Master of Science HES-SO in Engineering

Av. de Provence 6

CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation : Technologies de l’information et de la communication (TIC)

TITRE DE VOTRE TRAVAIL DE MASTER

Fait par

Quentin Forestier

Sous la direction de

Prof. Yassin Aziz, REKIK

Dans le laboratoire/institut de/ à l’école HEPIA

[Expert externe (Prénom et nom, titre de l’expert lors de la défence du projet de traval de Master)]

Lieu, HES-SO//Master, 2024

Accepté par la HES‑SO//Master (Suisse, Lausanne) sur proposition de

Prof. Yassin Aziz REKIK, conseiller de travail de Master

[Xyz, Expert principal]

Lausanne, le … 20 ??

|  |  |
| --- | --- |
| Prof. xxx  Conseiller |  |

# Table des matières

[Table des matières iii](#_Toc156228635)

[Remerciements vii](#_Toc156228636)

[Nomenclature ix](#_Toc156228637)

[Résumé x](#_Toc156228638)

[1. Introduction 1](#_Toc156228639)

[1.1. Cahier des charges 1](#_Toc156228640)

[1.2. Planification du projet 1](#_Toc156228641)

[1.2.1. Réseau 1](#_Toc156228642)

[1.2.2. Implémentation de la structure des scénarios 1](#_Toc156228643)

[1.2.3. Gestion de scénario intelligente 2](#_Toc156228644)

[1.2.4. Implémentations de la structure des influenceurs 2](#_Toc156228645)

[1.2.5. Mise en place des interactions 2](#_Toc156228646)

[1.2.6. Implémentation des scénarios existants 2](#_Toc156228647)

[1.2.7. Monitoring des simulations 2](#_Toc156228648)

[1.2.8. Génération du feedback 3](#_Toc156228649)

[1.2.9. Ancrage du monde virtuel 3](#_Toc156228650)

[1.3. Organisation du projet 3](#_Toc156228651)

[2. Analyse de l’existant 5](#_Toc156228652)

[2.1. Le projet HOST 5](#_Toc156228653)

[2.1.1. Application de modération 5](#_Toc156228654)

[2.1.2. Application HoloLens 2 5](#_Toc156228655)

[2.1.3. Limitations connues 6](#_Toc156228656)

[2.2. Couche réseau 6](#_Toc156228657)

[2.2.1. Netcode for GameObject 6](#_Toc156228658)

[2.2.2. Assets Unity 7](#_Toc156228659)

[2.2.3. Home made 7](#_Toc156228660)

[2.3. Scénario 7](#_Toc156228661)

[2.4. Indices et perturbateurs 7](#_Toc156228662)

[2.5. Casque de réalité augmentée 8](#_Toc156228663)

[2.6. Monitoring 8](#_Toc156228664)

[3. Technologies 9](#_Toc156228665)

[3.1. Casque de réalité augmentée 9](#_Toc156228666)

[3.1.1. HoloLens 2 9](#_Toc156228667)

[3.1.2. Meta Quest Pro 9](#_Toc156228668)

[3.1.3. Meta Quest 3 10](#_Toc156228669)

[3.1.4. Choix effectué 10](#_Toc156228670)

[3.2. Version d’Unity 10](#_Toc156228671)

[3.2.1. Unity 2020 LTS 10](#_Toc156228672)

[3.2.2. Unity 2021 LTS 10](#_Toc156228673)

[3.2.3. Unity 2022 LTS 10](#_Toc156228674)

[3.2.4. Unity 2023 11](#_Toc156228675)

[3.2.5. Choix effectué 11](#_Toc156228676)

[4. Conceptions et implémentations 13](#_Toc156228677)

[4.1. Couche réseau 13](#_Toc156228678)

[4.1.1. FMETP\_Stream 13](#_Toc156228679)

[4.1.2. Host Network Manager 13](#_Toc156228680)

[4.1.3. Host Network Object 13](#_Toc156228681)

[4.1.4. Host Network RPC 14](#_Toc156228682)

[4.1.5. Outils utilitaires 14](#_Toc156228683)

[4.2. Scénario 14](#_Toc156228684)

[4.2.1. Réflexion 14](#_Toc156228685)

[4.2.2. Structure 14](#_Toc156228686)

[4.2.3. Scénarios 16](#_Toc156228687)

[4.2.4. Énigmes 16](#_Toc156228688)

[4.2.5. Eléments d’une énigme 17](#_Toc156228689)

[4.3. Gestion intelligente du jeu 18](#_Toc156228690)

[4.3.1. Réflexion 18](#_Toc156228691)

[4.3.2. Scenario Manager 18](#_Toc156228692)

[4.3.3. Paramètres de scénario 18](#_Toc156228693)

[4.3.4. Calculateur d’indice d’influence 18](#_Toc156228694)

[4.4. Influenceurs 18](#_Toc156228695)

[4.4.1. Structure 18](#_Toc156228696)

[4.4.2. Indice 18](#_Toc156228697)

[4.4.3. Perturbateur 18](#_Toc156228698)

[4.4.4. Récepteur d’influence 18](#_Toc156228699)

[4.5. Interactions 18](#_Toc156228700)

[4.6. Monitoring 18](#_Toc156228701)

[4.6.1. Live Stream 18](#_Toc156228702)

[4.6.2. Interface 18](#_Toc156228703)

[4.7. Génération de feedback 18](#_Toc156228704)

[4.8. Ancrage du monde virtuel 18](#_Toc156228705)

[5. Problèmes rencontrés 19](#_Toc156228706)

[6. Conclusion 21](#_Toc156228707)

[7. Bibliographie 23](#_Toc156228708)

[8. Liste des figures 24](#_Toc156228709)

[9. Annexes 25](#_Toc156228710)

# Remerciements

Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper.

# Nomenclature

|  |  |
| --- | --- |
| RPC | Remote Procedure Call, une méthode appelée sur un hôte distant |
| BBB | 2em lettre de l’alphabète |
| CCC | …………….. |
| DDD | …………….. |
| EEE | ……………… |
| FFF | ……………… |
| GIC | ……………… |
| FAD | ……………… |
| IR | ……………… |
| KOH | ………………. |
| MLR | ………………. |
|  |  |
|  |  |
|  |  |

# Résumé

Key Words: Key, word.

# Introduction

## Cahier des charges

L'objectif de ce projet est de concevoir et développer une application d'énigmes en réalité augmentée, en partant de la solution existante (HOST). Actuellement, la solution HOST comprend deux scénarios jouables via une application de modération et une application dédiée aux HoloLens 2. Les énigmes sont conçues pour encourager le travail en équipe en combinant des indices numériques et réels.

Pour optimiser la flexibilité du projet, il est nécessaire de créer une application générique, permettant une évolution aisée. Étant donné qu’HOST présente une faible modularité, l'objectif est de concevoir une solution moins contraignante.

De plus, il est requis de décharger le maître de jeu de la responsabilité des indices et perturbateurs en intégrant un outil de monitoring automatique, voire intelligent, imaginé et implémenté dans le cadre de ce projet.

Compte tenu de l'annulation du développement du HoloLens 3 par Microsoft, il est essentiel de réfléchir à la sélection judicieuse du matériel à utiliser pour les prochaines phases du projet.

En conclusion, il est demandé de valider l'ensemble en implémentant les deux scénarios préexistants afin de démontrer les capacités du programme, tout en certifiant que toutes les fonctionnalités ont été prises en compte et que rien n'a été omis.

## Planification du projet

Le projet s'étale sur une période de vingt semaines, décomposé en neuf points clés afin de répondre aux exigences du cahier des charges. Idéalement, chaque point devrait être achevé en deux semaines maximum. En envisageant deux semaines supplémentaires, cela offre une marge de manœuvre pour résoudre d'éventuels problèmes.

Conjointement avec le responsable du projet, M. Rekik, il a été convenu qu'à la fin de chaque période de deux semaines, une présentation fonctionnelle du travail serait réalisée. Ceci permet de valider chaque étape avant de passer à la suivante.

### Réseau

La couche réseau occupe probablement une place prépondérante, étant le fondement du projet tout entier. Cela implique la recherche de la technologie réseau la mieux adaptée. Il est impératif que les participants puissent se connecter entre eux, et idéalement, que l'application des joueurs puisse s'auto-connecter au serveur sans nécessiter une saisie manuelle d'adresse IP ou autre.

Le transfert du flux vidéo des joueurs vers le maître du jeu intensifie les échanges.

Afin d'améliorer l'expérience utilisateur, il serait judicieux de synchroniser la position des objets entre les participants, ce qui rendrait les interactions plus crédibles et immersives.

En résumé, cela nécessite donc :

- L'implémentation de RPC,

- La synchronisation des objets,

- Une découverte automatique des serveurs,

- La gestion du flux vidéo.

### Implémentation de la structure des scénarios

Uniformiser la structure des différents scénarios présente l'avantage de réduire considérablement la charge de travail tout en facilitant la création de nouveaux scénarios. Il est crucial d'étudier et comprendre cette structure pour choisir l'architecture appropriée et comprendre pourquoi elle est la meilleure option. Dans la mesure où le projet vise initialement à être plus modulable, l'architecture doit être à la hauteur de cette ambition, favorisant un développement simplifié. L’idée étant d’offrir une solution a un comportement courant, sans pour autant la rendre spécifique.

Il est important de garder à l'esprit que la structure des scénarios doit s'aligner sur la couche réseau discutée précédemment, en tirant le meilleur parti de ses avantages sans pour autant les rendre étroitement dépendants.

### Gestion de scénario intelligente

Le cahier des charges spécifie la nécessité de décharger le maître du jeu, lui permettant ainsi de se concentrer pleinement sur la prise de notes. La mise en place d'une gestion de scénario intelligente vise à éliminer la responsabilité du maître du jeu concernant l'envoi d'indices et de perturbateurs.

Il est impératif que ces éléments soient envoyés automatiquement, en fonction de la progression des joueurs dans l'énigme en cours et dans l'ensemble du scénario. Pour ce faire, une analyse exhaustive des informations, notamment disponibles grâce à la structure du scénario, sera nécessaire afin d'extraire les données pertinentes.

Ensuite, il sera crucial de travailler ces données pour leur donner un sens, facilitant ainsi la prise de décision et l'orientation appropriée dans le déroulement du scénario.

### Implémentations de la structure des influenceurs

Les influenceurs représentent à la fois les indices et les perturbateurs, pouvant revêtir des formes textuelles, imagées, vidéo, sonores, visuelles, voire une combinaison de plusieurs catégories. Il sera nécessaire d'identifier les similitudes entre les divers influenceurs afin d'extraire leurs caractéristiques, facilitant ainsi la création d'une structure générique.

### Mise en place des interactions

Une fois le casque de jeu sélectionné, il sera nécessaire d'explorer les interactions possibles et d'en extraire le maximum afin de les rendre conviviales pour le processus de création d'un scénario. De plus, ces interactions doivent être aussi naturelles que possible, éliminant ainsi la nécessité pour les joueurs de réfléchir à la manipulation des objets lors de la résolution d'une énigme.

La mise en place d'un tutoriel permettant d'instruire les joueurs sur la réalisation de tous les types d'interactions serait idéale.

### Implémentation des scénarios existants

Cette phase implique l'intégration des scénarios du projet précédent dans le nouveau. En utilisant les éléments créés au cours des étapes antérieures du projet, la recréation des scénarios devrait s'effectuer de manière simple. La validation de ces scénarios confirmerait ainsi le bon fonctionnement et la généricité des étapes précédentes.

### Monitoring des simulations

Cette étape implique la création de plusieurs interfaces destinées au maître du jeu. La première interface doit permettre de sélectionner un scénario et de modifier ses paramètres. La principale interface de jeu doit inclure la vue en streaming des casques des joueurs ainsi qu'un espace dédié à la prise de notes. De plus, elle doit offrir la possibilité d'envoyer des messages aux joueurs.

Enfin, une interface de fin de simulation doit faciliter le débriefing avec les joueurs en affichant les performances sur chaque énigme et également d’ouvrir les rapports PDF et vidéos générés.

### Génération du feedback

Il sera nécessaire d'enregistrer une vidéo capturant la zone principale de l'interface du maître de jeu pour créer un récapitulatif incluant les commentaires. De plus, la création d'un rapport PDF est requise. Chaque commentaire sera accompagné de son image correspondante extraite de la vidéo, présentée dans le document.

### Ancrage du monde virtuel

Étant donné que ce projet se situe dans le domaine de la réalité augmentée, il est crucial de parvenir à une superposition harmonieuse entre le monde réel et virtuel. Pour ce faire, il est nécessaire d'analyser les possibilités offertes par le casque choisi, de les comparer, et de sélectionner celles qui semblent les plus prometteuses.

Cependant, il convient de garder à l'esprit que la restriction des actions des joueurs en ce qui concerne cette configuration simplifiera considérablement l'accessibilité de l'application.

## Organisation du projet

Le code du projet est accessible sur GitHub via le lien suivant : https://github.com/Host-Project/HostProject2.0. Contrairement à son prédécesseur, une décision a été prise en faveur de l'utilisation d'un unique projet pour les applications AR et de monitoring, conduisant ainsi à la création d'un mono référentiel (mono repository).

Le suivi du projet est réalisé à l'aide d'un kanban et des issues GitHub. Le dossier GitHub est intégré dans une organisation GitHub, n'étant ainsi pas lié à un compte spécifique. Cela simplifiera le transfert des droits aux développeurs suivants.

# Analyse de l’existant

## Le projet HOST

### Application de modération

L'application Windows développée pour la modération du jeu propose plusieurs fonctionnalités cruciales. En premier lieu, elle permet l'envoi de messages personnalisés aux participants, favorisant ainsi l'interaction et la transmission d'informations ou d'instructions spécifiques. De plus, elle offre la possibilité d'incorporer des perturbations dans le jeu pour enrichir l'expérience et la rendre plus immersive.

Une autre fonctionnalité majeure de l'application est la capacité à fournir des indices aux participants au besoin, les aidant à avancer dans les énigmes ou à surmonter des obstacles. Ces indices peuvent être envoyés de manière ciblée aux joueurs qui en ont besoin.

L'application permet également la visualisation du flux vidéo en direct provenant des HoloLens portés par les participants. Cette fonctionnalité permet aux modérateurs de suivre les actions des joueurs, d'observer leur progression et de comprendre leur point de vue. Ces vidéos, agrémentées de commentaires, sont enregistrées et peuvent être consultées ultérieurement lors des séances de débriefing.

Dans le but de faciliter le débriefing, l'application génère un fichier PDF associant des paires d'images extraites des vidéos aux commentaires correspondants. Cette approche permet une analyse approfondie de l'expérience de jeu, facilitant ainsi les discussions et les évaluations post-session.

En dernier lieu, l'application joue le rôle de serveur central auquel les HoloLens 2 se connectent. Elle supervise les sessions de jeu et les différentes étapes, garantissant la synchronisation entre les participants et le bon déroulement de l'expérience. Il est important de noter que chaque scénario a son application dédiée, ce qui signifie qu'il n'est pas possible de choisir le scénario directement dans l’application.

### Application HoloLens 2

L'application HoloLens 2 constitue la plateforme sur laquelle se déploie le scénario du jeu. Elle se connecte avec le serveur, à savoir l'application de modération, pour échanger des données et garantir la synchronisation entre les participants.

Cette application est responsable de la visualisation du Game World, l'environnement virtuel dans lequel les joueurs évoluent. Elle offre aux participants la possibilité de voir et d'interagir avec les divers objets et éléments du jeu.

L'application HoloLens 2 gère également les interactions physiques avec les objets virtuels en utilisant les capteurs et les fonctionnalités spécifiques des HoloLens 2. Cela lui permet de détecter les mouvements et les gestes des joueurs, leur permettant ainsi d'effectuer des actions telles que saisir, déplacer ou manipuler des objets virtuels.

Durant le déroulement du jeu, l'application HoloLens 2 communique avec le serveur pour mettre à jour sur la progression des énigmes. Lorsqu'une énigme est résolue, elle envoie un signal au serveur pour indiquer que l'étape est achevée, déclenchant éventuellement la mise à disposition de nouvelles informations ou énigmes.

Il est essentiel de noter que, de la même manière que pour l'application de modération, chaque scénario est associé à une application distincte. Cela signifie qu'il est nécessaire de changer d'application pour passer d'un scénario à un autre.

En résumé, l'application HoloLens 2 constitue l'interface immersive où se déroule le scénario du jeu. Elle assure la communication avec le serveur de modération, affiche le Game World, gère les interactions avec les objets virtuels et informe le serveur de la progression des énigmes.

### Limitations connues

#### Difficulté à créer des nouveaux scénarios

Comme décris plus haut, le projet est fait pour un seul et unique scénario. Pour en avoir un deuxième, il a fallu cloner le projet entier, et de modifier ensuite selon les besoins. On a donc deux projets similaires, se basant sur les mêmes technologies et logiques mais n’ayant aucun lien entre eux. Ce qui veut également dire que si un des projets est mis à jour, il faudra refaire le travail sur le deuxième.

Un autre problème lié est la difficulté d’avoir un fil conducteur. Les joueurs sont lancés dans la simulation avec des indications sur qu’elle devrait être leurs prochaines actions. Cependant, aucun fil conducteur n’est présent. Les joueurs ont la possibilité d’interagir et de terminer toutes les différentes énigmes dans l’ordre qu’ils le souhaitent. Cela peut être une bonne, mais également un problème si l’on souhaite avoir un scénario un peu plus linéaire.

Dans le cas du deuxième scénario créé, une linéarité a été introduite, mais au prix d’un travail chargé et très peu réutilisable. Ce qui veut dire que, si l’on devait créer un troisième scénario, cela serait tout autant compliqué que pour le deuxième.

#### Multiples connexions TCP et UDP sont utilisées

Le serveur et les clients ont chacun plusieurs connexions TCP et UDP actives. En effet, une première connexion via un Asset Unity (FMETP\_STREAM, qui utilise UDP) est utilisé pour avoir de la découverte automatique. Cette connexion n’est pas fermée par la suite, et n’est pas non plus utilisée pour les échanges.

Une connexion TCP "self made" est en place pour les échanges clients/serveurs. Elle utilise l’IP découverte par la connexion UDP afin de se connecter au serveur.

#### Inscriptions des méthodes RPC

Les RPC sont également faits « à la main ». C’est-à-dire qu’ils sont en fait des messages JSON, incluant un ID, un nom de méthode et des paramètres. Ce message est ensuite traduit à l’arriver, et utilisant la réflexion fournie par le C#, il est possible d’invoquer une méthode sur un objet à partir de son nom.

Cependant, il est nécessaire d’avoir inscrit la méthode au préalable. Ce qui veut dire qu’un script doit contenir une liste de méthodes, ayant chacune un ID et que cette même liste soit présente, avec les mêmes ID dans un script de l’autre projet.

Ce n’est pas une méthode fiable étant donné qu’il est très facile de modifier à un endroit en oubliant de le faire dans l’autre projet.

#### Taille du monde virtuelle

Les différentes simulations ayant un monde virtuel fixe, il peut parfois être plus grand que la place disponible dans le monde réel. Étant donné que les joueurs ne peuvent pas se déplacer dans le jeu avec l’aide de Joystick, cela peut être un problème car ils pourraient ne pas avoir accès à tous les endroits.

## Couche réseau

### Netcode for GameObject

Netcode for GameObject est la solution propriétaire d’Unity. Cette solution possède une documentation conséquente et facile à mettre en place. Elle est cependant possédée d’une multitude de fonctionnalité qui ne sont pas utile au projet, tout du moins dans un futur proche. C’est donc une technologie facile à utiliser, mais compliqué à maitriser.

L’architecture de cette solution est Client Serveur. Le serveur commande tout, et les clients lui font simplement des demandes. Ce qui veut dire que l’espace de jeu doit également être disponible sur le serveur alors que dans le cas de ce projet, il n’en a pas forcément besoin.

Le grand point fort de cette solution est qu’il facilite grandement les RPC. En effet, il suffit simplement de décorer une méthode pour que celle-ci deviennent utilisable depuis une autre machine.

Cependant, il n’est pas possible d’effectuer de la découverte de serveur automatique. Il serait alors nécessaire de faire entrer l’IP sur serveur au joueur sur le casque, ce qui nuirait à l’accessibilité de l’application.

Il n’existe pas non plus de moyen pré-fabriqué pour envoyer un flux vidéo.

### Assets Unity

Une autre possibilité serait d’utiliser une librairie disponible sur l’Unity Store. En effet, le projet actuel utilise la librairie FMETP\_Stream, qui permet des connexions TCP/UDP/Websocket entre plusieurs clients. Cette solution a, à la base, été développé pour envoyer des flux vidéo. Elle serait donc adaptée aux besoins du projet.

De plus, elle permet la découverte automatique des serveurs, ce qui faciliterait grandement la tâche. Il est également possible d’adapter l’asset selon les besoins, étant donné que nous avons le code source C# directement dans le projet.

Cependant, cette librairie ne permet pas les RPC, ou encore la synchronisation complète des objets. Il faudra alors rajouter cette surcouche afin de satisfaire les besoins du projet.

### Home made

Il est tout à fait envisageable de créer la couche réseau de toute pièce. Cette méthode permettrait que le serveur soit exactement designé pour les besoins du projet, mais demanderait un temps considérable, ainsi que de potentiels problèmes d’optimisation, voir même de surcharge réseau.

## Scénario

Dans le projet actuel, le scénario est une chose ficelée intrinsèquement avec le cœur du code. En effet, il n’existe pas de réelle séparation entre les mécanismes globaux, par exemple le réseau, et le scénario. Tout est en fait en un seul fichier, voir même dans des mêmes fonctions.

Ceci rend la généricité compliquée, étant donné qu’il est nécessaire de récrire des fonctions entières, afin d’arriver à nos fins. Pour donner un exemple, la synchronisation des énigmes est faite au même endroit que l’envoie et la réception d’indices. Or, ces deux choses n’ont rien à voir, à part qu’elles utilisent le réseau.

Une partie des informations pour un scénario, les pertubateurs et le nom du scénario, sont stockés sur une base de données. Cette base de données est construite au lancement de l’application, ce qui veut dire que les informations sont hardcodées dans des scripts. Cependant, pas toutes les informations sont dans cette base, et certaines informations sont même réécrite à un autre endroit, sans utiliser celle dans la base.

Il est donc compliqué de naviguer à travers le code, et de savoir ou il est utile de modifier et d’améliorer les choses.

Créer un nouveau scénario demande de reprendre le code de zéro, analyser toutes les interactions entre les scripts, en déduire lesquels sont essentiels, déterminer quelles données sont les bonnes pour enfin modifier et ajouter les informations nécessaires à notre nouveau scénario.

## Indices et perturbateurs

Dans l’état actuelle des choses, il n’y a pas de gestionnaire de jeu intelligent. Le maitre de jeu doit, par lui-même, déterminer si les joueurs sont en avance, ou en retard, et envoyer des indices ou perturbateurs. Ce système, mais malheureusement, il devient compliqué de prendre des notes en même temps.

De plus, cela demande que le maitre de jeu connaisse à l’avance les indices/perturbateurs pour savoir à quelle intensité celui-ci aide ou perturbe car il peut parfois être difficile de vraiment comprendre ce qu’il va s’afficher avec uniquement un petit texte de quelques mots.

Enfin, l’affichage des indices est quelque peu compliqué, car il est nécessaire de trouver la bonne énigme dans une liste déroulante, puis de trouver le bonne indice à envoyer dans une autre liste déroulante.

D’un point de vue développement, il est nécessaire d’harcoder les indices dans l’application de monitoring, en spécifiant le même id que dans l’application de jeu pour que tout fonctionne. Cela peut engendrer des problèmes difficiles à débugger.

## Casque de réalité augmentée

Le projet HOST utilise le casque de réalité augmentée de Microsoft, le HoloLens 2. Il permet des interactions simples, toucher et attraper. Microsoft fourni un SDK permettant les interactions en réalité augmentée pour le développement d’application sur Unity.

## Monitoring

L’application de monitoring comporte une multitude de fonctionnalité aidant le suivi d’une partie. Tout d’abord il y a le livestream joueurs, permettant de voir ce que les utilisateurs sont en train de faire dans le monde virtuel.

Ensuite, il est possible de prendre des notes, sous forme de commentaire, qui seront enregistrés dans une base de données, accompagné d’une image représentant la vidéo affichée dans la fenêtre principale et d’un timestamp du moment t auquel le commentaire a été pris.

Une vidéo est également enregistré tout du long de la simulation, permettant ainsi de créer des sous-titres à la fin de la simulation, avec les commentaire enregistrés au préalable.

Un rapport PDF est également généré grâce aux commentaires / images enregistrés.

# Technologies

## Casque de réalité augmentée

### HoloLens 2

Le HoloLens 2 est un casque créé par Microsoft permettant la réalité augmentée. Il possède des lentilles holographiques transparentes permettant d’afficher des images virtuelles, tout en voyant le monde réel. Il possède une multitude de capteur permettant le positionnement du casque dans l’espace, ainsi que du suivi des mains.

#### Avantages

Le projet HOST a été conçu pour le HoloLens 2. Les différentes interactions sont donc déjà connues et seraient plus simple à réutiliser.

Le casque étant tout d’abord axé AR, la visualisation du monde réel est de très bonne qualité, étant donné que l’on voit comme à travers des lunettes.

#### Inconvénients

Microsoft a annoncé que les HoloLens 3 ne verront jamais le jour. On peut en déduire que la technologie est en fin de vie. Il serait alors préférable de changer de technologie, pour en trouver une qui continuerait à être active.

De plus, le prix du casque est très élevé. L’application étant multijoueur, il faudrait alors avoir plusieurs casques, ce qui démultiplierait les coûts.

Enfin, l’affichage des éléments virtuels n’est pas très bon. En effet, dans une pièce trop lumineuse, il est difficile de distinguer les différents éléments qui s’affichent à l’écran.

### Meta Quest Pro

Le Meta Quest Pro est développé par Meta, anciennement Facebook. C’est un casque de réalité virtuelle, permettant la réalité augmentée. Ce qui veut dire que l’on regarde dans un écran, et quand on active le mode AR, on voit le flux des caméras. Il est possible de l’utiliser avec des manettes, mais également directement avec les mains, car il est capable de les traquer. Il est également capable de se repérer dans l’espace, mais uniquement de manière limitée.

#### Avantages

Le Meta Quest Pro permet l’utilisation de la plupart des fonctionnalités d’AR Fondation. Cette librairie permet une multitude de chose en AR, et est utilisé également pour les smartphones. Elle est donc robuste, et hautement utilisée.

Les éléments virtuels affichés sur l’écran sont de très bonne qualité. Il est très simple de les distinguer, même dans une pièce sombre.

Il est possible d’utiliser les manettes pour encore plus d’interactions. Cela laisserait une palette de possibilités plus grande.

#### Inconvénients

Étant donné que la réalité est vue à travers des caméras et un écran, elle n’est pas de très bonne qualité. Il est impossible de voir correctement une zone très éclairée par exemple, et également impossible de pouvoir lire sur une feuille.

### Meta Quest 3

Tout comme le Meta Quest Pro, le Meta Quest 3 a été crée par Meta. Il n’est pas disponible à la vente au jour du début de ce travail, mais il le deviendra environ 1 mois plus tard. C’est également un casque axé pour la réalité virtuelle, mais ayant de fort caractéristiques pour la réalité augmentée.

Il est capable de se repérer dans l’espace, de traquer les mains, de mapper une pièce afin de pouvoir en connaitre les murs, plafonds, sols, meubles et pouvoir les utiliser dans les applications.

#### Avantages

Ce casque sortant peu de temps après le début de ce travail, il sera donc au top de la technologie actuelle. De plus, c’est le moins chère des trois. Il est capable d’afficher des éléments virtuels de très bonne qualité, et le mode « réalité augmenté » est de bien meilleure qualité. En effet, il est maintenant possible de lire sur des feuilles, et il est même possible de déchiffrer quelques mots sur un téléphone.

De plus, grâce à sa gestion de l’environnement, il est encore plus capable de réalité augmentée. On pourrait imaginer créer les pièces des scénarios en fonction des meubles disponible dans le monde réel.

#### Inconvénients

Le casque n’est pas disponible dès le début du travail.

### Choix effectué

Étant donné que l’HoloLens 3 ne verra jamais le jour, continué sur cette technologie n’est pas judicieux. De plus, le projet reprenant à partir de zéro, c’est le bon moment pour changer de technologie. Dans cette même optique, il parait judicieux d’utiliser le dernier casque sortie pour être au top de la technologie.

Attendre un certain délai avant de pouvoir utiliser le casque n’est pas un problème, car il est tout a fait possible de commencer le développement sur le Meta Quest Pro, qui permet la plupart de même fonctionnalité, mais en moins bonne qualité.

## Version d’Unity

### Unity 2020 LTS

Version actuelle du projet. Permet d’utiliser les HoloLens 2, ainsi que le Meta Quest Pro. Cependant, cette version est trop ancienne pour développer sur le Meta Quest 3.

Il est intéressant de noter que des librairies pouvant apporter de nouvelles fonctionnalités, tel que Vuforia, qui permet l’analyse d’image et détection d’objets, sont également indisponible pour leur dernière version.

### Unity 2021 LTS

Cette version permet de développer sur tous les casques cités précédemment. Cette version est toujours supportés activement par Unity, ainsi que par des librairies diverses. Cependant, Unity a annoncé la fin du support de cette version pour 2024.

### Unity 2022 LTS

Le HoloLens n’est pas pleinement supporté sur cette version. D’après la documentation de Microsoft, il peut y avoir certains bugs d’affiche avec le pipeline URP d’Unity. Le projet ne l’utilisant pas, il serait possible de choisir cette version, étant donné que c’est la version LTS la plus récente, et qu’elle ne restreint que très peu le choix du casque.

### Unity 2023

Cette version ne possède pas de LTS. Il est possible d’avoir un manque de stabilité, mais cela se compense par un ajout de fonctionnalité nouvelle qui pourrait être intéressante.

Meta a mis à disposition des SDK pour le développement sur ces casques en réalité augmentée, uniquement pour cette version. Le SDK est expérimental, mais permettrait une meilleure gestion de l’application, notamment avec de l’occlusion, ou encore avec une meilleure gestion du mon réel.

Il est important de noter qu’il n’est pas possible de développer pour les HoloLens 2 sur cette version.

### Choix effectué

Le projet ayant pour but d’être expérimental, il a été décidé conjoitement avec le responsable du projet qu’Unity 2023 serait la version choisie.

Ceci permettra d’avoir une application supportée le plus longtemps possible, tout en ayant accès aux différentes librairies expérimentales, qui sont en train d’émerger pour la réalité augmentée.

# Conceptions et implémentations

## Couche réseau

### FMETP\_Stream

J’ai décidé de partir sur l’asset FMETP\_Stream pour plusieurs raisons. En premier lieu, cet asset a permis au projet actuel de fonctionner. De ce fait, un exemple fonctionnel est disponible, ce qui me permet de m’inspirer et/ou de reprendre certaine partie qui me paraisse bonne.

La découverte automatique de serveur enlève une grande épine du pied. En effet, l’implémenté aurait demandé un broadcast UDP sur le réseau, ce qui aurait largement augmenté la complexité du projet sur ce point-là, ce qui n’était pas l’objectif.

L’asset mettait également à disposition les échanges clients/serveurs. Il me restait donc uniquement à ajouter une couche par-dessus, pour faire les échanges avec les messages contenant les informations que je souhaite.

Le streaming vidéo étant le point fort de l’asset, cela m’a permis de ne pas me soucier des soucis de performance, ni même d’aucune gestion de stream que ce soit. Leur bonne documentation m’a permis d’implémenter ce point important en un temps record.

Petit point bonus, un début de synchronisation de positions des objets étant présent dans l’asset. Bien que non complet, il permettait de s’appuyer dessus afin d’avoir une base fonctionnelle. Il ne me restait qu’à agrémenter la fonctionnalité avec les besoins lié au projet pour arriver à un état fonctionnel pour le projet, tout en restant simple à utiliser.

### Host Network Manager

Le script Host Network Manager représente la surcouche réseau pour le projet. C’est ici que tous les liens avec l’asset FMETP\_Stream sont fait et que les méthodes de communications sont réseaux sont écrites. C’est le point central du réseau pour ce projet. Chaque interaction passe forcément par lui, et c’est ce manager qui va rediriger à la bonne entité en fonction du message reçu.

Il est donc possible de changer d’assets réseau en changeant uniquement ce manager, étant donné qu’aucun autre script n’a besoin ou ne fait de référence directement à l’asset réseau.

Au démarrage de l’objet contenant ce script, il recherche tous les objets dérivant de HostNetworkRPC afin de créer une liste des entités ayant la possibilité de recevoir des instructions à distance. Cette population est également faite lorsqu’une scène change.

Ces attributions sont finalement :

* Créer la liste des entités ayant des RPC ;
* Envoyer/Recevoir les RPC ;
* Envoyer/Recevoir les requêtes de synchronisation d’objets ;
* Gérer les connexions/déconnexions ;
* Gérer la liste de objets synchronisés ;
* Connaitre la liste des IPs connectés.

### Host Network Object

Ce script permet de rendre un objet synchronisé. La synchronisation comprend uniquement l’intégrité de la position, rotation et de l’échelle entre les différents clients.

Ce script détecte également lorsqu’une de ces valeurs a été modifiée chez un client, et envoie par lui-même une demande de synchronisation, spécifiant les nouvelles valeurs. Le serveur doit alors la gérer, effectuer le changement localement, puis la synchronisation chez tous les participants peut finalement être faite.

Tous ces objets ont un ID défini à la compilation. Un helper est disponible pour que les IDs soient automatiquement générés, permettant alors au développeur de ne pas avoir de doublon.

### Host Network RPC

Tout d’abord il faut bien comprendre ce que représente des RPC dans ce projet. Ce sont principalement des messages JSON contenant :

* Un ID ;
* Un nom de méthode ;
* Des paramètres.

Il est également important de savoir que toutes les instances de HostNetworkRPC ont un ID unique, permettant de les départager. Le manager n’a donc plus qu’à parcourir sa liste d’entité RPC, trouver laquelle a l’ID correspondant à celui dans le message, et enfin appliquer une réflexion pour appeler la bonne méthode avec les paramètres.

Cette méthode reste tout de même limitée, dans la mesure ou il est compliqué de debugger si le nom de la méthode n’est pas le bon. L’IDE ne montrera pas d’erreur, étant donné que l’on passe le nom de la méthode dans un string, et que l’IDE ne peut pas comprendre que cela fait lien à une méthode.

Cependant, en étant rigoureux, il est très facile de créer de nouvelles entités RPC et d’appeler leurs méthodes.

### Outils utilitaires

Une classe contenant des méthodes statiques est disponible pour toutes les méthodes utiles utilisées à plusieurs endroits pour el réseau. On peut notamment y retrouver les méthodes pour sérialiser/désérialiser les messages et les structures des messages envoyé sur le réseau.

## Scénario

### Réflexion

Le point le plus important du projet étant la généricité d’une structure de scénario, il a fallu passer du temps sur ce point-là.

Tout d’abord, il faut définir ce qui compose un scénario :

* Le scénario dans son ensemble ;
* Les énigmes du scénario ;
* Les éléments déclencheurs d’une énigme.

Il convient alors de connaitre points principaux de chaque composante. Il est important de gérer :

* Le début d’un scénario ;
* Le début d’une énigme ;
* La complétion d’un élément déclencheurs ;
* La fin d’une énigme ;
* La fin d’un scénario ;

Ces différents événements permettent de mettre en place le scénario selon l’envie. En effet, il est possible que certains scénarios aient besoin d’avoir un événement de démarrage globale afin de mettre en place des éléments du décors.

Pareillement pour le début d’une énigme, ou l’on pourrait avoir un élément scénaristique qui entre en jeu, par exemple une télévision qui s’allume, pour donner un élément de réflexion aux joueurs.

### Structure

J’ai donc pensé à deux manières de pouvoir représenter et utiliser ces événements de manières génériques.

#### Event Based

Cette structure permet d’avoir des événements dans les différents scripts, donnant alors accès à leur déclenchement. De ce fait, il ne reste donc plus qu’à écouter un événement, pour savoir quand effectuer les actions désirées.

Cela permet d’avoir une structure fixe, à laquelle on vient se « connecter » pour y ajouter les éléments propres au scénario.

Cette manière est probablement la plus simple à mettre en place, ne demandant qu’un seul script « squelette ». Cependant, utilisés les événements peut demander un surcoût en termes de performances.

Une image contenant texte, diagramme, croquis, dessin

Description générée automatiquement

Figure 1 : Schéma du déroulé d'un scénario

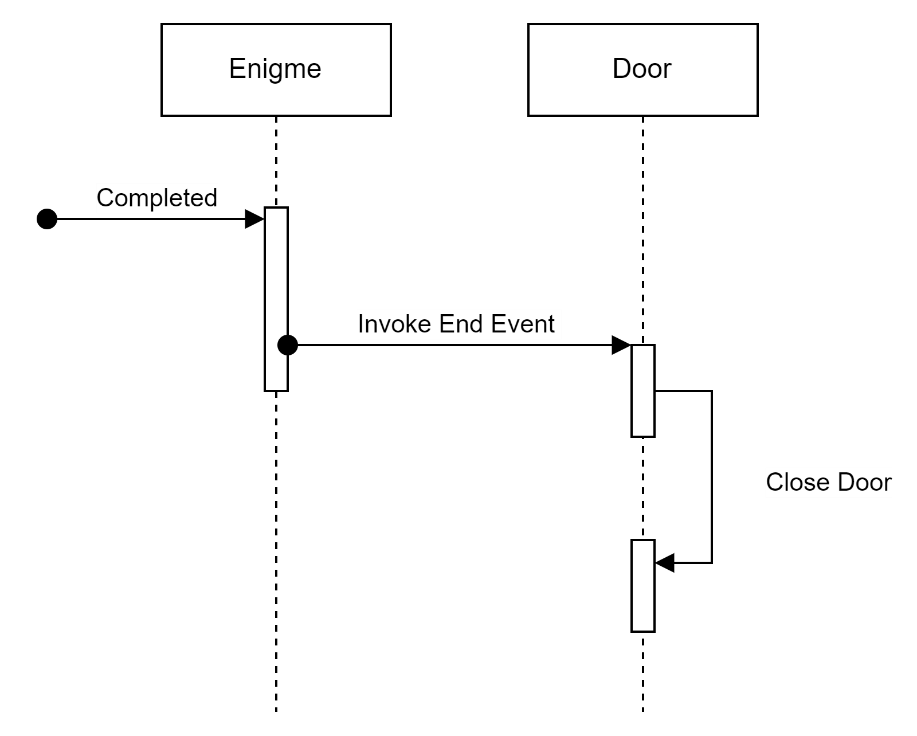


Figure 2 : Diagramme simple pour exemplifier la structure avec événements

#### Spécialisation

Au lieu d’avoir des événements qui sont invoqués aux différents stades correspondants, on aurait des fonctions effectuant le code désiré. Il s’agirait alors d’avoir un squelette de base, effectuant les actions obligatoires et nécessaire au fonctionnement en réseau, puis dans un deuxième temps, de faire des spécialisations de ces différents scripts pour pouvoir override les fonctions de manière à faire fonctionner un scénario spécifique.

L’avantage de cette méthode est la performance, mais cela se compte en quelques millisecondes.

Il est cependant nécessaire de créer une spécialisation pour la plupart des cas, ce qui augmente drastiquement le nombre de scripts, et la complexité du projet.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 3 : Diagramme simple pour exemplifier la structure avec spécialisation

#### Choix effectué

Finalement, il a été décidé que ce serait la structure event based qui allait être le cœur de la structure.

Ceci nous permet d’avoir un squelette entièrement fonctionnel, qu’il n’y aura pas besoin de changer par la suite.

Cela réduit également le nombre de scripts, permettant alors une clarté plus grande lors du développement.

### Scénarios

Au vu du choix pris, le scénario correspond à un gestionnaire d’événements. Il se comporte plutôt comme un aiguillage, sans logique propre à un scénario.

#### Début Scenario

Cette méthode lance le scénario. A ce moment, il va s’abonner aux différentes énigmes le composant, ainsi qu’appeler leur méthode de début global, qui sera détaillé dans la partie « Énigmes ».

Il va également se réinitialiser afin de partir sur de bonne base dans le cas où on relancerait le scénario.

Enfin, on va lancer spécifiquement la première énigme dans la liste.

Pour terminer, on va notifier, via l’événement, que le scénario a commencé afin que tous les abonnés puissent effectuer leurs propres actions.

Il est important de noter que ce n’est que le serveur qui doit effectuer ses actions, étant donné que les clients vont simplement recevoir dans un deuxième temps quelles méthodes effectuées.

#### Complétion d’une énigme

Lorsqu’une énigme se complète, il peut y avoir deux possibilités :

1. Il reste encore des énigmes, il faut alors commencer la suivante ;
2. Il ne reste plus d’énigme, le scénario est terminé.

Dans le premier cas, il suffit de démarrer l’énigme suivante en appelant sa méthode. Dans le deuxième, on doit notifier les abonnées que le scénario est terminé.

### Énigmes

Les éngimes suivent la même structure que le scénario. On a toujours l’idée d’un aiguillage sans logique propre, permettant alors d’être complétement décollé des scénarios précis.

#### Démarrage global de l’énigme

Ceci intervient au tout début du scénario, et non de l’énigme. Cette étape est importante si l’on veut effectuer des actions qui seront visible dès le début par les utilisateurs. On peut notamment penser à afficher des codes dans le décor, qui ne serviront qu’une fois l’énigme lancée, mais qui pourrait être trouvé au préalable.

Il est donc important, pour chaque énigme, de proposer un événement notifiant le script abonné du début global du scénario afin de départager la responsabilité et ne pas tout mettre dans l’événement de démarrage du scénario.

#### Démarrage d’une énigme

On doit tout d’abord notifier les clients que l’énigme commence. Il faut donc effectuer un RPC, puis laissé chacun notifier leurs abonnés que l’énigme à commencer.

On peut également prévoir qu’une énigme pourrait être terminé avant même d’être commencé, c’est pourquoi il faut également vérifier à ce moment si tous les éléments déclencheurs sont complétés et dans ce cas là, terminer l’énigme en invoquant l’événement correspondant.

#### Complétion d’un élément déclencheurs

Si tous les éléments sont terminés, on peut aller notifier que cette énigme est également terminée. Dans le cas contraire, le jeu continu sans changement.

### Éléments d’une énigme

Les éléments d’une énigme attestent de l’avancer. En effet, ce sont eux qui peuvent se compléter, ce qui se répercutera sur les énigmes, puis sur le scénario.

Ces éléments ne sont pas obligatoirement à effectuer dans l’ordre. Il est tout à fait possible de forcer l’ordre, par exemple en désactivant certains avant d’avoir compléter un certain point.

Il possède également un événement afin de savoir quand est-ce que celui-ci est complété. On peut par exemple imaginer que lorsqu’un code est trouvé, l’élément est donc complété et il faut notifier un coffre pour que celui-ci s’ouvre.

La complétion d’un élément est bien sûr traitée sur le serveur. Les clients ne peuvent qu’effectuer des requêtes de complétion, mais c’est le serveur qui va broadcaster à tous les utilisateurs que l’élément a été complété.

## Gestion intelligente du jeu

### Réflexion

La gestion intelligente du jeu demande beaucoup d’imagination quant à comment la rendre agréable. Du point de vue développeur, il faut être clair sur comment l’avancement du jeu se fait. On est donc obligé d’avoir des triggers digitaux pour attester du bon fonctionnement du scénario.

Si les joueurs ouvrent des cadenas réels, on ne peut pas savoir qu’ils ont réussi l’étape. On est donc obligé de trouver un moyen pour valider cette étape.

Cela demande donc une réelle réflexion, qui ne peut pas être ignoré. Une possibilité serait d’utiliser de l’analyse d’images, permettant alors d’avoir des QR codes dans les endroits bloqués, permettant alors de savoir lorsque cet endroit a été ouvert. Cependant, Meta ne permet pas l’utilise du flux vidéo, et cela n’est donc pas une solution envisageable.

Du point de vue des joueurs, il ne faut en aucun cas que les influences soient frustrantes. Donner un indice directement après avoir fini une énigme pourrait frustrer les joueurs.

J’ai donc imaginé un moyen de réduire ces potentiels frustrations, et imaginant un système basé sur deux variables clés.

L’une est le temps passé depuis le début du scénario, et donc connaitre si l’on est plutôt en avance, ou en retard par rapport au scénario en entier.

L’autre est le temps passé depuis le début d’une énigme. Ce temps est plus important que le temps global, étant donné que c’est vraiment sur une énigme en particulier que la frustration peut survenir.

Une image contenant diagramme, texte, cercle, ligne

Description générée automatiquement

Figure 4 : Réflexion sur la manière d'influencer les joueurs

L’impact a envoyé aux joueurs serait donc une corrélation entre le temps global, et le temps de l’énigme en cours. Sauf que le temps global pourrait influencer entre -2 et 2.

L’énigme en cours quand à elle serait de -5 à 5, et en additionnant les deux, on arriverait au niveau d’influence à envoyer.

Cependant, il est important qu’en fonction de l’avance / retard sur une énigme seulement, on puisse avoir toutes les influences. On peut donc avoir des influences potentiels entre -7 et 7, mais avoir les influences réels entre -5 et 5.

Cela amène plusieurs scénarios, dont voici les plus importants et significatifs.

#### Énigme et scénario en retard

Il est évident que lorsque l’équipe est en retard sur les deux aspects, on doit envoyer des indices percutants.

On a donc la possibilité d’être entre -1 et -7. Étant donné que le niveau maximal d’influence réel est de +/- 5, cela ne servirait pas d’arriver à -6 ou -7, mais l’addition des deux chiffres permettent d’arriver à un gros niveau d’influence encore plus rapidement.

#### Énigme en retard ; Scénario en avance

Si l’énigme est en retard, on doit pouvoir donner des bons indices aux joueurs pour éviter la frustration. Cependant, ils ont été performant précédemment, donc on peut leur laisse du temps à cogiter. On a donc le retard sur l’énigme entre -5 et -1, mais l’avance sur le scénario entre 1 et 2.

En faisant l’addition, on arrive donc à un intervalle de -3 à 1, ce qui limite les indices trop évidents.

#### Énigme en avance ; Scénario en retard

Ce scénario est exactement le même que le précédent, à la différence ou il est inversé. Avec la même justification, on arrive à un intervalle de -1 à 3, ce qui limite les perturbateurs et permet même d’envoyer des pistes si les joueurs sont vraiment perdus.

#### Énigme et scénario en avance

Tout comme le premier des quatre scénarios, si tout est en avance, on a un intevalle entre 1 et 7. On doit donc arriver à perturber les joueurs le plus possible afin de les remettre dans le temps souhaité.

### Scenario Manager

Le scénario manager est le chef d’orchestre. C’est lui qui possède les chronomètres et les paramètres de la simulation. C’est lui qui a la responsabilité de décider si des influenceurs doivent être envoyé.

Évidemment, il est nécessaire que les responsabilités soient partagées, et c’est pour cela qu’il ne sert uniquement que de liens entre chaque chose, et qu’il demande à chaque script, décris plus bas, les informations nécessaires à la prise de choix.

Deux états, un pour le scénario et un pour l’énigme sont disponible, permettant ainsi de déléguer les responsabilités et d’éventuellement de pouvoir créer plusieurs manières de calculer l’indice d’influence.

### Paramètres de scénario

Les paramètres de scénario et d’énigmes connaissent le temps que devraient prendre chacun. Ceux-ci sont calculé à partir de poids attribués à chaque énigme, puis en calculant le rapport de poids de l’énigme sur le total.. Ils sont capables de faire de multiples calculs permettant d’avoir les informations nécessaires, comme la différence de temps par rapport au temps souhaité.

En connaissant l’état d’avancement d’une énigme, et son temps actuel, on peut déterminer de combien de temps elle est en retard ou en avance.

Le temps actuels passés sur une énigme est également gardées afin de pouvoir afficher un récapitulatif du temps passer sur chaque étape à la fin de la partie.

### Calculateur d’indice d’influence

Le calculateur d’indice d’influence prend en compte les informations disponibles dans les paramètres de scénario et d’énigmes.

A l’heure actuelle, l’indice retourné est arbitraire, car il ne provient d’aucune source, mais uniquement de test pour avoir un bon feeling. Il serait imaginable de transformer ceci en apprentissage continue, adaptant ainsi les bonnes valeurs de retour.

## Influenceurs

### Structure

Les influenceurs suivent également la structure event based. En effet, ils ne sont qu’un squelette auquel on doit abonner un autre script qui devra être appelé via l’événement.

Les influenceurs, qu’ils soient indice ou perturbateur possèdent un niveau d’influence, entre -5 et -1 pour le perturbateur, et 1 et 5 pour les indices.

### Récepteur d’influence

Les récepteurs d’influence sont une petite aide qui permet d’afficher les différentes influences. Par exemple, un des récepteurs d’influence est pour les images.

Il suffit alors que l’on programme qu’au moment où l’événement de l’indice arrive, on effectue l’affichage de l’image sur le récepteur.

Ce récepteur va de lui-même gérer comment afficher l’information, peut également effectuer des actions au début et à la fin de l’affichage. Cela permet, par exemple, de jouer un son au début de l’indice.

## Interactions

En me basant sur le SDK de Meta, j’ai pu implémenter différent type d’interactions, en y ajoutant des événements pour savoir l’état actuel de l’objet. Ceci permet de mieux gérer ce qu’il se passe au niveau du réseau.

Grâce à ces événements, on peut également compléter certains des éléments d’une énigme.

### Objet déplaçable

Les objets déplaçables sont l’intéraction la plus facile à mettre en place, mais probablement la plus compliqué à avoir un feeling bon.

Il s’agit simplement d’ajouter un rigidbody et un collider à l’objet, afin que la physique d’Unity fasse sa magie. Cependant, il est nécessaire de bien tester cette méthode, étant donné que cela peut être une interaction très frustrante si elle n’est pas bien réalisée.

J’ai ajouté un script gérant les événements, en se basant sur la vélocité de rotation et de déplacement afin de savoir si l’objet :

* Commence à bouger
* Est en train de bouger
* A fini de bouger

### Objet attrapable

Les objets attrapables sont fournies par le SDK de Meta, et il m’a simplement fallu rajouter un script permettant de gérer les événements de grab qui sont :

* Commencer à attraper
* Objet tenu
* Objet relâché

### Objet regardable

Comme son nom l’indique, un objet peut savoir s’il est regardé. Pour cela j’ai utilisé le raycasting depuis la caméra principale afin de savoir si l’objet est dans son champ de vision.

Des événements sont également disponibles pour pouvoir gérer au mieux ces objets :

* Commencer à regarder
* Regarder depuis un temps x

### Objet touchable

Cette interaction est également basée sur le SDK de Meta. Cela permet à l’utilisateur d’avoir un effet de bouton qui peut être pressé.

L’événement de clique est disponible grâce à un script rajouter par mes soins.

## Monitoring

Le monitoring n’a pas beaucoup évolué par rapport à la version existante du projet. Cependant il est tout de même nécessaire d’expliciter ce qui a été fait.

### Live Stream

Le live stream des joueurs est disponible sur l’écran de l’application de monitoring. Il est possible de basculer entre les vues afin de capturer celle choisie.

Pour streamer, c’est la librairie FMETP\_Stream qui est utilisé. Il suffit d’utiliser leur Game View Encoder et Game View Decoder pour que cela fonctionne.

Pour différencier les flux vidéo, il est nécessaire de définir un ID unique pour chaque pair d’encodeur/décodeur. Cet ID est défini dès que les joueurs sont dans la scène de jeu.

### Interface

#### Lobby

Dans le lobby, le moniteur peut dès à présent choisir le scénario. Il peut mieux le reconnaitre grâce à son nom et son image.

On peut également voir que l’on peut paramétrer chaque énigme pour faire correspondre à nos besoins le scénario.

Les valeurs paramétrables sont les suivantes :

* Le temps que la simulation devrait prendre ;
* Le temps minimal entre chaque indice ;
* Le temps minimal entre chaque perturbateur ;
* L’intervalle de temps dans lequel on considère être « dans les temps ».

On peut également retrouver la liste des joueurs qui se sont connectés.

#### Écran principal

Sur l’écran principal, on retrouve un espace pour écrire des commentaires. Ces commentaires seront utile pour le Feedback.

Dans la partie de droite, on peut voir la caméra du PC, qui filme la salle, et la vue de chaque joueur. Il est possible d’alterner entre les vues simplement en cliquant sur celle désirée.

Lorsqu’une vue est sélectionnée et afficher en grand, elle est enregistrée afin d’être présente dans le feedback de fin.

Il est également possible de cliquer sur la bulle en bas à droite de l’écran afin d’ouvrir une fenêtre permettant d’envoyer un message aux joueurs.

## Génération de feedback

### Vidéo sous titrées

Lorsque le scénario se lance, un codec externe est utilisé pour capturer la vidéo. Cette vidéo capture uniquement la vue principale de l’écran.

Lorsque le scénario se termine, les commentaires pris par le maitre de jeu sont lu et transcrit dans un fichier srt, permettant alors d’avoir des sous-titres à la vidéo avec les commentaires pris.

### Rapport PDF

Lorsque le maitre de jeu écrit un commentaire, l’application effectue une capture d’écran afin de garder l’imagine qui était visible sur la vue principale.

A la fin de la simulation, un rapport PDF est généré contenant la pair commentaire/image.

## Ancrage du monde virtuel

### TODO

# Problèmes rencontrés

# Conclusion

# Bibliographie

1. Cussler, C. (s.d.). *Sahara.*
2. Hergé. (s.d.). *Tintin en amérique.*

# Liste des figures

[Figure 1 **Erreur ! Signet non défini.**](#_Toc382399344)

# Annexes

Liste des annexes

#### Annexe I

Annexe II