# Introduction

## Cahier des charges

# Analyse de l’existant

## Host

### Points fondamentaux

Basé sur Serveur TCP avec une surcouche « HostNetwork » -> UDP est généralement utilisé pour les jeux, changé ça pour TM ?

### Limitations

Vraiment fait pour 1 seul scénario

Pas vraiment de scénario a proprement parlé, toutes les énigmes faisables à tout moment, donc pas de trame. Donc il a fallu callé un manager de scénario. -> grosse amélioration pour TM

Le système de communication client-serveur est minimaliste, et a du mal avec de l’envoie de donnée fréquente (typiquement pour synchronisé la position d’un objet)

Les méthodes RPC doivent être inscrites sur les 2 app, même si une des 2 les utilisent pas car ne fait que l’invoker chez l’autre

# Scénario

## Contexte

## Nouvelles mécaniques de jeu

Affiche d’objet sur QR Code

Synchronisation accrue entre les HoloLens, donc plus de collaboration

## Énigmes

Ici faut mettre des infos sur comment ont été faite les énigmes, et également comment modifier les codes ou autre pour matcher la réalité -> Ce serait d’ailleurs un super point à simplifier pour le TM

## Indices

## Perturbateurs

# Choix des technologies

## Détection d’images ou QR codes

Premiere tentative avec la lib QRCode de Microsoft -> infructueux, même les exemples de Microsoft n’ont pas fonctionné. Limitation net également avec la taille très peu pratique des QR Code.

Du coup Vuforia, qui permet de faire ce qu’on veut, même si limiter à al version 9 car le projet est sous unity 2020

# Planification du projet

Code dispo sur Github (lien)

Utilisations des issues et d’un kanban directement sur github

Les repos sont dans une organisation, donc simple d’ajouter des collaborateurs, donner les droits à la fin du projet, etc..

## Ordre des jalons principaux

1. Écriture du scénario
2. Mise en place des mécaniques des énigmes
3. Création du Game World
4. Liaison entre les énigmes et le Game World
5. Scénarisation des énigmes
6. Indices et perturbateurs

# Développement du projet

## Game world

Le monde a été crée avec les assets « … ». Le but étant d’être un hôpital désaffecté, n y retrouve de vieux objets tel que :

* Projecteur
* Seringue
* …

## Énigmes

### Détection d’images

Utilisation de Vuforia pour détecter une image et placer un objet sur celle-ci.

### Prise en compte des éléments réels

Création du message crypté. Une partie de la clé de chiffrage doit être un élément réel, et donc non aléatoire, besoin de penser un algorithme qui ne modifie pas certaines lettres, cf code CryptedMessage pour mieux expliquer

## Networking

### Utilisation de HostNetworkManager

### Synchronisation des HoloLens

Mode client/serveur ou l’app de monitoring est le serveur et les hololens sont les clients -> on pourrait peut être passé à peer to peer, mais c’était plus simple pour le temps à dispo.

## HoloLens

### Différence avec la simulation Unity

Plus lent d’envoyer/lire buffer -> problème

### Déploiement

Ne pas oublier le certificat

Possible de build and run via Unity avec le device portal manager (mais nécessite de s’être connecté une fois via browser pour créer une sorte de user)

# Problèmes rencontrés

## Networking avec les HoloLens

Message Json successif sous force "{…}{…}" n’est pas un format reconnu, problème juste à cause du buffer qui était pas vide avant d’écrire. Ça marchait sur simulation unity car probablement instantané et donc buffer était vide à chaque fois.

Exemple de message qui posait problème : {"MessageType":0,"NetworkTarget":0,"SourceIP":"0.0.0.0","TargetIP":"","Data":"eyJPYmplY3RJZCI6MSwiUnBjSWQiOjExLCJQYXlsb2FkIjpbZmFsc2VdfQ=="}{"MessageType":0,"NetworkTarget":0,"SourceIP":"0.0.0.0","TargetIP":"","Data":"eyJPYmplY3RJZCI6MSwiUnBjSWQiOjEwLCJQYXlsb2FkIjpbNF19"}

# Conclusion

# Annexe

Code sur Github

Dossier de l’exécutable de l’app de monitoring

HoloLens avec l’application