FAREVA, Site de Mirabel,
Route de Marsat, 63200
Riom

Rapport d'élève ingénieur

Projet de 2e année

Filière 1 : Informatique des systèmes interactifs pour l'embarqué, la robotique et le virtuel.

Module de formation virtuelle pour FAREVA

Présenté par : **Hugo BARRANDON et Loup RUSAK**

Responsable ISIMA : M. Emmanuel MESNARD

Responsable Entreprise : M. Hervé RICHARD

Jeudi 21 mars 2023 10h

Campus des Cézeaux, 1 rue de la Chébarde, TSA 60125, 63178 Aubière CEDEX

Remerciements

En premier lieu, nous tenons tout particulièrement à remercier l'entreprise **FAREVA** pour avoir accordé leur confiance à l'ISIMA ainsi qu'à nous, étudiants, pour mener à bien la réalisation de ce projet en lien avec leur activité.

Nous remercions chaleureusement notre responsable entreprise **M. Hervé RICHARD**, manager de projet chez FAREVA, pour nous avoir partagé ses souhaits et ses attentes sur l'ambition de ce projet novateur pour l'entreprise. Après rencontres et échanges avec lui sur les modalités de ce projet, nous le remercions pour sa disponibilité, sa sympathie et également pour sa confiance accordée.

Nous tenons à remercier notre responsable ISIMA **M. Emmanuel MESNARD**, pour son aide, sa grande sympathie, sa disponibilité, et surtout pour nous avoir partagé son indéniable expérience dans ce domaine, nécessaire à la réussite de ce projet.

Par ailleurs, nous remercions ceux ayant participé à la préparation de ce rapport et de notre soutenance, notamment **Mme. Murielle MOUZAT** et **M. Mamadou KANTÉ**.

Table des matières

Table des figures et illustrations

Résumé

Il nous a été demandé pour la réalisation de ce projet, de concevoir un outil logiciel en **réalité virtuelle et immersive** avec casque pour **formation d'opérateurs** au montage d'un **module de process pharmaceutique**.

Par la manipulation et **l'assemblage scénarisé** de pièces mécaniques au sein d'un **environnement virtuel*** se rapprochant le plus possible du type « laboratoire pharmaceutique », le produit fini développé sous **Unity 3D** avec le langage de programmation **C#**, servira de **preuve de concept** pour la société FAREVA.

Il permettra à l'entreprise de poursuivre sa **transformation digitale** en repensant complétement ses méthodes de formations, pouvant s'avérer longues et couteuses en moyens et en fonds, surtout dans un milieu aussi règlementé et pointu qu'est **l'industrie pharmaceutique**.

Mots-clés : Réalité virtuelle, réalité immersive, formation d'opérateurs, module de process pharmaceutique, assemblage scénarisé, environnement virtuel, Unity 3D, C#, preuve de concept, transformation digitale, industrie pharmaceutique.

Abstract

We were asked for this project, to design a **virtual and immersive** tool in virtual reality headset for **training** of **pharmaceutical process module** assembly operators.

By **scripted handling and assembling** mechanical parts inside a **virtual environment** such as pharmaceutical laboratory, the product, developed with **Unity 3D** and the **C#** will serve as a **proof of concept** for FAREVA.

It will allow the company to pursue its own **digital transformation** by rethinking its training methods that could be long and costly in terms of resources and funds, especially in an environment as regulated and specialized as **pharmaceutical industry**.

Keywords: Virtual and immersive reality, operators training, pharmaceutical process module, scripted assembly, virtual environment, Unity 3D, C#, proof of concept, digital transformation, pharmaceutical industry.

Introduction

Dans le cadre de notre projet de deuxième année à l'ISIMA, nous avons été mis en contact avec l'entreprise luxembourgeoise FAREVA, leader mondial de la sous-traitance industrielle dans le secteur pharmaceutique et parapharmaceutique mais également cosmétique et d'hygiène. Au sein de ses laboratoires et sites de production, FAREVA élabore, fabrique et conditionne des solutions innovantes pour répondre aux besoins des clients dans ce secteur. La société continue de croitre, d'embaucher et de former de nouveaux opérateurs pour sa production.

Ces formations, longues et couteuses en moyens et ressources, durent en moyenne un an. Une année pendant laquelle les nouveaux salariés doivent effectuer dans des conditions très spécifiques des manipulations pratiques sur les machines de production, tout en étant supervisés. Le domaine pharmaceutique, étant extrêmement réglementé et secret, ces formations impactent donc directement sur la productivité de l'entreprise et gagnent à être bouleversées.

C'est dans cette optique-ci qu'il nous a été demandé de réaliser une preuve de concept et plus précisément de concevoir un outil logiciel 3D de formation virtuelle* et immersive au montage d'un module de production pharmaceutique. Conformément au codes et pratiques dans un développement en réalité virtuelle, l'outil se doit de reproduire les interactions physiques entre l'opérateur et les outils et pièces mécaniques mises à disposition.

Ainsi la problématique principale de ce sujet a été de proposer un résultat vraisemblablement réaliste et fidèle, que cela soit dans les aspects mécaniques et logiques, dans l'interaction, dans la scénarisation, sans oublier l'environnement virtuel.

Ces exigences et nécessités ont été les lignes directrice du développement de notre projet, nous allons donc vous présenter dans un premier temps l'entreprise FAREVA, son activité, comment est venue l'idée de s'orienter vers la formation virtuelle* et pourquoi cela deviendra nécessaire. Dans un second temps nous aborderons le matériel et les outils utilisés pour mener à bien ce projet, les différentes étapes de notre production ainsi que les méthodes utilisées. En troisième partie, nous rentrerons dans le détail du produit fini et nous discuterons sur les solutions non implémentées et les améliorations possibles. Nous finirons par dresser un bilan de ce projet, ses perspectives d'avenir et ses potentielles améliorations.

I – Contexte du projet

1 – Présentation de l'entreprise

1.1 – Son histoire

FAREVA est une entreprise de droit luxembourgeoise, leader mondial de la sous-traitance industrielle depuis sa création en 1981 dans l'Ardèche. Elle joue le rôle de façonnier dans le secteur pharmaceutique, parapharmaceutique, cosmétique et de l'hygiène.

Figure 1 : Logo de l'entreprise FAREVA

Durant cette année 1981, Bernard FRAISSE, âgé de 25 ans crée une petite entreprise de conditionnement nommée RCI. En s'entourant d'une petite équipe, l'unité devient la Fabrication Chimique Ardéchoise (FCA) et diversifie rapidement ses activités, en intégrant la recherche, la formulation et la production de produits chimiques.

Pendant les années suivantes, FCA prospère et élargit encore son champ d'activités en réalisant des produits ménagers et des aérosols. Puis suivent la cosmétique et enfin la production pharmaceutique au sens large. Ne cessant de croître, FCA devient en 2004 une entreprise internationale et change ainsi de nom. Fort de ses racines, Bernard FRAISSE la rebaptise FAREVA qui signifie « faire rêver » en patois ardéchois.

Trente ans plus tard, la société est devenue une multinationale et le leader mondial de la sous-traitance industrielle dans l'univers industriel et ménager, de la beauté, du maquillage, de la pharmaceutique et notamment des substances et principes actifs pharmaceutiques associés aux excipients.

1.2 – Son activité

Grâce à ses nombreuses équipes de recherche et développement et de marketing, FAREVA conçoit et met au point les produits de marques dans le monde entier, en décryptant l'évolution de ce marché. Ils proposent donc des produits innovants et répondant aux attentes des clients et des consommateurs. La

diversité des technologies utilisées, la maitrise des procédés industriels et le haut niveau d'exigence qualité et réglementaire leur permet d'industrialiser et fabriquer à grande échelle de nombreuses formules.

La mission d'origine de l'entreprise étant le conditionnement, elle garantit pour l'ensemble de sa production un conditionnement adapté, la disponibilité et la livraison du produit fini dans les temps.

FAREVA a durant les années acquéri de nombreux sites de productions et filiales partout en Europe puis en Amérique, ce qui lui a permis de nouer des relations fortes avec de nouveaux clients.

Elle emploie à l'heure actuelle plus de 13 000 personnes répartis sur plus de 40 sites dans le monde dont le dernier acquis est le site de Mirabel à Riom (Puy-de-Dôme) en 2021, appartenant auparavant à la filiale française du groupe américain MSD. C'est en rapport avec les nouvelles activités de ce site local que nous avons réalisé ce projet. Son dernier chiffre d'affaires s'élève à 1.8 milliard d'euros.

Figure 2 : Photo du site Mirabel de FAREVA (Riom)

1.3 – Son organisation

Le comité exécutif de l'entreprise est composé de 8 Vice-Présidents et de 1 CEO avec, à leur tête, Bernard FRAISSE, fondateur et président. Cependant, chaque site de production ou de recherche possède en quelque sorte son autonomie, que ce soit dans la clientèle et les projets. Le site pharmaceutique local et historique de Mirabel emploie environ 350 personnes et possède sa propre direction.

Dans la cadre de la future introduction d'un nouveau produit, le site de Riom et plus particulièrement M. Hervé RICHARD, chef de projet, a contacté l'ISIMA et a proposé un projet en lien avec son activité et celle du site de production. Nous sommes très heureux d'avoir pu contribuer à leurs projets et nous les remercions de nous avoir accordé leur confiance.

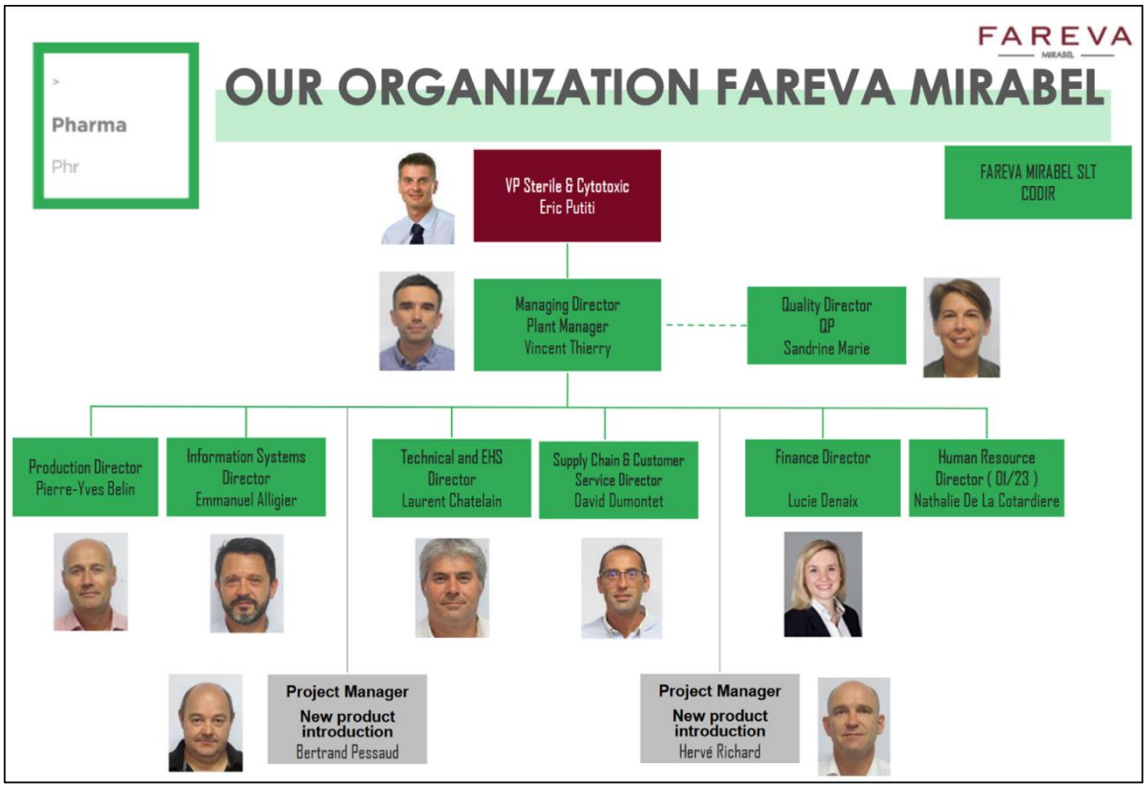

Figure 3 : Organisation sociale du site FAREVA Mirabel

2 – Étude du problème et de l'existant

2.1 – Connaissance du problème

Dans un domaine comme l'industrie pharmaceutique, la logistique de gestion des usines, des centres de recherche et développement mais également du travail des employés est primordiale afin d'assurer un cycle de production, de conditionnement et d'approvisionnement constant aux clients.

FAREVA est une entreprise en forte croissance et continue d'embaucher de nouveaux opérateurs pour sa production en les formant aux méthodes et outils de la société. Ces formations, concernant aussi bien le lancement de la production, son suivi que la maintenance, s'avèrent dans la plupart des cas longues et couteuses en moyens financiers et logistiques.

Se déroulant en moyenne pendant une année entière pour un opérateur, ces formations posent actuellement un problème pour la production de l'entreprise. En effet, une supervision lors des manipulations pratiques des machines de travail est nécessaire pour la transmission du savoir et des compétences. De plus, par mesure de sécurité et d'engagements de qualité des produits, la chaine de production se doit dans certains cas d'être arrêtée, ce qui représente un manque à gagner pour l'entreprise et des couts supplémentaires pour une prochaine remise en route.

2.2 – Analyse du problème

Au cœur de sa formation, le futur opérateur passe des heures et des heures à s'entrainer à répéter sans erreurs les techniques et les manipulations, en salle blanche et souvent sous gants, afin d'atteindre une précision des plus fines tout en minimisant le taux d'erreurs et la perte de ressources. Il doit connaitre parfaitement le fonctionnement de ses machines de travail et savoir comment les entretenir.

Ces machines étant extrêmement couteuses et volumineuses, une grande partie des entreprises ne peuvent pas se permettre d'en réserver certaines rien qu'à la formation, et c'est bien entendu également le cas pour FAREVA.

Plusieurs solutions ont été envisagées :

- Faire appel à des société extérieures pour réaliser la formation. Le problème soulevé a été le manque de similitudes avec la production propre de l'entreprise et donc une connaissance trop vague des procédés en plus d'un coût de sous-traitance élevé.
- Augmenter drastiquement le temps d'apprentissage théorique et par conséquent réduire la pratique pour moins impacter le cycle de productions. La solution n'a pas été viable car les opérateurs réalisaient trop d'erreurs lors de leur incursion dans la production.
- Diminuer le nombre de superviseurs et augmenter le nombre d'apprentis en charge. A l'instar de la solution évoquée précédemment, les contraintes de qualités n'étaient pas toujours respectées.

Aucune solution exposée ci-dessus n'est donc conforme aux attentes de l'entreprise et des nécessités du secteur.

2.3 – Étude d'une solution

Comme toute entreprise de nos jours, FAREVA cherche à poursuivre sa transformation digitale afin de s'adapter aux changements du monde industriel et de la recherche. Ainsi une autre solution répondant aux problèmes de formations a été envisagée : remplacer une importante partie de la formation pratique sur machine par une formation virtuelle, profitant donc du réalisme et de la liberté que les outils informatiques peuvent offrir[1].

Notre travail a donc été d'explorer cette solution déjà adoptée par de nombreuses entreprises dans le monde et de créer un outil 3D virtuel et immersif de formation au montage d'un module de production pharmaceutique se rapprochant du produit futur fabriqué et lancé par le site de Mirabel. L'objectif de FAREVA est d'arriver à trois quarts de formation virtuelle* et un quart de formation pratique réelle, le virtuel ne pouvant totalement remplacer le réel.

Le produit fini se présente sous la forme d'une preuve de concept que M. RICHARD présentera à sa direction comme une nouvelle approche de ces formations. Sa fonction précise sera de former des opérateurs à la manipulation et à l'assemblage de pièces, au sein d'un environnement de type laboratoire pharmaceutique.

Le fonctionnement de l'outil doit reproduire les différentes interactions physiques entre l'opérateur et ce qu'il peut actionner, bouger ou plus généralement ce qu'il peut manipuler. La solution doit également prendre en compte les contraintes réelles d'étapes d'actions dues à la scénarisation d'une procédure industrielle.

Pour la réalisation de ce projet, FAREVA et M. RICHARD ont tenu à imposer certaines contraintes réelles de ce métier pour que l'outil soit des plus réaliste et fidèle. Le cahier des charges donné comporte donc ces points essentiels :

- La scène virtuelle se doit de ressembler à l'environnement réel de production.
- L'utilisateur doit être capable de situer ses mouvements et ses actions dans l'environnement.
- Il doit être capable de manipuler et d'interagir avec des éléments du décor, des objets ou des outils.
- Le scénario doit inclure des contraintes pour le montage ainsi que des contraintes physiques régissant les éléments.
- La vue immersive doit intégrer une interface utilisateur permettant l'affichage d'informations de déroulé et retours d'erreurs.
- La scène doit comporter une paillasse sur laquelle sont posées des pièces à assembler.

La liste n'est bien entendu pas la plus exhaustive et FAREVA a été ouvert à toute proposition de notre part concernant des compléments d'interactivité.

II – Mise en œuvre et réalisation

1 – Ressources matérielles et logicielles

1.1 – Matériel de réalité virtuelle

Pour mettre en œuvre le développement d'un outil virtuel immersif, il est nécessaire de se pencher sur le matériel de réalité virtuelle à utiliser. Dans notre cas, nous avons dû nous conformer au cahier des charges présenté : l'utilisateur doit pouvoir visualiser son environnement virtuel* et interagir avec des objets et éléments du décor. Nous nous sommes donc naturellement tournés vers les **casques de réalité virtuelle**, assurant une liberté de mouvements, une facilité d'utilisation et des performances largement suffisantes pour notre projet.

Un casque de réalité virtuelle est donc un dispositif électronique porté sur la tête de l'utilisateur lui permettant de s'immerger dans un monde virtuel, c'est-à-dire un environnement réel récréé en images de synthèses. Ces équipements sont équipés d'un visiocasque stéréoscopique, comprenant un écran et un système optique permettant de récréer une vision en relief, un système de son stéréo et bien entendu

une multitude de capteurs de position. Ces capteurs, pouvant être des gyroscopes, des accéléromètres ou des caméras, concernent les mouvements de la tête mais également ceux des membres avec des contrôleurs sous forme de manettes, permettant également de simuler les interactions.

Il existe deux types de casques de réalité virtuelle : les casques à **suivi interne** et les casques à **suivi externe**. Pour ceux en suivi externe (outside out tracking), des capteurs de position externe au casque sont nécessaires et l'informatique permettant le fonctionnement du programme est dans un ordinateur connecté au casque par un fil. En ce qui concerne les casques a suivi interne (inside out tracking), tout est embarqué au sein du casque, que ce soit les capteurs de position ou bien l'informatique permettant son fonctionnement.

Nous avons comparé ces deux solutions et il en découle une différence significative dans la précision du suivi des mouvements, en effet le suivi externe est beaucoup plus précis que le suivi interne, malgré une liberté un peu plus restreinte. Sur le marché de ces types de casques, nous retrouvons principalement deux grands constructeurs : **Occulus** et **HTC**. Bien que très similaires, ils embarquent des technologies différentes, certaines propriétaires, et sont plus ou moins ouvertes aux solutions open source développées dans le domaine.

Ayant nécessité pour ce projet du plus de précision possible, un réalisme accru et moins de restrictions en puissance de calcul, nous donc avons développé et testé notre projet sur des casques à suivi externe que nous pouvions utiliser à l'ISIMA, à savoir **l'Occulus Rift** et le **HTC Vive**.

Figure 5 : Casque HTC Vive et accessoires

Figure 5 : Casque Occulus Rift et accessoires

Ne connaissant pas le matériel qui sera potentiellement adopté par FAREVA, nous avons durant ce projet, fait en sorte de développer notre outil en le rendant compatible avec tout type de casque. Pour réaliser cela, nous avons utilisé des logiciels et solutions adaptées à cet aspect universel.

1.2 – Outils logiciels

Pour la création de ce genre d'outils de réalité virtuelle et immersive, plusieurs logiciels gratuit et open source sont disponibles sur le marché, comme **Unreal Engine**, **Google AR et VR** ou **Unity**. Unity et Unreal Engine sont par défaut des outils plus adaptés pour le jeu-vidéo, cependant on observe ces dernières années, une tendance à se spécialiser de plus en plus dans la création 3D virtuelle au sens large et les expériences innovantes en réalité immersive. Il est clair qu'entre ces deux logiciels, le plus démocratisé et le plus utilisé dans le domaine de la VR reste Unity. Après quelques cours effectués sur ce logiciel, c'est celui qui nous a été conseillé pour le projet et c'est celui que nous avons sélectionné.

Unity est donc un moteur de jeu multiplateforme l'un des plus répandus dans l'industrie du jeu vidéo, que cela soit pour de grands ou petits studios du fait de sa simplicité de prise en main, de sa rapidité de prototypage et de son adaptabilité à tout support. Le logiciel a la particularité d'utiliser du code écrit en **C#** afin de régir les éléments constitutifs et scripter leurs comportements et interactions selon la volonté du créateur, ceci en plus de tout ce qui existe déjà au sein du moteur de jeu. Pour notre projet nous avons utilisé la **version 2021.3.12F1 LTS**, donc une version **Long Term Support** dont le suivi de la part de la firme Unity est assuré sur une période plus longue. Dans le monde du génie logiciel il est nécessaire d'opter pour ce genre de version pour créer des outils qui continueront d'évoluer et resteront un certain temps au sein de l'entreprise.

L'interface d'Unity est assez simple et intuitive. Elle comporte tout d'abord la **Scène** virtuelle dans laquelle il est possible de visualiser tous les éléments 3D constitutifs de l'environnement virtuel* regroupés en une hiérarchie (**Hierarchy**) d'objets appelés **GameObjects***. Nous avons dans la continuité de cette scène **l'Inspector,** qui lui permet de visualiser et de modifier toutes les propriétés d'un GameObject*, notamment en lui rattachant de nouveaux « composants* » (**Components**), souvent sous la forme de scripts. Dans Unity, tout est GameObject* ; que ce soit le modèle 3D d'un objet virtuel, un script d'interaction ou de comportement associé à un objet ou encore une image. Cette particularité permet une utilisation homogène, intuitive et simple pour tout le développement d'une solution.

Concernant la création en réalité virtuelle et immersive, nous venons rajouter à Unity des modules d'extension que l'on appelle « **plugins** ». Il existe un plugin propriétaire Unity appelé « **XR Interaction Toolkit** », que nous avons utilisé dans un premier temps avant de passer au plugin « **Steam VR** », permettant de réaliser encore plus. Comme celui de Unity, la grande force de Steam VR est de permettre une adaptation simple et polyvalente d'une solution VR sur n'importe quel casque et matériel de réalité virtuelle.

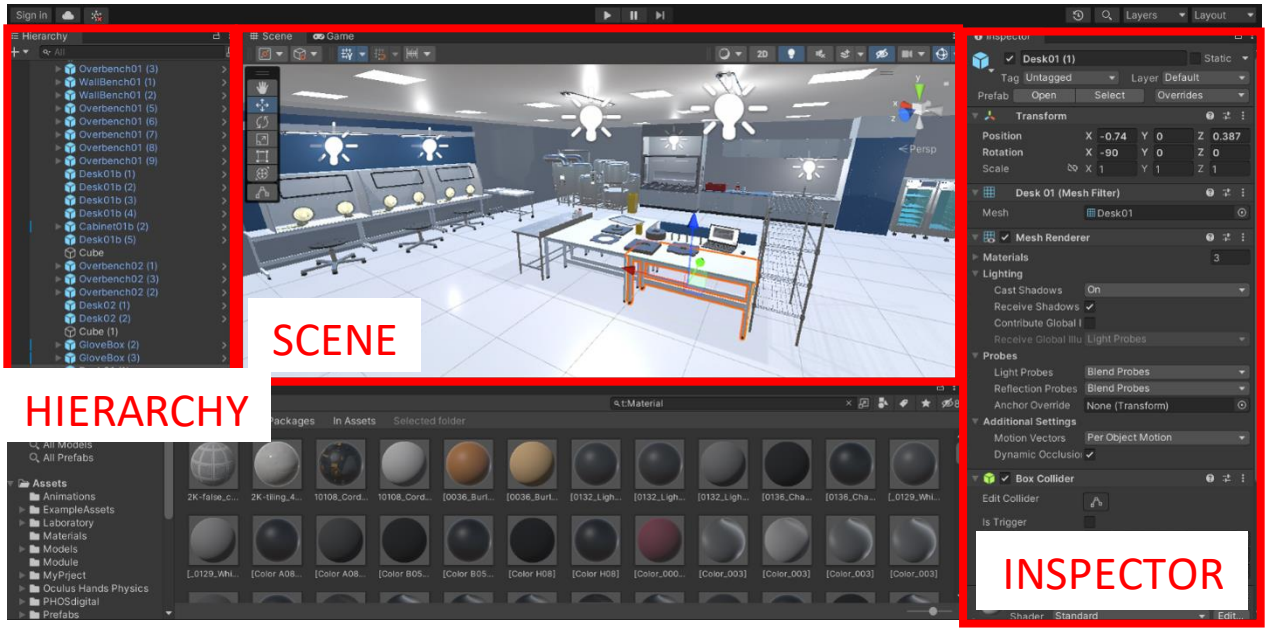

Figure 6 : Interface Unity

2 – Planification du travail

Afin de réaliser ce projet dans de bonnes conditions et n'ayant pas eu dès le début toutes les informations de la part de l'entreprise, nous avons décidé de diviser notre travail en deux phases.

La première phase a été une période d'expérimentation durant laquelle nous avons effectué beaucoup de recherches (modèles 3D et tutos scripting) et avons testé d'implémenter des concepts de système d'interactions et de montages simplifiés, seulement en 3D et sans aspect immersif. C'est grâce au travail réalisé durant cette phase que nous avons pu présenter une ébauche du concept, adaptée rapidement en réalité virtuelle, à M. RICHARD de FAREVA lors de notre **rencontre du 15 décembre**.

À la suite de cet échange, il nous a donné une multitude d'informations supplémentaires, avec des photos nous servant d'exemples afin de mieux visualiser les environnements et les conditions de travail dans ce domaine. Mais aussi des précisions sur le nouveau module qui sera créé pour que nous puissions, sans le copier, puisque tout est confidentiel, en créer un aussi réalisable et dans des caractéristiques similaires.

C'est à partir de ce moment que nous avons entamé notre deuxième phase de production, durant laquelle nous avons effectué à nouveau des recherches de modèles 3D pour la création d'un environnement de type laboratoire, créé de toute pièce le nouveau module à assembler et amélioré et affiné nos systèmes d'interactions et de montage en VR pour le rendu final.

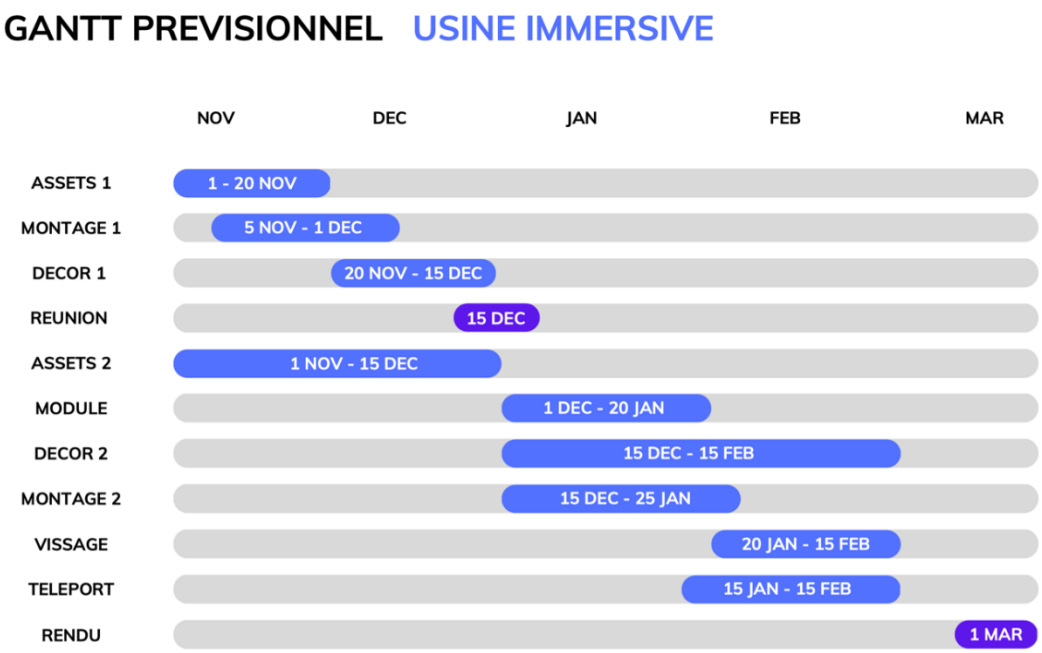

Figure 7 : Gantt prévisionnel du projet

Comme vous pouvez le voir sur le diagramme ci-dessus, nous avons bien deux phases distinctes ; avant et après notre rencontre avec M. RICHARD. La deuxième partie étant celle la plus conséquente en charge de travail. Nous avions une première idée de ce qu'il fallait réaliser grâce à une première rencontre avec notre tuteur ISIMA M. MESNARD, mais nous avons eu beaucoup de nouvelles informations et attentes supplémentaires. Il nous a été difficile de prévoir à l'avance ce que nous avions à faire durant cette deuxième phase de production et certaines tâches ont pris beaucoup plus de temps que prévu.

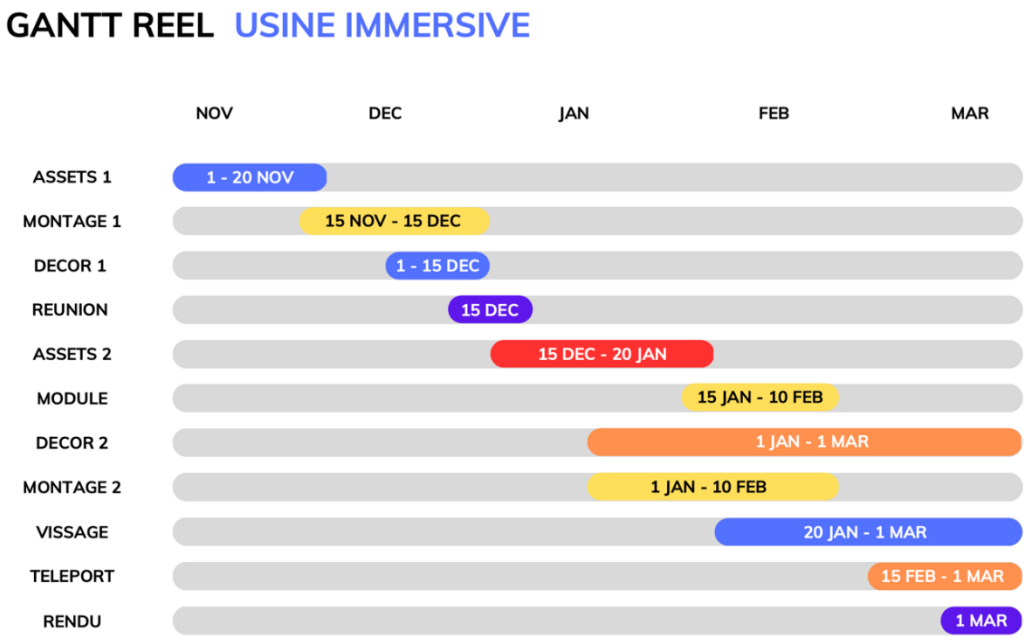

Figure 8 : Gantt réel du projet

3 – Conception de la solution

Concernant la conception de la solution et son développement, nous avons recensé deux types de tâches : des tâches techniques de développement pour les interactions et scénarisations, et des tâches plus graphiques associée à l'immersion en réalité virtuelle et l'espace 3D environnant.

3.1 – Interactions et scénarisations

3.1.1 – Logique d'interactions

La logique d'interactions de notre projet se base sur les « **Colliders** » qui sont les boites de collisions de Unity. Ce sont des composant* que l'on peut attacher à un GameObject*. Il en existe plusieurs, différenciables principalement par leur forme. On a notamment des boites de collisions en forme de capsule et de sphère que nous avons beaucoup utilisé, ainsi que le « **Mesh Collider** » qui permet de générer procéduralement une boite de collisions* à partir d'un rendu visuel d'un objet 3D.

Ces boites de collisions émettent un signal lorsqu'elles rencontrent une autre boite de collisions*, et se signale va se propager dans la hiérarchie en commençant par l'objet contenant la boite de collisions* et en remontant ses parents jusqu'à être capté par un composant* de type « **RigidBody** ». Le RigidBody va ensuite utiliser ce signal pour calculer la physique entre les deux boites de collisions et l'appliquer aux deux objets.

Il est possible d'utiliser les boites de collisions suivant deux modes[2] :

- **Collision physique** : la collision existe réellement, les objets sont impactés par cette collision et s'entrechoquent dans la simulation. La boite de collision* émet un signal de type « Collision ».
- **Collision logique** : la collision n'est pas simulée, les objets ne sont pas physiquement impactés par la collision. Ce mode sert principalement à détecter des positions et envoi un autre type de signal nommé « Trigger ».

Chaque signal a des méthodes dédiées appelées par le RigidBody au moment où il détecte le signal. On a trois fonctions par signal[2] :

- **Enter** : appelée quand on entre en collision.
- **Stay** : appelée régulièrement quand on est en collision.
- **Exit** : appelée quand on sort de la collision.

Chacune de ces six méthodes peuvent donc être redéfini dans un ou plusieurs scripts attachés au même objet que le RigidBody, et seront appelées automatiquement par Unity.

Au début du projet, nous avions simplement utilisé les Mesh Collider en mode collision physique pour avoir rapidement une boite de collision* qui correspondait approximativement aux pièces du module à monter. On utilisait donc les méthodes « onCollisionEnter » et « onCollisionExit » pour savoir si deux

objets de notre module sont entrés en collision et on les assemblait s'ils le pouvaient, ce qui correspond à un système de clipsage. Ce système assez simple n'était malheureusement pas très réaliste, on avait une impression « d'aimant » dû au fait que lors de l'assemblage les deux objets se positionnaient automatiquement à la bonne position relative, peu importe le point de collision, pour des raisons que nous détaillerons dans la partie dédiée à la logique d'assemblage.

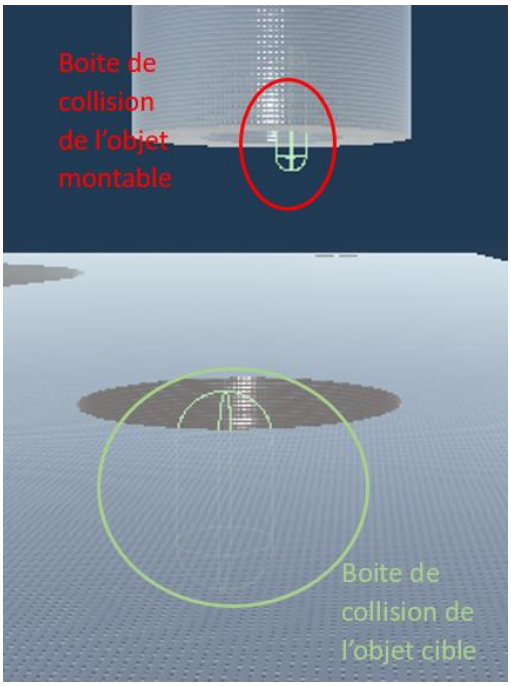

Figure 9 : Exemple du mécanisme de collisions pour montage

Pour améliorer ce système, nous avons commencé à réfléchir sur l'ajout de points de liaison qui nous permettraient de détecter si les deux objets sont suffisamment bien positionnés l'un par rapport à l'autre pour être assemblé. Pour cela, nous avons ajouté des objets fils aux parties de notre nouveau module. Ces GameObject* vides n'avaient que deux composants* : un **Transform** (imposé par Unity pour tout GameObject*[2]) et une boite de collision* en mode collision logique. En ajoutant ces points de liaison à chaque objet du module, on peut avoir une simulation beaucoup plus réaliste et ne liant deux objets que si les points de liaisons sont en collision. Comme on peut le voir sur la figure 9, les boites de collisions sont suffisamment petites pour forcer l'utilisateur à être précis lors de l'assemblage. Comme les points n'ont pas de RigidBody, tous les signaux sont automatiquement transmis à l'objet père, ce qui permet de centraliser la gestion des collisions et de conserver un maximum le système d'assemblage déjà mis en place.

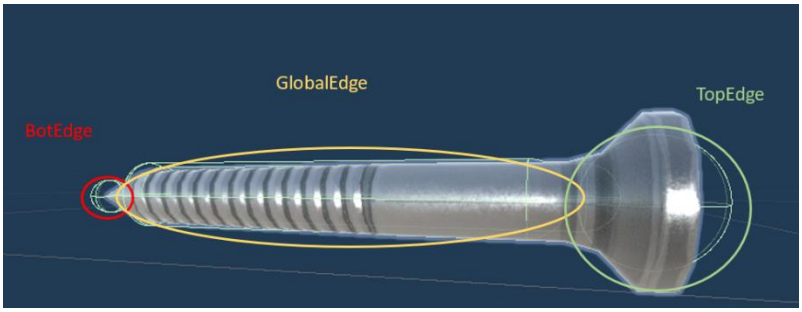

Figure 10 : Boites de collisions logiques d'une vis

Nous avons également ajouté un système de vissage. La vis a une boite de collisions* en mode physique qui sert uniquement à générer des collisions physiques, et trois boites de collisions en mode logique mises en évidence dans la figure ci-dessus :

- **BotEdge** : cette boite de collisions* située à la pointe de la vis permet de savoir si elle est plantée dans un objet.
- **TopEdge** : cette boite de collisions* en haut de la vis permet de savoir si un objet, un tournevis par exemple, est en contact avec le haut de la vis.
- **GlobalEdge** : cette boite de collisions* couvre l'ensemble de la vis entre les deux autres boites de collisions. Elle permet notamment de savoir si au moment où la pointe de la vis sort d'un objet, on a simplement retiré la vis (Body n'est pas en collision avec l'objet) ou si on l'a vissée à travers l'objet et que le bout et ressorti de l'autre coté (Body est encore en collision avec l'objet).

Pour fonctionner, la vis a besoin d'être plantée dans un objet où on peut la planter. On regarde le **Tag** de l'objet lors de la collision avec BotEdge et s'il a le tag « screwable », alors on peut planter la vis. Ensuite la vis ne bouge plus (on annule sa physique en mettant le RigidBody en « isKinematic »). Si TopEdge détecte un tournevis, alors on applique la variation de rotation du tournevis à la vis sous forme de translation. Pour choisir la direction de la translation, nous avons fait en sorte que le vecteur direction « Up » de la vis représente toujours la droite passant par BotEdge et TopEdge, on translate donc la vis suivant ce vecteur.

3.1.2 – Logique d'assemblage

La logique d'assemblage représente la façon dont doit être monté le module. Le système doit être capable d'imposer un ordre d'assemblage, c'est pourquoi nous avons utilisé une **machine d'état**. Une machine d'état est **patron comportemental*** qui sert à modéliser un **automate fini**, ainsi on peut créer un état par étapes d'assemblage, ce qui nous permettra de connaitre à tout moment où on en est dans le processus de montage et quelles sont les prochaines étapes.

La machine d'état repose sur deux scripts principaux :

- **StateMachine** : qui représente la machine d'états. Elle contient la liste des états et les administre pour connaitre les états initiaux, courants et les changements d'états.

- **State** : qui représente un état. Ce script définit les fonctions de base communes à tous les états et ajoute une classe mère commune à tous ces états, ce qui permet une meilleure gestion dans la machine d'état.

De base, le patron machine d'état ne prend qu'un seul état courant, nous l'avons donc adapté à nos besoins en transformant la variable unique d'état courant en liste d'état. Nous avons également modifié la classe State pour ajouter une liste d'états suivants, permettant de connaitre les dépendances entre les états et permettant d'obliger la finition de plusieurs étapes avant de débloquer une nouvelle étape. Grâce à cette liste d'états suivants, on peut avec un algorithme récursif simple, définir dans chaque état le nombre d'états précédents à réaliser, et décrémenter ce nombre à chaque état précédent réalisé pour savoir à quel moment une étape est réalisable.

Pour compléter la machine d'état, nous avons créé le script **Mountable,** qui permet de définir si un objet peut être utilisé dans le montage ou non.

Dans sa première version, les états de la logique d'assemblage comportaient seulement deux objets avec un script Mountable chacun : un définissant l'objet à déplacer et l'autre définissant l'objet sur lequel le premier doit être monté. Grâce à ces deux objets, l'étape peut donner à l'objet à déplacer son objet « cible ». Ainsi lorsque l'objet à déplacer entre en collision avec l'objet cible, on peut appliquer l'assemblage. L'assemblage consiste simplement à supprimer le RigidBody de l'objet à déplacer et le faire devenir enfant de l'objet cible, l'objet à déplacer devient ainsi un composant* à part entière de l'objet cible. Avec la suppression de son RigidBody, la collision de l'objet à déplacer envoi directement les informations au RigidBody de l'objet cible.

Cette logique d'assemblage fonctionne assez bien, mais elle ne permet pas de faire un certain nombre d'actions :

- Le désassemblage : ce problème n'a malheureusement pas pu être résolu. Pour désassembler un objet, il faudrait au moment où l'utilisateur attrape le montage, pourvoir savoir quel objet est attrapé précisément, et pouvoir appliquer une force différente à cet objet et au reste du montage. Le problème est que pour détecter quel objet est attrapé, il faut garder le RigidBody sur l'objet au moment de l'assemblage. Nous avons essayé cette solution en reproduisant tous les mouvements appliqués sur l'objet au reste du montage, mais cela a généré énormément de latence, au point de ne plus être supportable dans le casque.
- Avoir plusieurs pièces pouvant aller à un même point de montage : Prenons l'exemple d'une base où on peut insérer 4 montants, chaque montant peut être insérée dans chaque trou, on ne peut donc pas utiliser la logique d'étape liant un objet à une cible. Pour corriger ceci, nous avons améliorer notre "State" en transformant les deux objets en deux listes d'objets. Ainsi on a une liste comprenant les objets utilisables et une liste comportant les objets cibles. Ce système nous permet également de connaitre l'avancement d'une tache, par exemple "2/4 montants fixées".

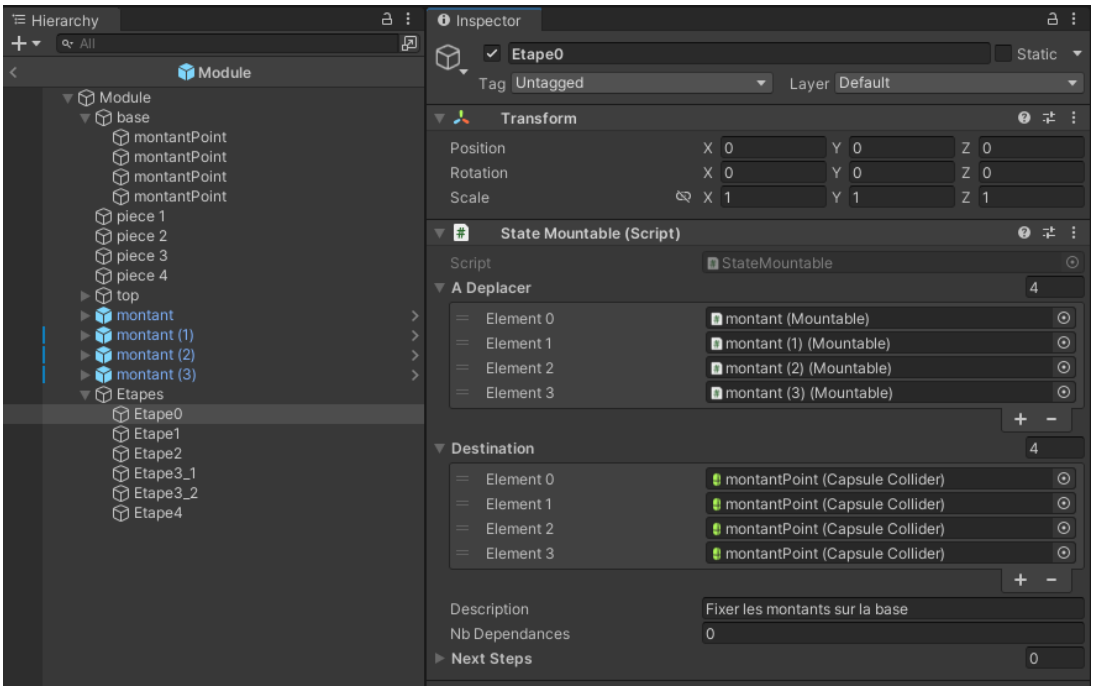

Figure 11 : Exemple d'une étape de montage

La figure ci-dessus montre l'étape prise en exemple juste avant et la vue que l'on a depuis l'Inspector. On aperçoit les différents montants et les boites de collisions que doivent atteindre les montants.

3.2 – Environnement graphique et virtuel

3.2.1 – Interface utilisateur

L'interface utilisateur est un point important de notre projet. Lors d'une formation en réalité virtuelle, il est très important d'avoir un suivi de son avancement. Cette information est fournie par les interfaces utilisateurs.

Au début, nous avions choisi d'afficher la liste des étapes suivantes à un point précis de la scène Unity. Lors de la rencontre de mi-décembre avec M. RICHARD, nous nous sommes rendu compte que ce n'était pas le plus intuitif et loin d'être le plus visible. M. MESNARD nous a conseillé de déplacer l'information en bas de l'interface utilisateur car, avec un casque nous avons plus tendance à baisser la tête. De plus lors de notre assemblage, on baisse la tête pour voir la table et les pièces.

Nous avons donc modifié notre affichage pour le transformer en un bandeau situé au niveau du bassin de l'utilisateur. Nous avons choisi de le faire suivre la position du joueur, son orientation mais pas la hauteur de son regard car cela limite trop le champ de vision.

Lors de ce changement, nous avons dû modifier quelques éléments : au départ, nous avions juste un objet de type « UI » (User Interface : interface utilisateur) dans notre scène. Mais pour que la nouvelle version suive le joueur, nous l'avons ajouté en enfant de l'objet « Player ».

Lors du déplacement du joueur, il arrivait que l'interface rentre dans le décor et ne soit donc plus visible. Pour corriger ce problème nous avons dû ajouter une caméra au Player qui ne filme que la couche UI et retire le rendu de cette couche à la caméra de base. La nouvelle camera effectue également son rendu sur le casque et on peut modifier le paramètre « Depth »[2] : on peut définir la distance entre l'objet et le rendu et donc d'afficher le layer UI très proche de la caméra et ainsi devant les autres objets.

Pour récupérer les informations à afficher, on donne simplement l'objet affichant le texte en paramètre à notre module et il s'occupe de le mettre à jour quand on avance dans le montage.

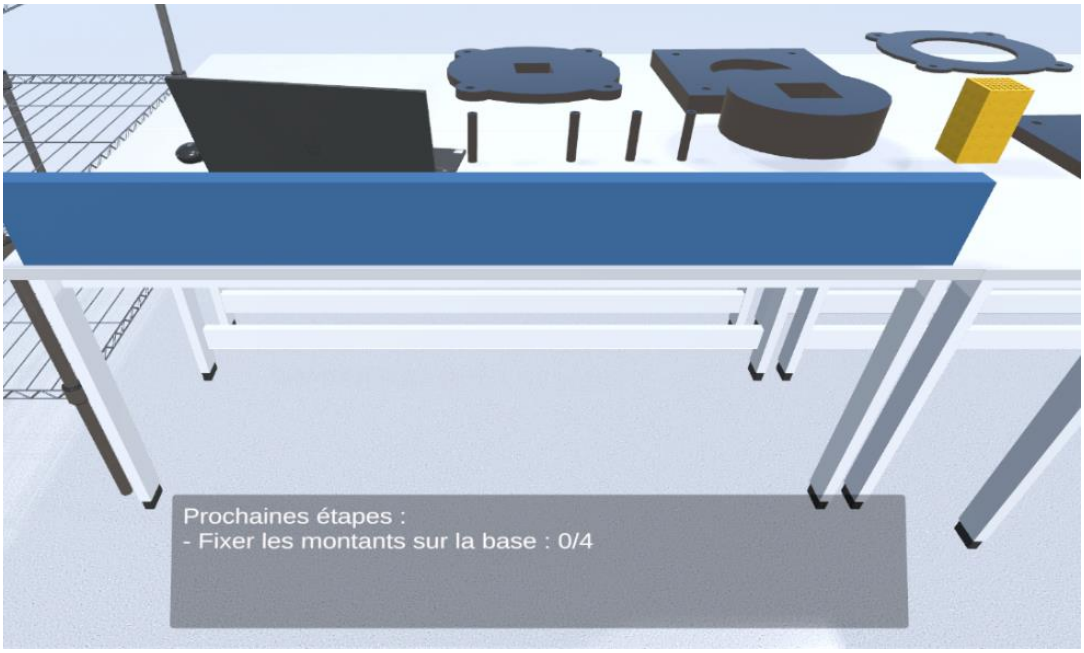

Figure 12 : Interface d'aide contextuelle en simulation

3.2.2 – Module d'assemblage

Le cœur de notre système de montage, expliqué ci-dessus, est le module de montage et donc le nouveau process intermédiaire de production que FAREVA design et met en place pour mener à bien la chaine de production de leur nouveau produit.

Lors de notre rencontre avec M. RICHARD, nous avons eu beaucoup d'informations supplémentaires sur ce nouveau produit et les méthodes de production qui seront adoptées. Le nouveau process de fabrication nécessite de l'outillage spécifique, développé, produit et assemblé spécialement pour celui-ci. C'est à ce niveau-là que la création de notre outil intervient.

Bien que tout cela soit confidentiel, nous avons pu visionner quelques images de ces outils de process, et nous nous en sommes, sans copier, grandement inspiré afin de rester le plus fidèle possible aux caractéristiques générales du montage.

Pour ce faire, nous avons dans un premier temps réalisé une première ébauche de ce module sur dessin papier. Nous avons tenu à dessiner intégralement plusieurs pièces et leurs dimensions dont une base, des emboitements, des pas de vis, un joint, un filtre et une dernière pièce se positionnant sur le dessus. Par la suite, par un souci de réalisme et de fidélité, nous avons rajouté des trous dans les pièces intermédiaires ainsi que des embouts pouvant accueillir des tuyaux de liquides dans les pièces inférieures et supérieures.

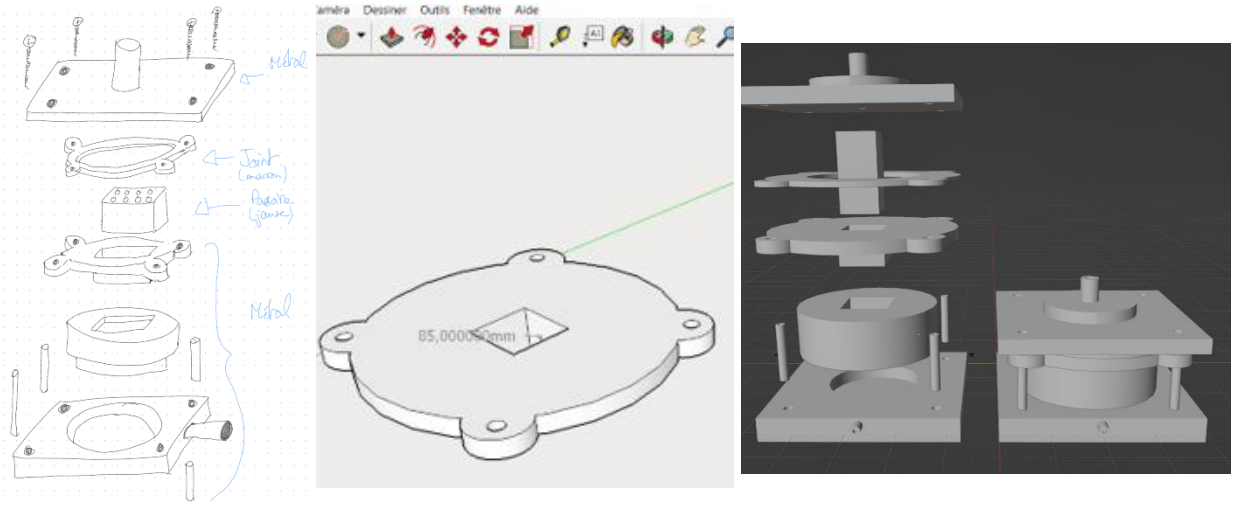

Figure 13 : Modélisation 3D du module d'assemblage

Une fois les différentes pièces dessinées et les côtes de celles-ci décidées, nous avons dans un deuxième temps réalisé la **modélisation 3D** grâce aux logiciels **Sketchup** et **Blender 3D**. Durant cette phase, nous avons dû régulièrement revoir nos dimensions et certaines formes trop complexes.

3.2.3 – Laboratoire virtuel

Maintenant la conception de notre module d'assemblage terminée, nous nous sommes attelés à la création du décor type laboratoire. Ayant également eu la possibilité de visionner quelques photos des laboratoires de FAREVA en plus de beaucoup de recherches sur internet, nous avons pu avoir une idée de l'ambiance général et du type de matériel présent dans ces environnements.

Figure 14 : Photo type de laboratoire pharmaceutique

Recréer un environnement similaire sans modèles 3D directement applicables est compliqué et demande beaucoup de temps. Nous avons donc dû rechercher ce qu'on appelle des « **Assets** ». Ce sont des ressources digitales 3D complètes pouvant être ajoutées dans un projet très simplement. Elles sont souvent composées d'un modèle 3D ou **maillage 3D** dt d'une **texture** à appliquer. Dans le cadre de Unity, il existe des fichiers « **Prefab** ». Ces Prefabs sont des assets tout en un, où la texture, l'éclairage et souvent le comportement sont déjà fusionnés et ajustés pour créer un seul objet simple.

Pour trouver ce genre de ressources, de nombreux sites « banque d'assets » existent sur internet. Sur ces sites, la majorité des modèles sont payants et peuvent couter très cher en fonction de ce que l'on désire, du niveau de finition et de complexité du maillage 3D et des textures. Les assets gratuits sont donc rares, souvent de mauvaise qualité ou dans certains cas ils nécessitent des ajustements pouvant prendre du temps comme la gestion des collisions avec les colliders.

Unity étant une solution très complète et polyvalente, le logiciel propose sa propre banque d'assets appelée « AssetStore ». Il est disponible depuis l'interface du logiciel et également depuis le web et fonctionne en lien avec notre compte Unity pour effectuer l'importation des achats au sein des projets.

Pour le projet, nous avons trouvé un package* d'assets type laboratoire très complet, qui nous a permis d'économiser beaucoup de temps. Celui-ci contient 3 dossiers :

- Un dossier pour les **pièces d'architecture** comme les murs, les sols, les fenêtre et les éclairages.
- Un autre pour les « **furnitures** » (meubles en français).
- Un dernier pour les « **props** » (accessoires en français).

Voici mis à plat sur une scène vide, tous les assets en notre possession pour la construction du décor :

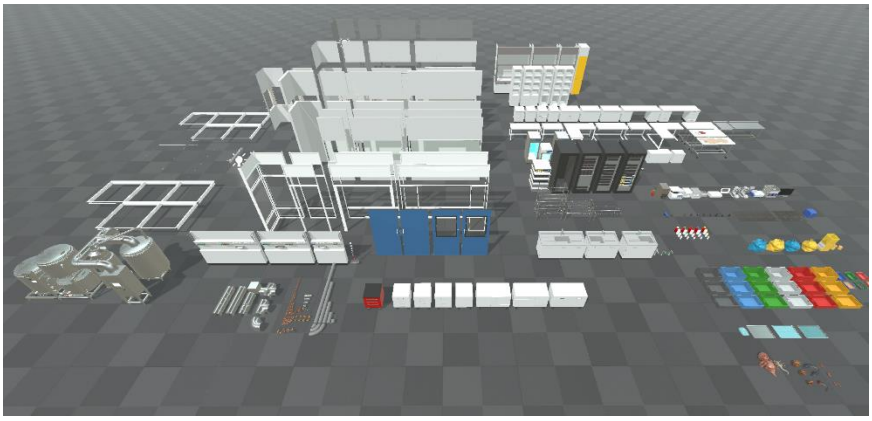

Figure 15 : Inventaire des assets décor de laboratoire

La recherche achevée, nous sommes passés à la construction de notre environnement 3D. Il a déjà fallu réfléchir aux dimensions voulues de la pièce sachant que le monde dans Unity est sans unités, les dimensions à adopter pour les différents objets sont donc relatives.

Le sol de notre laboratoire est un GameObject* de type « **Plane** » (plan) ayant la particularité de n'avoir qu'une seule face rendue graphiquement. Nos murs sont tirés de notre package* d'assets et sont également des objets plans, avec rotation et texture ajoutée. Enfin le plafond est également un objet plan, cette fois retourné afin d'être visible de dessous dans la scène virtuelle et transparent du dessus pour faciliter la vision en mode édition.

Nous avons pensé par la suite à l'agencement même de la pièce, quels meubles ajouter, en quels nombre et comment créer au sein d'une même salle des espaces à vocations différentes. Des premiers croquis ont été réalisés et nous avons effectués des premiers tests de forme sur Unity. Nous avons dû à plusieurs reprises diminuer la taille de notre pièce car nous avions mal jugé la perception qu'aurait l'utilisateur dans le casque. C'est également par soucis d'ameublement et de répétitivité que nous avons fait le choix de réduire les dimensions, même si les laboratoires réels s'avèrent beaucoup plus grands.

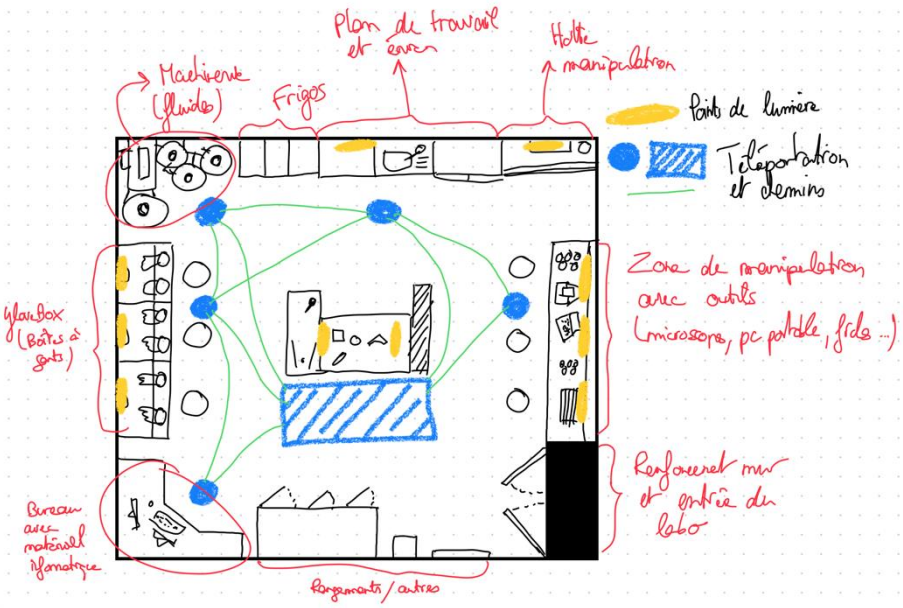

Figure 16 : Schéma de l'agencement du décor

Comme vous pouvez le voir sur ce schéma final, nous avons choisi de placer la zone de montage du module au centre de la pièce, étant l'objectif principal de cet outil de formation, autour de laquelle gravitent des zones annexes avec une utilité pharmaceutique propre. Nous retrouvons dans les labos beaucoup de tuyauterie et machines imposantes, nous avons donc disposé quelques machines grand format avec cuves de fluides dans le coin haut gauche de la pièce, en vision direct depuis le poste de montage. M. RICHARD nous ayant indiqué que certains montages ou expérimentations exigent plus de contrôle et sont souvent réalisés sous gants dans des boites sous vide appelées « glove box » disposées en ligne, nous avons fait de même dans la partie latérale gauche. Nous avons agrémenté ces réalisations par des grandes zones de manipulations avec matériel scientifique (microscopes, incubateurs, spectrophotomètres…) avec beaucoup de pièces de détail sur la partie latérale droite. Pour finir, nous avons ajouté une zone de nettoyage avec évier sur la partie haute, accompagnée d'une hotte pour produit volatils dangereux, des rangements situés dans toute la pièce et enfin un bureau d'opérateur dans le coin inférieur gauche.

Afin de donner plus de réalisme au rendu de notre laboratoire, nous avons ajoutés des systèmes de lumières, positionnés au plafond sous forme de plaques de LEDS blanche pour l'éclairage principal. En hauteur du plan de montage mais également sur les zones annexes de manipulation nous avons rajoutés des lumières de proximité sous forme d'assets de néons, permettant un éclairage local vif ainsi que des effets de réflexion sympathiques sur les textures des structures. Il a été nécessaire d'ajuster la couleur de lumière de toutes ces sources de lumières afin de ne pas avoir un rendu trop bleuté mais plus chaud, tout en respectant l'éclairage neutre de ce genre d'environnement. Dans le même principe, nous avons fait en sorte par leur disposition, que ces éclairages ne génèrent pas d'ombres.

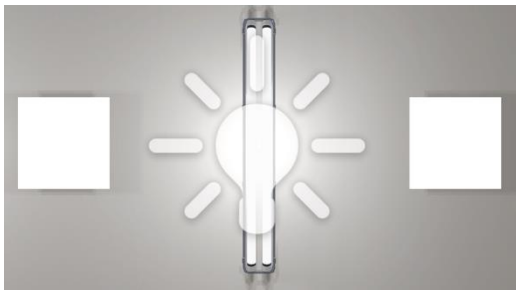

Figure 17 : Sources de lumière virtuelle

Par le nombre d'éléments de structure et d'éléments décoratifs, nous avons voulu allier l'utile au réalisme. Positionner le montage au centre de la pièce permettra à l'utilisateur sous casque de ne pas se sentir à l'étroit, en ayant dans son champ de vision fixe les autres parties de la pièce. Grâce au degré de liberté qui lui sera offert, et s'il souhaite se mouvoir physiquement il pourra en découvrir de nouvelles et regarder de plus près les détails du décor.

III – Résultats et améliorations

1 – Logique d'interaction et d'assemblage

Pour conclure sur les mécaniques d'interaction et d'assemblage, l'utilisation des boites de collisions, en collision logique dans notre système, permet d'avoir des points de montage assez précis rendant l'assemblage plus réaliste qu'au début où nous avions uniquement les boites de collisions des objets. Le fait de modifier le parent des pièces lors de l'assemblage permet d'avoir un comportement cohérent, car le montage devient un unique objet comme dans la vie réelle. Cependant, cela ne permet pas d'avoir une logique de désassemblage, à cause des conflits créés par les interactions entre les Colliders, les RigidBody et les scripts qualifiants les objets d'attrapables. Voici comment l'utilisateur est immergé dans le casque lors de la phase de montage :

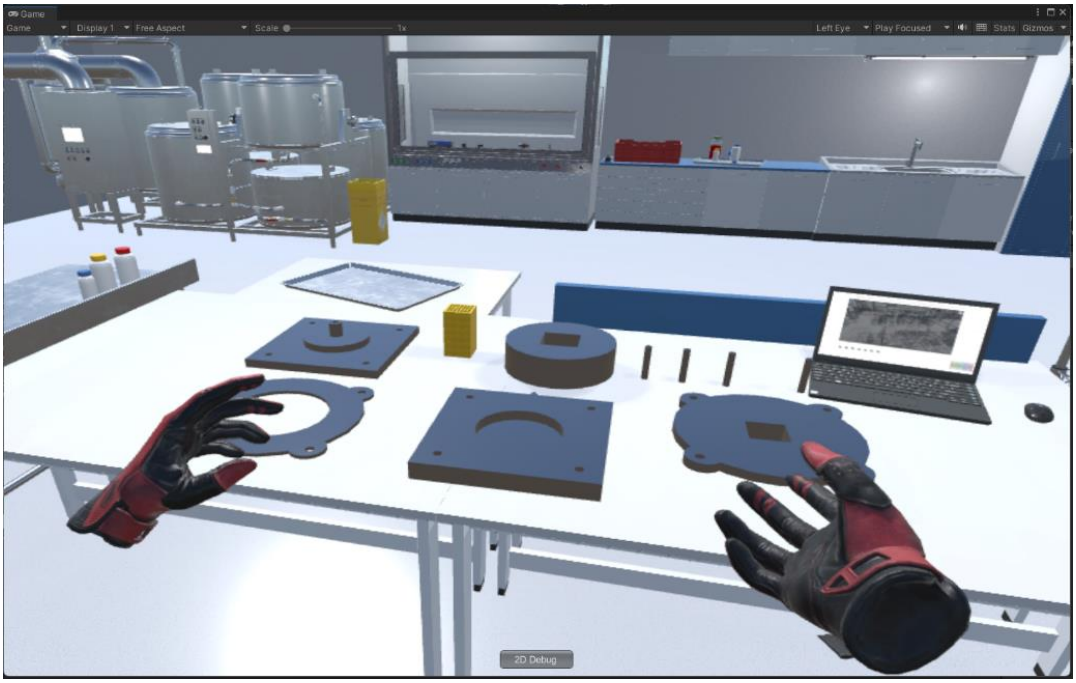

Figure 18 : Immersion dans le casque lors du montage

2 – Environnement

Concernant le décor de notre outil de formation, comme expliqué précédemment, nous avons passé du temps à concevoir et réaliser un environnement aussi réaliste et crédible que possible. Après ajustements des objets 3D, des meubles, des textures et des sources de lumière, le résultat se présente ainsi :

Figure 19 : Vue plan de montage du décor

Figure 20 : Vue globale du dessus du décor

Comme vous pouvez le voir, nous sommes restés fidèles à notre conception initiale, avec finalement peu de changements mais des ajustements. On remarque bien les différentes zones annexes que voici en détail :

Figure 21 : Vue des boites de manipulation à gants

Ici la zone des « glove box », présentes au nombre de 3. Un éclairage de proximité sous forme de néon est disposé juste au-dessus et nous avons ajouté un pan de mur bleu afin de proposer un contraste visuel pour plus de confort.

Figure 22 : Vue zone de manipulations scientifiques

Ici vous pouvez observer la zone réservée aux manipulation complexes de laboratoire. Des machines comme des microscopes, des incubateurs et autres sont disposées sur des paillasses, avec des détails de prises électriques et de robinets d'appoints.

Figure 23 : Vue machinerie et zone de nettoyage du décor

Enfin nous avons ici la zone avec machinerie et cuves de fluides, avec à leur droite la hotte et ensuite la zone de nettoyage avec évier pour finir par des rangements et des ensembles réfrigérants.

A l'heure actuelle, n'ayant pas encore pu rencontrer à nouveau M. MESNARD et M. RICHARD pour le projet, nous n'avons pas eu de retours sur la partie environnement 3D. Bien que nous ayons meublé la pièce de façon variée grâce aux supports en notre possession, il reste de l'espace non utilisé et certains détails ne sont peut-être pas des plus réalistes et cohérents. Il est évident que certaines modifications seront nécessaires en fonction de ces retours.

3 – Idées non implémentées

Nous avons passé beaucoup de temps à essayer de développer des idées qui malheureusement n'ont pas pu être implémentées dans le projet final.

3.1 – Main plus réalistes

Nous avons notamment passé plus de 20 heures à essayer de créer une main plus réaliste. Nous ne sommes pas satisfaits de la façon dont la main attrape les objets, ce n'est pas réaliste et impacte la précision des mouvements de l'utilisateur.

Nous avons tout d'abord essayé de modifier le point d'accroche des objets mais ce point est unique. Nos parties de modules étant assez grandes, si nous voulons être réaliste il faudrait différents points suivant la position de la main pour les attraper. Nous avons essayé de déplacer le point d'accroche de l'objet dynamiquement en fonction d'où se situe la main et en faisant un lancer de rayon sur l'objet depuis le centre de la main. Malheureusement cela a compliqué énormément la maniabilité car l'objet avait une rotation assez imprévisible et il fallait s'y prendre à plusieurs fois pour avoir une rotation utilisable.

Nous sommes donc passé sur la technologie « Auto-Hand » trouvée sur Internet et disponible sur l'Asset Store de Unity. Elle permet de générer plusieurs points d'accroche sur un objet et utilise la génération d'animation procédurale pour calculer la position des doigts. Cette main donnait vraiment un rendu très réaliste, mais elle utilisait des classes propres au package* et malgré énormément d'efforts pour créer une interface entre le package* et notre logique d'interactions nous n'avons pas réussi à obtenir de résultats satisfaisants.

3.2 – Meilleurs retours d'informations

Comme dit plus haut dans la partie dédiée à l'interface utilisateur, il est important d'avoir beaucoup de retours lors de l'assemblage. Nous avions prévu de faire une prévisualisation de la position exacte des pièces lors de l'assemblage, en affichant un modèle « fantôme » à la bonne position par exemple. Ainsi l'utilisateur pourrait avoir une idée d'où et comment agencer les pièces. Une extension de cette idée est de calculer la différence de position et de rotation entre la prévisualisation et la position de la pièce posée par l'utilisateur et ainsi de pouvoir quantifier la qualité du montage et afficher les informations à l'utilisateur.

3.3 – Coulissement et vissage

Ces deux fonctionnalités avaient pour but de créer un assemblage plus complet et réaliste. Malheureusement, malgré les nombreuses heures passées dessus, le coulissement n'a pas donné de résultats convenables. Notre idée se basait sur la même idée que la logique d'assemblage, mais au lieu

de supprimer le RigidBody de l'objet à la collision, on ajoute une étape intermédiaire qui verrouille le déplacement et la rotation libre de la pièce et qui l'oblige à ne bouger autour d'un seul axe.

Ceci est facile à faire dans le cas de la sélection par souris, car il suffit de projeter le déplacement qu'on applique sur l'objet sur l'axe choisi, mais cela devient beaucoup plus compliqué dans le cas d'une sélection par main virtuelle, car que ce soit par Steam VR ou XR interaction, nous n'avons pas accès à la force qu'applique le mouvement de la main à l'objet et donc nous ne pouvons pas le limiter.
Pour le vissage, nous avons réussi à créer cette fonctionnalité, comme expliqué plus haut, cependant nous n'avons pas eu le temps de modifier la logique d'assemblage pour l'implémenter. La solution envisagée était d'ajouter un paramètre à notre étape qui définissait si l'étape était un emboitement ou un vissage. Dans le cas du vissage on vérifiait si la vis est entièrement vissée et si elle lie les différents objets en regardant si ces objets sont dans la liste de la vis.

4 – Améliorations possibles

Les améliorations possibles pour notre projet concernent évidemment les deux aspects principaux de celui-ci : la logique d'interactions et de scénarisation ainsi que l'environnement virtuel* immersif.

D'abord, nous pourrions améliorer la logique d'interactions en recréant nous même les mains virtuelles, cela nous permettrait un contrôle total sur la gestion des déplacements d'un objet attrapé et donc nous pourrions plus facilement implémenter le coulissement d'une pièce sur une autre. Ce contrôle des déplacements permettrait également de mieux gérer les déplacements d'un module composé de plusieurs pièces, et possiblement permettre d'appliquer les déplacements à l'ensemble du module sans supprimer les RigidBody lors de l'assemblage, ce qui permettrait d'envisager une logique de désassemblage.

Dans un second temps, le décor de notre scène peut tout à fait être modifié ou réinventé. Nous sommes conscients que l'environnement que nous avons produit peut manquer de cohérence ou de réalisme. Nous pourrions créer des paillasses plus grandes et plus complètes avec des outils supplémentaires, des bureaux avec plus de détails et d'éléments avec lesquels on peut interagir ou bien encore changer les textures et ajuster les lumières pour encore plus d'effet et de réalisme sans pour autant oublier le souci de latence de calcul pour une solution en réalité virtuelle.

Enfin, nous avons voulu améliorer la simulation en implémentant Steam VR, ce qui permettrait une meilleure portabilité (utilisable sur plusieurs types de casques). Cela nous aurait également permis d'ajouter rapidement plusieurs fonctionnalités déjà existantes dans le package. Malheureusement, l'importation a causé de gros problèmes de compatibilité avec le package* utilisé jusqu'à présent et à rendu notre projet inutilisable. Malgré nos tentatives, nous n'avons pas réussi à revenir sur une précédente version, nous avons donc demandé exceptionnellement à M. MESNARD de rendre la partie code de notre projet un peu après la deadline pour pouvoir corriger ce problème.

Conclusion

Pour conclure, notre objectif pour ce projet était de concevoir un outil de formation pharmaceutique en réalité virtuelle et immersive, conçu avec Unity 3D, développé avec le langage C# et testé sur casque Occulus Rift. Ce projet, servant de preuve de concept pour l'entreprise FAREVA, est dédié à la formation d'opérateur de montage dans un environnement virtuel. Il est composé de plusieurs éléments : un environnement laboratoire pharmaceutique réaliste facilitant l'immersion de l'utilisateur et d'un module à assembler en suivant des étapes définies par des scripts. Conformément aux objectifs fixés, il est facile de modifier l'ordre d'assemblage et les éléments pouvant être assemblés.

La mise en place de notre système d'interaction et de scénarisation a été complexe à mettre en œuvre, du fait des multiples directions possibles de développement et des conflits entre les différentes fonctionnalités et contenus fournis. C'est en faisant preuve de volonté et de persévérance tout au long du projet, que nous pouvons avec du recul lister les point positifs et négatifs de notre travail, relativiser sur les choix que nous avons fait ainsi que sur les améliorations possibles. Il serait intéressant d'améliorer l'immersion en affinant le réalisme du montage et en modélisant le corps complet de l'utilisateur dans la simulation. Nous avons également pensé aux possibles extensions de la réalité virtuelle au-delà de la formation, comme la simulation des réactions chimiques dans les machines.

En réalisant ce projet de A à Z, nous avons travaillé sur de nombreux domaines, dont certain que nous n'avions pas envisagé de traiter. Nous avons également énormément appris sur le fonctionnement d'une simulation en réalité virtuelle : la nécessité de la fluidité, les mouvements naturels et comment les utiliser, ainsi que les limites que cela impose comme les mouvements et interactions limités.

Nous espérons que notre travail constitue une première preuve de concept assez conséquente et qu'elle répondra aux attentes de l'entreprise et de M. RICHARD. Les perspectives d'améliorations sont nombreuses, le domaine d'application vaste et nous sommes persuadés que ces nouveaux outils de formation se démocratiseront bientôt au sein de FAREVA et ailleurs.

Références bibliographiques

[1] D. MA, J. GAUSEMEIER, X. FAN, M. GRAFE, *Virtual reality & augmented reality in industry*, Printemps 2011 [Online].
Disponible sur : http://ndl.ethernet.edu.et/bitstream/123456789/17387/1/474.pdf
[Consulté le: 13-dec-2022]

[2] Unity Technologies, *Documentation Unity 2021.3*, Avril 2022 [Online].
Disponible sur : https://docs.unity3d.com/Manual/index.html.
[Consulté le : 25-fév-2023]

Lexique

Boite de collisions : Zone permettant de détecter la présence d'une autre boite de collision.

Composant (Unity) : Élément permettant d'ajouter des caractéristiques à un GameObject modifiant son comportement lors de la simulation.

Environnement virtuel : Mondes artificiels plus ou moins réaliste créer et gérer par un ordinateur. Des utilisateurs peuvent y interagir un peu comme dans la réalité.

Formation virtuelle : Forme d'apprentissage où l'apprenant est plongé dans un environnement virtuel où il est formé à des gestes visant à être reproduit dans la réalité.

GameObject : Élément de base de Unity, il constitue la base de tout objet créé dans une simulation sous Unity.

Package (Unity) : Ensemble de GameObjects, composants et ressources Unity pouvant être importés dans différents projets.

Patron comportemental : Meilleure solution connue à un problème de conception récurrent lié à l'interaction entre les classes.

Prefab (Unity) : GameObject contenant différents GameObjects pouvant être importés dans différentes simulations et dont les modifications effectuées dessus seront appliquées sur leur version dans les simulations.

Réalité virtuelle ou immersive : Simulation de la présence physique d'un utilisateur dans un environnement artificiellement généré par des logiciels. La réalité virtuelle (ou immersive) crée un environnement avec lequel l'utilisateur peut interagir, elle reproduit donc une expérience sensorielle.

Tag (Unity) : Attribut d'un GameObject permettant de le caractériser.

Transform (Unity) : Composant Unity définissant la position, la rotation et l'échelle d'un GameObject dans la simulation.

Unity : Moteur de jeu développé par Unity Technologies. Il permet de créer un environnement virtuel et simuler la physique de cet environnement pour immerger un utilisateur dans cet environnement au travers d'un casque de réalité virtuelle.