Hes·so VALAIS WALLIS
Haute Ecole de Gestion
Hochschule für Wirtschaft

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2022

Interagir avec des objets réels en réalité virtuelle

Etudiant : Samuel Wenger

Professeur : Antoine Widmer

Résumé

Ce travail traite de l'utilisation de la réalité virtuelle à des fins pédagogiques. Nous cherchons à déterminer si son utilisation permet de repenser les moyens d'enseignement traditionnels pour améliorer leur impact et leur efficacité auprès des étudiants.

Dans ce travail, nous nous intéressons plus particulièrement aux technologies permettant la détection d'objets réels en réalité virtuelle. Nous cherchons à déterminer quelle technologie actuelle répond le mieux à nos besoins et à valider son choix.

Sur la base de nos recherches, nous avons pu établir un état de l'art sur le sujet. Ainsi, nous avons démontré l'utilité de l'utilisation de la réalité virtuelle dans le domaine de l'enseignement et ses avantages sur l'engagement des étudiants.

Nous avons également effectué une analyse des technologies existantes permettant la détection d'objets et nous avons déterminé celles que nous souhaitions utiliser dans le cadre de la réalisation d'un prototype d'application. En appliquant la méthodologie Scrum, nous avons mis en place un prototype qui a permis de valider notre choix technologique et de démontrer que la détection d'objets en réalité virtuelle est possible.

Finalement, nous avons exposé les perspectives d'évolution de ce projet et mis en avant quelques limites techniques détectées pendant la mise en place du prototype.

Mots-clés : réalité virtuelle, VR, interactions, objets réels, enseignement.

Avant-propos et remerciements

La thématique traitée dans ce rapport est la mise en place d'une interaction avec des objets réels dans la réalité virtuelle à travers la réalisation d'un prototype.

Ce rapport est réalisé dans le cadre du module 656-1 « Travail de Bachelor » de la filière Informatique de Gestion de la Haute Ecole de Gestion. Le thème a été proposé par le professeur M. Antoine Widmer, en collaboration avec l'Orif, organisation romande qui a notamment pour mission la formation et l'intégration socioprofessionnelle de personnes atteintes dans leur santé ou en difficulté.

Les potentiels de cette technologie étant très large, le travail réalisé dans ce rapport consiste principalement à faire un état de l'art des technologies existantes afin de déterminer lesquelles pourront êtres utilisées dans la réalisation de ce projet. Ce choix sera validé par la mise en place d'un prototype. L'objectif de ce rapport est de présenter le contexte du projet ainsi que de documenter les étapes de sa réalisation et les problèmes rencontrés.

Nous souhaitons remercier M. Antoine Widmer, professeur responsable du projet, pour ses conseils, son accompagnement et son soutien technique tout au long de la réalisation du travail, ainsi que toutes les personnes ayant contribué à la rédaction de ce document ou à sa relecture.

Hes·so VALAIS WALLIS
Haute Ecole de Gestion
Hochschule für Wirtschaft

Table des matières

LISTE DES TABLEAUX **1**

LISTE DES FIGURES **2**

LISTE DES ABRÉVIATIONS **4**

1. INTRODUCTION **5**

2. OBJECTIFS ET PROBLÉMATIQUE **6**

2.1. CONTEXTE DU PROJET 6

2.2. ETAPES PRÉVUES POUR LA RÉALISATION DU PROJET 7

2.3. MÉTHODOLOGIE UTILISÉE 7

3. ETAT DE L'ART **8**

3.1. LA RÉALITÉ VIRTUELLE ET SES UTILISATIONS ACTUELLES 8

3.1.1. Qu'est-ce que la réalité virtuelle ? *8*

3.1.2. Utilisations actuelles *9*

3.1.3. La réalité virtuelle dans le cadre pédagogique *9*

3.2. LA DÉTECTION D'OBJETS EN RÉALITÉ VIRTUELLE 10

3.2.1. Détection avec des marqueurs *11*

3.2.2. Détection sans marqueurs *13*

3.3. TECHNOLOGIES EXISTANTES 14

3.3.1. Vuforia *15*

3.3.2. OpenCV *16*

4. RÉALISATION D'UN PROTOTYPE **18**

4.1. CHOIX TECHNOLOGIQUES 18

4.1.1. Unity *18*

4.1.2. OpenCV *19*

4.1.3. ArUco Markers *19*

4.2. GESTION DE PROJET 22
4.2.1. Product Backlog 22
4.3. PHASES DE DÉVELOPPEMENT 22
4.3.1. Sprint 0 – Mise en place 23
4.3.2. Sprint 1 – Création de marqueurs ArUco 25
4.3.3. Sprint 2 – Obtenir la position et l'orientation d'un marqueur 32
4.3.4. Sprint 3 – Détection simultanée de plusieurs marqueurs 36
4.3.5. Sprint 4 – Détection des marqueurs dans une scène de réalité virtuelle 42
4.3.6. Sprint 5 – Détection d'objets réels dans une scène de réalité virtuelle 47
5. CONCLUSION 51
5.1. COMPÉTENCES DÉVELOPPÉES ET DIFFICULTÉS RENCONTRÉES 51
5.2. PERSPECTIVES D'ÉVOLUTION 52
5.3. LIMITES DÉTECTÉES 52
ANNEXES 53
ANNEXE I – MARCHE À SUIVRE POUR LA CRÉATION D'UN NOUVEAU MARQUEUR 53
ANNEXE II – PROCÉDURE POUR RECONNAÎTRE UN NOUVEAU MARQUEUR DANS LA SCÈNE 55
ANNEXE III – EXEMPLES DE MARQUEURS 57
ANNEXE IV – PRODUCT BACKLOG 58
ANNEXE V – RELEASE ROADMAP AND PROJECT VELOCITY 59
ANNEXE VI – BURNDOWN CHARTS 60
BIBLIOGRAPHIE 62
DÉCLARATION DE L'AUTEUR 64

Hes·so VALAIS WALLIS
Haute Ecole de Gestion
Hochschule für Wirtschaft

Liste des tableaux

Tableau 1 - Comparatif des solutions existantes pour l'utilisation de ArUco et OpenCV 21

Liste des figures

Figure 1 – Atelier de la section construction métallique de l'Orif de Sion 6
Figure 2 - Utilisation de la réalité virtuelle dans l'enseignement 10
Figure 3 - Exemple de marqueurs spécialisés 11
Figure 4 - Exemple de marqueurs image classique 12
Figure 5 - Exemple de RA sans marqueurs chez Ikea 13
Figure 6 - Exemple de RA sans marqueurs basée sur la localisation 14
Figure 7 - Logo Vuforia 15
Figure 8 - Logo OpenCV.....16
Figure 9 - Logo Unity.....18
Figure 10 - Marqueurs ArUco.....20
Figure 11 - Capture d'écran de Unity 2017.2 23
Figure 12 - Capture d'écran du package « ArUcoUnity ».....24
Figure 13 - Extrait du Product Backlog pour le Sprint 1 25
Figure 14 - Capture d'écran de la configuration d'un marqueur 26
Figure 15 - Capture d'écran de la calibration de la caméra 26
Figure 16 - Capture d'écran de la configuration de la détection de marqueurs 27
Figure 17 - Capture d'écran du Package Manager de Unity.....28
Figure 18 - Capture d'écran de l'écran de création d'un marqueur.....29
Figure 19 – Capture d'écran de la détection d'un marqueur avec l'identifiant n°1 30
Figure 20 - Capture d'écran de la console affichant l'identifiant du marqueur détecté 30
Figure 21 – Extrait du Product Backlog pour le Sprint 2 32
Figure 22 - Capture d'écran de la configuration d'un objet 3D 32
Figure 23 - Capture d'écran de la console affichant la position et l'orientation du marqueur 34
Figure 24 - Capture d'écran du code permettant de définir les matrices de transformation 34
Figure 25 - Capture d'écran du code permettant d'appliquer la matrice de transformation 35
Figure 26 - Capture d'écran du placement d'un objet 3D sur le marqueur détecté.....35
Figure 27 - Extrait du Product Backlog pour le Sprint 3 36
Figure 28 - Capture d'écran de la détection simultanée de deux marqueurs 36

Figure 29 – Capture d’écran de la configuration d'un marqueur à détecter dans la scène 37
Figure 30 - Capture d'écran de la structure de la scène et ses objets.....38
Figure 31 - Capture d'écran de la configuration du premier script dans l'objet Quad.....38
Figure 32 - Capture d'écran de la configuration du second script dans l'objet Quad 39
Figure 33 - Capture d'écran du code déplaçant l’objet en fonction de son marqueur 40
Figure 34 - Capture d'écran des objets 3D placés selon leur marqueur respectif.....40
Figure 35 - Extrait du Product Backlog pour le Sprint 4 42
Figure 36 - Capture d'écran de la structure des objets de la scène en AR.....42
Figure 37 - Capture d'écran de la structure des objets de la scène en VR.....43
Figure 38 - Capture d'écran de la configuration de la caméra.....44
Figure 39 - Capture d'écran de la configuration du premier script sur la MainCamera.....45
Figure 40 - Capture d'écran de la configuration du second script sur la MainCamera.....45
Figure 41 - Capture d'écran des objets 3D placés selon leur marqueur respectif en VR 46
Figure 42 - Extrait du Product Backlog pour le Sprint 5 47
Figure 43 - Objets 3D utilisés pour le prototype.....48
Figure 44 - Capture d'écran des trois objets 3D représentés dans une scène de VR simple 48
Figure 45 - Capture d'écran de la représentation des trois objets 3D dans la scène de VR finale .49

Liste des abréviations

Abréviations issues de la langue française

RV	Réalité virtuelle
RA	Réalité augmentée
2D	2 dimensions
3D	3 dimensions

Abréviations issues de la langue anglaise

VR	Virtual Reality
AR	Augmented Reality
SDK	Software Development Kit
QR Code	Quick Response Code

1. Introduction

A l'heure où de grandes sociétés telles que Meta (anciennement Facebook) ou Microsoft investissent de manière importante dans le développement du métavers, un monde virtuel régulièrement décrit comme la future version d'internet, l'intérêt de la réalité virtuelle est une nouvelle fois mis en avant. Selon des études du secteur, les marchés de la réalité virtuelle et de la réalité augmentée devraient atteindre près de 600 milliards de dollars d'ici 2025. L'apparition de nouveaux produits développés par des entreprises telles que Sony, Lenovo ou HTC suggère une forte demande des consommateurs pour des produits qui modifient la réalité. (Bidar, 2022)

Dans l'enseignement, les outils numériques sont de plus en plus utilisés pour revoir certains moyens d'enseignement traditionnels. En plus d'améliorer l'engagement des élèves ou des étudiants, l'utilisation du numérique complète les supports de cours actuels et modernise la manière d'amener du contenu aux étudiants.

Dans ce contexte, nous faisons l'hypothèse que la réalité virtuelle permet d'améliorer l'efficacité des moyens d'enseignement traditionnels.

Dans ce rapport, nous commencerons par présenter le contexte du projet et les étapes de sa réalisation. Dans un deuxième temps, nous nous intéresserons à la réalité virtuelle et au potentiel que représente cette technologie. Nous présenterons les solutions disponibles pour la réalisation du projet et nous chercherons à identifier celles qui correspondent le mieux aux besoins. Dans la troisième partie, nous détaillerons les étapes du développement de notre prototype. Finalement, en guise de conclusion, nous tirerons un bilan du travail réalisé et nous présenterons les perspectives d'évolution du prototype.

2. Objectifs et problématique

2.1. Contexte du projet

L'Orif, Organisation Romande d'Intégration et de Formation, est active dans toute la Suisse Romande. Elle a notamment pour mission l'orientation, la formation et l'intégration professionnelle durable du plus grand nombre de personnes en difficulté dans le marché du travail. Les formations dispensées sur le site de Sion comprennent les métiers du bâtiment (carrelage, peinture, maçonnerie, menuiserie, sanitaire, construction métallique, paysagisme), de la restauration (cuisine, service) et de service (intendance, conciergerie, assistant en soin, vente commerce de détail). (Orif, s.d.)

Figure 1 - Atelier de la section construction métallique de l'Orif de Sion

Source : https://www.orif.ch/fr/sites-orif/valais/orif-sion

Dans le cadre de la formation des constructeurs métallique, les maîtres socio-professionnels ont émis le souhait de pouvoir utiliser la réalité virtuelle pour la formation de leurs étudiants. L'utilisation de cette technologie dans le cadre de la formation offrirait la possibilité d'appréhender les principaux gestes du métier, de les apprendre et de les entraîner dans des conditions proches de la réalité. Il est souhaité que des outils réels tel que des scies, des poinçonneuses ou des foreuses puissent être utilisés par les participants et être visualisés dans la réalité virtuelle pour rendre l'immersion la plus complète possible.

Le projet initié avec la HES-SO Valais-Wallis vise donc à mettre en place une application de réalité virtuelle permettant de simuler l'activité des constructeurs métallique à des fins de formation. Le projet réalisé dans le cadre de ce travail de Bachelor est un sous-projet de la réalisation de cette application. Il consiste en la réalisation d'un prototype qui permette de détecter la position et l'orientation d'un objet réel afin de le visualiser en temps réel dans la réalité virtuelle.

2.2. Etapes prévues pour la réalisation du projet

La première étape de la réalisation de ce travail consiste à s'intéresser à la technologie VR et à effectuer une analyse des technologies existantes pour la détection d'objets en réalité virtuelle. Il conviendra d'en analyser plusieurs et de déterminer celle qui convient le mieux aux besoins du projet. Dans un deuxième temps et afin de valider le choix de technologie, un prototype sera mise en place.

2.3. Méthodologie utilisée

Dans le cadre du développement du prototype, nous utiliserons la méthodologie Scrum. Un Product Backlog sera mis en place pour lister les fonctionnalités nécessaires à la bonne marche de l'application. La réalisation de ce projet s'est étendue du 21 février 2022 au 12 août 2022. Le travail de développement du prototype a débuté en mai 2022 et s'est réalisé sur cinq Sprints de trois semaines.

3. Etat de l'art

3.1. La réalité virtuelle et ses utilisations actuelles

3.1.1. Qu'est-ce que la réalité virtuelle ?

La réalité virtuelle est un ensemble de technologies informatiques qui permet de plonger une personne dans un environnement virtuel créé par des logiciels. On parle alors d'immersion dans un monde artificiellement créé. Il peut s'agir d'une reproduction du monde réel ou d'un univers totalement imaginaire. La réalité virtuelle se distingue de la réalité augmentée qui elle consiste à ajouter des éléments virtuels sur un environnement réel. On parle dans ce cas d'amélioration de la réalité. (Futura, explorer le monde, s.d.)

Le plus souvent, l'immersion se fait au moyen d'un casque qui place un système d'affichage 3D devant les yeux de l'utilisateur. Le dispositif doit permettre d'envoyer l'utilisateur dans un environnement virtuel et lui faire croire que les scènes qui lui sont présentées sont réelles. Si la vue est sollicitée, l'expérience est bien souvent complétée par l'utilisation d'autres sens tels que l'ouïe ou le toucher pour améliorer l'immersion. (Futura, explorer le monde, s.d.)

Les premiers casques de réalité virtuelle datent des années 1970. La NASA a joué un rôle important dans le développement de cette technologie en donnant une importante impulsion dans le cadre d'un programme de recherche. Des casques développés pour le grand public, principalement dans le domaine des jeux vidéo, ont également fait leur apparition dans les années suivantes. Cependant, leur prix prohibitif et l'inconfort de ces solutions a surtout mené à des échecs commerciaux. L'arrivée de Palmer Luckey en 2009 a largement contribué à la démocratisation de la réalité virtuelle. Ce précurseur de la VR s'est lancé dans un projet de casque de réalité virtuelle et en a développé plusieurs prototypes. Grâce à des financements participatifs et en partageant ses avancements sur internet, il a réussi à étendre sa notoriété. Face au succès rencontré, il a alors créé sa propre structure *Oculus VR* en 2012 qui a mené plus tard à la commercialisation de l'*Occulus Rift* qu'on connaît aujourd'hui. Sa société a par la suite été vendue à Facebook pour plusieurs milliards de dollars. (Artefacto, s.d.)

3.1.2. Utilisations actuelles

Pour le grand public, la réalité virtuelle est aujourd'hui largement associée aux jeux vidéo. S'il est vrai que la réalité virtuelle a dans un premier temps été développée pour améliorer l'expérience vidéo de divertissement par une immersion totale, l'utilisation qui en est faite aujourd'hui est bien plus large. Les barrières connues dans le passé, notamment le prix ou d'autres freins techniques, disparaissent peu à peu et les utilisations sont de plus en plus diverses. Aujourd'hui, la réalité virtuelle est par exemple utilisée dans le domaine militaire pour l'entraînement des pilotes d'avion ou dans le domaine de la santé pour des campagnes de sensibilisation ou le traitement de phobies. Son utilisation se démocratise également de plus en plus dans l'immobilier, l'événementiel, la muséographie ou la formation. (Artefacto, s.d.)

3.1.3. La réalité virtuelle dans le cadre pédagogique

L'utilisation de la réalité virtuelle dans l'enseignement s'est largement démocratisée dans les dernières années. Les établissements de formation cherchent à redynamiser leurs enseignements et la réalité virtuelle est une alternative populaire que certaines écoles ont décidé de creuser. La démocratisation de cette technologie et la baisse des coûts de mise en place ont permis de rendre bien plus abordable l'accès à des contenus de qualité, ce qui explique également l'essor de cette technologie dans le cadre pédagogique.

Des études ont mis en avant le fait que l'utilisation de technologies numériques dans l'enseignement permet de maintenir un meilleur engagement des étudiants. En effet, elles permettent de capter leur attention pendant une plus longue période de temps qu'avec des moyens d'enseignement traditionnels. La réalité virtuelle est une des options disponibles et les recherches montrent qu'elle est spécialement efficace pour impliquer les enfants de la génération actuelle. (Larmand, 2022)

Figure 2 - Utilisation de la réalité virtuelle dans l'enseignement

Source : https://www.nfp77.ch/fr/portfolio/l-utilite-de-la-realite-virtuelle-dans-l-enseignement-des-sciences-naturelles/

Cette technologie, lorsqu'elle est utilisée comme outil pédagogique, permet aux enseignants d'immerger les élèves dans le contenu qui leur est enseigné car elle permet d'illustrer de manière vivante les propos de l'enseignant. Directement impliqués, les élèves ont alors plus de facilité à reconnaître le sens de ce qu'ils apprennent. L'utilisation de la réalité virtuelle complète donc le matériel d'enseignement et permet un apprentissage plus approfondi. (Larmand, 2022)

L'utilisation de la réalité virtuelle dans le cadre pédagogique ouvre de nombreuses possibilités et représente de nombreux avantages. Elle permet notamment de réaliser des tâches sans se mettre en danger. Les erreurs, souvent décrites comme formatrices dans le domaine de la formation, deviennent alors possibles et cela sans que la sécurité des étudiants ne soit mise en cause. La VR permet de reproduire ou simuler des conditions rares ou extrêmes et de modéliser des terrains d'entraînements habituellement inaccessibles (lieux inaccessibles ou vues aériennes). (Lourdeaux, 2004)

3.2. La détection d'objets en réalité virtuelle

Dans le cadre de ce travail, nous cherchons à détecter des objets réels afin de les représenter dans la réalité virtuelle. En réalité augmentée et en réalité virtuelle, plusieurs technologies existent et permettent de répondre à ce besoin. Si les technologies se basent souvent sur l'utilisation de

marqueurs semblables à des QR-Codes placés sur les objets, il existe également des solutions qui n'utilisent aucun marqueur et qui se basent uniquement sur la reconnaissance de l'objet réel.

3.2.1. Détection avec des marqueurs

Pour la détection d'objets, la réalité augmentée et la réalité virtuelle se basent souvent sur une technologie permettant de reconnaître un marqueur dans un flux vidéo. Ce marqueur permettra de déterminer la position et l'orientation de l'objet pour le reconnaître dans l'espace. Cette manière de fonctionner requiert de connaître préalablement l'objet pour pouvoir le reconnaître par la suite. C'est pour cette raison qu'on parle de marqueurs. Dans l'absolu, tout objet peut devenir un marqueur, mais il en existe différents types. (3d émotion, s.d.) (Paladini, 2018)

Marqueurs spécialisés

Figure 3 - Exemple de marqueurs spécialisés

Source : https://www.v-cult.com/blog/2017/10/02/la-realite-augmentee-quest-ce-que-cest/

Il s'agit du marqueur historique lorsqu'on parle de reconnaissance en réalité virtuelle ou en réalité augmentée. Les marqueurs spécialisés prennent la forme d'un code en noir et blanc proche d'un QR-Code. Ils ont l'avantage d'être facilement reconnus et peuvent être facilement orientés. C'est celui qui fournit clairement les meilleurs résultats en terme de tracking dans un flux vidéo. De plus, le coût en terme de calculs est faible, ce qui rend possible de suivre plusieurs objets en même temps sans rencontrer de difficultés. Cependant, le désavantage principal de ce marqueur est son design qui ne passe pas inaperçu. Cela peut représenter un obstacle dans certaines situations. (3d émotion, s.d.)

Marqueurs image classique

Figure 4 - Exemple de marqueurs image classique

Source : https://realityxmedia.com/best-augmented-reality-for-events-conferences-business/

Cette méthode consiste à utiliser une image traditionnelle comme marqueur. Il peut s'agir d'une photo, d'une affiche ou d'une couverture de magazine. Ces marqueurs permettent de s'affranchir des codes en noir et blanc et des problèmes d'ergonomie qu'ils peuvent représenter. Cependant, il y a de nombreuses précautions à prendre avec l'utilisation de cette méthode. Les images doivent être facilement reconnaissables, c'est-à-dire qu'elles doivent comporter des variations dans les teintes, avoir un contraste élevé et comporter beaucoup de détails. Le coût des calculs est bien plus important et la puissance nécessaire est plus conséquente. De plus, cette méthode est moins fiable que la précédente et les décrochements sont plus fréquents. (3d émotion, s.d.)

Marqueurs objets

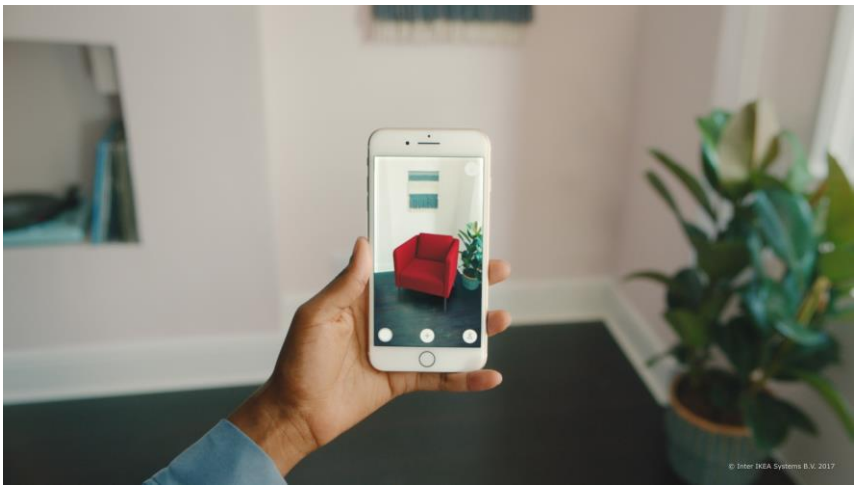
Si les marqueurs prennent le plus souvent la forme de codes ou d'images, ils peuvent aussi être des objets. Les algorithmes qui permettent de les identifier sont en fait l'évolution en 3D des algorithmes pour la 2D. Cette méthode requiert donc de posséder une version numérique en 3D de l'objet pour pouvoir reconnaître l'objet. Cela représente une barrière supplémentaire car elle nécessite de scanner les objets en 3D. Cependant, les progrès dans ce domaine sont nombreux et des algorithmes capables d'enregistrer des objets réels pour les utiliser ensuite comme marqueurs apparaissent. La puissance de calcul nécessaire avec cette méthode est encore plus importante, ce qui a un impact considérable sur le poids final de l'application. (3d émotion, s.d.)

3.2.2. Détection sans marqueurs

Il existe également des applications de réalité augmentée ou de réalité virtuelle qui n'utilisent pas de marqueurs et qui se basent sur d'autres principes pour fonctionner.

Une alternative aux marqueurs consiste à détecter les caractéristiques de l'environement dans lequel on se situe. En effet, le flux vidéo est analysé par des algorithmes et des formules mathématiques qui sont par exemple aujourd'hui en mesure de détecter facilement des surfaces planes telle que des murs ou des sols afin de les modifier ou d'y incruster un objet.

Figure 5 - Exemple de RA sans marqueurs chez Ikea



Source : https://fr.media.ikea.ch/pressrelease/se-meubler-intelligemment-avec-la-realite-augmentee/2862/

Les applications de réalité augmentée ou de réalité virtuelle qui fonctionnent sans marqueurs se basent aussi bien souvent sur d'autres caractéristiques, telles que la localisation ou la position. Elles utilisent d'autres données comme le GPS, la boussole numérique ou l'accéléromètre de l'appareil pour fonctionner. Les applications sont ensuite capables de placer les objets comme s'ils étaient ancrés à des emplacements précis du monde réel. (Unity Documentation, 2018)

Figure 6 - Exemple de RA sans marqueurs basée sur la localisation

Source : https://www.v-cult.com/blog/2017/10/02/la-realite-augmentee-quest-ce-que-cest/

3.3. Technologies existantes

Nous souhaitons maintenant nous intéresser aux librairies et solutions existantes qui pourront être utilisées dans le cadre de ce projet pour implémenter la détection d'objets dans le développement de notre prototype.

Les principales technologies permettant la détection d'objets que nous avons analysées lors de nos recherches sont utilisées dans le cadre de projets de réalité augmentée. Nous partons du constat que les applications de réalité augmentée et de réalité virtuelle se basent sur les mêmes technologies et que les mêmes kit de développement peuvent donc être utilisés.

3.3.1. Vuforia

Figure 7 - Logo Vuforia

Source : https://docs.unity3d.com/2017.2/Documentation/Manual/vuforia-sdk-overview.html

Vuforia ou Vuforia Engine est le kit de développement logiciel le plus utilisé pour le développement en réalité augmentée. Il supporte une grande majorité des téléphones, tablettes et lunettes connectées et permet de créer des expériences de réalité augmentée qui interagissent de manière réaliste avec les objets et l'environnement. Dévoilé pour la première fois en 2011, il existe une grosse communauté autour de ce SDK. (Vuforia, s.d.)

Vuforia utilise la technologie de vision par ordinateur pour reconnaître et suivre des images planes et des objets 3D simples. Le SDK fait correspondre des images prises par une caméra avec une image de référence prédéfinie. Pour accélérer cette comparaison, il se base sur les points caractéristiques spécifiques de l'image (tâches, courbes ou forts contrastes). Il est donc important que l'image de référence possède de nombreux éléments caractéristiques pour que la détection puisse se réaliser. (Bobeshko, 2017)

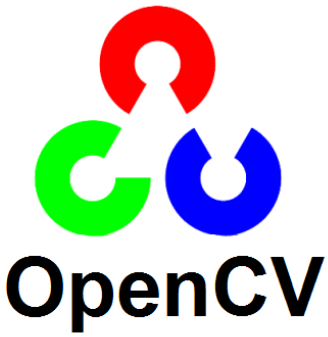
Vuforia va enregistrer les positions relatives de ces points caractéristiques et va chercher à les extraire dans les images qui proviendront de la caméra de l'appareil. Si la majorité des points caractéristiques de référence se retrouvent dans l'image de la caméra, alors l'image est reconnue comme le marqueur. Il est aussi possible de reconnaître des objets réels en 3D avec ce SDK, mais les mêmes contraintes subsistent. Les objets doivent notamment avoir une coloration contrastée afin de pouvoir se baser sur diverses caractéristiques facilement reconnaissables pour la détection. Un objet qui se déforme ne sera pas reconnu par Vuforia. (Bobeshko, 2017)

En conclusion, il s'agit d'un SDK rapide et facile à utiliser mais ses fonctionnalités sont limitées. Il y a aussi une limite sur le nombre d'objets qui peuvent être reconnus simultanément. Les

marqueurs qui peuvent être utilisés doivent respecter des tailles strictes pour être reconnus. S'ils sont partiellement masqués, la détection se fera avec difficulté. (Bobeshko, 2017)

3.3.2. OpenCV

Figure 8 - Logo OpenCV



Source : https://opencv.org/

Open Source Computer Vision est une bibliothèque open-source et multiplateforme principalement destinée à la vision par ordinateur et au traitement d'images en temps réel. Pour la détection, elle se base principalement sur la détection des bords, de cercles et de lignes, le lissage, le flou, la perspective et l'extraction de points caractéristiques. (Bobeshko, 2017)

OpenCV dispose de plugins qui lui confèrent des fonctionnalités supplémentaires, notamment pour le suivi d'objets en temps réel. Ces plugins permettent par exemple la détection d'objet basée sur des marqueurs, la détection de plusieurs objets en simultané, la reconnaissance faciale ou le suivi de la position des mains. (Bobeshko, 2017)

OpenCV a l'avantage considérable d'être très personnalisable. Il est alors possible d'étendre des fonctionnalités existantes ou d'en ajouter de nouvelles en passant par quelques lignes de code. Les algorithmes disponibles pour le traitement des images sont nombreux, y compris pour détecter des images avec peu de points caractéristiques. De plus, il permet de travailler avec des images de toutes les tailles. (Bobeshko, 2017)

Cependant, l'utilisation de OpenCV peut se révéler complexe. Une tâche qui va au-delà de la reconnaissance d'image nécessite l'écriture de code supplémentaire et peut rapidement demander beaucoup de temps de développement.

4. Réalisation d'un prototype

4.1. Choix technologiques

En tenant compte des éléments décrits ci-dessus et de nos recherches sur le sujet, notre choix technologique s'est porté sur la bibliothèque open-source OpenCV.

Bien que Vuforia présente des avantages non-négligeables en ce qui concerne la documentation disponible, la facilité de mise en place et la communauté existante autour du SDK, les limites présentées semblent contraignantes pour la mise en place de notre projet.

OpenCV est une librairie complète et puissante que nous pourrons plus facilement ajuster à nos besoin, bien que cela puisse demander un travail de développement plus important.

Pour la mise en place de notre prototype, nous utiliserons les outils présentés ci-après :

4.1.1. Unity

Figure 9 - Logo Unity

Source : https://unity.com/fr

Unity est un environnement de développement intégré et un moteur de jeu multiplateforme 2D et 3D largement répandu dans l'industrie du jeu vidéo. Cet outil facile d'utilisation met à disposition des développeurs un éditeur visuel intuitif qui permet de déposer les éléments directement dans une scène et de les manipuler selon leur souhait. Unity a l'avantage de proposer une version disponible gratuitement pour les plus débutants et de disposer d'une grande communauté, ce qui facilite sa prise en main. (R., 2021)

Le choix de cet environnement de développement pour la réalisation de ce projet est cohérent dans la mesure où les développements dans Unity s'effectuent en C#, langage que nous maitrisons et que nous aurons plus de facilité à appréhender.

De plus, on retrouve sur internet de nombreuses ressources qui permettent d'accélerer ou de rendre plus confortable le processus de développement avec Unity. Il peut s'agir de modèles d'objets 2D ou 3D, de textures, de matériaux, de scènes, mais aussi d'autres outils permettant d'étendre les fonctionalités de Unity. Dans Unity, ces ressources portent le nom d'*Assets* et elles peuvent se retrouver facilement dans l'*Asset Store Unity*.

4.1.2. OpenCV

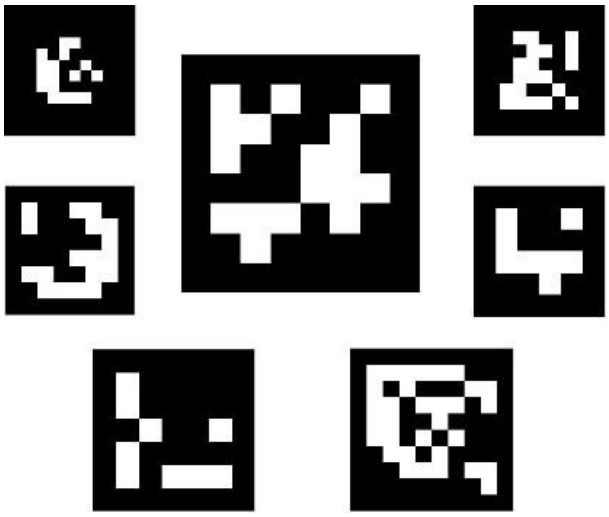
Comme précédemment présenté dans ce rapport, OpenCV est une bibliothèque open-source qui permet la vision par ordinateur et le traitement d'images en temps réel. Dans le cadre de notre projet, nous l'utiliserons pour détecter les objets réels.

4.1.3. ArUco Markers

Comme précédemment présenté, la détection d'objets en réalité augmentée ou en réalité virtuelle est un processus basé sur la correspondance entre des caractéristiques de l'environnement réel et une image 2D ou un objet 3D. Les marqueurs permettent de faciliter ce processus car ils sont plus faciles à détecter et ils ont l'avantage de fournir des informations précises permettant d'obtenir plus facilement la pose de la caméra. (OpenCV, s.d.)

Nous utiliserons le module ArUco disponible pour OpenCV qui nous permettra de générer des marqueurs et de les détecter. Un marqueur ArUco est un carré composé d'une large bordure noire et d'une matrice binaire à l'intérieur qui permet de déterminer son identifiant. La taille du marqueur définit la taille de la matrice interne. De plus, chaque coin doit donc pouvoir être identifié de manière précise car il doit être possible de détecter un marqueur peu importe la position dans laquelle il se trouve. (OpenCV, s.d.)

Figure 10 - Marqueurs ArUco



Source : https://docs.opencv.org/4.x/d5/dae/tutorial_ArUco_detection.html

Tous les marqueurs qui peuvent être détectés dans l'application sont enregistrés dans ce qu'on appelle un dictionnaire de marqueurs. Le dictionnaire est défini par une taille qui correspond au nombre de marqueurs qui le compose. (OpenCV, s.d.)

Pour intégrer l'utilisation des marqueurs ArUco, nous devons faire appel à un wrapper qui nous permettra d'utiliser les fonctionnalités du module ArUco dans Unity. Plusieurs solutions sont disponibles sur le marché et nous devons sélectionner celle qui correspond le mieux à nos besoins. Dans un premier temps, nous cherchons à identifier, en se basant sur une liste de critères, la solution qui semble le mieux convenir à nos besoins.

Le tableau ci-dessous présente une analyse des quatre solutions que nous pouvons imaginer utiliser dans le cadre de notre projet. L'analyse se base sur les informations disponibles sur internet ainsi que sur les essais que nous avons pu effectuer. Les critères auxquels nous avons apporté le plus d'importance sont les suivants : la compatibilité avec Unity, la possiblité de détecter des marqueurs ArUco, la facilité à mettre en place la solution et la documentation disponible sur internet.

Tableau 1 - Comparatif des solutions existantes pour l'utilisation de ArUco et OpenCV

Critères	Pond.	ArucoUnity by NormandErwan		OpenCV for Unity by Enox Software		OpenCV plus Unity by Paper Plane Tools		AR Marker Detector by Paper Plane Tools	
Installation et mise en place									
Facilité de téléchargement	3	3	**9**	3	**9**	2	**6**	2	**6**
Facilité de mise en place	4	3	**12**	3	**12**	2	**8**	1	**4**
* Compatibilité avec Unity	4	2	**8**	3	**12**	2	**8**	1	**4**
Date de parution	2	1	**2**	3	**6**	2	**4**	1	**2**
Produit									
Prix	3	3	**9**	0	**0**	3	**9**	3	**9**
* Détection des marqueurs ArUco	4	3	**12**	3	**12**	3	**12**	2	**8**
Fonctionnalités disponibles	1	2	**2**	2	**2**	2	**2**	1	**1**
Prérequis technologiques	3	2	**6**	3	**9**	2	**6**	1	**3**
Documentation									
Documentation disponible	4	3	**12**	2	**8**	1	**4**	1	**4**
Notoriété et communauté	3	3	**9**	3	**9**	1	**3**	0	**0**
TOTAL			**81**		**79**		**62**		**41**

* = critère éliminatoire

0 = insuffisant - 1 = moyen - 2 = satisfaisant - 3 = très satisfaisant - SO = sans objet

Source : Données de l'auteur

Cette analyse montre que la solution qui semble le mieux correspondre à nos besoins est la solution « ArUcoUnity » proposée par NormandErwan. Cette solution clé en main a l'avantage de fournir tous les outils nécessaires à l'utilisation du module ArUco dans Unity. La mise en place est facilitée grâce à un tutoriel qui présente les différentes étapes de la configuration. La documentation disponible est complète et permet de configurer facilement les objets dans Unity grâce aux scripts fournis.

Il convient toutefois de souligner que la solution « OpenCV for Unity » de Enox Software obtient un nombre de points très proche au terme de notre analyse. Son désavantage principal réside dans le fait qu'il s'agit d'un *Asset* payant, ce qui n'est pas le cas des autres solutions analysées. Sa date de parution et sa compatibilité avec Unity obtiennent toutefois une meilleure note.

Nous prenons la décision de débuter le travail de développement avec la solution « ArUcoUnity », en gardant cependant en tête que « OpenCV for Unity » reste une solution intéressante.

4.2. Gestion de projet

La méthodologie Scrum sera utilisée pour la mise en place du prototype. Selon la définition, cette méthodologie permet « d'appréhender des problèmes complexes en fournissant des produits de la plus haute valeur possible » (Scrum, s.d.). Dans le cadre de ce projet, nous fonctionnerons avec des phases de développement de trois semaines, appelées des Sprints. Les étapes de développement et les fonctionnalités demandées seront détaillées dans un Product Backlog qui permettra d'assurer l'organisation et le suivi du travail de développement.

4.2.1. Product Backlog

Un Product Backlog a été mis en place pour la réalisation de ce projet (annexe IV). Il comprend une dizaine de user stories qui correspondent chacune à une fonctionnalité attendue dans le prototype de notre application. Le travail de développement, pour un total de 44 story points, est réparti sur cinq Sprints.

4.3. Phases de développement

Dans les pages qui suivent nous détaillons, par Sprint, le processus et les étapes de développement de notre prototype.

4.3.1. Sprint 0 – Mise en place

Le Sprint 0 était un Sprint de préparation. L'objectif de ce Sprint initial était de mettre en place un projet test afin de configurer l'environnement de développement et de comprendre la structure d'un projet Unity et son fonctionnement. La première étape consistait à installer le logiciel Unity. Nous avons ensuite intégré le module ArUco à cet environnement un installant l'*Asset* « ArUcoUnity ».

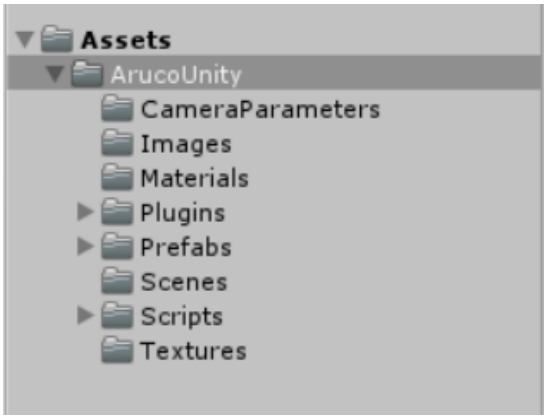
La documentation du package « ArUcoUnity » recommande d'installer la dernière version stable de l'*Asset*. Dans notre cas, il s'agit de la version 1.2.0 qui fonctionne avec les versions 2017.x et 5.x de Unity. Nous avons fait le choix d'installer la version 2017.2 de Unity qui est celle avec laquelle la solution a été testée. La documentation disponible se base également sur cette version. (NormandErwan, 2019)

Figure 11 - Capture d'écran de Unity 2017.2

Source : Données de l'auteur

Nous avons ensuite téléchargé la version 1.2.0 de « ArUcoUnity », puis créé un nouveau projet dans Unity à l'intérieur duquel nous avons importé le package téléchargé.

Figure 12 - Capture d'écran du package « ArUcoUnity »



Source : Données de l'auteur

Nous avons également profité de ce premier Sprint de mise en place pour nous familiariser avec Unity, la réalité augmentée et la réalité virtuelle qui sont des technologies et des concepts qui nous étaient encore inconnus.

4.3.2. Sprint 1 – Création de marqueurs ArUco

Le but de ce premier Sprint était de se familiariser avec l'utilisation de OpenCV et de ArUco à travers Unity. Nous avions comme objectifs principaux de créer des marqueurs ArUco et de réussir à les reconnaître dans la réalité augmentée.

Les user stories ci-dessous sont extraites du Product Backlog et sont concernées par ce Sprint. La user story n°2 sera utilisée comme référence ce qui signifie que sa valeur en story point sera de 1 et qu'elle servira à estimer le poids des user stories suivantes.

Figure 13 - Extrait du Product Backlog pour le Sprint 1

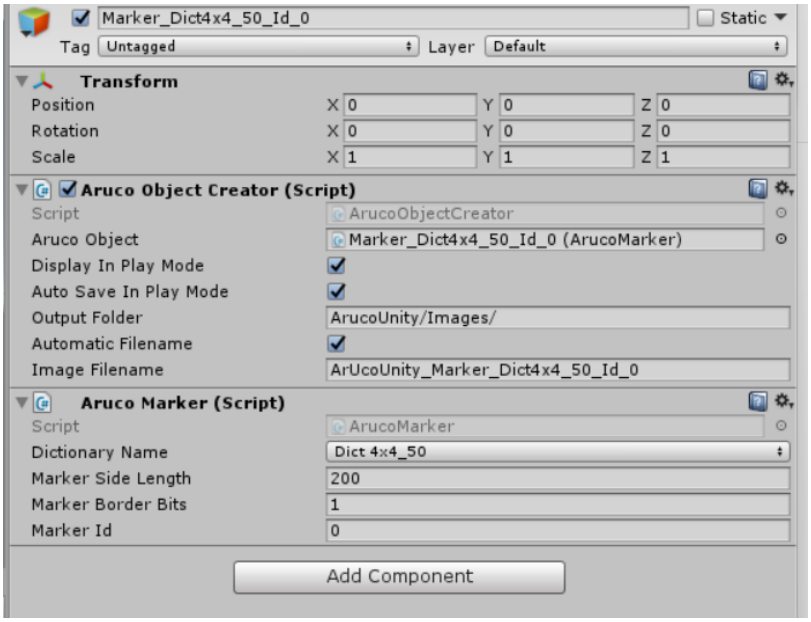
US Nr	En tant que ...	Je veux ...	car/pour ...	Acceptance Criteria	Priorit	Size	Sprint	MoSC
2	Développeur	créer des markers	pouvoir les imprimer, les utiliser et les reconnaître	- Marker AruCo	900	1	1	MUST
3	Utilisateur	décoder un marker en réalité augmentée	reconnaître le marker scanné et l'identifier	- id du marker	850	5	1	MUST

Source : Données de l'auteur

Pour utiliser des marqueurs ArUco, nous avons dû commencer par les créer et les imprimer. Le package « ArUcoUnity » fournit des scripts permettant de créer facilement différents styles de marqueurs (Grid board, ChArUco board ou Diamond marker).

En se basant sur la documentation disponible, nous avons créé un objet vide dans la scène. Nous avons ensuite ajouté les scripts *ArUcoMarker* et *ArUcoObjectCreator* qui nous ont permis de générer un marqueur, de l'enregistrer puis de l'imprimer. Nous avons répété cette opération pour en générer plusieurs. Dans l'exemple ci-dessous, nous avons généré un marqueur de 4 bits sur 4 avec l'identifiant 0.

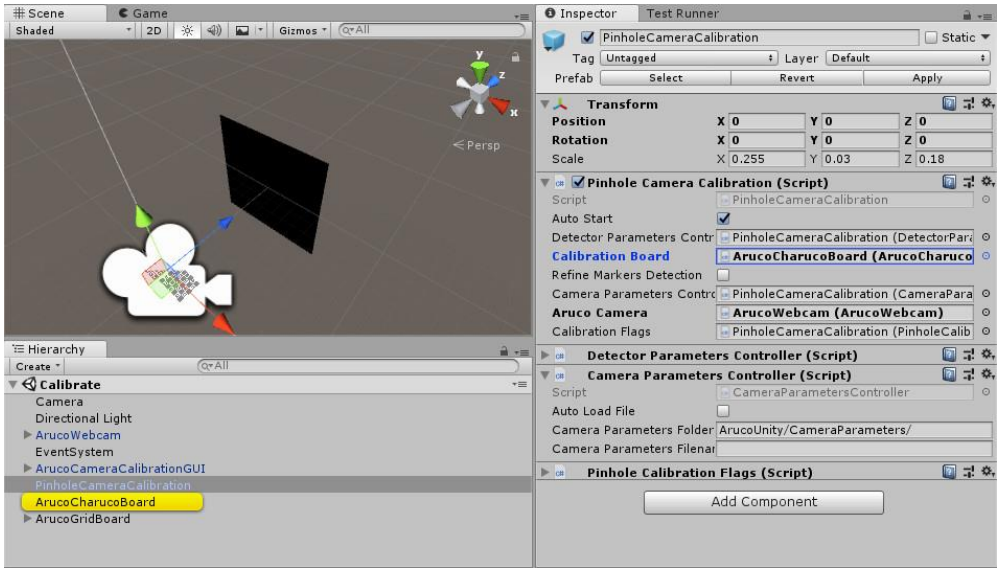
Figure 14 - Capture d'écran de la configuration d'un marqueur



Source : Données de l'auteur

L'étape suivante consistait à calibrer la caméra. Pour cela, une scène *CalibrateCamera* était fournie avec le package. Pour l'utiliser, nous avons d'abord dû créé un GridBoard de marqueurs et le déplacer devant la webcam afin de la calibrer correctement. Un fichier XML de paramètres de calibration s'est automatiquement enregistré.

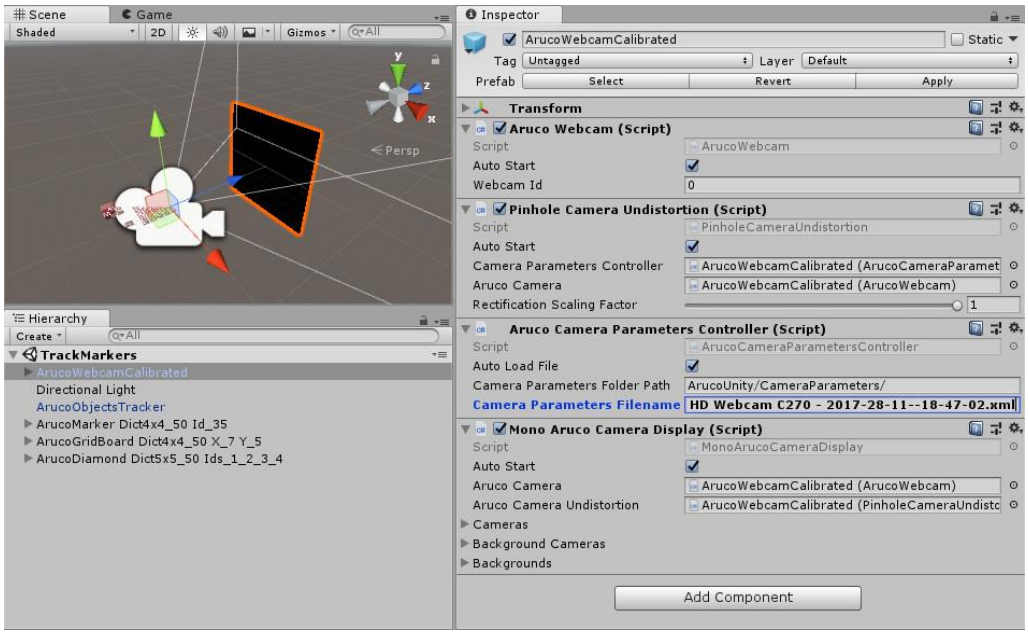
Figure 15 - Capture d'écran de la calibration de la caméra



Source : Données de l'auteur

Nous avons ensuite pu traquer les marqueurs précédemment créés grâce à la scène *TrackMarker* qui était également disponible dans le package. Il a fallu fournir le fichier XML de calibration comme paramètre de la caméra pour que la reconnaissance se fasse correctement. Cette étape a permis de vérifier que tout fonctionnait bien.

Figure 16 - Capture d'écran de la configuration de la détection de marqueurs



Source : Données de l'auteur

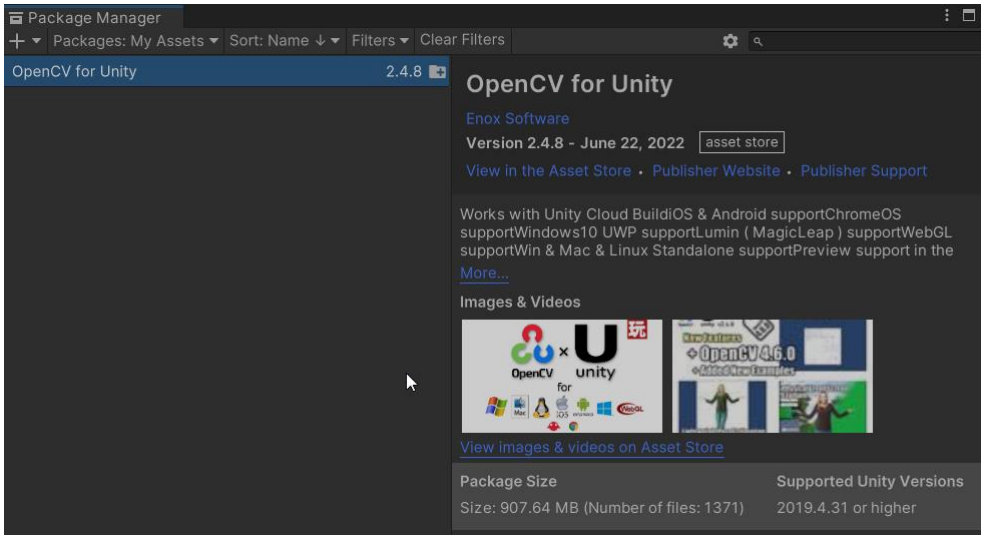
Bien que nous ayons réussi à reproduire l'environnement de démonstration, à générer des marqueurs et à les reconnaître, nous avons remarqué que la version 2017.2 de Unity présentait de grandes différences avec les versions plus récentes du logiciel. Les solutions présentées sur internet se basaient souvent sur des versions plus actuelles de Unity et ne permettaient pas de résoudre les problèmes rencontrés avec notre version du logiciel.

Nous avons estimé que l'utilisation d'une ancienne version de Unity pouvait représenter un potentiel problème pour la mise en place de notre projet, en particulier pour le passage de notre projet en réalité virtuelle. Pour cette raison, nous avons pris la décision de nous tourner vers la solution « OpenCV for Unity » présentée au point 4.1.3.

Nous avons donc décidé d'utiliser la version 2021.3.5 de Unity. L'*Asset* a été acheté depuis l'*Asset Store Unity* et l'importation dans un nouveau projet s'est faite depuis le *Package Manager*

du projet Unity. Nous avons commencé par utiliser les exemples fournis dans le package pour comprendre le fonctionnement et la logique de ce nouvel *Asset*.

Figure 17 - Capture d'écran du Package Manager de Unity



Source : Données de l'auteur

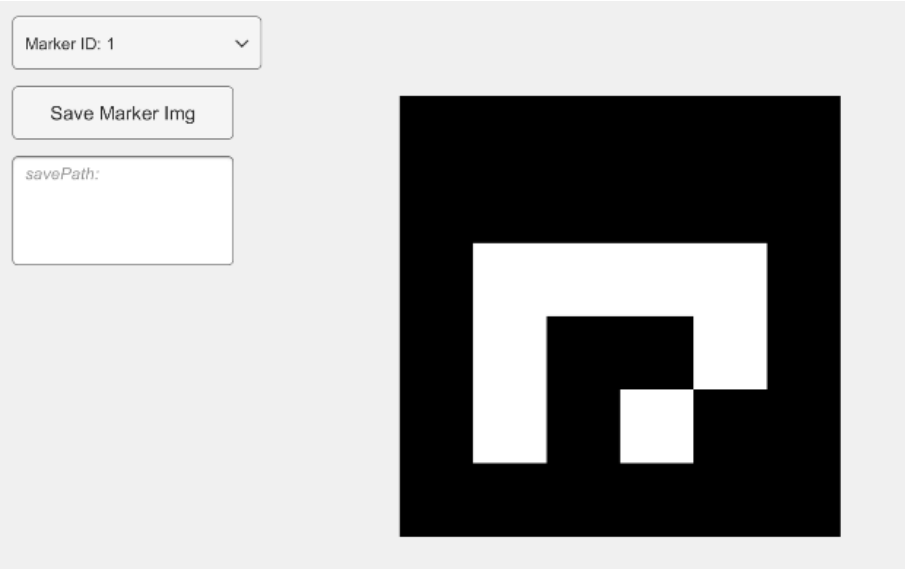
Création de marqueurs

La scène d'exemple *ArUcoCreateMarkerExample* permet la création de marqueurs. Nous avons décidé de la simplifier légèrement et de la réutiliser pour la création des marqueurs dans notre projet. Cela nous a permis de valider la user story n°2 de notre Product Backlog.

C'est principalement la méthode ci-dessous de ArUco qui est appelée et qui permet la création du marqueur. L'utilisateur peut ensuite enregistrer l'image sur son ordinateur pour l'imprimer.

```
ArUco.drawMarker(dictionary, (int)markerId, markerSize, markerImg, borderBits);
```

Figure 18 - Capture d'écran de l'écran de création d'un marqueur



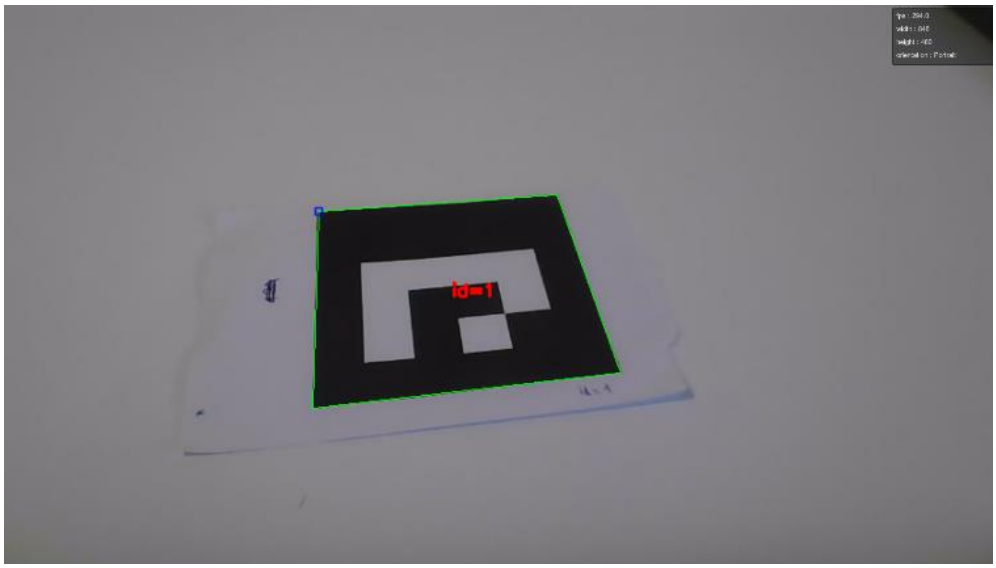
Source : Données de l'auteur

Détection et identification de marqueurs

Pour la détection des marqueurs et afin de répondre à la user story n°3, nous nous sommes basés sur la scène *ArUcoWebCamTextureExample* et nous l'avons adapté pour la simplifier. Dès qu'un marqueur est reconnu, ses bordures et son identifiant s'affichent. Nous avons également affiché l'identifiant du marqueur détecté dans les logs. La reconnaissance des marqueurs dans le flux de la caméra s'effectue dans la méthode *Update()* qui est appelée à chaque frame grâce notamment à l'appel des méthodes suivantes :

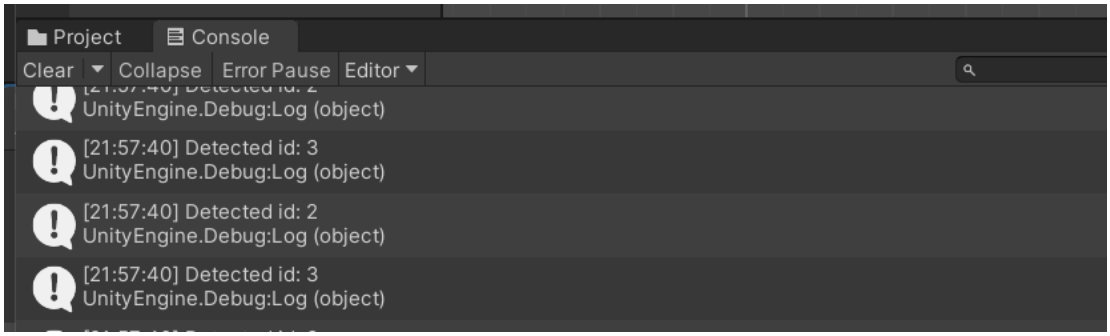
```
ArUco.detectMarkers(rgbMat, diction Ary, corners, ids, detectorParams, rejectedCorners);
ArUco.drawDetectedMarkers(rgbMat, corners, ids, new Scalar(0, 255, 0));
```

Figure 19 - Capture d'écran de la détection d'un marqueur avec l'identifiant n°1



Source : Données de l'auteur

Figure 20 - Capture d'écran de la console affichant l'identifiant du marqueur détecté



Source : Données de l'auteur

Bilan

Au terme de ce premier Sprint, nous sommes capables de créer des marqueurs pour les enregistrer en tant qu'image imprimable et d'identifier un marqueur scanné en réalité augmentée. Nous avons donc valider les user stories n°2 et 3 de notre Product Backlog pour un score total de 6 en termes de story points.

Ce Sprint nous a également permis de valider le choix final de l'*Asset* utilisé pour intégrer l'utilisation de OpenCV dans Unity. En effet, nous avons estimé qu'il était préférable d'utiliser une version plus récente de Unity pour éviter de futurs problèmes lors du développement. Nous avons donc accordé plus d'importance qu'initialement prévu au critère relatif à la date de parution et à la compatibilité avec une version actuelle de Unity.

4.3.3. Sprint 2 – Obtenir la position et l'orientation d'un marqueur

L'objectif principal de ce deuxième Sprint était de créer un objet 3D qui puisse s'afficher sur le marqueur détecté. Pour pourvoir placer notre objet, nous avions besoin de connaître la position et l'orientation d'un marqueur par rapport à la caméra.

Les user stories n°4, 5 et 6 de notre Product Backlog sont concernées par ce Sprint. Le travail pour ce Sprint est estimé à 9 story points.

Figure 21 - Extrait du Product Backlog pour le Sprint 2

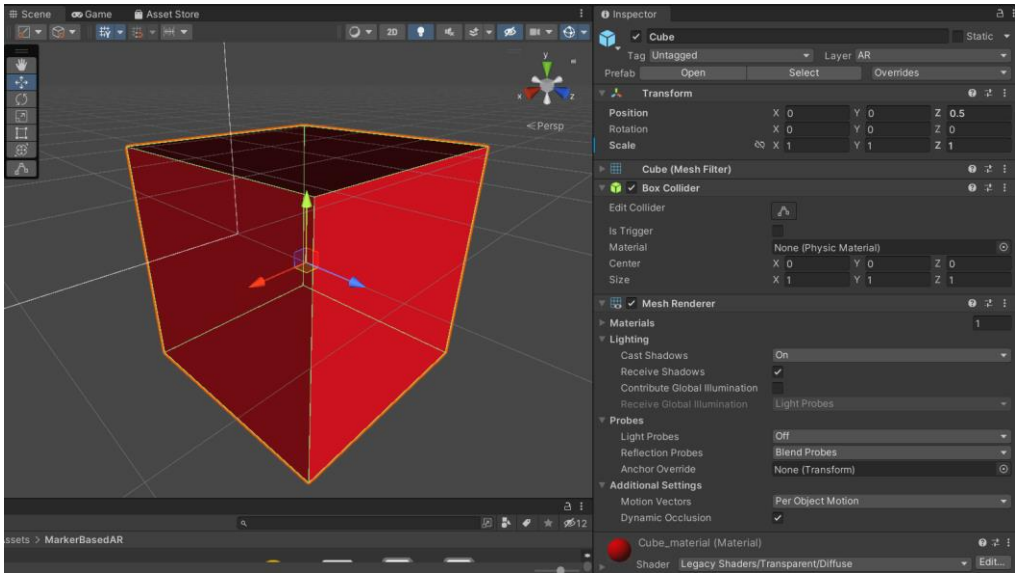
US Nr	En tant que ...	Je veux ...	car/pour ...	Acceptance Criteria	Priorit	Size	Sprint	MoSc
4	Développeur	créer un objet en réalité augmentée associé au marker	le visualiser/représenter dans la réalité augmentée	- objet qui s'affiche	840	3	2	MUST
5	Utilisateur	connaitre la position d'un marker par rapport à la caméra	pouvoir déplacer un objet virtuel selon la pose du marker	- position correcte de l'objet - déplacement en temps réel possible	830	3	2	MUST
6	Utilisateur	connaitre l'orientation d'un marker par rapport à la caméra	pouvoir déplacer un objet virtuel selon la pose du marker	- orientation correcte de l'objet - rotation en temps réel possible	820	3	2	MUST

Source : Données de l'auteur

Création d'un objet 3D

Nous avons commencé par créer un nouvel objet 3D que nous avons ensuite placé dans notre scène en fonction de la position des marqueurs détectés. A partir des objets 3D disponibles dans Unity, nous avons créé un cube auquel nous avons attribué une nouvelle matière de couleur rouge. Ainsi, nous avons obtenu un cube rouge.

Figure 22 - Capture d'écran de la configuration d'un objet 3D



Source : Données de l'auteur

Connaître la position et l'orientation d'un marqueur

Nous nous sommes ensuite intéressés au code de l'*Asset* « OpenCV for Unity » pour comprendre comment la position et l'orientation d'un marqueur présenté à notre webcam étaient récupérées. Pour cela, il a fallu se plonger dans le code pour comprendre les différents appels de méthodes et détecter celui qui permettait de récupérer ces informations.

Nos recherches sur le sujet nous ont permis de savoir ce que nous devions rechercher. OpenCV et ArUco utilisent un modèle de caméra sténopé. Le point d'origine est définit comme le centre de projection du modèle de caméra et toutes les coordonnées ArUco sont définies dans ce système de coordonnées. En résumé, les informations retournées par ArUco correspondent en fait à la position des marqueurs par rapport à la caméra.

ArUco nous permet d'identifier les marqueurs dans notre image et de retourner leur position et leur orientation dans le système de coordonnées 3D définit par notre caméra. Ces informations sont contenues dans les matrices suivantes :

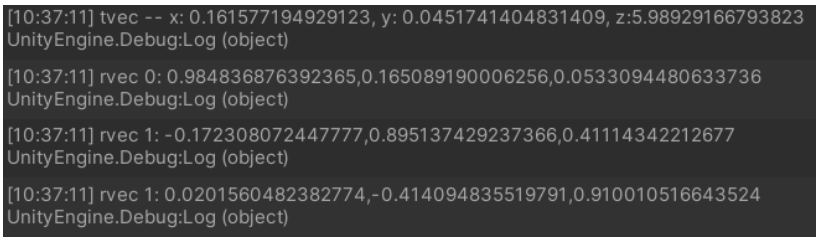
- ***tvecs*** qui contient les vecteurs de translation, soit les coordonnées x, y et z des marqueurs par rapport à l'origine,
- et ***rvecs*** qui contient les vecteurs de rotation, qui définit un axe de rotation et l'angle de rotation autour de cet axe.

Ces informations peuvent être obtenues facilement grâce à l'appel de la méthode ci-dessous. En plus de détecter les marqueurs présents sur l'image, ArUco va retourner leur position et leur orientation dans les matrices de vecteurs **tvecs** et **rvecs**.

```
ArUco.estimatePoseSingleMarkers(corners, markerLength, camMatrix, distCoeffs, rvecs,
                                tvecs);
```

Ci-dessous, un exemple de ce que ArUco est capable de retourner lorsqu'on lui présente un marqueur. Ces informations correspondent à la position et à l'orientation du marqueur par rapport à la caméra.

Figure 23 - Capture d'écran de la console affichant la position et l'orientation du marqueur



Source : Données de l'auteur

Déplacement de l'objet

Les matrices sont utilisées pour déplacer les objets dans notre scène. Pour cela, elles doivent être transformées en une matrice de transformation qui sera appliquée sur notre objet. Lors de la détection d'un marqueur, le code ci-dessous définit la matrice de transformation enregistrée dans le marqueur m :

Figure 24 - Capture d'écran du code permettant de définir les matrices de transformation

```
// Marker transformation with regards to the camera
public Matrix4x4 transformation;
```

```
m.transformation.SetRow(0, new Vector4((float)rotMat.get(0, 0)[0], (float)rotMat.get(0, 1)[0], (float)rotMat.get(0, 2)[0], (float)Tvec.get(0, 0)[0]));
m.transformation.SetRow(1, new Vector4((float)rotMat.get(1, 0)[0], (float)rotMat.get(1, 1)[0], (float)rotMat.get(1, 2)[0], (float)Tvec.get(1, 0)[0]));
m.transformation.SetRow(2, new Vector4((float)rotMat.get(2, 0)[0], (float)rotMat.get(2, 1)[0], (float)rotMat.get(2, 2)[0], (float)Tvec.get(2, 0)[0]));
m.transformation.SetRow(3, new Vector4(0, 0, 0, 1));
```

Source : Données de l'auteur

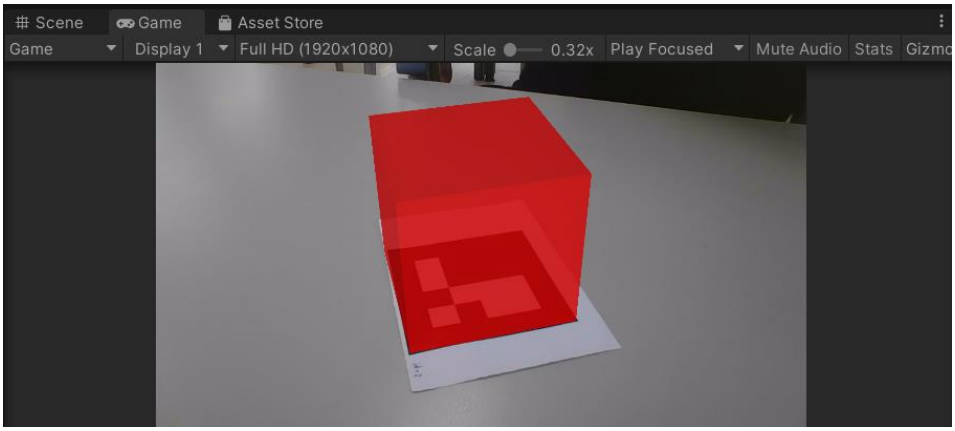
Dans la méthode *Update()* appelée à chaque nouvelle frame, la matrice de transformation du marqueur est récupérée. La formule mathématique permet de rendre compatible le sytème de coordonnées avec Unity en inversant l'axe Y. La méthode *ARUtils.SetTransformFromMatrix()* applique cette transformation sur l'objet 3D ce qui permet de le positionner à l'emplacement du marqueur.

Figure 25 - Capture d'écran du code permettant d'appliquer la matrice de transformation

```
Matrix4x4 transformationM = marker.transformation;

// right-handed coordinates system (OpenCV) to left-handed one (Unity)
// https://stackoverflow.com/questions/30234945/change-handedness-of-a-row-major-4x4-transformation-matr
Matrix4x4 ARM = invertYM * transformationM * invertYM;

// Apply Y-axis and Z-axis refletion matrix. (Adjust the posture of the AR object)
ARM = ARM * invertYM * invertZM;

ARM = ARCamera.transform.localToWorldMatrix * ARM;

Debug.Log("ARM: " + ARM.ToString());

GameObject ARGameObject = settings.getARGameObject();
if (ARGameObject != null)
{
    ARUtils.SetTransformFromMatrix(ARGameObject.transform, ref ARM);
```

Source : Données de l'auteur

Figure 26 - Capture d'écran du placement d'un objet 3D sur le marqueur détecté



Source : Données de l'auteur

Bilan

Au terme de ce deuxième Sprint, nous sommes capables de récupérer la position et l'orientation d'un marqueur par rapport à notre caméra, grâce à l'utilisation de OpenCV et de ArUco. Nous avons également créé un objet 3D sur lequel nous avons appliqué notre matrice de transformation, ce qui nous a permis de déplacer notre objet dans la scène en fonction de la position de notre marqueur.

Nous avons donc validé les user stories n° 4, 5 et 6 de notre Product Backlog qui correspondent à un effort total de 9 story points.

4.3.4. Sprint 3 – Détection simultanée de plusieurs marqueurs

Dans ce troisième Sprint, notre objectif principal était dans un premier temps de rendre notre solution capable de détecter simultanément plusieurs marqueurs sur la même image. Dans un deuxième temps, nous souhaitions créer de nouveaux objets associés à nos différents marqueurs. Nous souhaitions modifier l'aspect visuel de nos objets pour les différencier et les déplacer dans la scène en fonction de chaque marqueur.

Les user stories n°7 et 8 du Product Backlog sont concernées par ce Sprint pour un poids estimé à 8 story points.

Figure 27 - Extrait du Product Backlog pour le Sprint 3

US Nr	En tant que ...	Je veux ...	car/pour ...	Acceptance Criteria	Priorit	Size	Sprint	MoSC
7	Utilisateur	détecter plusieurs objets en même temps	les visualiser ensemble dans la réalité augmentée	- afficher au moins 2 objets en même temps - respect des dimensions et espacements	800	5	3	MUST
8	Utilisateur	modifier les aspects visuels des objets détectés	les différencier dans la réalité augmentée	- aspects différents des objets	750	3	3	MUST

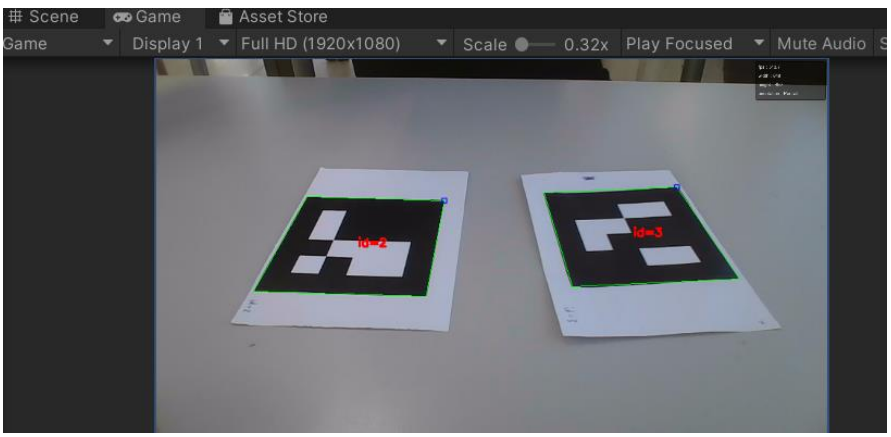
Source : Données de l'auteur

Détection simultanée de plusieurs marqueurs

Dans le cadre du projet, nous sommes amenés à détecter plusieurs objets simultanément. Nous avions donc le besoin de les détecter, mais également de les identifier afin de pouvoir différencier l'objet 3D lié au marqueur.

La méthode *ArUco.detectMarkers()* utilisée au Sprint 1 détectent plusieurs marqueurs simultanément. La méthode *drawMarker()* permet de les afficher :

Figure 28 - Capture d'écran de la détection simultanée de deux marqueurs



Source : Données de l'auteur

Les identifiants des marqueurs détectés sont contenus dans une matrice *ids*. Il est possible d'accéder au nombre total des marqueurs détectés grâce à la méthode *ids.total()* et nous pouvons parcourir la matrice pour trouver les identifiants détectés.

Modification de l'aspect visuel d'un objet en fonction du marqueur détecté

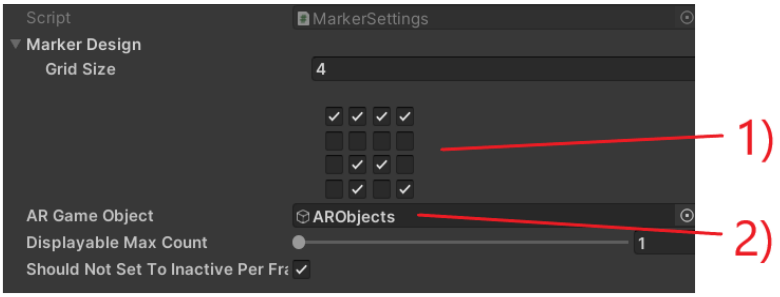
Notre difficulté réside dans le fait que nous souhaitons différencier l'objet qui sera affecté par la matrice de transformation de chaque marqueur. Chaque objet doit être associé à un marqueur et être déplacé uniquement s'il s'agit du marqueur auquel il est associé.

Nos premiers essais n'ont pas été très concluants et nous avons eu des difficultés à naviguer dans la matrice pour effectuer un test et déplacer uniquement l'objet associé. Lors de nos premiers essais, si l'ordre dans lequel les marqueurs étaient détectés et enregistrés dans la matrice subissait un changement entre deux appels de la méthode *Update()*, alors les objets ne se plaçaient plus sur le bon marqueur jusqu'à ce que l'ordre soit une nouvelle fois inversé.

Lors de nos recherches sur internet, nous avons pu nous appuyer sur l'*Asset* « MarkerBased AR Example » proposé par le même développeur que l'*Asset* « OpenCV for Unity » et qui présente une utilisation similaire à nos besoins. Nous avons installé la version 1.2.4 de juin 2022 pour nous intéresser à son fonctionnement et nous avons gardé ce qui était utile pour notre projet.

La configuration des marqueurs détectables s'effectue maintenant depuis la scène grâce à un objet vide et à l'utilisation du script *MarkerSettings* qui permet de définir le marqueur à reconnaître (1) et un groupe d'objet qui lui est associé (2). Nous créons un objet par marqueur à reconnaître.

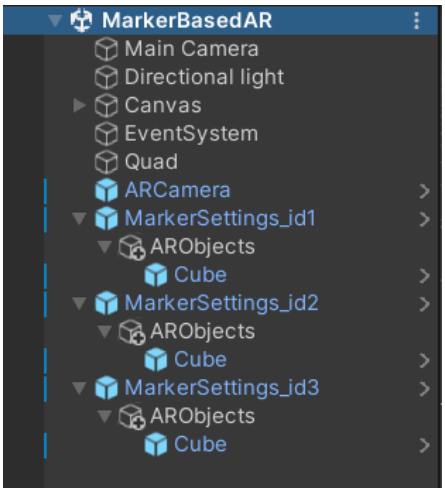
Figure 29 - Capture d'écran de la configuration d'un marqueur à détecter dans la scène



Source : Données de l'auteur

Nous avons ensuite configuré trois objets 3D différents. Nous leur avons attribué des textures différentes pour que les couleurs ne soient pas les mêmes et qu'on puisse visuellement les distinguer. Nous les avons ensuite associé chacun à un marqueur.

Figure 30 - Capture d'écran de la structure de la scène et ses objets

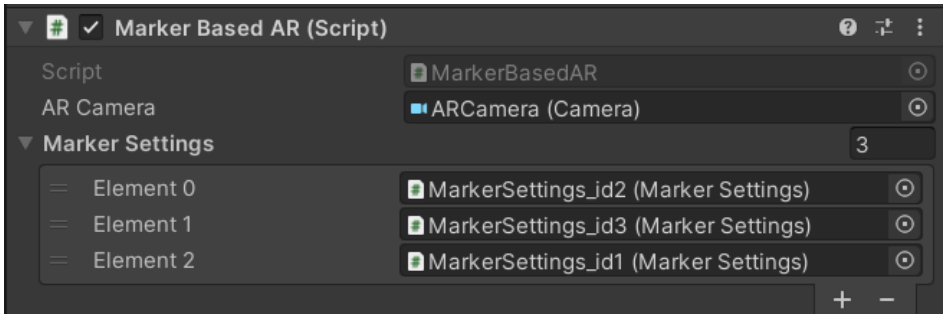


Source : Données de l'auteur

Le reste de la configuration s'effectue grâce aux scripts *MarkerBasedAR* et *WebCamTextureToMatHelper* qui permettent d'afficher notre caméra et de visualiser les marqueurs dans l'objet *Quad*.

Nous avons ajouté les objets configurés pour chaque marqueur dans les paramètres du script *MarkerSettings*. Ainsi, les objets peuvent être repris et utilisés dans le code pour être déplacés.

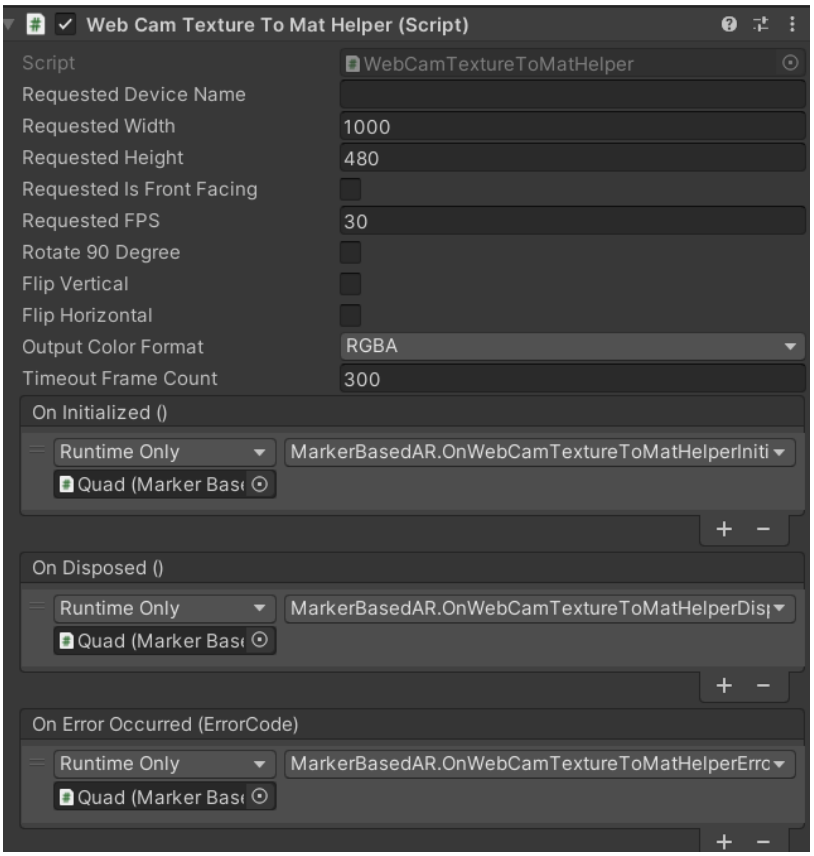
Figure 31 - Capture d'écran de la configuration du premier script dans l'objet Quad



Source : Données de l'auteur

Le script *WebCamTextureToMatHelper* permet simplement de définir la taille de l'image captée par la caméra et de définir quelles actions doivent être exécutées au lancement, à l'arrêt ou lors d'une erreur.

Figure 32 - Capture d'écran de la configuration du second script dans l'objet Quad



Source : Données de l'auteur

Au niveau du code, les marqueurs configurés sont récupérés dans un tableau d'objets *markerSettings*. Pour chaque marqueur trouvé dans l'image, le marqueur est comparé avec les marqueurs enregistrés dans la liste. Si une correspondance est trouvée entre les deux marqueurs, alors l'objet est déplacé grâce à la matrice de transformation.

Figure 33 - Capture d'écran du code déplaçant l'objet en fonction de son marqueur

```
List<Marker> findMarkers = markerDetector.getFindMarkers();
for (int i = 0; i < findMarkers.Count; i++)
{
    Marker findMarker = findMarkers[i];

    foreach (MarkerSettings settings in markerSettings)
    {
        if (findMarker.id == settings.getMarkerId())
        {
            Matrix4x4 transformationM = findMarker.transformation;

            // right-handed coordinates system (OpenCV) to left-handed one (Unity)
            // https://stackoverflow.com/questions/30234945/change-handedness-of-a-row-major-4x4-transformation-matrix
            Matrix4x4 ARM = invertYM * transformationM * invertYM;

            // Apply Y-axis and Z-axis refletion matrix. (Adjust the posture of the AR object)
            ARM = ARM * invertYM * invertZM;

            ARM = ARCamera.transform.localToWorldMatrix * ARM;

            Debug.Log("ARM: " + ARM.ToString());

            GameObject ARGameObject = settings.getARGameObject();
            if (ARGameObject != null)
            {
                ARUtils.SetTransformFromMatrix(ARGameObject.transform, ref ARM);
```

Source : Données de l'auteur

Comme présenté dans l'exemple ci-dessous, lorsque les trois marqueurs sont présentés devant la caméra, les trois cubes s'affichent. Si les marqueurs sont déplacés alors les objets suivent leur marqueur.

Figure 34 - Capture d'écran des objets 3D placés selon leur marqueur respectif

Source : Données de l'auteur

Bilan

Au terme de ce troisième Sprint, nous sommes capables de détecter plusieurs marqueurs ArUco en simultané et d'afficher leurs identifiants. Après plusieurs recherches et divers essais, nous avons également été capables de lier des objets visuellement différents à un marqueur spécifique et d'afficher et déplacer l'objet uniquement en fonction de la position du marqueur.

Nous avons donc validé les user stories n° 7 et 8 de notre Product Backlog qui correspondent à un effort total de 8 story points.

4.3.5. Sprint 4 - Détection des marqueurs dans une scène de réalité virtuelle

L'objectif principal de ce Sprint était de rendre compatible notre projet actuel avec la réalité virtuelle. Jusqu'au Sprint 3, nous avons travaillé sur un projet de réalité augmentée car nous visualisions notre caméra en temps réel et les objets 3D venaient s'ajouter sur notre image. A l'inverse, en réalité virtuelle, la caméra ne doit plus s'afficher. En d'autres termes, nous souhaitons remplacer l'image de la caméra par une scène virtuelle en 3D.

Les user stories n° 9 et 10 sont concernées par ce Sprint pour un effort estimé à 8 story points.

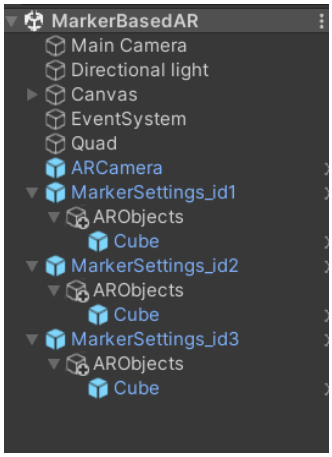
Figure 35 - Extrait du Product Backlog pour le Sprint 4

US Nr	En tant que ...	Je veux ...	car/pour ...	Acceptance Criteria	Priority	Size	Sprint	MoSC
9	Développeur	créer une scène de réalité virtuelle	transformer mon projet AR en VR	- scène de VR	720	3	4	MUST
10	Développeur	adapter la configuration de ma scène	pouvoir reconnaître, afficher et déplacer mes objets 3D dans la scène en fonction de mes marqueurs	- objets qui s'affichent et se déplacent dans la scène	700	5	4	MUST

Source : Données de l'auteur

En réalité augmentée, nous avions une *ARCamera* qui permettait de gérer l'utilisation de notre webcam. Cet objet était utilisé dans l'objet *Quad* pour récupérer l'image et l'afficher. La *MainCamera* servait simplement à afficher le *Quad*.

Figure 36 - Capture d'écran de la structure des objets de la scène en AR



Source : Données de l'auteur

Pour transformer le projet, nous avons donc commencé par créer une nouvelle scène de réalité virtuelle dans notre projet. Nous avons repris les éléments de configuration de nos marqueurs et les objets 3D associés à ces marqueurs.

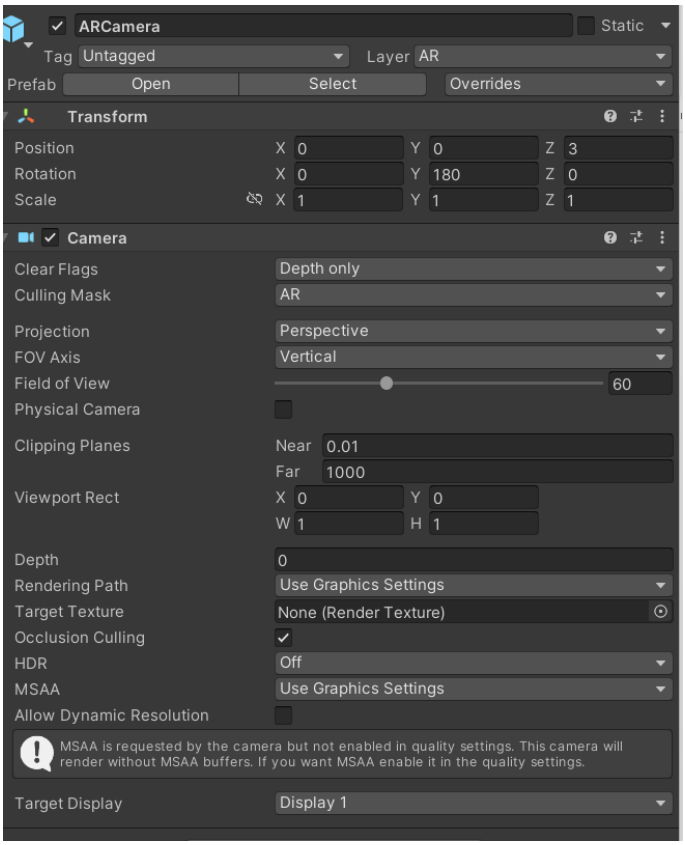
Figure 37 - Capture d'écran de la structure des objets de la scène en VR

Source : Données de l'auteur

Dans le projet de réalité augmentée, l'image était capturée par une *ARCamera*, puis transformée en matrice grâce au script *WebCamTextureToMatHelper* afin d'effectuer ensuite la détection des marqueurs et afficher l'image dans le *Quad*.

Dans notre projet de réalité virtuelle, nous souhaitons continuer à utiliser notre webcam pour capturer l'image et c'est pour cette raison que nous avons ajouté une *ARCamera* à notre scène.

Figure 38 - Capture d'écran de la configuration de la caméra

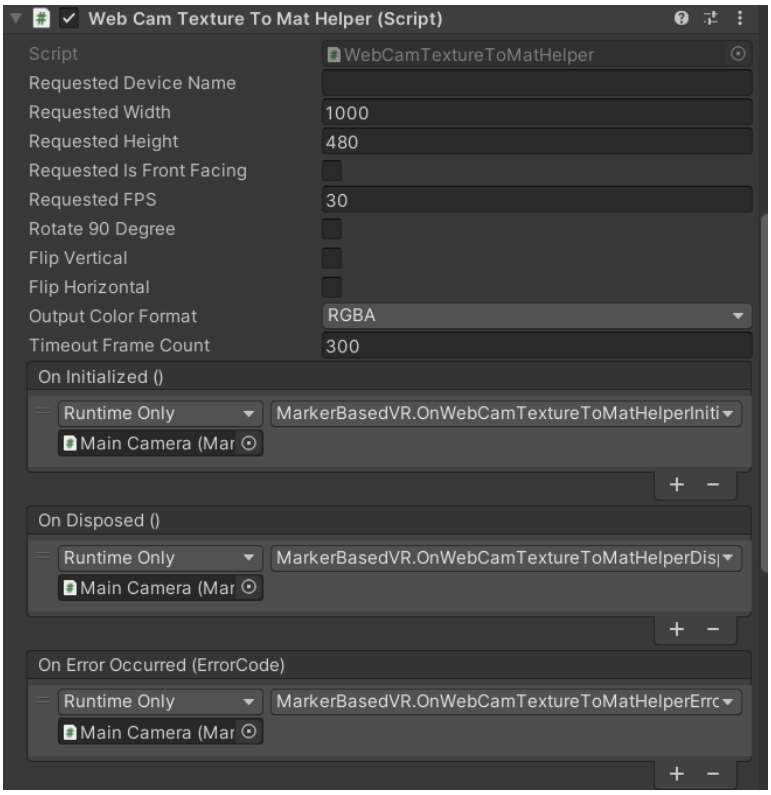


Source : Données de l'auteur

Comme nous ne souhaitons plus afficher notre caméra, nous n'avons plus la nécessité d'utiliser un *Quad*. Il reste cependant nécessaire de transformer l'image captée par notre *ARCamera* en matrice afin de pouvoir la traiter et obtenir la position des marqueurs détectés.

Pour cette raison, nous avons déplacé sur la *MainCamera* les scripts qui se trouvaient sur le *Quad*. La configuration reste similaire. Dans un premier temps, nous avons configuré le script *WebCamTextureToMatHelper* qui permet la conversion en matrice de notre image. Nous devons définir la taille de l'image à détecter et les méthodes qui sont appelées au lancement, à l'arrêt ou lorsqu'une erreur se produit.

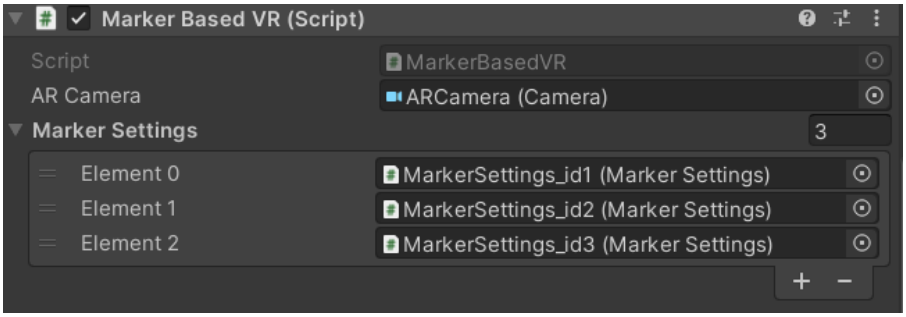
Figure 39 - Capture d'écran de la configuration du premier script sur la MainCamera



Source : Données de l'auteur

Dans un deuxième temps, nous avons configuré le script *MarkerBasedVR* en définissant la caméra utilisée et les paramètres des marqueurs à détecter. C'est ce script qui va appeler les méthodes ArUco permettant la détection des marqueurs.

Figure 40 - Capture d'écran de la configuration du second script sur la MainCamera

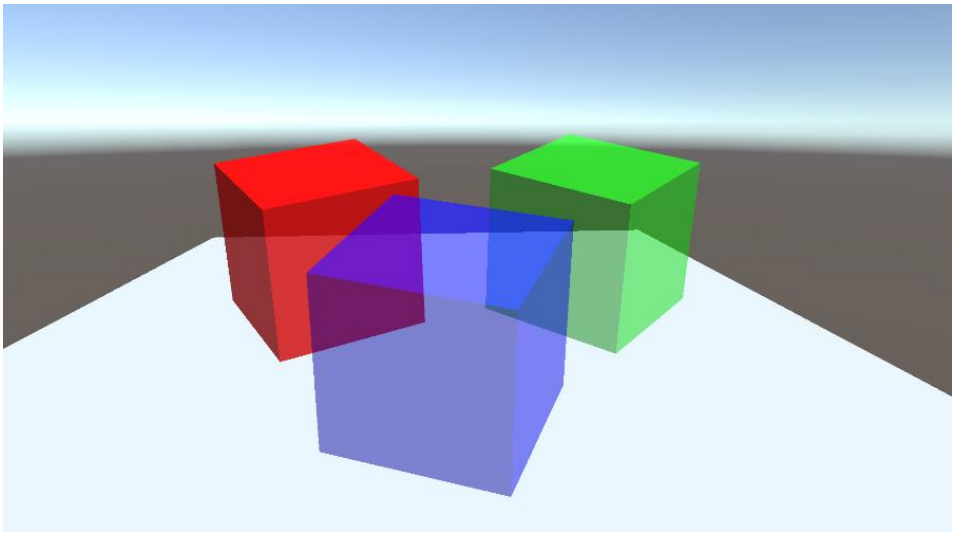


Source : Données de l'auteur

Cette configuration nous a permis de transformer notre projet de réalité augmentée en projet de réalité virtuelle tout en conservant la même logique au niveau du code. Désormais, le projet

utilise toujours la webcam pour capturer l'image, mais celle-ci n'est plus affichée. A la place, une scène virtuelle s'affiche et les objets associés aux marqueurs détectés s'affichent.

Figure 41 - Capture d'écran des objets 3D placés selon leur marqueur respectif en VR



Source : Données de l'auteur

Bilan

Au terme de ce quatrième Sprint, nous avons réussi à transformer notre projet de réalité augmentée en projet de réalité virtuelle dans le sens où une scène virtuelle remplace désormais l'image capturée par la caméra. Les marqueurs continuent à être détectés et les objets associés s'affichent correctement.

Nous avons donc validé les user stories n° 9 et 10 de notre Product Backlog qui correspondent à un effort total de 8 story points.

4.3.6. Sprint 5 – Détection d'objets réels dans une scène de réalité virtuelle

L'objectif principal de ce Sprint final était de rendre possible l'utilisation du prototype avec des objets réels afin de démontrer de manière concrète que la détection d'objets réels est possible avec les technologies choisies. Pour ce faire, nous devions améliorer l'interface graphique de notre scène afin de la rendre plus réaliste et aboutie et configurer des objets virtuels associés à des objets réels grâce à des marqueurs. Nous avions également comme objectif l'élaboration d'une procédure pour la création et la configuration des marqueurs afin de faciliter la reprise de notre projet.

Les user stories n° 11, 12 et 13 de notre Product Backlog sont concernées par ce Sprint. L'effort pour ce Sprint est estimé à 13 story points.

Figure 42 - Extrait du Product Backlog pour le Sprint 5

US Nr	En tant que ...	Je veux ...	car/pour ...	Acceptance Criteria	Priorit	Size	Sprint	MoSC
11	Utilisateur	tester mon application avec des objets réels	les afficher en temps réel dans la réalité virtuelle pour en faire une démonstration concrète	- 3 objets 3D qui s'affichent	350	5	5	SHOULD
12	Développeur	améliorer l'interface graphique	rendre le projet VR plus concret et réaliste	- UX	300	5	5	SHOULD
13	Développeur	être capable de reprendre le projet facilement	afin de pouvoir ajouter de nouvelles fonctionnalités	- documentation - commentaires et nettoyage du code	250	3	5	SHOULD

Source : Données de l'auteur

Utilisation avec des objets réels

L'objectif principal de ce travail était de démontrer qu'il était possible de détecter des objets réels dans la réalité virtuelle à l'aide des marqueurs ArUco. Afin de valider notre travail, nous devions créer un prototype. Nous avons donc apporté des modifications à notre scène pour qu'elle puisse détecter des objets réels à partir de marqueurs. Pour cette démonstration, nous avons remplacé nos cubes de couleur par des objets 3D que nous avons recherché dans l'*Asset Store Unity*. Nous avons décidé d'utiliser les trois objets suivants pour notre travail :

Figure 43 - Objets 3D utilisés pour le prototype

Source : Données de l'auteur

Nous avons donc ajouté ces trois objets à notre scène et supprimé nos cubes de couleur. Nous avons ajusté la taille et l'orientation de nos objets pour que le rendu soit le plus réaliste possible. Nous avons également veillé à ce que les proportions entre les objets de différentes tailles soient respectées.

Nous avons ensuite imprimé et collé nos marqueurs sur des objets réels. Nous avons mené quelques tests pour trouver une taille idéale pour nos marqueurs. En effet, les petits marqueurs sont idéals pour être placés sur des objets réels mais ils sont plus difficiles à détecter. Les grands marqueurs sont plus facilement détectables et permettent d'améliorer la fiabilité du tracking. Nous avons donc dû trouver un équilibre à ce niveau.

Nous avons opté pour des marqueurs de 5cm de large. Un exemple de ces marqueurs à la bonne dimension se trouve en annexe de ce rapport. En présentant nos marqueurs à notre webcam, les objets 3D s'affichent dans la scène de réalité virtuelle.

Figure 44 - Capture d'écran des trois objets 3D représentés dans une scène de VR simple

Source : Données de l'auteur

Amélioration de la scène

Afin de rendre le projet le plus réaliste possible, nous avons amélioré l'interface graphique de notre scène et simulé l'intérieur d'une pièce.

Nous avons opté pour l'intérieur d'un garage afin de se projeter dans la finalité du projet, à savoir un atelier de travail de constructeurs métallique. Nous avons trouvé un projet de scène sur l'*Asset Store Unity* et nous avons pu reprendre les éléments qui nous étaient utiles. Nous avons également ajouté une table sur l'avant de la scène pour que les objets puissent y être déposés.

Voici, ci-dessous, un aperçu de notre scène finale avec les trois objets.

Figure 45 - Capture d'écran de la représentation des trois objets 3D dans la scène de VR finale

Source : Données de l'auteur

Élaboration de procédures

Nous avons également profité de ce Sprint pour élaborer des procédures pour la création de nouveaux marqueurs ArUco depuis une scène Unity et pour la configuration d'un nouveau marqueur à reconnaître dans une scène. Celles-ci se retrouvent en annexe à ce rapport.

Bilan

Au terme de ce Sprint final, nous avons répondu aux user stories n° 11, 12 et 13 qui demandaient d'intégrer l'utilisation d'objets réels dans le projet et d'améliorer l'interface de la scène pour rendre la démonstration finale le plus réaliste possible. Nous avons désormais une scène de réalité virtuelle qui permet d'afficher trois objets réels. Nous avons également créé deux procédures permettant de faciliter la reprise du projet.

5. Conclusion

Dans ce rapport, nous avons commencé par présenter le contexte du projet, les étapes prévues pour sa réalisation et la méthodologie de travail utilisée. Dans un deuxième temps, nous nous sommes intéressés aux technologies de la réalité augmentée et de la réalité virtuelle et à leurs utilisations. Nous avons mis en avant l'utilité de l'utilisation de ces technologies dans l'enseignement car elles permettent de favoriser les apprentissages des élèves, notamment en améliorant leur engagement.

Nous nous sommes ensuite intéressés à la détection d'objets en réalité virtuelle. Nous avons analysé les solutions existantes qui permettent de mettre en place un prototype implémentant la détection d'objets en réalité virtuelle. Notre choix final s'est porté sur la bibliothèque open-source OpenCV et le module ArUco permettant la détection basée sur des marqueurs.

Finalement, nous avons détaillé les étapes qui nous ont permis de mettre en place un prototype avec une scène capable de détecter trois objets réels et de les représenter en réalité virtuelle à l'aide des technologies choisies.

5.1. Compétences développées et difficultés rencontrées

Grâce à la réalisation de ce travail, nous avons acquis de nouvelles compétences dans le domaine de la réalité augmentée et de la réalité virtuelle, en particulier en ce qui concerne l'utilisation du logiciel Unity et de ses extensions. En effet, nous n'avions aucune connaissance dans ce domaine avant la réalisation de ce travail. Nous avons donc du appréhender ces technologies et ces outils, ce qui a demandé un investissement en temps conséquent et représenté une contrainte importante pour la mise en place du prototype. Nous avons également dû nous appuyer sur plusieurs exemples de projets existants pour compléter nos connaissances et comprendre comment intégrer OpenCV et ArUco dans notre projet et les utiliser.

A travers la réalisation de ce projet, nous avons également pu améliorer nos compétences en gestion de projet, notamment en ce qui concerne l'utilisation de la méthodologie de travail Scrum.

Nous avons dû commencer par analyser et comprendre les besoins afin de mettre en place un Product Backlog qui a ensuite guidé notre travail de développement.

5.2. Perspectives d'évolution

A l'issue de ce travail, le prototype fonctionne selon nos attentes. En effet, nous disposons d'une scène de réalité virtuelle capable de détecter trois objets réels et de les afficher. Ainsi, nous avons démontré que les technologies choisies permettent de détecter des objets réels en réalité virtuelle.

Comme stipulé au point 2.1 de ce rapport, ce sous-projet sera intégré dans un projet plus global. Selon nous, les prochaines étapes consistent à implémenter la détection d'objets sur plusieurs faces, à rendre compatible l'application avec un casque de réalité virtuelle, puis à implémenter des interactions plus approfondies avec les objets.

5.3. Limites détectées

Pendant notre développement et nos tests, nous avons détecté quelques limites techniques. Ces points devront être testés dans les conditions réelles d'utilisation de l'application finale.

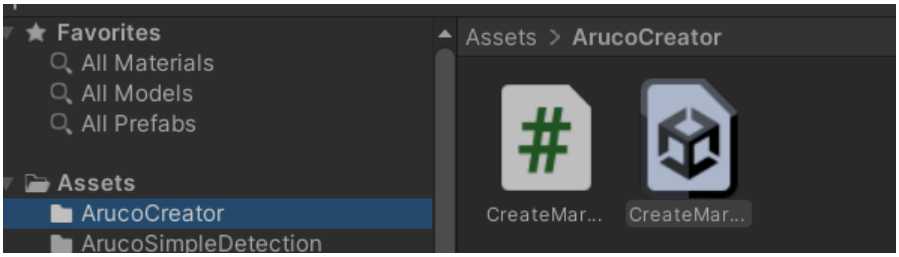
Premièrement, nous avons constaté que l'éclairage joue un rôle très important sur la fiabilité avec laquelle les marqueurs sont détectés. En effet, si la lumière n'est pas suffisante, nous remarquons que les marqueurs peinent à être détectés. Au contraire, si l'éclairage est trop direct ou trop important, alors les marqueurs peuvent parfois briller ou se refléter ce qui rend également leur détection compliquée. L'éclairage naturel est celui avec lequel nous avons obtenu les meilleurs résultats.

Deuxièmement, nous avons également constaté que la taille des marqueurs influe très fortement sur la qualité de la détection. En réduisant la taille des marqueurs pour les placer plus facilement sur de petits objets, nous avons constaté une baisse de l'efficacité de notre prototype. Les grands marqueurs sont plus difficiles à placer sur des objets, mais ils assurent une meilleure détection. Il faut donc trouver un équilibre entre la taille des marqueurs et les performances de l'application.

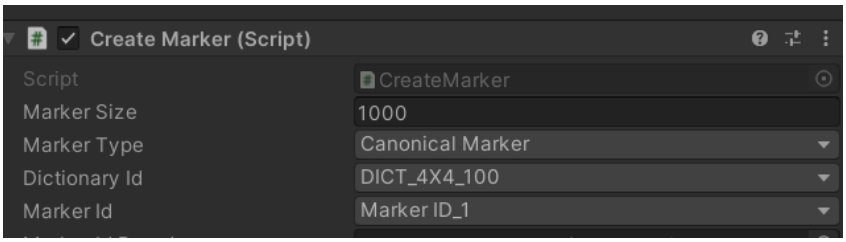
Annexes

Annexe I – Marche à suivre pour la création d'un nouveau marqueur

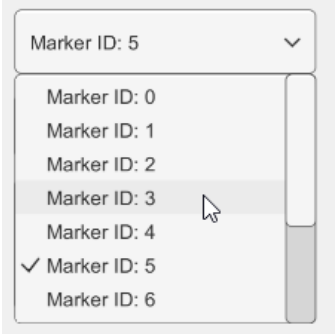
Cette marche à suivre décrit les étapes à suivre pour créer un nouveau marqueur. Elle a pour objectif de faciliter la reprise du projet. Pour créer des marqueurs supplémentaires, la scène *CreateMarker* qui se trouve dans le dossier *ArUcoCreator* peut être utilisée.



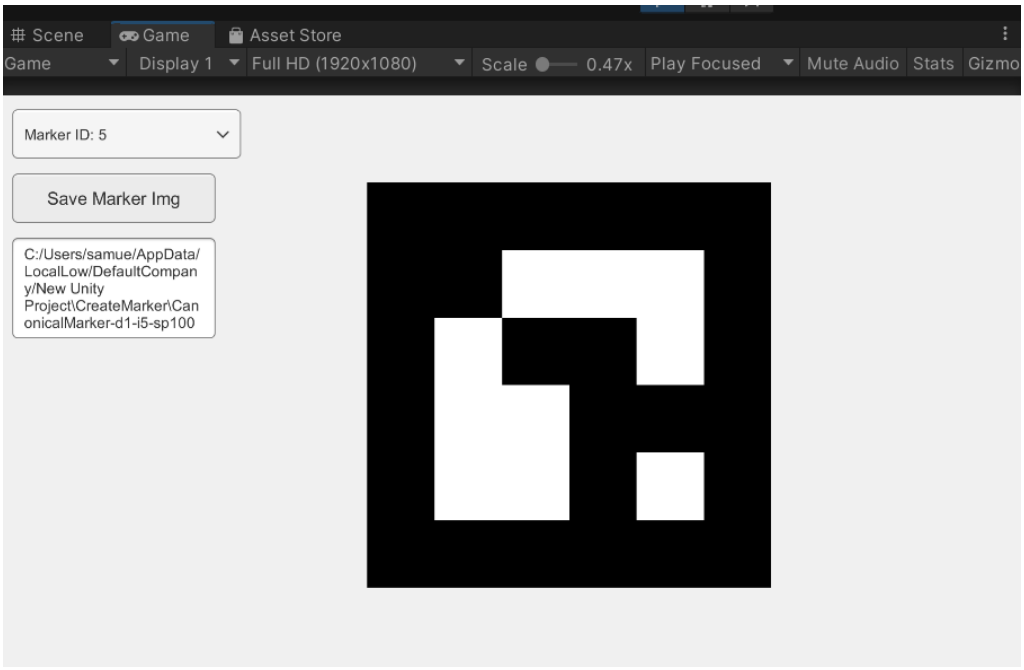
Par défaut, les marqueurs créés sont de type *Canonical Marker* (simple) et utilisent le dictionnaire *DICT_4X4_100* qui se base sur des marqueurs de 4 bits sur 4. Si nécessaire, ces réglages peuvent être modifiés sur l'objet *Quad*.



Une fois la scène exécutée, l'utilisateur peut choisir un identifiant dans la liste déroulante. Au besoin, la liste peut être complétée avec des identifiants supplémentaires jusqu'à atteindre la limite du nombre de marqueurs disponibles dans un dictionnaire.



Le nouveau marqueur s'affiche sur la droite de l'écran. Il est possible d'enregistrer le marqueur créé comme image en appuyant sur le bouton « Save Image Img ».

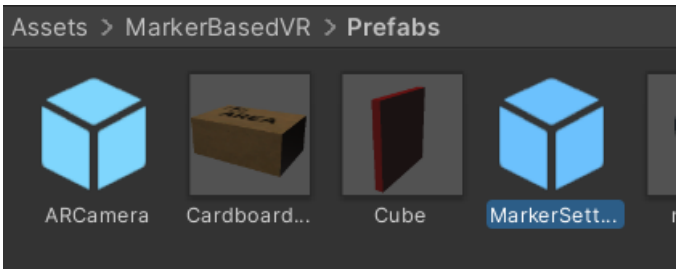


Le chemin d'accès au fichier s'affiche. Il est possible de le copier pour accéder à l'image créée.

Annexe II - Procédure pour reconnaître un nouveau marqueur dans la scène

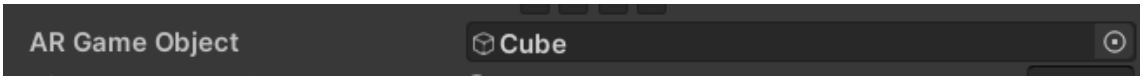
Cette marche à suivre décrit les étapes à suivre pour configurer un nouveau marqueur à détecter dans une scène. Elle a pour objectif de faciliter la reprise du projet.

Pour ajouter un nouvel objet à détecter, il faut commencer par ajouter un nouvel objet *MarkerSettings* dans la scène. L'objet se retrouve dans le dossier *Prefabs*.

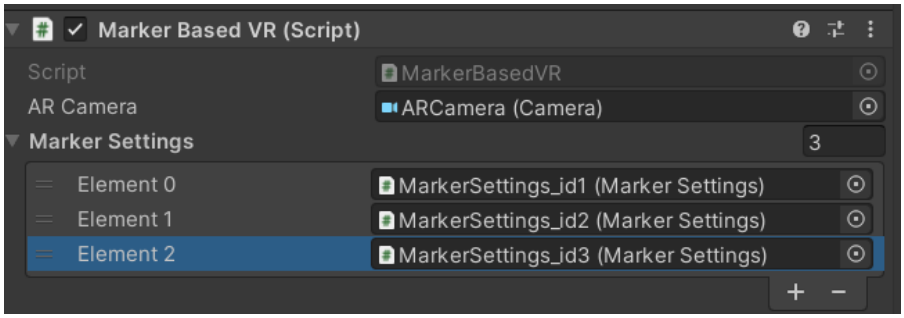


Il faut ensuite définir la taille en bits du marqueur que l'on souhaite détecter. Une grille s'affiche et permet de définir le marqueur à détecter. Il suffit de marquer les zones noires de notre marqueur dans le tableau.

L'objet *MarkerSettings* doit ensuite être associé à un objet 3D de notre scène. Il suffit de glisser l'objet dans la zone correspondante.



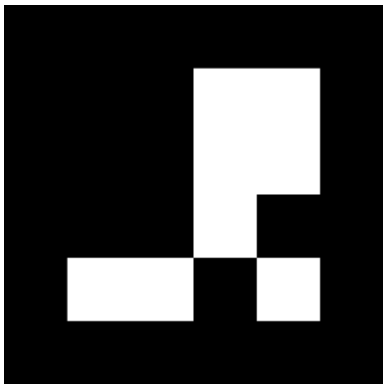
Il faut ensuite ajouter l'objet *MarkerSettings* aux paramètres d'appel du script *MarkerBasedVR* qui effectuera la détection. Pour cela, il suffit d'ajouter une ligne dans le paramètre *Marker Settings* et d'y glisser le nouvel objet *MarkerSettings* précédemment créé.



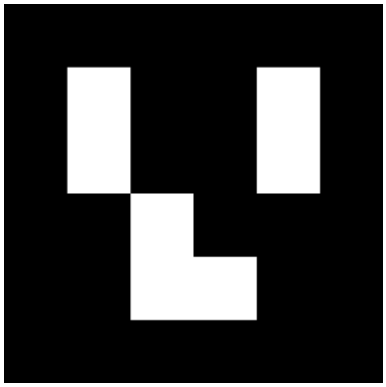
Annexe III – Exemples de marqueurs

Identifiant 1 (mug) :

Identifiant 2 (canette) :



Identifiant 3 (boîte) :

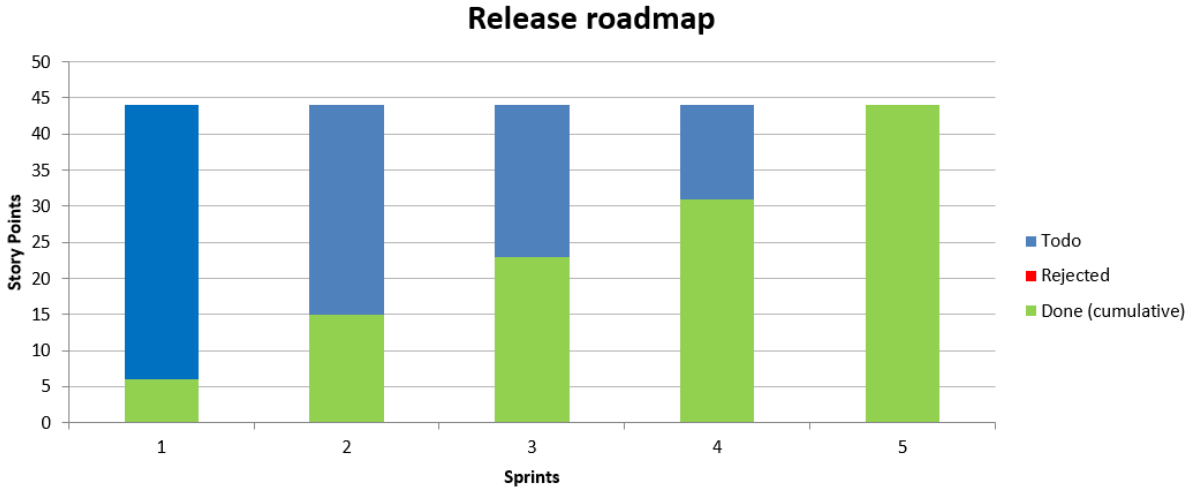


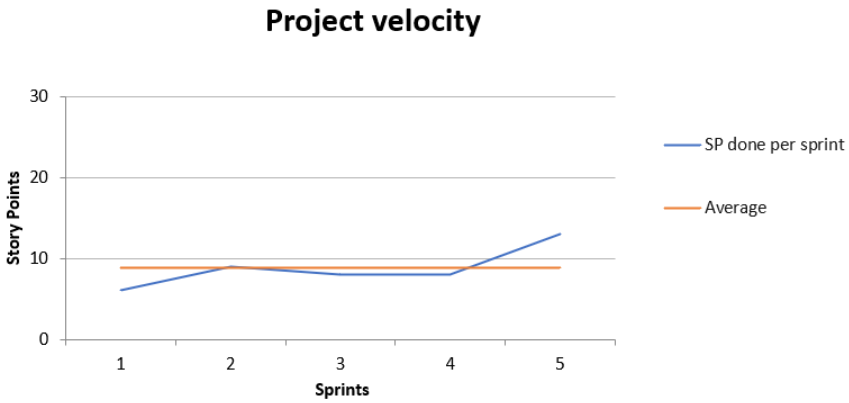
Annexe IV - Product Backlog

US Nr	En tant que ...	Je veux ...	car/pour ...	Acceptance Criteria	Priorit	Size	Sprint	MoSC
1	Développeur	mettre en place un projet test	configurer l'environnement de développement et comprendre la structure d'un projet	- projet simple qui tourne	1000	2	0	MUST
2	Développeur	créer des markers	pouvoir les imprimer, les utiliser et les reconnaître	- Marker AruCo	900	1	1	MUST
3	Utilisateur	décoder un marker en réalité augmentée	reconnaître le marker scanné et l'identifier	- id du marker	850	5	1	MUST
4	Développeur	créer un objet en réalité augmentée associé au marker	le visualiser/représenter dans la réalité augmentée	- objet qui s'affiche	840	3	2	MUST
5	Utilisateur	connaitre la position d'un marker par rapport à la caméra	pouvoir déplacer un objet virtuel selon la pose du marker	- position correcte de l'objet - déplacement en temps réel possible	830	3	2	MUST
6	Utilisateur	connaitre l'orientation d'un marker par rapport à la caméra	pouvoir déplacer un objet virtuel selon la pose du marker	- orientation correcte de l'objet - rotation en temps réel possible	820	3	2	MUST
7	Utilisateur	détecter plusieurs objets en même temps	les visualiser ensemble dans la réalité augmentée	- afficher au moins 2 objets en même temps - respect des dimensions et espacements	800	5	3	MUST
8	Utilisateur	modifier les aspects visuels des objets détectés	les différencier dans la réalité augmentée	- aspects différents des objets	750	3	3	MUST
9	Développeur	créer une scène de réalité virtuelle	transformer mon projet AR en VR	- scène de VR	720	3	4	MUST
10	Développeur	adapter la configuration de ma scène	pouvoir reconnaître, afficher et déplacer mes objets 3D dans la scène en fonction de mes marqueurs	- objets qui s'affichent et se déplacent dans la scène	700	5	4	MUST
11	Utilisateur	tester mon application avec des objets réels	les afficher en temps réel dans la réalité virtuelle pour en faire une démonstration concrète	- 3 objets 3D qui s'affichent	350	5	5	SHOULD
12	Développeur	améliorer l'interface graphique	rendre le projet VR plus concret et réaliste	- UX	300	5	5	SHOULD
13	Développeur	être capable de reprendre le projet facilement	afin de pouvoir ajouter de nouvelles fonctionnalités	- documentation - commentaires et nettoyage du code	250	3	5	SHOULD

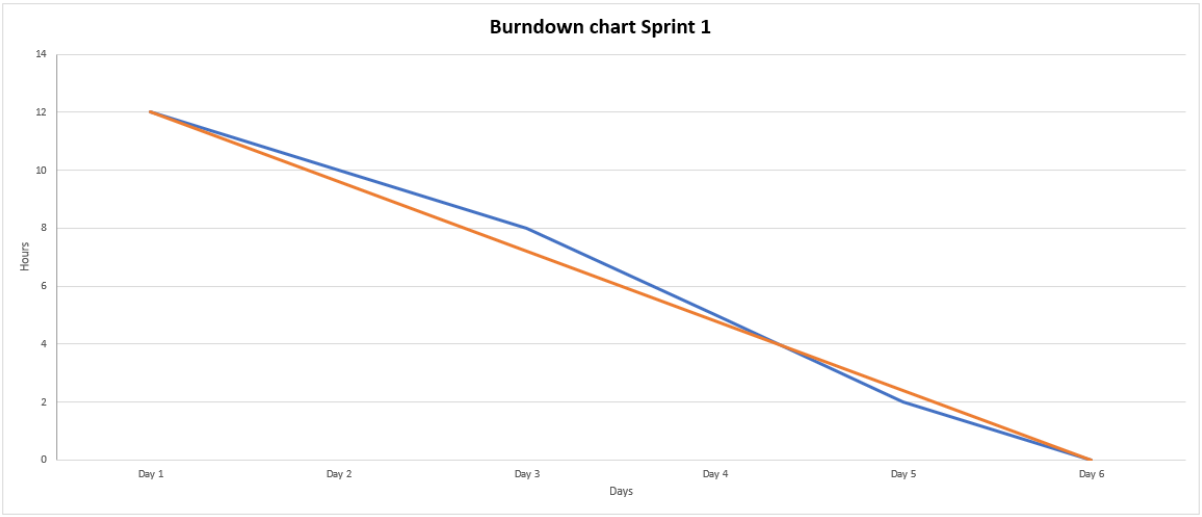
Annexe V - Release roadmap and project velocity

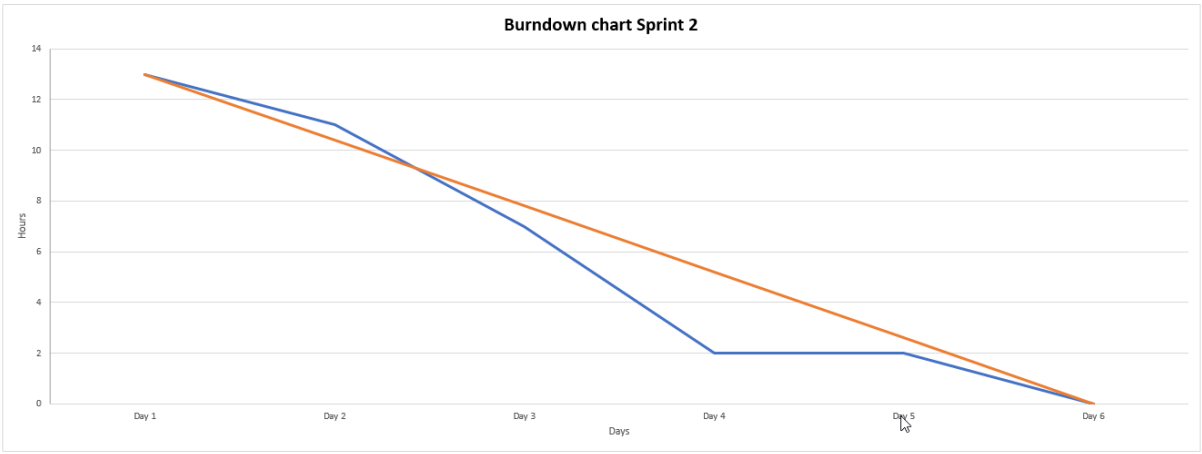
	Sprints 1	2	3	4	5
SP done per sprint	6	9	8	8	13
Average	9	9	9	9	9
Todo	38	29	21	13	0
Rejected	0	0	0	0	0
Done (cumulative)	6	15	23	31	44

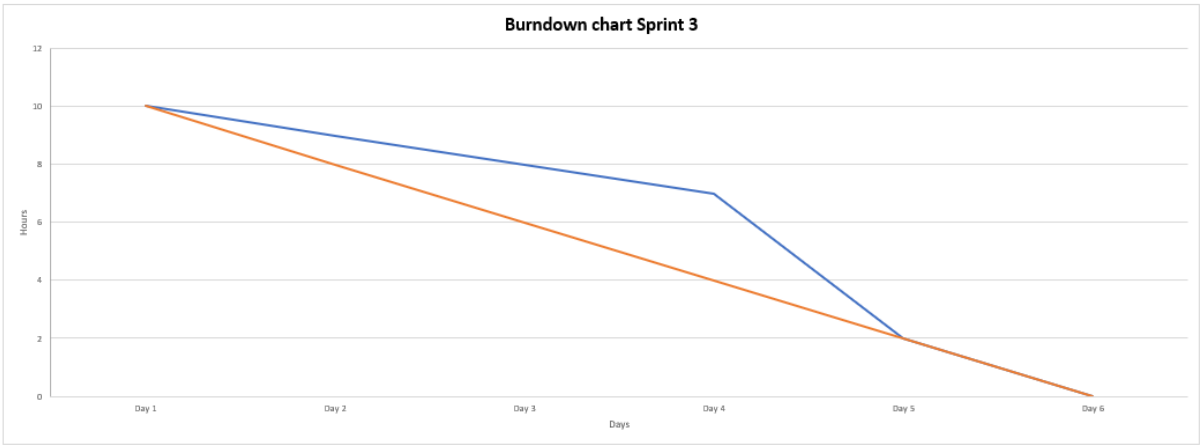


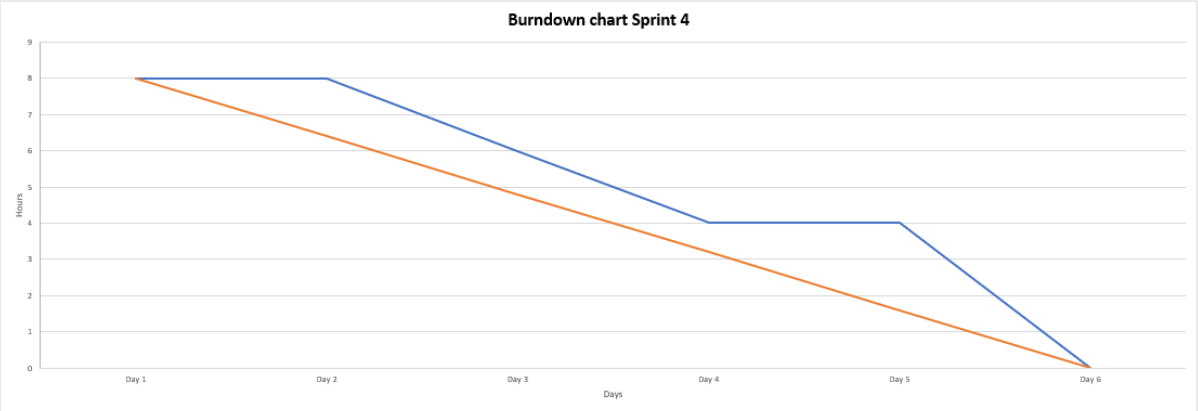


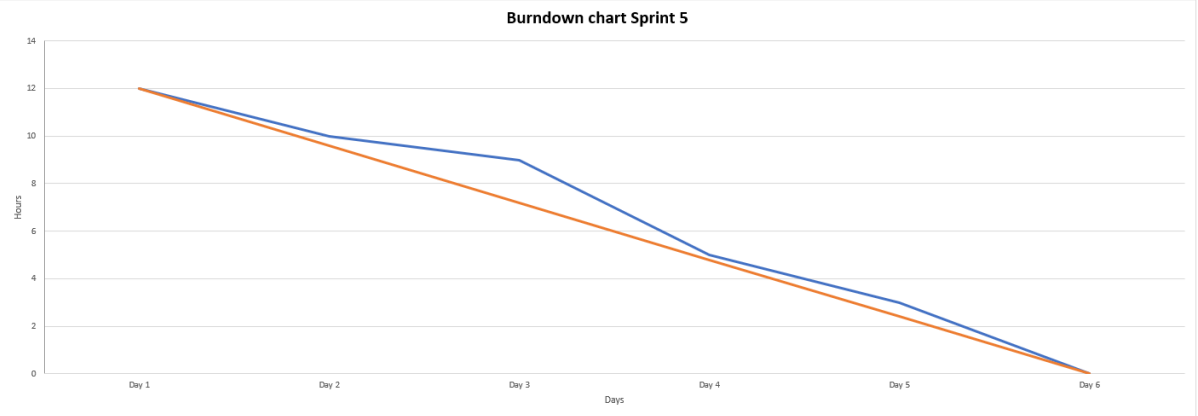
Annexe VI - Burndown charts








Burndown chart Sprint 4
Hours
Days
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6


Burndown chart Sprint 5
Hours
Days
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6

Bibliographie

3d émotion. (s.d.). *La réalité augmentée centrée*. Récupéré sur 3d émotion: https://www.3demotion.net/realite-augmentee-centree/

Artefacto. (s.d.). *Définition de la réalité virtuelle*. Récupéré sur Artefacto Augmented reality: https://www.artefacto-ar.com/realite-virtuelle/

Bidar, M. (2022, janvier 7). *Why 2022 is the year virtual and augmented reality will come to life*. Récupéré sur CBS News: https://www.cbsnews.com/news/2022-ces-virtual-reality-augmented/

Bobeshko, A. (2017, 04 07). *Object Recognition in Augmented Reality*. Récupéré sur Virtual Reality Pop: https://virtualrealitypop.com/object-recognition-in-augmented-reality-8f7f17127a7a

Futura, explorer le monde. (s.d.). *Réalité virtuelle : qu'est-ce que c'est ?* Récupéré sur Futura Sciences.

Larmand, A. (2022, 03 06). *Using Virtual Reality in Education*. Récupéré sur eduporium: https://www.eduporium.com/blog/eduporium-weekly-ar-and-vr-in-education

Lourdeaux, D. (2004). Réalité Virtuelle et Formation : Conception d'Environnements Virtuels Pédagogiques.

NormandErwan. (2019, 05 08). *ArucoUnity*. Récupéré sur ArucoUnity

OpenCV. (s.d.). *Detection of ArUco Markers*. Récupéré sur OpenCV: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

Orif. (s.d.). *Orif Sion*. Récupéré sur Orif: https://www.orif.ch/fr/sites-orif/valais/orif-sion

Orif. (s.d.). *Origines et mission*. Récupéré sur Orif: https://www.orif.ch/fr/a-propos/origines-et-mission

Paladini, M. (2018, 08 14). *3 DIFFERENT TYPES OF AR EXPLAINED: MARKER-BASED, MARKERLESS & LOCATION*. Récupéré sur blippAR: https://www.blippar.com/blog/2018/08/14/marker-based-markerless-or-location-based-ar-different-types-of-ar

R., S. (2021, 05 25). *Unity : tout ce qu'il faut savoir sur le puissant moteur de jeu*. Récupéré sur realite-virtuelle.com: https://www.realite-virtuelle.com/unity-tout-savoir/

Scrum. (s.d.). *What is scrum?* Récupéré sur Scrum.prg: https://www.scrum.org/resources/what-is-scrum

Unity Documentation. (2018, 03 28). *Vuforia - Unity Manual*. Récupéré sur Unity Documentation: https://docs.unity3d.com/es/2018.3/Manual/vuforia-sdk-overview.html

Vuforia. (s.d.). *Getting Started*. Récupéré sur Vuforia Developer Library: https://library.vuforia.com/

Hes·so VALAIS WALLIS
Haute Ecole de Gestion
Hochschule für Wirtschaft

Déclaration de l'auteur

Je déclare, par ce document, que j'ai effectué le présent travail de Bachelor seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du responsable de filière et du professeur chargé du suivi du travail de Bachelor.

Sierre, le 12 août 2022

Samuel Wenger