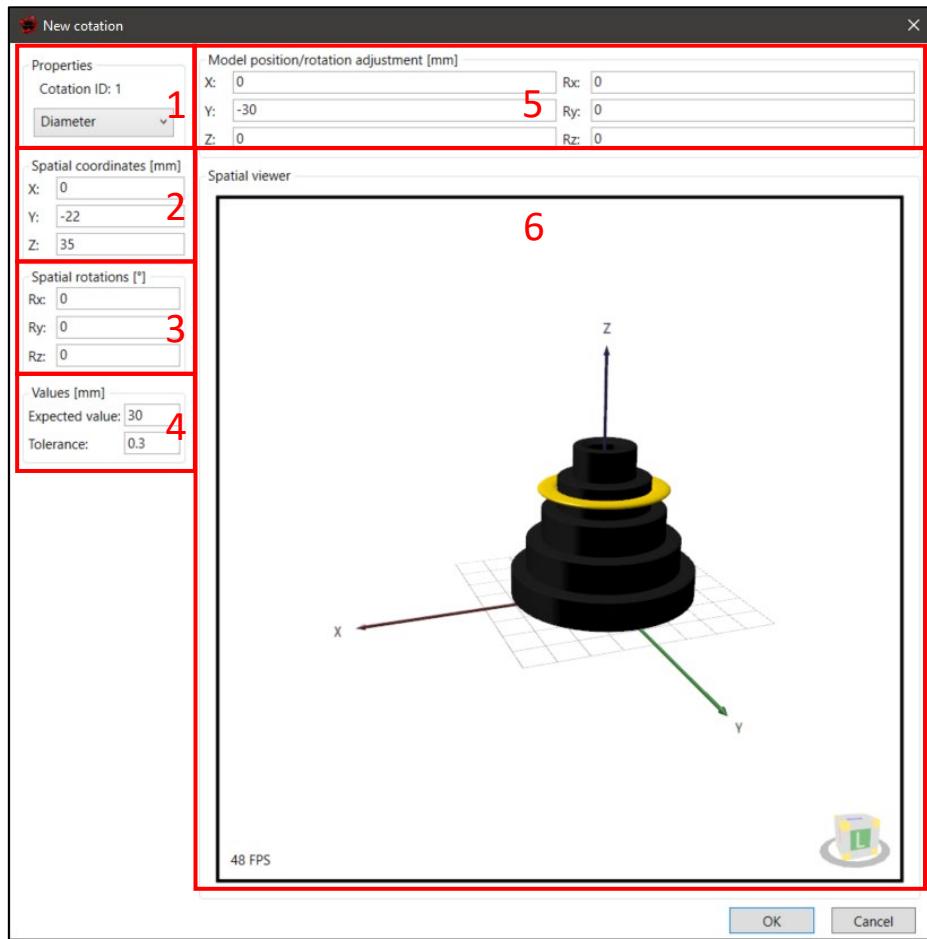


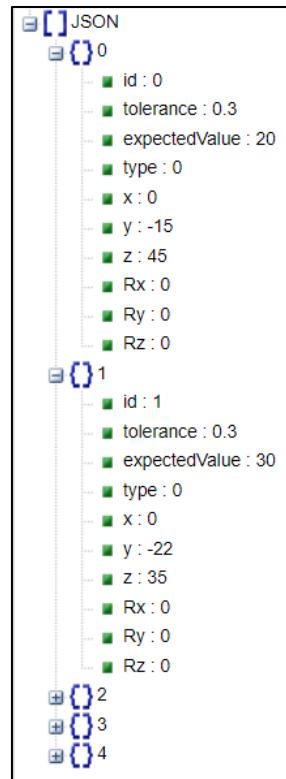
Figure 52: Holo2FactoryConfigurator (détails ProductionWindow)

1. Choix du type de cotation (Diamètre, rectiligne, trou)
2. Insertion des coordonnées spatiales
3. Rotation de la cotation
4. Valeur attendue de mesure et tolérance de mesure
5. Ajustement de la position et de l'orientation du modèle
6. Visualisateur 3D

Les hologrammes présents dans Unity sont de type fbx. Helix ne prenant pas en compte ce format, il est nécessaire de convertir le modèle. Cette conversion influence parfois l'origine de la pièce. C'est la raison pour laquelle il est possible d'ajuster la position et l'orientation du modèle.

4.8.7 Résultat au format JSON

Lors de la sauvegarde de la liste des cotations, un fichier est généré au format JSON. Il est possible de visualiser le résultat exporté à l'aide d'un JSON viewer :



```
[{"id": 0, "tolerance": 0.3, "expectedValue": 20, "type": 0, "x": 0, "y": -15, "z": 45, "Rx": 0, "Ry": 0, "Rz": 0}, {"id": 1, "tolerance": 0.3, "expectedValue": 30, "type": 0, "x": 0, "y": -22, "z": 35, "Rx": 0, "Ry": 0, "Rz": 0}, {"id": 2, "tolerance": null, "expectedValue": null, "type": null, "x": null, "y": null, "z": null, "Rx": null, "Ry": null, "Rz": null}, {"id": 3, "tolerance": null, "expectedValue": null, "type": null, "x": null, "y": null, "z": null, "Rx": null, "Ry": null, "Rz": null}, {"id": 4, "tolerance": null, "expectedValue": null, "type": null, "x": null, "y": null, "z": null, "Rx": null, "Ry": null, "Rz": null}]
```

Figure 53: Résultat de la liste de cotations exportée au format JSON

4.9 Convention de codage

4.9.1 Analyse du code

Afin de vérifier les usages et les bons principes du langage de programmation C#, il m'a été conseillé d'utiliser le "l'exécution d'analyse du code" de Visual Studio. Cet outil analyse le programme et signale les violations des règles de nommage, des syntaxes obsolètes ou des variables non utilisées.

Selon les [conventions de codage](#) Microsoft, un membre privé devrait commencer par une lettre minuscule préfixée d'un *underscore*. Lors de l'exécution de l'analyse du code, l'accesseur/setter rend cette convention inopérante. Dans mes codes sources, les variables privées possédant un getter/setter commenceront par une majuscule et ne posséderont pas *d'underscore*.

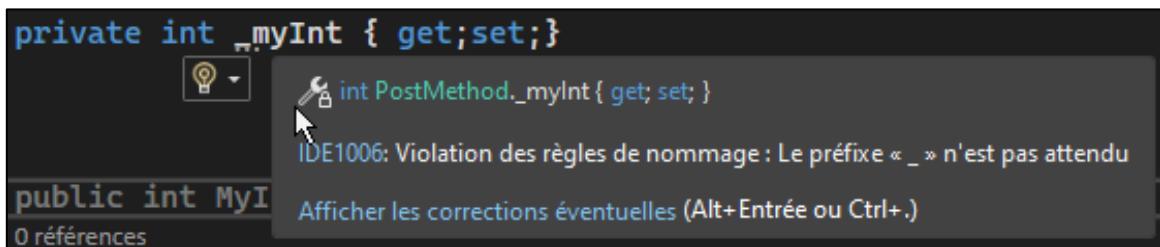


Figure 54: Convention de codage

Les scripts de développement Unity utilisent le langage C#, mais de nombreux standards ne sont pas applicables. Les versions .NET étant différentes entre mes applications, certaines syntaxes diffèrent. Linstanciation d'un objet se réalise conventionnellement de cette manière.

```
MyObject Obj = new MyObject();
```

Extrait de code 30: Syntaxe C# new obsolète

Cependant, il est possible dès la version de langage C# 9.0 de simplifier cette syntaxe.

```
MyObject Obj = new();
```

Extrait de code 31: Syntaxe C# new dotnet 9.0

4.9.2 Incohérences de compatibilité Unity

L'analyseur de code se base sur la version du langage pour déterminer si la syntaxe est optimale ou non. Malheureusement, Unity n'accepte certains de ces changements. Cette découverte a été réalisée lors de l'exécution du programme, un temps précieux a donc été utilisé pour remettre en état le code. Unity utilise également des fonctions catégorisées de "Message", c'est le cas des fonctions *Start()* et *Update()*. Des *warnings* indiquent que ces méthodes ne sont pas utilisées, pourtant, ces fonctions sont bien appelées.

Voici quelques exemples de syntaxes inconnues malgré une version *dotnet* permissive :

```
//remove using warning
#pragma warning disable IDE0063
//remove new warning-> SYNTAX AINT'T WORKING IN UNITY
#pragma warning disable IDE0090
//remove new warning-> SYNTAX AINT'T WORKING IN UNITY
#pragma warning disable IDE0017
//remove "is not used" because CodeAnalysis can't see the call
#pragma warning disable IDE0051
```

Extrait de code 32: Désactivation des warnings

4.10 Tests unitaires/protocole de tests

4.10.1 Tests unitaires dans l'application WPF

La vérification du bon fonctionnement de code peut se réaliser à l'aide de tests unitaires. Dans Visual Studio, il est possible de créer un second projet de type *UnitTest* dans la solution à contrôler. Il est dès lors possible de créer des tests et d'analyser les résultats. En voici un exemple pour la méthode *DetectNetwork* de la classe *ControllerClient*.

```
public void TestDetectNetwork()
{
    try
    {
        ControllerClient.DetectNetwork("127.0.0.1");
        Assert.IsTrue(true);
    }
    catch
    {
        Assert.IsFalse(false);
    }
    try
    {
        ControllerClient.DetectNetwork("Misoune");
        Assert.IsFalse(false);
    }
    catch
    {
        Assert.IsTrue(true);
    }
}
```

Récapitulatif des détails du test

- ✓ TestDetectNetwork
- Source: [UnitTest1.cs](#) ligne 12
- Durée: 60 ms

Extrait de code 33: Tests unitaires

Figure 55: Test unitaire du code

Les tests de l'application Holo2factoryConfigurator se font à l'aide d'un protocole de test de fonctionnalité disponible en annexe.

4.10.2 Couverture du code

Des tests unitaires avec un taux de couverture de 100% signifie qu'ils couvrent la complexité cyclomatique d'un code. Ceci dénote qu'il faudrait vérifier l'ensemble des combinaisons possibles atteignable avec le programme. Ce processus est particulièrement chronophage et les applications développées dans ce TB ne garantissent pas ce niveau de couverture. Certains cas ne sont pas atteignables de par l'abstraction logicielle.

Il n'y a pas de règles précises concernant le pourcentage de couverture de code à atteindre. Couvrir 100% n'est pas une bonne approche car elle requiert excessivement de temps et il devient nécessaire d'introduire des niveaux d'abstraction supplémentaires. Le pourcentage idéal varie selon l'application mais il est toutefois conseillé d'atteindre un minimum de 60%.⁸

Il est possible de calculer le taux de couverture depuis Visual Studio. Dans le cas de l'application WPF, la classe *ControllerClient* est couverte à 74.36%, ce qui est acceptable.

Hiérarchie	Non couverts (blocs)	Non couverts (% blocs)	Couverts (blocs)	Couverts (% blocs)
unitests.dll	10	25,64%	29	74,36%
UnitTests	10	25,64%	29	74,36%
UnitTest1	10	25,64%	29	74,36%
TestDetectNetwork()	4	28,57%	10	71,43%
TestPutRequest()	6	28,57%	15	71,43%
UnitTest1()	0	0,00%	3	100,00%
get_ControllerClient()	0	0,00%	1	100,00%

Figure 56: Couverture du code

Les tests de fonctionnement des contrôles WPF ont été réalisés à l'aide d'un protocole de tests disponible en annexe.

4.11 Performance du repérage spatial

Différentes approches de repérage spatial ont été étudiées. La première était un posage fixe qui consistait à placer la pièce dans un emplacement prédéfini. L'utilisateur devait ensuite communiquer à l'application où se trouvait le posage en y plaçant un hologramme à l'aide de ses mains à l'origine du plateau. La seconde approche implémente le *tracking* dynamique de la pièce grâce au SDK de Vuforia.

⁸ Carlos Arguelles, Marko Ivanković, and Adam Bender (2020)

4.11.1 Posage fixe

Ce système comprend une contrainte prééminente : il est nécessaire de placer le repère manuellement à chaque lancement de l'application. Cette opération est laborieuse et la précision n'est pas optimum. Le plateau est conçu pour deux pièces prédéfinies, il ne permet donc pas d'accueillir de nouvelles pièces. Il faudrait réaliser un plateau ajustable dans lequel la pièce sera placé dans un coin. Un autre inconvénient concerne l'orientation de la pièce : si l'objet à mesurer possède un perçage sur la face cachée, l'application devra indiquer la rotation à effectuer à l'opérateur.

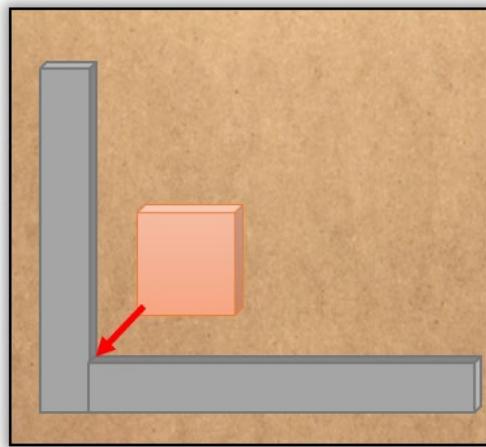


Figure 57: Prototype visuel posage adaptatif

Ses avantages essentiels sont la précision et la stabilité. Les composants étant statiques, le positionnement spatial des hologrammes est très fiable.

4.11.1.1 Code QR

Une alternative au placement fixe était le repérage à l'aide de code QR. Une phase de recherche conséquente a été réalisée mais l'implémentation s'est résulté en échec. Les versions proposées par Microsoft sont *outdated* et ne fonctionnent pas dans Unity. Une seconde librairie a été testée, c'est le contenu communautaire [StereoKit](#). Ce dernier s'est correctement téléchargé dans le casque mais aucune détection n'a été remarquée. D'autres tentatives de librairies, modifications de codes n'ont pas abouti.

4.11.1.2 Microsoft Azure Anchor

Microsoft propose une fonction multiplateforme appelée communément Anchor. Le nom Azure fait référence aux éléments Cloud de Microsoft. Ceci signifie qu'une Anchor (hologramme placé dans l'espace à une position fixe) peut être utilisée par différents appareils en même temps. Ceci peut être utile pour collaborer entre plusieurs appareils.

Ces *anchors* pourraient être utilisées comme référence spatiale mais le positionnement n'est pas très précis. Dans le cas d'étude, le niveau de précision est au millimètre.

4.11.2 Tracking Vuforia

L'opération d'apprentissage est fastidieuse, elle requiert la création du modèle 3D, la configuration du modèle dans l'application Vuforia, l'apprentissage en DL de la pièce sur le service *Cloud* puis l'implémentation dans l'application Unity. Ce service requiert également une licence qui coûterait aux alentours de 500 CHF par année. C'est une solution intéressante si le nombre de nouvelle pièce est faible. Dans le cas contraire, la réalisation de ce processus ne serait pas rentable. L'avantage du suivi dynamique est la possibilité à l'opérateur de saisir la pièce dans sa main et de la manipuler librement. Les performances seraient toutefois supérieures si les modèles avaient été plus complexes. Les résultats ci-dessous démontrent qu'une simple rotation de la pièce influence la position de la cotation malgré un ajustement logiciel.

4.11.3 Résultats

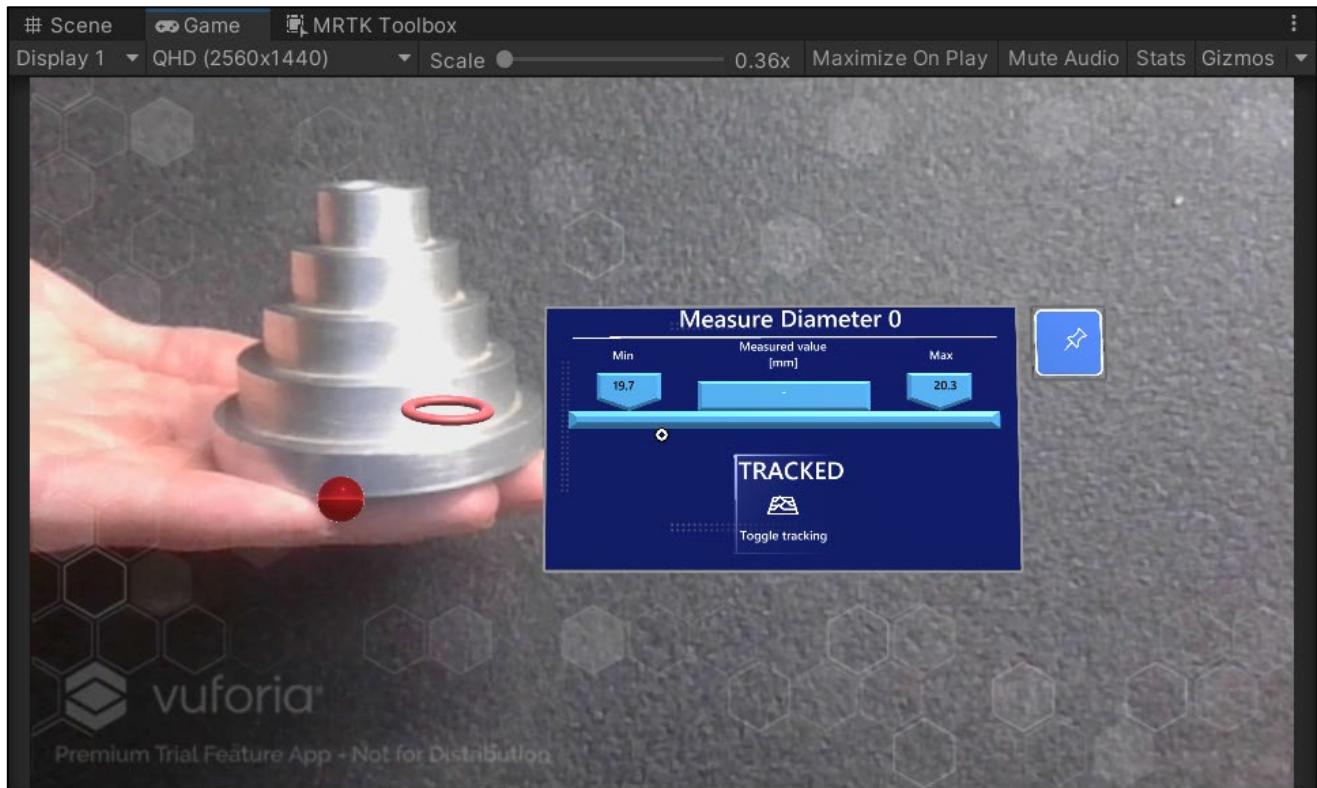


Figure 58: Placement de la cotation en fonction du tracking

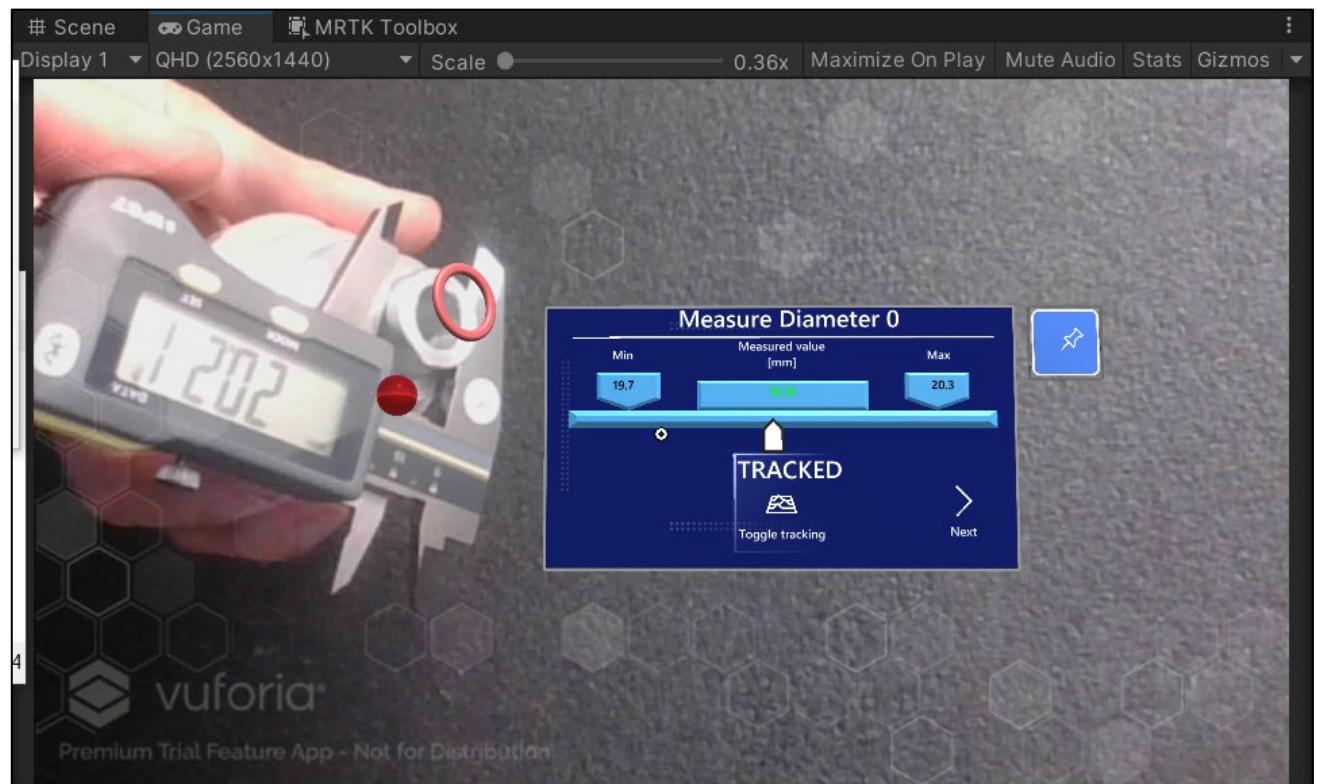


Figure 59: Placement de la cotation en fonction du tracking 2

4.12 Alternative intéressante (Dynamics 365 Guides)

4.12.1 Connexion à Guides

Microsoft propose une application nommée "*Dynamics 365 Guides*", elle a pour but de réaliser des séquences de guidage personnalisables pour un opérateur.

J'aurais souhaité réaliser une petite séquence de test avec la licence d'essai de *Microsoft Dynamics*. Il est nécessaire de posséder un compte Microsoft professionnel, le compte de la HES-SO remplit cette condition. Toutefois, il n'est pas possible de se connecter à l'application dans le casque avec ce compte. Un patch de correction résoudra probablement ce problème dans les mois à venir. Lors de la connexion, l'écran repasse au menu démarrage :

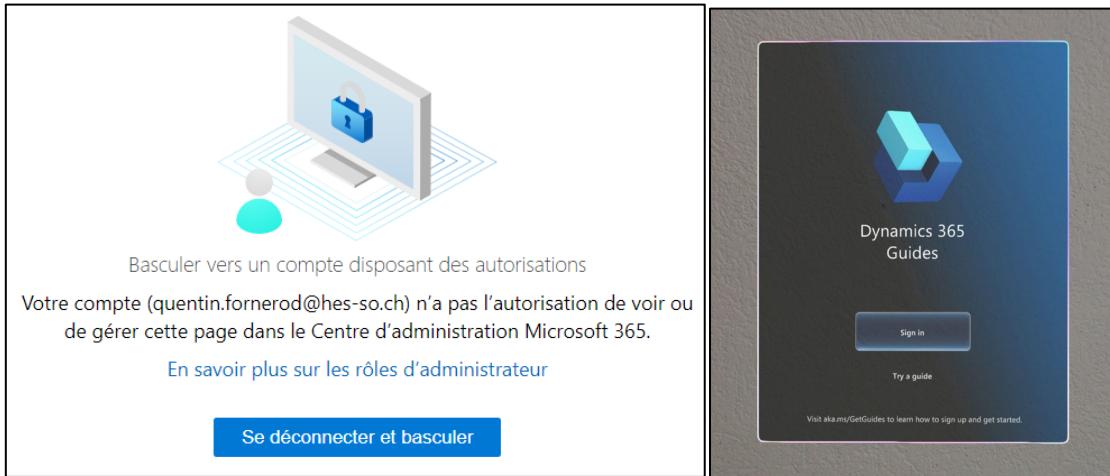


Figure 60: Connexion au compte professionnel

Figure 61: Connexion Dynamics Guides sur HoloLens 2

4.12.2 Démonstration des fonctionnalités

J'ai malgré cela suivi la démonstration mise à disposition. Voici un plan de travail (ici virtuel, c'est de la simulation) sur lequel repose un compresseur à réviser. Le concept de cette séquence est de guider l'opérateur en empruntant les bons outils et en effectuant les opérations effectives pour la réparation de la machine.



Figure 62: Aperçu visuel Dynamics Guides (instruction)

Un suivi statistique des opérateurs et des tâches est directement implémenté dans *Guide*



Figure 63: Aperçu Dynamics Guides (statistiques)

C'est un paradigme de guidage différent du cas d'étude développé, car ici tout est réalisé depuis le software de Microsoft. Ceci aura pour contrainte de restreindre les libertés logicielles, mais cela offrira davantage de stabilité et réduira considérablement le temps de développement. Le projet est uniquement une configuration d'outils, de messages etc., plus un développement logiciel.

Le repérage spatial se réalise à l'aide de code QR placé sur l'espace de travail.

4.12.2.1 Témoignage d'une entreprise

L'entreprise canadienne [Kruger](#) produisant des produits essentiels à partir de ressources renouvelables utilise à présent des HoloLens et le concept de *Dynamic Guides* dans le cadre de la production. Le développement du système a été réalisé par l'entreprise québécoise [AGC](#). J'ai retracé un extrait de témoignage⁹ en français.

Kruger, le spécialiste nord-américain des produits cartonnés et de l'énergie existe depuis environ 120 ans. Malgré cela, l'entreprise a adopté une approche technologique moderne, donnant à ses employés des outils innovants pour les rendre plus confiants et efficaces dans leur travail. Avec Microsoft Dynamic Guides, Kruger amène un guide de poche industriel avec des outils sophistiqués tel que des triggers spatiaux et des indications pour guider l'employé à l'emplacement de la tâche à effectuer. Désormais les informations dont auraient besoin les employés se trouvent au bout de leurs doigts. Cet accès rapide d'information booste la productivité et dirige les employés vers de nouveaux défis. Les travailleurs les plus expérimentés de chez Kruger peuvent facilement documenter leur expérience personnelle dans le fonctionnement et le dépannage d'une installation dans un document de réalité mixte. Ces solutions transforment la manière dont les opérateurs se forment et la manière dont ils exécutent leurs tâches tout en restant dans le flux du travail grâce à la capacité des HoloLens d'afficher de l'information holographique

Témoignage 1: Entreprise Kruger

⁹ Microsoft Community Dynamics (2021)

5 Synthèse

5.1 Problèmes rencontrés

J'ai rencontré abondamment d'anicroches lors de ce TB et certaines d'entre elles méritent d'être détaillées. Une partie de ces dernières ont déjà été explicitées dans leur chapitre respectif.

5.1.1 *Mise à jour Windows*

Microsoft a déployé mi-juin 2022 un *patch* ayant pour but de corriger des problèmes. Malheureusement, ce dernier a rendu le partage de Wi-Fi (*hotspot*) invalide.

Ceci pourrait paraître bénin mais le chargement du *build* dans le casque devient impossible. Aucune indication d'erreur m'a orienté vers la cause qui était cette mise à jour. J'ai remis en question mon code, ma configuration de projet, le casque HoloLens et d'autres composants logiciels. J'ai tenté de recharger des backups, recréer des solutions et ceci m'a considérablement péjoré dans l'avancement de mon projet.

Un patch fin juin a corrigé ce problème.¹⁰

5.1.2 *Version expérimentale de MRTK*

Bien que MRTK 2 existe depuis 2019, le développement paraît délaissé. Du contenu graphique est toujours au stade expérimental et aucun exemple existe que ce soit en français, en anglais ou en chinois.¹¹

Ceci m'a contraint à me documenter en relisant les lignes développées par la communauté et les employés de Microsoft. Malgré cela, Microsoft a annoncé la sortie de MRTK 3 en juillet 2022.

5.1.3 *Problèmes quelconques*

- Version .net WPF
- *Helix*
- Configuration Unity
- Version C# Unity différente
- Implémentation Vuforia
- Lecture des mesures du pied à coulisse
- Lecture protocole Bluetooth en C#
- Expert dans le domaine pour m'aider
- Srialisation
- Dans le cas des HoloLens, beaucoup de contenu provient de particuliers
- Compatibilité entre .net6 etc
- Remoting
- Unity ne tolère pas les bibliothèques importées en tant que lien

¹⁰ Lien du patch : <https://docs.microsoft.com/en-us/windows/release-health/resolved-issues-windows-10-21h2#2845msgdesc>

¹¹ Stade expérimental des *Dialogs* :
https://hololenscdev.github.io/MRTKDoc/Assets/MRTK/SDK/Experimental/Dialog/README_Dialog.html

5.2 Planification

Ce travail de Bachelor comporte de nombreuses étapes et il est nécessaire d'établir un planning dès la pré-étude. Cette planification se base sur le *Cycle en V*, c'est-à-dire une partie analyse, spécification et conception. Le processus est ensuite inversé, des tests unitaires vérifient la conception, la vérification contrôle les spécifications et la validation détermine si les besoins analysés sont complétés.

La pré-étude a duré environ 8 semaines. Ceci correspond à 90 heures (~11.25h par semaine). Ceci représente environ un quart du projet. L'effort estimé était de 91 heures et le temps utilisé est également de 91 heures. En réalité, le déroulement n'était pas parfait, certaines tâches ont demandé davantage de temps et d'autres, un peu moins. Cette *erreur* de planification nommée *delta* dans l'annexe est de 14 heures.

N°	Description	Détail	Effort estimé
T10	Pré étude		91
T10.10	Prise en main de la technologie		
T10.10.10	Installation Unity+ MRTK et Affichage d'un hologramme dans l'espace		10
T10.10.20	Communication pied à coulisse/PC (Bluetooth)		15
T10.10.30	Transfert de données entre PC/Hololens (Woopsa/ASP)		10
T10.10.40	Implémentation des interactions possibles avec l'utilisateur (Affichage, boutons)		10
T10.20	Tracking d'une pièce basique avec Visionlib/Vuforia SDK		8
T10.30	"Hello World" avec Blazor		4
T10.40	Définition de l'application		4
T10.20	Analyse de la fonctionnalité		
T10.20.10	Diagrammes d'activités		4
T10.20.20	Diagrammes de cas d'utilisation		4
T10.20.30	Modèle de domaine (diag. De classes)		10
T10.30	Rédaction des spécifications		4
T10.30.10	Conception de l'architecture logicielle		4
T10.30.20	Planification détaillée de la réalisation		4

Figure 64: Planning pré-étude avec heures réalisées

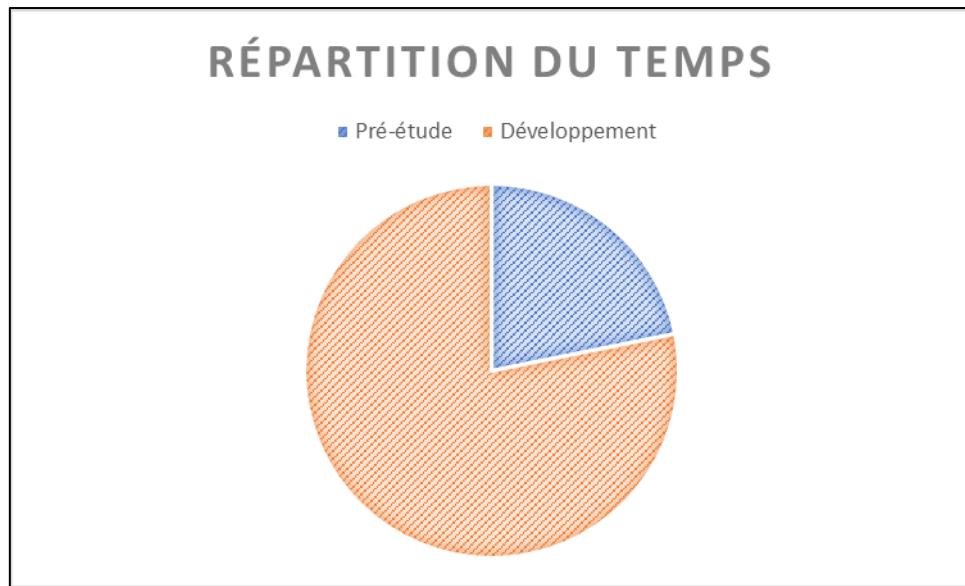
Malgré une structure séparant le développement du projet et la pré-étude, passablement de recherches ont été effectuées durant création des applications. Mants imprévus et aléas ont impliqué de la documentation supplémentaire.

Les noms des éléments dans le planning peuvent différer par rapport aux noms dans les codes sources car ce planning a été réalisé avant le développement et des modifications majeures ont eu lieu.

En résumé, 417 heures ont été consommées durant ce TB. L'outil de planification indique un *delta* de 56 heures, ceci correspond à l'*erreur* par rapport au temps travail estimé et réalisé par tâche. On peut remarquer dans l'annexe que le l'implémentation d'*Helix* et le *tracking* ont requis d'avantage d'attention que prévu. Les charges de l'application WPF ont été sous-estimées.

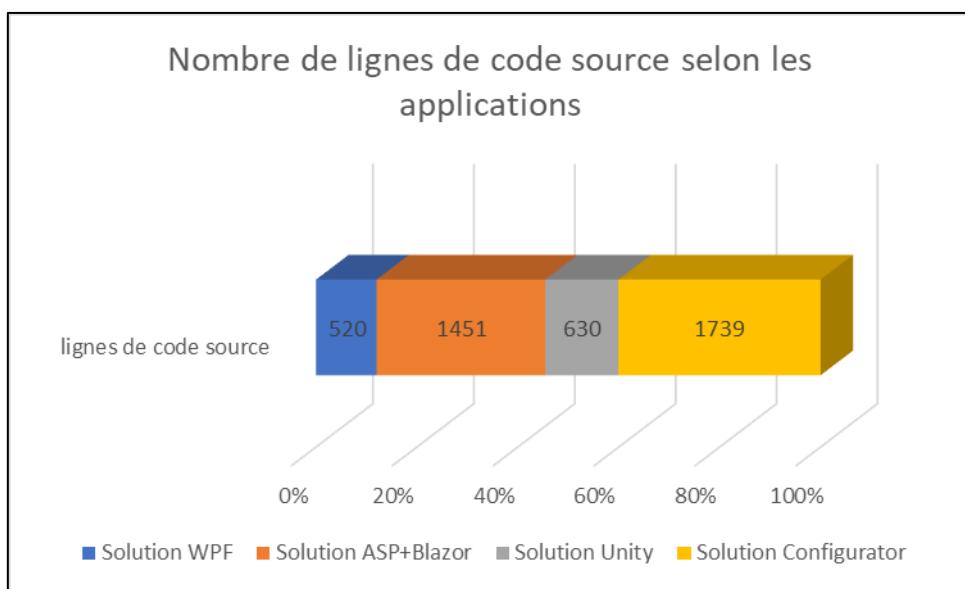
5.3 Statistiques du projet

Ce TB s'est décomposé en deux phases distinctes : la pré-étude ainsi que le développement des applications. En réalité, beaucoup de recherches ont été effectuées lors du développement, un prolongement de la pré-étude aurait été judicieux.



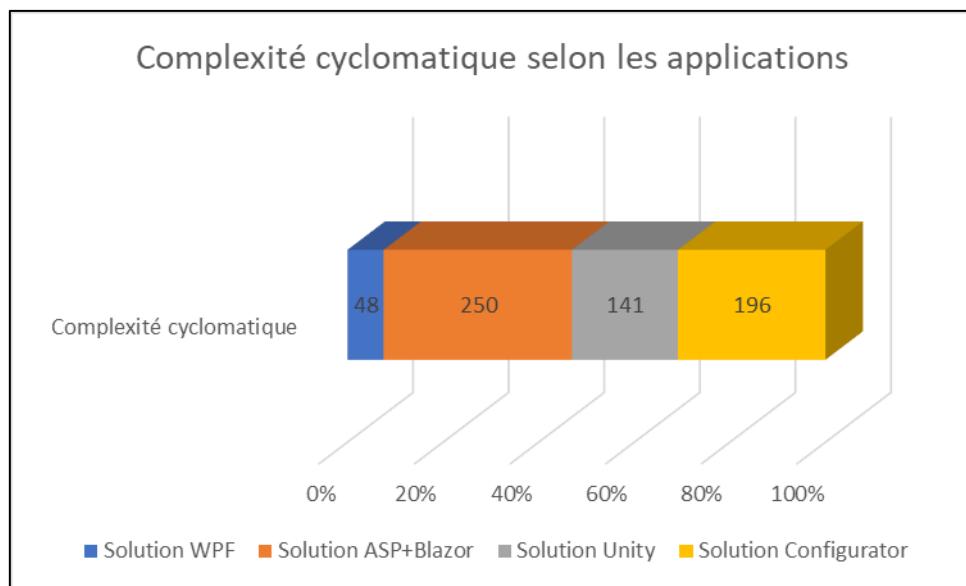
Graphique 1: Répartition du temps de travail

Ce graphique n'est pas représentatif du travail effectué, il illustre uniquement la répartition du code source dans les différentes applications développées.



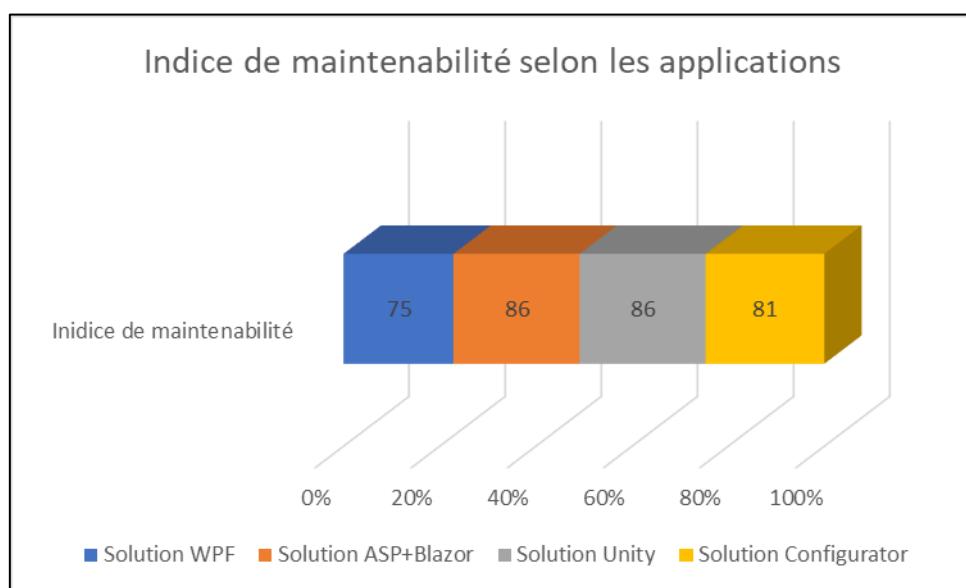
Graphique 2: Code source

Pour rappel, la complexité cyclomatique est un outil de métrique logicielle. Elle détermine le nombre de chemins possible d'exécution d'un code. Plus l'indice sera faible, plus sa couverture de code sera minime.



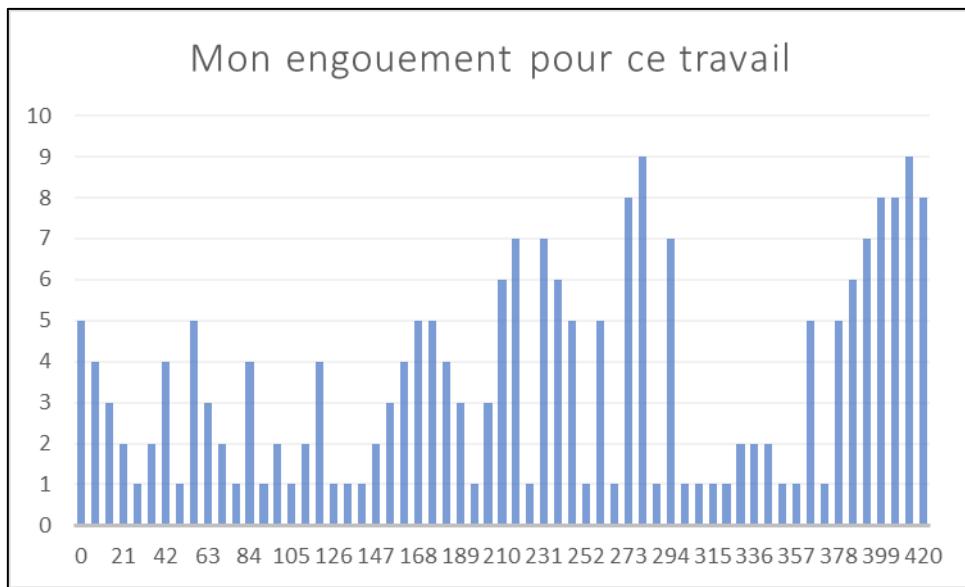
Graphique 3: Complexité cyclomatique

Cet indice reflète l'aisance à maintenir un code cohérent lors d'une modification sans devoir y consacrer beaucoup de temps. Dans Visual Studio, un indice inférieur à 10 est jugé faible, modéré entre 10 et 20 (non compris) puis vert entre 20 et 100.



Graphique 4: Indice de maintenabilité

Ce travail de Bachelor s'est déroulé du 21 février 2022 au 8 août 2022. Durant cette période, de nombreux aléas ont modifié mon ressenti vis-à-vis de ce TB. Malgré cela, ceci n'a pas impacté ma détermination ni les efforts fournis.



Graphique 5: Engouement pour ce travail

Quelques moments clés (- difficultés rencontrées, challengeant, + aboutissement, satisfaction)

- Découverte de Unity
- Test de lecture Bluetooth des mesures en C#
- *Enterprise Architect*
- Chargement d'une application dans le casque
- Unity
- + Test des HoloLens
- + Séances hebdomadaires
- Test Vuforia
- *Helix*
- Srialisation
- + NewtonSoft
- + *Tracking* fonctionnel
- Mise à jour Windows
- + Rédaction du rapport

5.4 Accomplissement des objectifs (à développer)

5.4.1 Vérification des spécifications de l'application Desktop

Spécification	Application Desktop	Étape	Etat	Commentaires
10.10.10	Le système doit être capable de lire les mesures d'un périphérique Bluetooth	1	Vert	Le système est capable de lire les mesures via l'application WPF
10.10.20	Un système de communication doit être établi entre le casque HoloLens et le PC de l'opérateur	1	Vert	L'échange de données se fait à l'aide de requêtes API.
10.10.30	La mesure émise par le pied à coulisse doit être affichée sur le casque AR	1	Vert	La mesure est reçue par l'application WPF, transmise au Controller puis récupérée par l'application Unity
10.10.40	Le système doit être capable de fonctionner pour au moins un type de pièce	1	Vert	L'application est bien capable d'indiquer les cotations sur le porte-stylo
10.10.50	Les données mesurées par l'opérateur doivent être sauvegardées dans un fichier	1	Vert	Ces données sont sauvegardées dans un fichier nommé "PiecesData.json"
20.10.10	Il doit être possible de mesurer plusieurs types de pièces	2	Vert	Le système a été testé avec deux pièces différentes mais il est possible d'en ajouter davantage
20.10.20	Les résultats des mesures doivent être transmis à l'application Blazor	2	Vert	L'application Blazor récupère les données de mesure à l'aide d'une requête API
30.10.10	Une nouvelle application devrait permettre de créer de nouveaux types de pièces	3	Vert	Holo2factoryConfigurator a été développé et il permet bien de créer de nouvelles recettes de cotations.

Tableau 17: Vérification des spécifications application Desktop

5.4.2 Vérification des spécifications de l'application Unity

Spécification	Application Unity	Étape	Etat	Commentaires
10.20.10	L'application Unity doit indiquer la cote à mesurer à l'opérateur	1	Vert	Un hologramme est placé pour indiquer la pièce à mesurer
10.20.20	L'application Unity doit implémenter une communication pour échanger des données entre le casque HoloLens et le PC de l'opérateur	1	Vert	L'échange de données entre le casque et l'application PC se fait par requête API
10.20.30	L'application Unity doit indiquer la plage de valeur attendue de la mesure	1	Vert	La plage est indiquée en utilisant la valeur minimale et maximale
10.20.40	L'application Unity doit indiquer si la mesure est conforme ou non	1	Vert	La couleur de la mesure détermine si la valeur est conforme ou non. Il y a également un <i>slider</i> qui se déplace en fonction de la mesure.
10.20.50	L'application Unity doit afficher le statut de la connexion avec le PC	1	Vert	Le statut n'est pas en permanence affiché mais un message d'erreur intervient dès la perte de connexion
10.20.60	L'application Unity doit afficher le statut de la connexion avec le pied à coulisse	1	Vert	L'application Unity n'affiche pas le statut de connexion car il est toujours visible sur le pied à coulisse
10.20.70	L'application Unity devrait contenir une aide contextuelle pour guider l'opérateur en cas de besoin	1	Vert	Il est possible d'accéder à l'aide en montrant la paume de sa main
10.20.80	L'application Unity doit être capable de se référencer spatialement à l'aide des QR code	1	Vert	Cette spécification est surclassée par le tracking
20.20.10	Les hologrammes changent de dimension en fonction de la cote à mesurer	2	Vert	Le <i>slider</i> se déplace en fonction de la valeur mesurée
20.20.20	L'application doit intégrer un système de sélection de type de pièce	2	Vert	L'opérateur peut choisir le type de pièce qu'il souhaite mesurer
30.20.10	Un système de tracking de pièce devrait être implanté	3	Vert	Le tracking est implanté
30.20.20	Les hologrammes d'indications de mesure devraient être placés en fonction du tracking	3	Jaune	Les hologrammes sont placés en fonction du tracking mais la précision laisse à désirer

Tableau 18: Vérification des spécifications Unity

5.4.3 Vérification des spécifications de l'application Blazor

Spécification	Application Blazor	Étape	Etat	Commentaires
20.30.10	L'application PC doit désormais héberger un serveur Blazor	2	Vert	Le serveur Blazor est inclus dans l'application ASP
20.30.20	Les données des mesures doivent être affichées sur un site web	2	Vert	Elles sont accessibles pour le superviseur sur le site web
20.30.30	L'application devrait intégrer un système de tri des mesures	2	Rouge	Pas implémenté (superficiel)
20.30.40	Ces mesures doivent être accessible pour une personne se trouvant sur le même réseau local	2	Vert	Quiconque peut accéder à ces mesures pour autant qu'il soit sur le réseau du serveur ASP
30.30.10	L'application devrait intégrer un affichage de mesures pour tout type de pièce	3	Vert	Les pièces sont distinguées par leur nom respectif

Tableau 19: Vérification des spécifications application Blazor

Les pourcentages présents dans le tableau sont approximatifs car l'accomplissement de certaines spécifications est discutable.

Nom de l'application	Spécifications obligatoires/facultatives	Taux de compléction ob./fac.
ASP	7/1	100%/100%
Blazor	3/2	100%/50%
Unity	9/3	88%/66%

Tableau 20: Synthèse de la vérification des spécifications

5.5 Perspectives de développement

Ce projet a abouti à un prototype fonctionnel qui remplit le concept de base. C'est-à-dire, de guider un opérateur dans une séquence de mesure. Le configIBUTEUR peut créer de nouvelles recettes de production et le superviseur a accès aux données de mesure.

Toutefois, il est possible d'y ajouter davantage de fonctionnalités.

Le *tracking* est fonctionnel mais sa précision laisse à désirer. Je ne pense pas qu'il soit possible d'améliorer les performances du SDK de Vuforia, les pièces ont été entraînées et le *dataset* provient d'un algorithme de *Deep Learning*. Je constate que le *tracking* n'est pas la meilleure approche dans ce cas d'étude. Il est plus adapté à détecter un élément fixe.

Une variante intéressante aurait été de combiner les deux solutions proposées dans ce travail. **Posage fixe** repéré par le *tracking*. Ceci fusionne les atouts des deux stratégies. La détection dynamique et la stabilité des hologrammes.

L'interface Blazor affiche les résultats des mesures, le site web pourrait inclure un espace de connexion et rendre ces données privées. Cette page web contient des données non sensibles, elles sont uniquement visionnables et ne peuvent donc pas être altérées par un tiers.

Le *toolkit MRTK 3* est sorti en juin 2022, ses designs sont retravaillés et les développeurs ont le focus sur cette technologie. Il serait intéressant de faire un *upgrade* de ce projet avec MRTK 3 et de développer d'avantage les fonctionnalités.

5.6 Conclusion

Ce travail de Bachelor résume l'étude et la conception de quatre applications distinctes.

Le développement d'une application de réalité augmentée avec le logiciel Unity. La lecture de mesures d'un appareil Bluetooth. La création d'une solution hébergeant un serveur ASP et Blazor permettant un échange de données à l'aide de requêtes API ainsi qu'une application habilitant un configurateur de recettes de générer une liste de cotations.

Durant ce Bachelor, je n'ai rarement été confronté à des projets de cette envergure. En effet, pendant notre cursus, nous réalisons un seul travail de semestre, et, en groupe. Nous ne sommes pas habitués à concevoir l'intégralité d'un projet. Ce travail est l'illustration parfaite d'un développement d'application. Il se décompose en différentes phases allant de l'analyse des besoins, la conception et le développement jusqu'à la vérification des spécifications.

J'ai acquis de nombreuses compétences dans le cadre de ce TB. L'apprentissage du logiciel Unity, les concepts de la réalité augmentée, le concept du *tracking*, l'implémentation de bibliothèques externes et de l'expérience en C#. Tout cela était fort enrichissant pour moi.

La réalité augmentée est un concept novateur qui est en train de se développer mais sa popularisation est pour l'instant restreinte. Le HoloLens 2 est une spécialisation d'un appareil de réalité augmentée, cela signifie que sa documentation est d'autant plus amoindrie. Durant ce travail, j'ai principalement effectué des prospections au sujet des technologies qui s'apprêtent au cas d'étude choisi et des recherches approfondies lors du développement des applications. Ceci a inéluctablement accru mon autonomie et mon indépendance.

Ce projet a utilisé de nombreux concepts étrennant, il était difficile de planifier correctement l'architecture de l'étude du projet. En effet, de nombreux événements inopinés se sont produits et ceci a impacté le déroulement des opérations. Mants changements de directions ont été réalisés et ces derniers, non planifiés, ont induit un retour à la réflexion UML. Il était nécessaire de réaliser un travail de cette envergure pour apprécier l'intérêt des diagrammes UML.

La réalité augmentée est une nouvelle technologie pleine de potentiel. Il est indubitable, elle suscite un réel intérêt pour l'industrie. Il est possible d'ajouter du contenu visuel pour un opérateur sans lui restreindre ses degrés de libertés. Les commandes vocales et oculaires sont un atout futuriste et deviendront, sans doute, un impératif. Sa popularité ne cesse de croître, notamment aux Etats-Unis et ce nouveau paradigme d'industrialisation s'instaurera très certainement dans les années à venir en Europe.

Quentin Fornerod



6 Bibliographie

6.1 Images

- [1] Microsoft, HoloLens 2. 2022 [en ligne] Adapté de <https://www.microsoft.com/en-us/hololens/buy>
- [2][3][4] Sylvac, Pied à coulisse. 2022 [en ligne] https://www.sylvac.ch/products/calipers/caliper-s_cal-evo-smart
- [5] Google Trends, HoloLens 2 trend. 2022 [en ligne] <https://trends.google.com/trends/?geo=CH>
- [6] OpenXR, Unifying Reality. 2022 [en ligne] <https://www.khronos.org/openxr/>
- [7] VisionLib, Augmented Reality. 2022 [en ligne] <https://visionlib.com/>
- [8] Vuforia, Developer Library. 2022 [en ligne] <https://library.vuforia.com/>

6.2 Documentation

Buildwagon, What happened to the Microsoft Hololens. 2022 [en ligne] <https://www.buildwagon.com/What-happened-to-the-HoloLens.html>

Woopsa, Woopsa. 2022 [en ligne] <https://woopsa.org/>

MRTK, What is Mixed Reality Toolkit 2. 2022 [en ligne] <https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05>

Témoignage Kruger, Kruger boosts employee productivity in factories using HoloLens 2 and Dynamics 365 Guides. 2021 [en ligne] <https://customers.microsoft.com/en-us/story/1356704042805187793-kruger-discrete-manufacturing-dynamics-365>

Googleblog, Code coverage best practices. 2020 [en ligne]. <https://testing.googleblog.com/2020/08/code-coverage-best-practices.html>

Toute image ou ressource non référencée provient de : Fornerod, Q. (2022), Holo2Factory

7 Abréviations/Définitions

Abréviation/définition	
AR	Augmented Reality (réalité augmentée)
TB	Travail de Bachelor
HoloLens	Casque de réalité augmentée de Microsoft
QR code	Quick Response code
.NET	Prononcé dotnet, c'est un framework développé par Microsoft
ASP	ASP.NET est un framework de Microsoft utilisé pour mettre en œuvre des applications web
Blazor	Blazor est un framework web open source de Microsoft permettant de créer des applications web en C#
UML	Unified Modeling Language (Diagrammes de visualisation dans les domaines du développement logiciel et en conception orientée objet)
Tracking	Action de suivre un élément dynamiquement dans l'espace (position, orientation)
API	Application Programming Interface: interface qui permet à des applications de communiquer entre elles.
API REST	Representational State Transfer (échanges de données basés sur HTTP)
CNC	Computer Numerical Control (Machine-outil à commande numérique)
HEIG-VD	Haute Ecole d'Ingénierie et de Gestion du canton de Vaud
EAI	Orientation du Bachelor en génie électrique : Electronique et Automatisation Industrielle.
MRTK	Mixed Reality Feature Tool, outil de configuration Microsoft
WPF	Windows Presentation Foundation: spécification graphique de Microsoft
Woopsa	Web Object-Oriented Protocol for Software and Automation
JSON	JavaScript Object Notation : format idéal pour transmettre/représenter un objet, des données.
DLL	Dynamic-link library : Bibliothèque Microsoft contenant du code ou des ressources pour une application
Razor	C'est la syntaxe de programmation pour des pages web en C#

FBX	Filmbox : format de fichier pour les modèles 3D incluant textures et caméra (point de vue de l'observateur)
OBJ	Format de fichier pour les modèles 3D basique sans textures
FPS	Frames Per Second : nombre d'images par seconde
MVC	Model View Controller: framework d'application web développé par Microsoft
UI	User Interface : c'est un contenu graphique pour l'utilisateur
ID	Identifiant
GAPS	Gaps is an Academical Planification System : plateforme web de la HEIG-VD
iAi	Institut d'Automatisation Industrielle : institut de la HEIG-VD
DL	Deep Learning : technique de Machine Learning utilisant des concepts avancés

Tableau 21: Abréviations/Définitions

8 Liste des figures

Figure 1: Aperçu de la réalité augmentée	19
Figure 2: Modèle du casque HoloLens 2 disponibles [1]	19
Figure 3: Prototype visuel étape 1.....	24
Figure 4: Protoype visuel étape 2 Figure 5: Prototype visuel Blazor	27
Figure 6: Prototype visuel étape 3.....	30
Figure 7: Schéma d'interaction entre les acteurs	34
Figure 8: Machine d'état de la séquence.....	35
Figure 9: Planning de la pré-étude (tâches à effectuer)	36
Figure 10: Pied à coulisse Sylvac [2]	36
Figure 11: Modèles de pieds à coulisse Sylvac [3]	37
Figure 12: Application Sylvac Anywhere [4]	37
Figure 13: Descriptif WPF	38
Figure 14: WPFAApp épurée.....	38
Figure 15: Recherches HoloLens Google Trends [5]	39
Figure 16: Toolbox MRTK Figure 17: Exemple UI MRTK	40
Figure 18: Overview OpenXR [6]	40
Figure 19: Génération de la solution Unity	41
Figure 20: Aperçu visuel Unity	41
Figure 21: Logo VisionLib [7] Figure 22: Logo Vuforia [8]	42
Figure 23: Posage fixe (plateau)	42
Figure 24: Résultat requête API dans le navigateur	43
Figure 25: Test des API avec ReqBin	44
Figure 26: Référentiel spatial avec Helix	45
Figure 27: Premier test avec Blazor	45
Figure 28: QR code GitHub Holo2Factory.....	46
Figure 29: Header application Blazor	57
Figure 30: Résultats des mesures Blazor	58
Figure 31: UI (sélection de l'utilisateur).....	60
Figure 32: UI (Message d'erreur)	60
Figure 33: UI (focus choix de la pièce) Figure 34: UI (choix de la pièce).....	61
Figure 35: UI (bannière de la séquence de mesure).....	61
Figure 36: UI (hand menu)	62
Figure 37: UI (hand menu) implémenté	62
Figure 38: UI (guide utilisateur)	63
Figure 39: Hiérarchie logicielle Unity.....	63
Figure 40: Model Target Generator (visuel)	67
Figure 41: Model Target Generator (viewpoint)	67
Figure 42: Tracking de la pièce	68
Figure 43: Création du dataset par DL en Cloud.....	68
Figure 44: Pièce de test (modélisation)	69
Figure 45: WPFAApp (réception d'une mesure) Figure 46: WPFAApp (en attente)	72
WPFAApp (initialisation)	72
Figure 48: WPFAApp (aide utilisateur).....	72
Figure 49: Holo2factoryConfigurator (MainWindow)	75
Figure 50: Holo2FactoryConfigurator (détails MainWindow)	76
Figure 51: Holo2FactoryConfigurator (ProductionWindow)	77
Figure 52: Holo2FactoryConfigurator (détails ProductionWindow)	78

Figure 53: Résultat de la liste de cotations exportée au format JSON	79
Figure 54: Convention de codage	80
Extrait de code 33: Tests unitaires Figure 55: Test unitaire du code.....	81
Figure 56: Couverture du code	82
Figure 57: Prototype visuel posage adaptatif.....	83
Figure 58: Placement de la cotation en fonction du tracking	84
Figure 59: Placement de la cotation en fonction du tracking 2.....	84
Figure 60: Connexion au compte professionnel Figure 61: Connexion Dynamics Guides sur HoloLens 2	85
Figure 62: Aperçu visuel Dynamics Guides (instruction)	85
Figure 63: Aperçu Dynamics Guides (statistiques)	86
Figure 64: Planning pré-étude avec heures réalisées.....	88

9 Liste des tableaux

Tableau 1: Cas d'utilisations	23
Tableau 2: Spécifications application Desktop étape 1	26
Tableau 3: Spécifications Unity étape 1	26
Tableau 4: Spécifications application Desktop étape 2	29
Tableau 5: Spécifications Unity étape 2	29
Tableau 6: Spécifications application Blazor étape 2	29
Tableau 7: Spécifications application Desktop étape 3	31
Tableau 8: Spécifications Unity étape 3	31
Tableau 9: Spécifications application Blazor étape 3	31
Tableau 10: Spécifications de rendu.....	31
Tableau 11: Synthèse des spécifications application Desktop	32
Tableau 12: Synthèse des spécifications Unity.....	32
Tableau 13: Synthèse des spécifications application Blazor	33
Tableau 14: Liste des URL du serveur ASP.....	43
Tableau 15: Liste des requêtes API disponibles.....	43
Tableau 16: Liste des requêtes API disponibles (complétée)	53
Tableau 17: Vérification des spécifications application Desktop	92
Tableau 18: Vérification des spécifications Unity.....	93
Tableau 19: Vérification des spécifications application Blazor	94
Tableau 20: Synthèse de la vérification des spécifications	94
Tableau 21: Abréviations/Définitions	99

10 Liste des diagrammes UML

Diagramme UML 1: Cas d'utilisations.....	20
Diagramme UML 2: Activités de l'opérateur	21
Diagramme UML 3: Séquence de mesure	21
Diagramme UML 4: Application PC étape 1	25
Diagramme UML 5: Application Unity étape 1.....	25
Diagramme UML 6: Application PC étape 2	28
Diagramme UML 7: Application Unity étape 2	28
Diagramme UML 8: Application PC étape 3	30
Diagramme UML 9: Application Unity étape 3	31

Diagramme UML 10: déploiement	33
Diagramme UML 11: ASP (partie 1).....	49
Diagramme UML 12: ASP (partie 2).....	50
Diagramme UML 13: Unity	59
Diagramme UML 14: WPFAApp	69
Diagramme UML 15: Holo2factoryConfigurator	73

11 Liste des témoignages

Témoignage 1: Entreprise Kruger	86
---------------------------------------	----

12 Liste des graphiques

Graphique 1: Répartition du temps de travail.....	89
Graphique 2: Code source	89
Graphique 3: Complexité cyclomatique	90
Graphique 4: Indice de maintenabilité	90
Graphique 5: Engouement pour ce travail	91

13 Liste des extraits de code

Extrait de code 1: Création d'une WebApplication	50
Extrait de code 2: Routing du Controller	50
Extrait de code 3: Création de la classe ProductionManager.....	51
Extrait de code 4: Récupération de l'instance ProductionManager.....	51
Extrait de code 5: Structure d'un singleton en C#	51
Extrait de code 6: Exemple d'une requête GET	52
Extrait de code 7: Exemple d'une requête PUT	52
Extrait de code 8: Démarrage de la séquence.....	53
Extrait de code 9: Initialisation des cotations mesurées	54
Extrait de code 10: Chargement des pièces mesurées.....	55
Extrait de code 11: Sauvegarde des pièces mesurées.....	55
Extrait de code 12: Chargement d'une recette selon le nom de la pièce	56
Extrait de code 13: Récupération des pièces mesurées (requête API)	56
Extrait de code 14: Requête HTTP pour la récupération des données de mesure	57
Extrait de code 15: Syntaxe "Virtualize" Blazor	57
Extrait de code 16: Syntaxe "Virtualize" Blazor (référence d'objet).....	58
Extrait de code 17: Adresse IP dynamique selon l'appareil.....	64
Extrait de code 18: Récupération des GameObject et affectation d'une fonction	64
Extrait de code 19: Récupération des TextMeshPro	64
Extrait de code 20: Requête cyclique	64
Extrait de code 21: Création de la requête PUT	65
Extrait de code 22: Envoi de la requête PUT	65
Extrait de code 23: Réponse à la requête PUT	66
Extrait de code 24: Affichage d'un message en cas d'erreur.....	66
Extrait de code 25: OkDialog (aperçu du code)	66
Extrait de code 26: Dispatcher timer WPF	70

Extrait de code 27: Contrôle de la connexion au serveur	70
Extrait de code 28: Ping vers le serveur ASP	71
Extrait de code 29: Récupération du statut de la requête	71
Extrait de code 30: Syntaxe C# new obsolète	80
Extrait de code 31: Syntaxe C# new dotnet 9.0.....	80
Extrait de code 32: Désactivation des warnings.....	81
Extrait de code 33: Tests unitaires Figure 55: Test unitaire du code.....	81

14 Annexes

Les annexes pdf présentes à la fin du rapport.

- Planning
- Guide utilisateur Holo2factoryConfigurator
- Mode d'emploi du pied à coulisse

Les annexes logicielles ont été déposées sur la plateforme GAPS. Elles sont également disponibles sur mon repository GitHub. Voir 3.10

- Projet Unity (le *build* du projet ayant une taille d'environ **8 Gb** ne sera pas remis)
- Solution Visual Studio WPF
- Solution Visual Studio ASP+Blazor
- Solution Visual Studio Holo2factoryConfigurator
- Diagramme UML
- Fichiers C# partagés (*shared*)

Note : l'ordre des éléments dans le dossier peut différer de la liste citée ci-dessus.

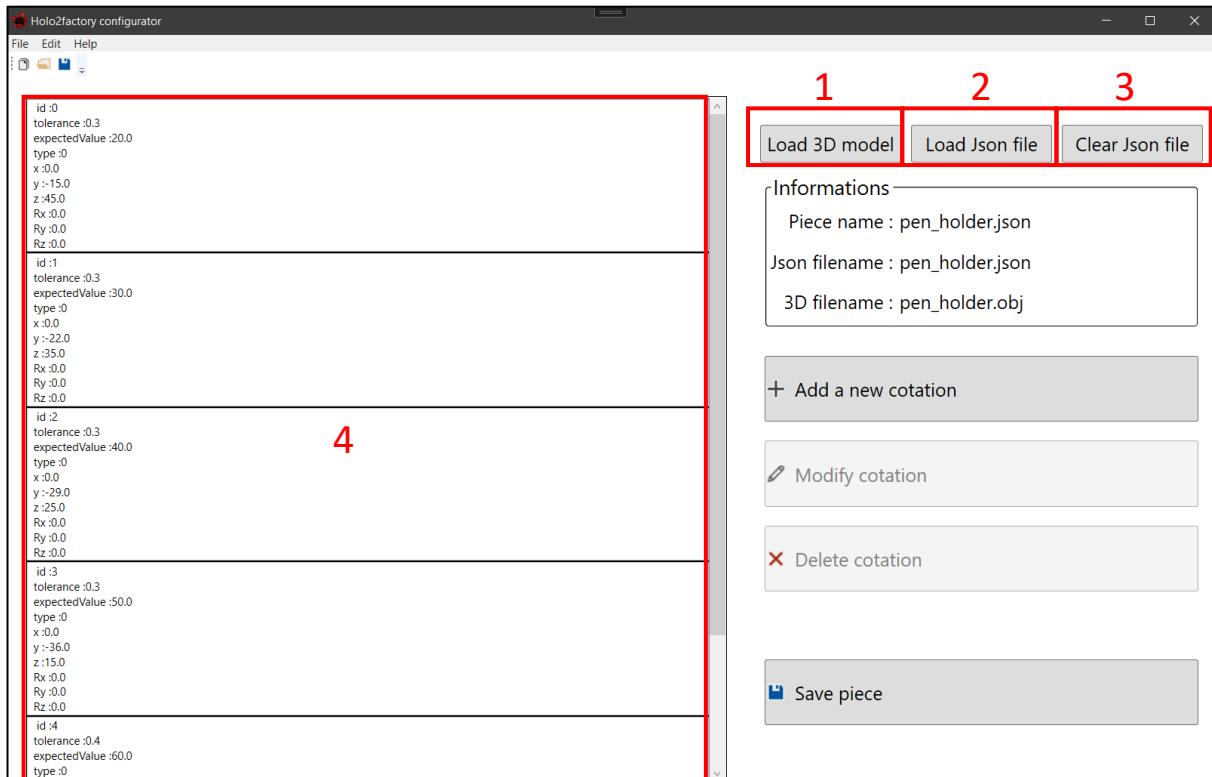
14.1 Planning

Description	Début	Effort estimé	s.1	s.2	s.3	s.4	s.5	s.6	s.7	s.8	s.9	s.10	s.11	s.12	s.13	s.14	s.15	s.16	s.17	s.18	s.19	s.20	s.21	s.22	Consommé	Reste à faire	Delta
Pré-étude																											
Prise en main de la technologie		91																									
Installation Unity - MRTK et Affichage d'un hologramme dans l'espace		10	8	2																						10	0.0
Communication pieuvre à coulisse/PC/Bluetooth		15	8	2																						12	3.0
Transfer de données entre PC+Hoblets (Wopsa/ASP)		10	1	2	4																					10	0.0
Implementation des interactions possibles avec l'utilisateur (Affichage, boutons)		10	1	2	4	3																			10	-0.0	
"Hello World" avec Blazor		8	4	1	6																				11	-3.0	
Définition de l'application		4	2	1	1	1																			3	1.0	
Analyse de la fonctionnalité		4	1	1	1	1																			4	0.0	
Diagrammes d'activités		4	2	2																					4	0.0	
Diagrammes de cas d'utilisation		4	2	2																					4	0.0	
Modèle de domaine (diag. De classes)		10	1	4	2	1																		8	2.0		
Rédaction des spécifications		4	1	1	1	3																		6	-2.0		
Conception de l'architecture logicielle		4	1	1	1	1																		3	1.0		
Planification détaillée de la réalisation		4	1	2	1	3																		6	-2.0		
Développement du projet																											
Etape 1																											
Application PC																											
Classe ProductionManager		8																							2	0.0	
Classe Production		4																						2	6	-2.0	
Classe ARController		10																						2	10	0.0	
Classe MyeRealtyServer		10																						3	10	0.0	
Classe MeasuredPiece		2																						2	0.0		
Classe Recipe		4																						3	7	-3.0	
Classe PieceComposition		4																						4	0.0		
Application Unity																											
Classe MainApp		16																						5	17	-1.0	
Classe HTFServer		8																						8	0.0		
Classe ActivePiece		4																						4	4	0.0	
Banner		4																						2	2	0.0	
Buttons		3																						1	1	3.0	
Labels		4																						0	4	0.0	
Repérage spatial		10																						3	10	-3.0	
Etape 2																											
Affichage des données Blazor		16																						0	0	0.0	
Classe Blazor		16																						8	16	0.0	
Affichage des résultats		16																						8	8	0.0	
Filtrage des données		8																						0	8	0.0	
Mise en forme		8																						4	4	0.0	
Application Unity		40																						10	49	-9.0	
Améliorations des hologrammes		5																						0	0	-5.0	
Etape 3																											
Création de pièce		16																						0	0	0.0	
Implémentation Hélix		40																						10	49	-9.0	
Interface WPF		16																						10	47	-7.0	
Applcation Unity		40																						2	0	0.0	
Implémentation du Tracking		40																						10	47	-7.0	
Préparation des livrables																											
Rédaction du rapport		30																						5	25	30	0.0
Réalisation de l'effice		8																						0	8	0.0	
Enregistrement vidéo de l'installation/montrage		16																						0	0	0.0	
Imprevus (10 %)		42																						0	0	0.0	
Gestion de Projet (-10 %)		40																						2	0	0.0	
Réalisation des plannings/séances hebdomadaires		0																						17	0	0.0	
TOTAL		420	10	14	11	14	13	15	14	9	11	12	9	10	40	38	35	42	33	42	47	0	560	0	0.0		

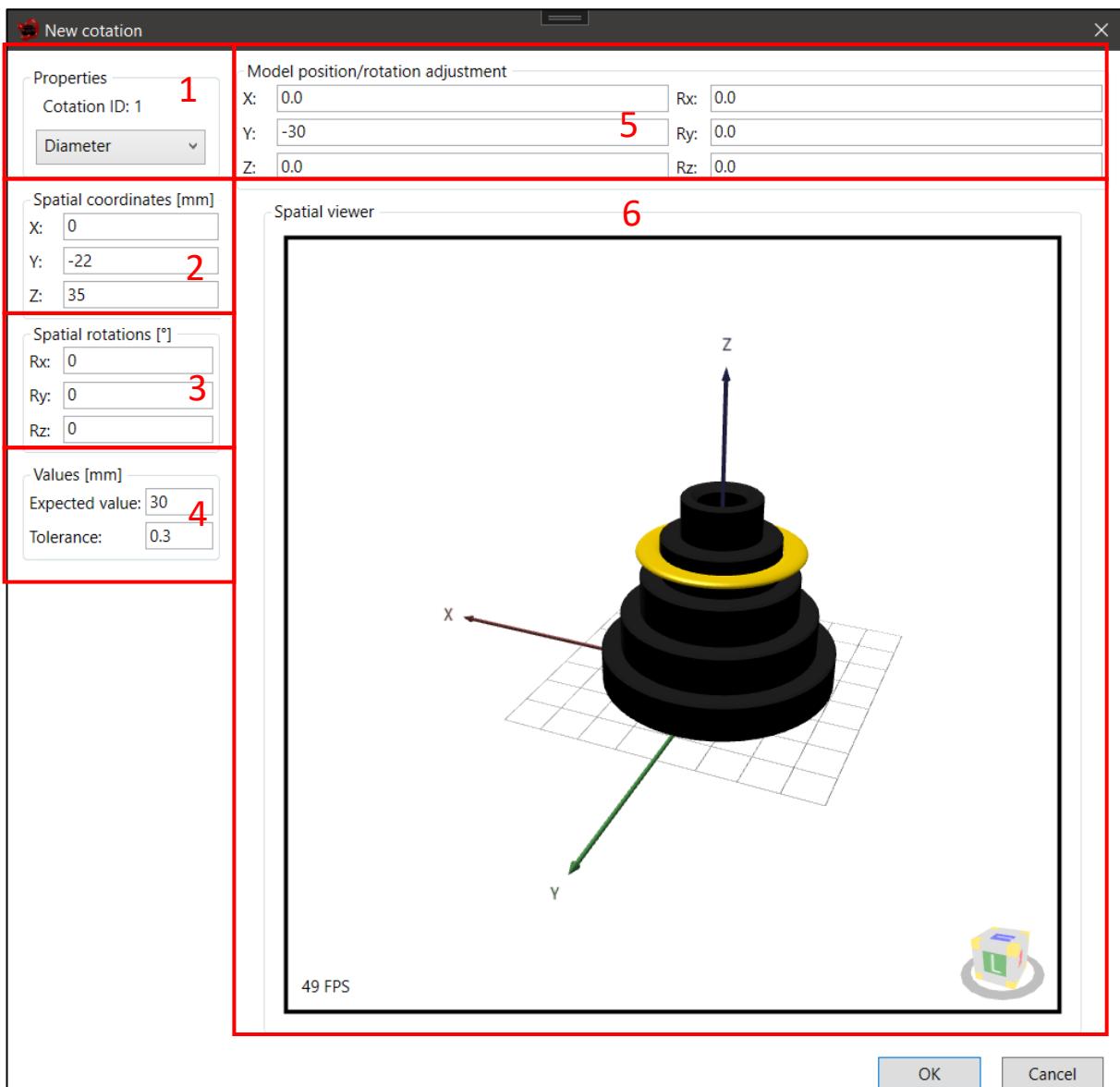
14.2 Manuel d'utilisateur Holo2FactoryConfigurateur

Holo2factory Configurator user guide

This application has as purpose to create a new list of cotations and set them correctly in the space. You have to correctly set the position and the orientation of the cotations. There is a 3D viewer to help you place the cotation and to have a brief renderer. The exported list will be used in the Unity application to display the cotations.



1. Load the 3D model, visible on the picture below
2. Open an existing json file (containing a list of cotations)
3. Clear the copy of the opened json file
4. List of cotations



1. Set the type of the cotation (Diameter, rectilign, hole)
2. Set the spatial coordinate of the cotation
3. Set the rotation of the cotation
4. Type the effective value of the cotation and the tolerance
5. Feel free to edit the position of the model
6. Preview, you can drag, rotate the model freely

14.3 Manuel d'utilisateur du pied à coulisse

S_Cal EVO Smart & Smart Micron

(Data/Favorite) Button
Bouton (Data/Favoris)
(Data/Favorite) Taste
Pulsante (Data/Preferto)
Botón (Data/Favoritos)

note (1)

DATA

MODE

SET

Bluetooth

Thumbwheel (depends on model)
Molette (selon modèle)
Antriebsrolle (je nach Modell)
Rotella (come modello)
Rueda dentada (según modelo)

Instructions Mode d'emploi Bedienungsanleitung Manuale d'uso Instrucciones de uso

sylvac

Replacing the battery
Changement de la pile
Auswechseln der Batterie
Sostituzione della batteria
Sustitución de la batería

Battery (CR2032)
Batterie (CR2032)
Batteria (CR2032)
Batería (CR2032)

23.45 → **PrE** → 10.00 → 0.00 → OFF → rESEt → MAC → Hld

(2) not available with Simple profile / non disponible avec profil Simple / nicht verfügbar mit Simple Profil / no disponible con perfil Simple

(3) Active configuration displays first / La configuration active s'affiche en premier / Die aktive Konfiguration wird zuerst angezeigt / La configurazione attiva viene visualizzata per prima / La configuración activa se muestra primera.

23.45 → **SET** → 0.00 → Hold → 2 SET → 3 PrE

> 2s. → **SET** → 0.00 → Hold → 2 SET → 3 PrE

> 4s. → (SIS) → OFF

> 6s. → OFF

(1) > 3s. → **MAC** → UnLoc

(1) > 3s. → **MAC** → Loc

off éteint / aus / spento / apagado	disconnected / déconnecté / keine Verbindung / scolligata / desconectado
blinking / clignote / blinkend / lampiggianti / parpadeante	advertising / détection / bereit zur Verbindung / scoperta / detección
on / allumé / stehend / acceso / encendido	connected / connecté / Verbindung hergestellt / collegata / conectado

rESEt clear pairing information / efface les informations d'appairage / Kopplung aufheben / cancellare le informazioni di accoppiamento / eliminar su información de emparejamiento

MAC display the MAC address / affiche l'adresse MAC / zeigt die MAC Adresse / visualizza l'indirizzo MAC / muestra la dirección MAC

SIMPLE profile without pairing / profil non appairé / Profil ohne Kopplung / profilo non accoppiato / perfil no apareado

PAIR paired and secured profile / profil appairé et sécurisé / Profil mit Kopplung und Verschlüsselung / profilo accoppiato e sicuro / perfil apareado y seguro

Hld virtual keyboard / clavier virtuel / virtuelle Tastatur / tastiera virtuale / teclado virtual

Frequency band / Bande de fréquence / Frequenzband / Banda di frequenza / Banda de frecuencia	2.4GHz (2.402 - 2.480GHz)
Modulation / Modulation / Modulation / Modulación	GFSK (Gaussian Frequency Shift Keying)
Max output power / Puissance max / Max Leistungsabgabe / Potenza di cresta / Potencia de pico	Class 3: 1mW (0dBm)
Range / Portée / Reichweite / Portata / Alcance	Open space/Espace Ouvert/Im Freien/Spazio aperto/Espacio abierto: <15 m Industrial environment/Environnement industriel/Industrielles Umfeld/Ambiente industriale/Entorno industrial: 1-5 m
Autonomy / Autonomie / Autonomie / Autonomía / Autonomia	Continuous: up to 2 months (Always connected with 4 values /s.) / jusqu'à 2 mois (Toujours connecté avec 4 valeurs /s.) / bis zu 2 Monaten (Immer verbunden mit 4 Werte /s.) / fino a 2 mesi (sempre connessi con i 4 valori /s.) / hasta 2 meses (siempre conectados con 4 valores /s.) Saver: up to 5 months (The instrument sends value only when the position has changed) / jusqu'à 5 mois (L'instrument envoie les valeurs quand la position change) / bis zu 5 Monaten (Das Gerät sendet die Werte nur wenn die Position geändert hat) / fino a 5 mesi (Lo strumento invia valore solo quando la posizione è cambiata) / hasta 5 meses (El instrumento envía valor sólo cuando la posición ha cambiado) Blind/Push: up to 7 months (Value is sent from the instrument (button) or requested from the computer) / jusqu'à 7 mois (La valeur est envoyée de l'instrument (bouton) ou demandée par ordinateur) / bis zu 7 Monaten (Der Wert wird von dem Gerät gesendet (Taste) oder vom Computer angefordert) / fino a 7 mesi (valore viene inviato dal (pulsante strumento) o richiesto dal computer) / hasta 7 meses (Valor se envía desde el (botón de instrumento) se puede solicitar a la computadora)

Technical data
Données techniques
Technische Daten
Specifiche
Especificación

Bluetooth

Resolution / Résolution / Auflösung / Risoluzione / Resolución	0.01 / .0005"	0.001 / .00005"	Specifications Spécifications Spezifikationen Specificazioni Especificaciones
Measuring range / Etendue de mesure / Messbereich / Campo di misura / Campo de medida	150 mm - 6" / 200mm - 8" / 300mm - 12"	150 mm - 6"	
Max. error / Erreur max. / Fehlertoleranz / Error max / Error máx	100nm; 20µm 100-300nm; 30µm (DIN862)	100nm; 15µm 100-150nm; 20µm (DIN862)	
Repeatability / Répétabilité / Wiederholbarkeit / Ripetibilità / Repetibilidad	10µm / .0005"	4µm @40mm / .00015" @1.5"	
Max slider speed / Vitesse max / Max Verteilgeschwindigkeit / Velocity max / Velocidad máx		Max 2.5m/s. - 100"/s.	
Display refresh rate / Nombre de mesure-s / Anzahl Messungen-s / Numero di misura-s / Número de medidas-s		> 10 / s.	
Data output / Sortie de données / Datenausgang / Uscita dati / Salida de datos		Bluetooth® 4.0 2.4GHz	
Mean power consumpt. / Consommation moyenne / Verbrauch / Consumo / Consumo		45 µA	
Battery life / Vie de la batterie / Batterie-Lebensdauer / Autonomia / Autonomía		Bluetooth® OFF : 8000h Bluetooth® ON : See Bluetooth® technical data	
Working temperature (storage) / Température de travail (stockage) / Arbeitstemperatur(Lagerung) / Temperatura operativa (stoccaggio) / Temperatura de trabajo (almacenamiento)		+5° to +40°C (-10° to +60°C)	
Weight / Poids / Gewicht / Peso / Peso	175g / 205g / 275g	175g	
Measuring system / Système de mesure / Messsystem / Sistema di misura / Sistema de medida		Sylvac system patented	
IP specification / Spécification IP / IP spezifikation / Specifica IP / Especificación IP		IP 67 (IEC60529)	

Advanced functions and additional information: refer to website www.sylvac.ch

CERTIFICATE OF CONFORMITY

We certify that this instrument has been manufactured in accordance with our Quality Standard and tested with reference to masters of certified traceability by the federal institute of Metrology.

CERTIFICAT DE CONFORMITÉ

Nous certifions que cet instrument a été fabriqué et contrôlé selon nos normes de qualité et en référence avec des étalons dont la traçabilité est reconnue par l'institut fédéral de métrologie.

QUALITÄTSZEUGNIS

Wir bestätigen, dass dieses Gerät gemäß unseren internen Qualitätsnormen hergestellt wurde und mittels Normalen mit anerkannter Rückverfolgbarkeit, kalibriert durch das eidgenössische Institut für Metrologie, geprüft worden ist.

CERTIFICATO DI CONFORMITÀ

Con il presente si certifica che questo strumento è stato prodotto secondo il nostro standard sulla qualità e controllato rispetto a campioni di riferibilità riconosciuta dall'istituto federale di metrologia.

CERTIFICADO DE CONFORMIDAD

Certificamos que este instrumento ha sido fabricado conforme a nuestras normas de calidad y ha sido controlado en relación con patrones de trazabilidad reconocida por la oficina nacional de metrología.

Certifications Certifications Zertifikat Certificado Certificado

Carefully dry all mechanical parts of the instrument after contact with liquids to ensure proper operation and avoid corrosion. Don't use aggressive products (alcohol, trichloroethylene or others) to clean plastic parts. Don't expose the instrument to direct sunlight, heat or humidity.

En cas de projections de liquides, essuyez les parties métalliques de l'instrument afin de garantir un bon fonctionnement mécanique, et évitez les problèmes de corrosion.
Ne pas utiliser de produits agressifs (alcool, trichloroéthylène, etc) pour le nettoyage des parties plastiques.
Ne pas entreposer l'instrument dans un endroit exposé au soleil, à la chaleur ou à l'humidité.

Nach Kontakt mit Flüssigkeit die mechanischen Teile gut trocken um ein einwandfreies Funktionieren zu garantieren und Rostprobleme zu vermeiden.
Keine aggressiven Produkte (Alkohol, Trichlorethylen und andere) für die Reinigung der Kunststoffteile benutzen.
Die Instrumente nicht an einem der Sonne, Hitze oder Feuchtigkeit ausgesetzten Ort aufzubewahren.

In caso di proiezione di liquidi, asciugare le parti metalliche dello strumento in maniera di garantire un buon funzionamento meccanico, ed per evitare i problemi di corrosione.
Non utilizzare prodotti aggressivi per la pulizia del parti in plastico.
Non erizzare lo strumento al sole, vicino a fudi di calore o all' umidità.

Seque cuidadosamente todas las partes metálicas del instrumento para evitar cualquier humedad y así garantizar un buen funcionamiento mecánico y evitar óxido.
No use productos agresivos (alcohol,tricloroetileno u otros) para limpiar las partes de plástico.
No guarde el instrumento en el sol, calor o humedad.

Maintenance Maintenance Unterhalt Avviso Mantenimiento

DESCRIPTION OF BLUETOOTH® MODULE:

This module is based on Nordic Semiconductor nRF8001 μBlue Bluetooth Low Energy Platform. The nRF8001 is a single chip transceiver with an embedded baseband protocol engine, suitable for ultra-low power wireless applications conforming to the Bluetooth Low Energy Specification contained within v4.0 of the overall Bluetooth specification. The nRF8001, used in the current revision of ISP091201, is a production product using a RoM for the baseband protocol engine.

Bluetooth® Certification

US/CANADA CERTIFICATION

Sylvac

m.n : S_Cal EVO

This device contains
FCC ID: 2AAQS-ISP091201
IC: 11306A-ISP091201

BRAZIL CERTIFICATION

Este equipamento opera em caráter secundário, isto é, não tem direito à proteção contra interferência prejudicial, mesmo de estações do mesmo tipo e não pode causar interferência a sistemas operando em caráter primário.



SOUTH KOREA CERTIFICATION

MSIP-CRM-INs-ISP091201

Class A Equipment (Industrial Use)

이 기기는 업무용(A급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정 외의 지역에서 사용하는 것은 목적으로 합니다.

JAPAN CERTIFICATION



R 001-A06167

TAIWAN CERTIFICATION



經型式認證合格之低功率射頻電機，非經許可，公司、商號或使用者均不得擅自變更頻率、加大功率或變更原設計之特性及功能。
低功率射頻電機之使用不得影響飛航安全及干擾合法通信，經發現有干擾現象時，應立即停用，並改善至無干擾時方得繼續使用。
前項合法通信，指依電信法規定作業之無線電通信。低功率射頻電機須忍受合法通信或工業、科學及醫療用電波輻射性電機設備之干擾。

MEXICO CERTIFICATION

La operación de este equipo está sujeta a las siguientes dos condiciones: (1) es posible que este equipo o dispositivo no cause interferencia perjudicial y (2) este equipo o dispositivo debe aceptar cualquier interferencia, incluyendo la que pueda causar su operación no deseada.

Contiene módulo inalámbrico
Marca: Sylvac
Modelo: ISP091201D
IFT: RCPSYIS14-0655

NOTICE:

Changes or modifications made to this equipment not expressly approved by Sylvac may void the FCC authorization to operate this equipment.

NOTICE:

This device complies with Part 15 of the FCC Rules and with RSS-210 of Industry Canada. Operation is subject to the following two conditions,

(1) this device may not cause harmful interference, and

(2) this device must accept any interference received, including interference that may cause undesired operation.

NOTE:

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

RADIOFREQUENCY RADIATION EXPOSURE INFORMATION:

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance of 20 cm between the radiator and your body.
This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.



Changes without prior notice
Sous réserve de toute modification
Änderungen vorbehaltlich
Soggetto a modifica senza preavviso
Reservados los derechos de modificación sin previo aviso

IP67

Edition : 2018.10 / 681.295.04
www.sylvac.ch

The Bluetooth® word mark and logos are registered trademarks owned by the Bluetooth SIG, Inc. and any use of such marks by Sylvac is under license. Other trademarks and trade names are those of their respective owners.

14.4 Protocole de test Holo2factoryConfigurator

Tests effectués sur l'application WPF Holo2factoryConfigurator

N° du test	Description	Résultat
1	Chargement d'un modèle 3D	OK
2	Chargement d'un fichier Json	OK
3	Chargement d'un fichier Json corrompu -> doit afficher une erreur	OK
4	Réinitialiser la recette en cours	OK
5	Ajout d'une cotation	OK
6	Erreur en cas d'ajout de cotation avec valeur incorrecte	OK
7	Modification d'une cotation seulement si une cotation est sélectionnée	OK
8	Suppression d'une cotation seulement si une cotation est sélectionnée	OK
9	Sauvegarde la recette en cas de changement	Sauvegarde sous... possible si aucun changement
10	Annulation de l'enregistrement	OK
11	Annulation du chargement d'un modèle 3D	OK
12	Annulation du chargement d'un fichier Json	OK
13	Déplacement du modèle dans le 3D viewer (x,y,z,Rx,Ry,Rz)	OK
14	Erreur si tolérance de mesure trop faible	OK
15	Erreur si mesure inférieure à zéro	OK
16	Annulation lors de l'ajout d'une cotation	OK
17	Annulation lors de la modification d'une cotation	OK
18	Erreur en cas de valeur d'offset du modèle incorrecte	OK