Projet S2 : TheAzureStone

Rapport première soutenance

Par le groupe BananaSplit

Adrien Mussat

Jules-Victor Lepinay

Amine Belhadj

Bob De Boisvilliers

Sommaire

1 Introduction

1.1 Membres du groupe

Notre groupe *BananaSplit* se compose de 4 membres :

- Chef de projet : Adrien Mussat (adrien.mussat@epita.fr) B1
- Jules-Victor Lepinay (jules-victor.lepinay@epita.fr) B1
- Amine Belhadj (amine.belhadj@epita.fr) B1
- Bob De Boisvilliers (bob.de-boisvilliers@epita.fr) C2

1.2 Rappel du projet

Comme indiqué en détaille dans le cahier de charges durant ce projet de S2, nous avons saisi l'opportunité qui nous était offerte de réaliser un jeu vidéo. Domaine qui nous ai cher étant tous amateur de ces derniers en particulier les jeux de stratégie et de simulation. The Azure Stone serait un jeu de gestion et stratégie basé exclusivement sur une ferme. Le joueur pourra se projeter ainsi pendant quelques heures dans une tout autre vie. Ce jeu s'inspirera sur certains jeux similaires auxquels nous avons joué tel que *Farmville* ou bien *SimCity* mais aussi des jeux mobiles pour leur style à la vue isométrique.

Dans ce jeu, le joueur intègre donc un fermier dont l'ultime but et de prospérer en gagnant de l'argent via la vente de ses productions.
Pour cela, il a la possibilité : d'embaucher des ouvriers agricoles pour l'aider à travailler dans ses champs et enclos, d'augmenter la taille de sa ferme, de diversifier ses cultures et les rendre plus rentables grâce à un système de recherche, et de choisir de jouer seul ou en coopération.
Comme expliqué plus haut, ce jeu rentre dans les catégories des jeu des gestion et stratégie, il sera donc composé avec une vue à la troisième personne et le contrôle d'un avatar dan le jeu.

2 Avancement du projet

2.1 Nouveau planning et répartitions des taches

Re-estimations des différents objectifs pour les soutenance :
Légendes:

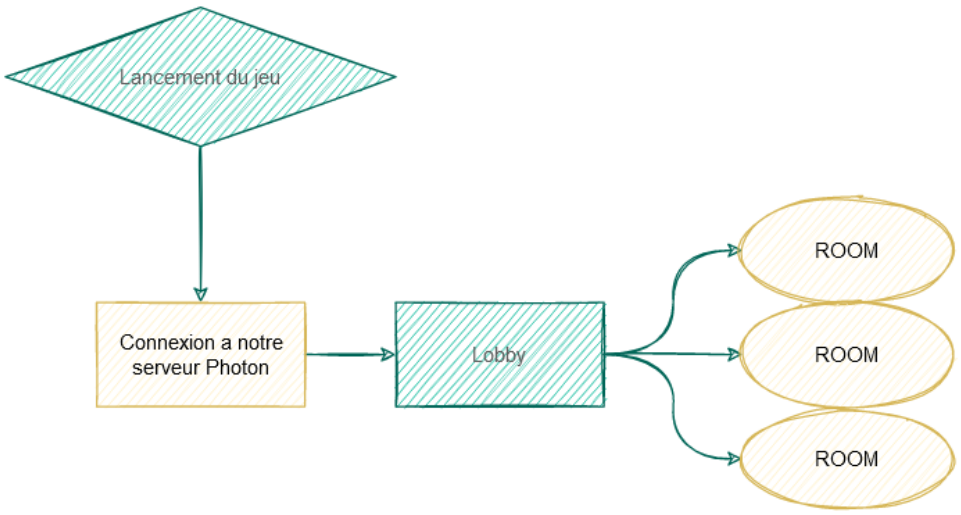
Tâche	Pourcentages original prévus lors du cahier des charges
	Pourcentages ré-évaluer avec l'avancer actuelle

Soutenances et parties	**Première soutenance**	**Soutenance intermédiaire**	**Dernière soutenance**
Multijoueur, RSS et Chat	30 à 40 %	70 à 80 %	100 %
	40 %	70 %	100 %
IA	< 20 %	40 %	100 %
	0 %	30 %	100 %
Gameplay	30 %	75 %	100%
	25 %	60 %	100%
Level Design (Asset, graphismes)	20 %	60 %	100 %
	10%	50%	100 %
UI	20 %	50 %	100 %
	10%	50%	100 %
SiteWeb	20 %	60 %	100 %
	20%	60%	100 %

2.2 Multijoueurs(1) RSS

- Responsable : Adrien
- Suppléant : Jules-Victor

Avancée actuelle :
Le multijoueur est possible dans notre jeu principalement grâce à l'intégration Photons dans Unity qui nous permet d'avoir un serveur de jeu gratuitement supportant jusqu'à 20 joueurs connectés en simultané.
Ainsi, lors du lancement de notre jeu, la connexion se fait automatiquement de la manière suivante :



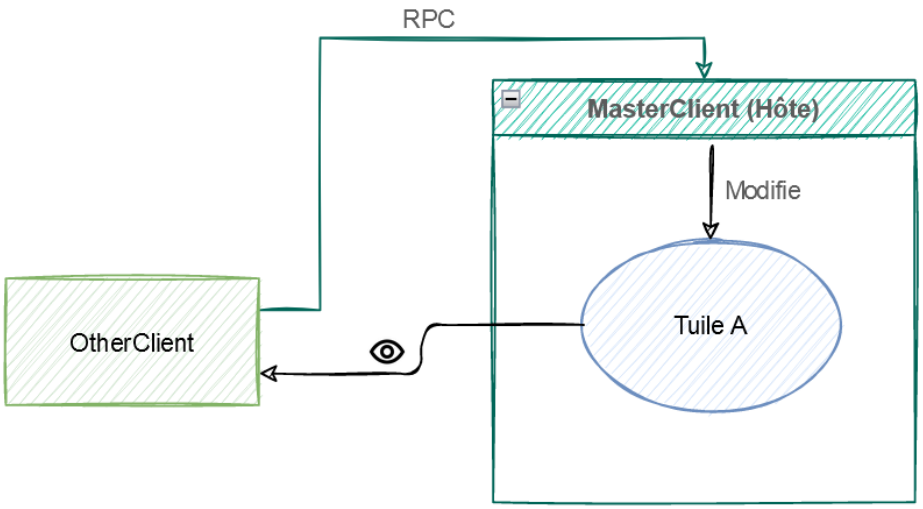
Arrivé dans le *Lobby* le joueur a alors le choix entre créer un nouvelle *ROOM* (il devient l'hôte.) ou alors en rejoindre une grâce à son nom.

La connexion à une même *ROOM* réussie il fallait ensuite synchroniser le jeu entre les différents clients connecté a celle-ci. Par défaut Photon, donne un système qui permet a chaque client connecter d'instancié des Objet sur la *ROOM* cependant cette vision du jeu était fortement incompatible avec le Gameplay, en effet le principe de notre jeu repose sur la modification d'un plateau, celui-ci ne peut donc pas être séparé sur plusieurs clients.

Nous avons donc trouvé une solution à ceci : les RPCs. Les *Remote Procedure Calls* sont des fonctions que rend possible Photons. Comme leur nom l'indique elle permettent d'appeler des fonctions chez d'autres Clients. On reconnaît les fonctions dans le code de la façon suivante :

```
[PunRPC]
void fun(string param)
{
    Debug.Log(param);
}
```

Nous avons donc décider que tout les *gameobject* "appartiennent" au client *Host* de la *Room*. Et tous les autres clients envoient des RPCs à ceux dernier pour interagir avec les tuiles (plus d'information sur les tuiles dans la section Gameplay).



Dans cet exemple, si l'hôte veut modifier la tuile A, il peut faire directement. Cependant si un autre client qui a rejoint la *Room* désire faire la modification, il doit envoyer une requête à l'hôte qui l'exécutera à ça place. Puis la "PhotonView" s'occupera de synchroniser l'objet et donc le client recevra un retour. Lors de l'envoi des RPC nous nous sommes heurté a une difficulté : il est impossible d'envoyé des références interne à Unity (un matériel, GameObjetc, etc..) il faut donc impérativement tout convertir en un Type compatible puis le reconvertir à son arrivée sur le client Master. Nous recherchons actuellement une méthode plus rapide et moins répétitive de conversion.

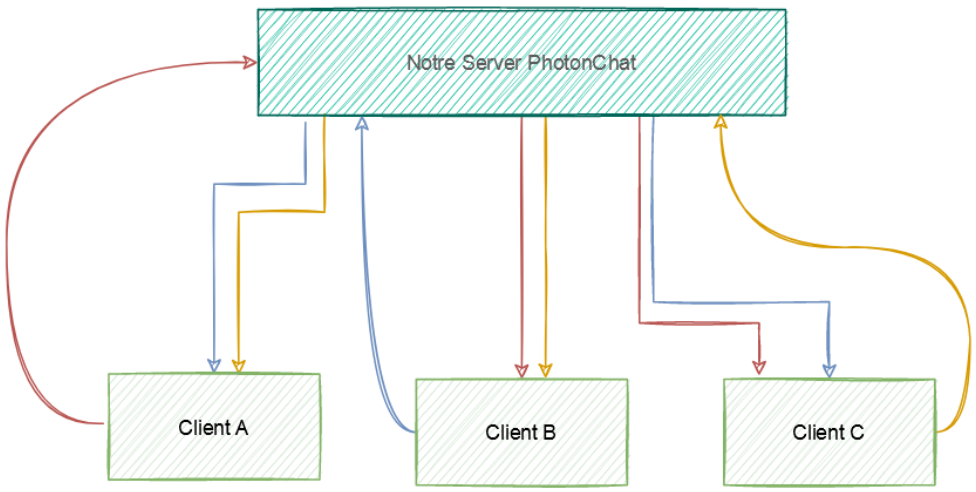
Ce qu'il reste a faire :
Nous avons déjà trouvé les méthodes utilisées pour la connexion et la synchronisation, nous estimons qu'il reste désormais à optimiser ce qui est possible et d'étendre ses méthodes pour couvrir tous les usages du Gameplay.

2.3 Multijoueurs(2) Chat

- Responsable : Jules-Victor
- Suppléant : Adrien

Ce qui a été fait :

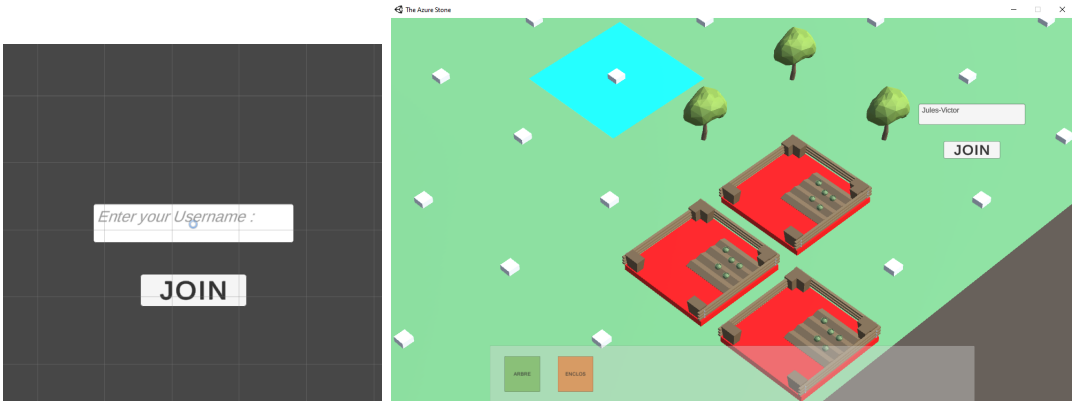
L'implémentation d'un chat a été effectué avec l'aide des plugins Photon Chat. Les plugins Photon Chat sont différents de ceux de Photon. Ce qui permet de pouvoir non seulement de discuter avec la personne présente sur la room de la partie, mais de pouvoir interagir avec tous les autres joueurs présents sur le jeu en temps réel. En effet, cette fonctionnalité est un point essentiel pour permettre un échange entre les joueurs lorsqu'un marché serra implémenté dans un futur proche.



Le chat est un des éléments du multijoueurs avec une interaction joueur-serveur-joueur. Pour l'instant le multijoueur est limité à une seule interaction celle du joueur-seveur-joueurs.

Puisque le plugin utilisé est un plugin différent du plugin du multijoueur, il faut se connecter en rentrant son nom. Ce qui permet ensuite au joueur de pouvoir interagir avec les autres joueurs. La fenêtre du chat prenant une place plutôt importante sur l'écran de jeu, il est possible de le faire disparaître.
Toutes les interactions du chat sont gérer par un unique script contenant toutes les fonctions : le "ChatManager" ayant comme paramètres, tous les boutons, champs de saisie et textes.

En effet, lorsque le joueur est dans la fenêtre de jeu principale, il peut se connecter une première fois au chat en appuyant sur la touche "T". Se faisant, il peut écrire des messages que les autres joueurs verront et lorsqu'il aura fini ou qu'il voudra faire disparaître le chat il n'aura qu'à appuyer sur la même touche "T". Or cette fois-ci, après la disparition de la fenêtre, il pourra la refaire apparaître, mais sans avoir besoin de se reconnecter.

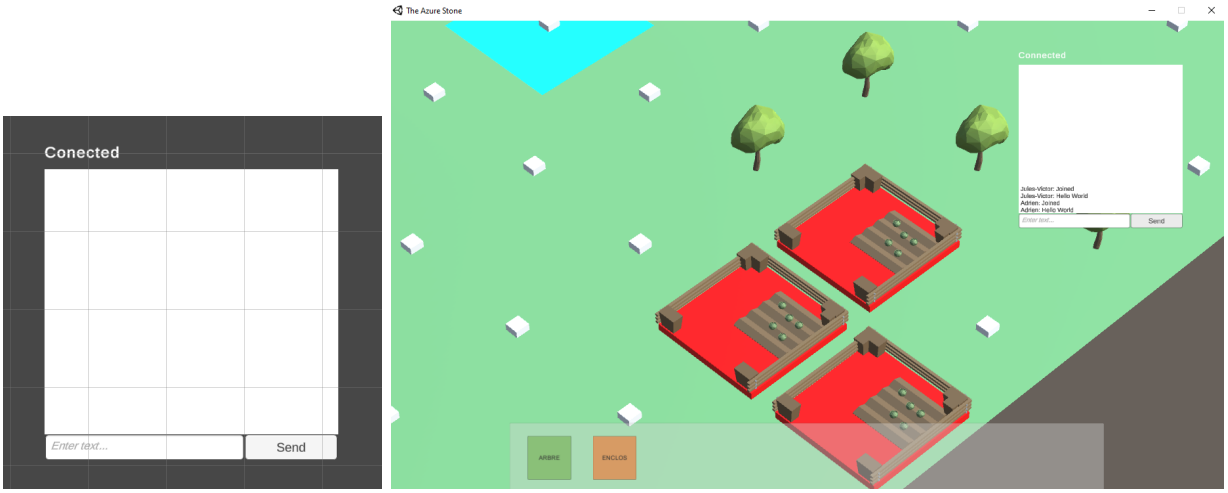


Ici, nous pouvons observer la fenêtre de connexion qui permet de rejoindre le chat global après la pression sur le bouton "JOIN". Le bouton fait appel à une fonction du ChatManager, la fonction "GetConnected". Le joueur est prié de rentrer son nom d'utilisateur pour pouvoir être identifié dans le chat.

```
// This function connect the player to the global chat by creating a new client
// which will use the internal setting of the game as parameters
public void GetConnected()
{
    Debug.Log( message: "Connected");
    chatclient = new ChatClient( listener: this);
    chatclient.Connect(PhotonNetwork.PhotonServerSettings.AppSettings.AppIdChat,
        PhotonNetwork.AppVersion,
        new AuthenticationValues(UserName.text));
}
```

Ici la fonction Getconnected

Cette fonction va permettre au joueur de se connecter au serveur de global du chat en créant un nouveau client. Ce nouveau client va se connecter avec en paramètre passé les paramètres de Photon présents dans le jeu, ce qui permet donc de se connecter uniquement aux joueurs possédant l'identifiant de PhotonChat.



Ici, nous pouvons observer la fenêtre de connexion qui permet d'écrire sur le chat ainsi que de recevoir les messages des autres joueurs. Cette fenêtre est découpée en quatre parties. La première qui est le statut de connexion au serveur chat, la deuxième étant la zone de chat contenant pour chaque message l'envoyeur suivie de son message.

Ce qu'il reste à faire :

L'avancement actuel du chat permet d'interagir avec tous les joueurs possédant le jeu. Or, à terme, le chat permettra d'envoyer des messages privés aux joueurs avec une liste des joueurs connectés. L'interface sera à revoir. En effet pour l'instant, bien que le fond blanc permette de bien distinguer les zones de textes et du reste du jeu. IL faut revoir le graphisme pour qu'il soit plus proche du level design du jeu.

2.4 IA

- Responsable : Bob
- Suppléant : Jules-Victor

Ce qui a été fait : Nous n'avons pas encore commencé la création d'IA, nous n'avons que des recherches et une voie de développement. Sois 2 types d'IA, des IA de PathFinding ainsi qu'une IA "d'environnement". C'est-à-dire que le jeu est soumis à des événements qui sont déterminés par une IA selon la situation de la partie.

Ce qui doit être fait : Comme ce qui a été expliquer plus haut, les IA sont à développer. Ces IA sont donc celles des "tracteurs" ainsi que des "ouvriers agricoles". Ces deux IA sont des IA de PathFinding, c'est-à-dire que ces IA doivent rechercher le chemin le plus court entre un point d'origine et un point de destination. Pour cela, le jeu est sur carte rempli de tuiles carrés, sur une majorité de tuiles, sera présent un bâtiment construit par le joueur et dont chaque bâtiment sera rempli de collision. Ce qui nécessitera a nos algorithmes de recherches de chemins de prendre en compte ces obstacles et donc de ne pas avoir comme chemin un trajet direct entre le point de départ et d'arriver traversant les bâtiments. On utilisera comme support pour créer nos algorithmes de l'algorithme de Dijkstra ainsi que de l'algorithme de A*(dit A étoile).

Le premier se base sur les graphes pondérés et leurs parcours entre deux sommets, ici, nous n'aurons pas de graphes, mais un système de coordonnées avec des coordonnées bannies, celles présentes dans les bâtiments ce qui permet d'ajouter de la complexité au parcours.

Le second se base sur la vitesse de calcul plutôt que l'exactitude du résultat et est donc très utilisé dans les jeux vidéos, nous nous concentrerons surtout sur celui-ci pour la réalisation de nos algorithmes. C'est un algorithme lui aussi basé sur les graphes, mais qui en cherche un le plus rapidement possible bien que ce ne soit pas le chemin le plus optimal.

L'IA d'environnement prendra elle en compte les différents éléments présents sur la partie, en effet pour vendre ses ressources le joueur passera par un bâtiment de vente. Pour réguler tout problème d'équilibrage et empêcher l'exploitation d'une ressource, cette IA examinera la quantité de production et le nombre de vente d'une ressource.

2.5 Gameplay

- Responsable : Jules-Victor
- Suppléants : Adrien, Bob

Ce qui à été fait :

Actuellement lors du lancement d'une map le client Host génère une grille avec différentes tuiles. Chaque tuile possède donc une position précise et va contenir toute l'interaction. Le ou les enfant d'une tuile va contenir les bâtiments et autre information relative à la case.

Le joueur à déjà la possibilité de choisir dans une bar d'items un bâtiment et de l'appliquer a une case.

Lors du passage de la souris sur une tuile, une couleur bleue apparaît à titre indicatif.

Ce qui doit être fait :

Pour ce qui est de la suite du jeu plusieurs notions essentielles qui doivent être implémenter.

- Inventaire :
  Nous avons pour commencer l'inventaire, qui sera un gros registre contenant toutes les ressources détenues par la ferme et auxquelles chaque joueur présent sur la partie pourra accéder. Ces ressources seront synchronisées entre chaque joueur par le biais du serveur Photon.
- Production :
  Ensuite, nous avons la production de ressources qui se fait par l'intermédiaire de bâtiments construits tel que les champs ou bien les enclos.
- Ressources :
  Ce qui suit la production, est les ressources. Nous avons donc comme première ressource, les graines, élément majeur dans ce jeu. Il y aura les graines les plus connues comme le blé, le riz, le maïs et les carottes, car elles sont facilement identifiables qu'importe l'échelle de résolution, pratique lorsque la carte est grande et que la caméra est éloigné du sol. Ensuite, nous avons les animaux de la ferme, c'est-à-dire des vaches, des cochons ainsi que des poules.

- IA :
  Enfin, nous avons les IA qui ont été décrites plus haut tel que les ouvriers agricoles et les tracteurs qui remplacent le joueur en effectuant les mêmes actions que le joueur, mais plus vite et constamment. Ces IA seront essentiellement des IA de PathFinding qui devront se diriger de la ferme au lieu de production choisi par le joueur.
  - Ouvriers agricole :
    Cette IA de PathFinding remplace le joueur dans tous les moyens de production. Sois la culture des graines ainsi que l'élevage des animaux. L'ouvrier agricole se déplace à la même vitesse que le joueur, mais produit un peu plus vite que celui-ci.
  - Tracteur :
    Le tracteur est lui aussi une IA de PathFinding, mais bien que l'ouvrier agricole fait la même chose que le joueur. Le tracteur lui ne fait que la culture de plante dans les champs en étant bien plus rapide que le joueur niveau déplacement et production.
  - IA d'environnement :
    Cette IA ne sert qu'à l'équilibrage de l'économie permettant à chaque partie d'être unique. Elle prend en compte les productions et ventes des joueurs sur la partie.

2.6 Level Design

- Responsable : Amine
- Suppléant : Jules-Victor

Ce qui a été fait :
Concernant les assets crées jusqu'à présent pour notre jeu, ils ont été modélisés et texturés sur logiciel de 3D pour la plupart, certains ont été récupéré sur Internet notamment les assets à maillage complexes comme les animaux (par soucis d'efficacité et de rapidité). Au fur et à mesure du temps, nous espérons que tous les assets aspireront à être entièrement designer par notre équipe.
Jusqu'à présent les assets ont été créé en jonglant entre Maya: un logiciel de 3D réputé pour ses images de synthèse (utilisé notamment au cinéma) et Blender: un logiciel de modélisation gratuit, très connu dans le domaine de la 3D.

La création de chaque asset suit le modèle suivant :

- modélisation
- sculpting (facultatif)
- création des UVs
- Texturing/Shaders
- Animation (facultatif)

Le design des assets est inspiré du célèbre jeu de la franchise rockstar : Red Dead Rédemption: un jeu mettant en scène des cowboys au Etats-Unis en 1899, le thème colle donc à notre idée de jeux qui a comme objectif principal de faire prospérer une ferme. Ainsi le style des assets est plutôt ancien (type XXe siècle), le bois sera préféré au plastique, les machines auront un design type « fin révolution industrielle».

Quelques comparaison entre nos asset et le jeux :

Figure 1: A gauche des rendus de nos asset et a droite des screen issue de RDR2

Vers la seconde soutenance: En vue de la seconde soutenance, les objectifs pour le level design seront d'une part d'avancer dans la modélisation, la création et la diversification des assets, d'autre part d'introduire les premières animations d'assets dans l'optique de vivifier notre gameplay puis d'envisager des cinématiques. Attention pour des questions de fluidité dans les tests la plupart des asset ne sont pas encore intègres au jeu.

2.7 UI

- Responsable : Amine
- Suppléant : Adrien

Ce qui a été fait :

Actuellement, les interfaces utilisateur sont minimales, elles ne servent uniquement pas à tester le fonctionnement des divers ajouts. On peut déjà compter une interface pour :

- Chargement de jeu
- Lobby (choix de la room)
- Bar avec les différents bâtiments
- Connexion au chat
- Chat

Ce qui reste a faire :

Certaines interfaces sont encore manquantes et une interface finales et plus adaptés dans le jeu est nécessaire. De plus, il faut améliorer et peaufiné chaque interface déjà existant pour les rendre le plus agréable possible a l'utilisateur. Un thème reste à déterminer pour chacune des fenêtres que l'on a créé.

2.8 Siteweb

- Responsable : Adrien (partie serveur)
- Suppléant : Bob (partie design)

Ce qui a été fait :

Le site web est actuellement propulsé par GitHub Pages : c'est une fonctionnalité comprise dans github qui permet de déployer automatiquement du code de manière sécurisé.
Le site peut se trouver à l'adresse suivante : *https://bananasplits.github.io/*
Nous avons décidé de coder l'intégralité du site pour avoir totalement la main dessus. Nous ferons donc usage ici de langage tel que l'HTML, CSS, et JavaScript, indispensable pour un site web.

Donc, le design du site web reprend le design de la carte du jeu ainsi que ces principes. C'est-à-dire que le site web est en forme de losange avec des tuiles cliquable permettant d'accéder aux informations telles que la présentation du groupe, du jeu ainsi que d'autres informations que nous implémenterons.

Le site à une architecture atypique avec le fait de cliquer sur les tuiles pour accéder aux diverses informations et au fait de scroller avec la souris pour quitter ces menus.

Ce qui reste à faire :

Continuer la construction du site pour obtenir un résultat propre et qui permet de présenter rapidement notre projets.
Pour plus de professionnalisme, il sera éventuellement possible de déployer un serveur DNS avec une redirection de type CNAME sur l'adresse existante.

3 Conclusion

Actuellement, le projet est dans un état très primaire, mais de solides base ont déjà été poser. Notamment avec un multijoueur fonctionnel qui permet d'ores et déjà de connecter plusieurs clients entre eux et de synchroniser la carte. Les premiers assets ont été modéliser qui permette de mieux nous situer dans l'UI. Le gameplay n'est encore qu'à ces balbutiements, nous avons comme premières interactions le fait de faire appaître des bâtiments sur la carte avec synchronisation entre les joueurs présents dans la room qui peuvent eux aussi ajouter des bâtiments.
Par rapport à ce qui était prévu initialement au cahier des charges nous pensons avons pris un peu du retard sur le LevelDesign et l'interface Utilisateur, mais maintenu voir devancer le reste des objectifs.
Nous possédons déjà une bonne idée du résultat recherché et des différents moyen d'y parvenir nous avons une bonne ligne directrice dans ce à quoi va ressembler le jeu. Nous avons en interne un google docs rassemblant toutes les informations lieux au futur du jeu sur lequel nous interagissons en y commentant nos capacités à la réalisation des tâches qui nous ont été donner.

4 Bibliographie

Quelques références qui nous on permis d'arriver jusqu'ici :
https://doc.photonengine.com/en-us/pun/current/gameplay/rpcsandraiseevent
https://doc-api.photonengine.com/en/pun/v2/class_photon_1_1_pun_1_1_photon_view.html
https://www.photonengine.com/en-US/Chat
https://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra
https://fr.wikipedia.org/wiki/Algorithme_A*