Rapport de Soutenance 2

FQTM

Mathieu Malabard
Maxence Gay
Baptiste Daumard
Téo Le Vern

SOMMAIRE

1 Introduction

Notre objectif durant cette deuxième période de travail a été de faire passer notre jeu d'un état de prototype abstrait, avec seulement les bases du jeu implémentées, à un stade plus avancé se rapprochant de notre objectif final.

De plus, depuis la première soutenance nous avons bien assimilé les bases de Unity et cela nous à permis d'être plus efficaces dans les tâches à faire et de savoir d'où viennent souvent les bugs.

Enfin, nous avons, au contraire de la première soutenance, prévu le code pour qu'il soit adapté au multijoueur. Cela a permis de ne pas avoir à refaire tout le code à la fin, ce qui avait été notre plus grosse erreur pendant la première période de travail.

2 Bibliographie

Bâtiment :

- Youtube : Code Monkey "Grid System in Unity (Heatmap, Pathfinding, Building Area)" https://www.youtube.com/watch?v=waEsGu–9P8
- Youtube : Unity Tutorial - Placing objects on a grid https://www.youtube.com/watch?v=D9ZU0mfuk

Inventaire :

- Youtube : InScope Studios "Unity RPG Tutorial - Inventory UI" https://www.youtube.com/watch?v=XhQ-hNbi-Lo
- Youtube : INVENTORY UI - Making an RPG in Unity (E05) https://www.youtube.com/watch?v=w6_fetj9PIw

Arbre de technologies :

- Youtube: Code Monkey Simple Skill Tree https://www.youtube.com/watch?v=_OQTTKkwZQY&t=883s
- Youtube: exemple d'un jeu indépendant pour l'arbre de technologie https://www.youtube.com/watch?v=kmcPaQUdL7I

IA :

- Youtube : Brackeys "2D PATHFINDING - Enemy AI in Unity" https://www.youtube.com/watch?v=jvtFUfJ6CP8
- Youtube : Code Monkey "The PERFECT Pathfinding!" https://www.youtube.com/watch?v=46qZgd-T-hk
- Youtube : Sebastian Lague "A* Pathfinding (E01: algorithm explanation)" https://www.youtube.com/watch?v=-L-WgKMFuhE

Multijoueur :

- Youtube : Lignus "Je fais les bases du networking - Unity & Mirror 2019" https://www.youtube.com/watch?v=WqJ3RlHEB4E
- Forum Unity/Mirror

Site web :

- OpenClassrooms : "Site web avec HTML 5 et CSS3" https://openclassrooms.com/fr/courses/1603881-apprenez-a-creer-votre-site-web-avec-html5-et-css3/1604192-decouvrez-le-fonctionnement-des-sites-web
- Youtube : Graven "CREER UN SITE ? " https://www.youtube.com/watch?v=J9w-cir5a6U&t=2s

3 Reprise de la dernière soutenance

3.1 Rappel du jeu

From Québec to Mars est un jeu de type Bac à sable où l'objectif est de créer une usine automatisée afin de produire des ressources en grande quantité. Ces ressources pourront être utilisées pour créer des objets ou pour rechercher des technologies.
Cependant, le joueur n'est pas seul sur Mars. Il sera régulièrement confronté à des vagues d'attaque d'extraterrestres agressifs. Le joueur devra donc s'équiper pour survivre.
Le jeu sera accessible en multijoueur en coopération afin d'obtenir de l'aide dans ce milieu hostile. Les joueurs partageront leur base ainsi que leurs technologies.

L'objectif final est de débloquer les technologies nécessaires à la création de la poutine dorée et de la créer évidemment.
Évidemment, les technologies nécessaires demanderont des quantités conséquentes de ressources.

3.2 Avancement

A l'issue de la première période de travail, beaucoup d'éléments du jeu marchaient, mais seulement indépendamment les uns des autres, l'exemple le plus concret étant le multijoueur.
Aujourd'hui, les éléments marchent ensemble, en multijoueur. Il est possible pour le joueur de construire des bâtiments, d'utiliser un arbre de technologie, de récupérer et lâcher des objets au sol, et tout cela compatible avec le réseau.
De plus, des ennemis sont capables de trouver leur chemin jusqu'au joueur en évitant les bâtiments et de lui faire perdre de la vie.

Le système de bâtiment a été implémenté de façon a ce que l'on puisse le relier le plus facilement possible avec la génération de carte et prévoir au mieux les problèmes qui pourraient arriver avec des bâtiments spéciaux (par exemple la foreuse qui ne peut être construite que sur du minerai).

L'inventaire a été amélioré. Les joueurs peuvent désormais "empiler" des objets dans des cases d'inventaire. De plus, l'inventaire est maintenant fonctionnel en réseau ce qui est une très bonne nouvelle. En effet, le multijoueur a posé énormément de problèmes lors des fusions des différentes implémentations mais maintenant que l'inventaire est compatible avec le système de réseau, nous avons la preuve que nous avons compris les bases du multijoueur. Nous sommes donc assez confiants pour la suite du projet au niveau du réseau.
Enfin, l'inventaire arbore une nouvelle apparence. Il est maintenant plus grand et n'apparaît que lorsque le joueur le demande.

L'arbre de technologie est fonctionnel mais il n'est absolument pas terminé. Il est compatible avec les placements de bâtiments, il peut changer les paramètres du joueur et est commun à tous les joueurs, étant donné qu'il s'agit d'un jeu de coopération. Cela signifie que les joueurs partagent les mêmes améliorations. Lorsqu'un joueur débloque une technologie, tous les autres la débloque aussi.

Les ennemis n'apparaissent pas encore automatiquement, mais ils sont capables, grâce à un système de pathfinding, de trouver leur chemin, à travers un labyrinthe de bâtiments, jusqu'au joueur
Le site internet a été créé et il est déjà bien avancé. Un emplacement a été prévu pour le trailer et des informations sur le jeu et l'équipe de développement y figurent déjà.

Il y a cependant un point négatif, il était précisé dans le planning que l'implémentation d'un système électrique ainsi que le système de fabrication étaient prévues pour fin Avril. Cependant, la complexité du système multijoueur a demandé énormément de temps et il n'a pas été possible pour Mathieu (en charge du multijoueur), de développer ces systèmes. Pour le moment, ils sont reportés aux mois à venir. Ce temps perdu est aussi du temps gagné dans les résolutions prévues en juin.

4 Tâches

4.1 Bâtiments

Les bâtiments sont les objets qui permettront aux joueurs d'évoluer dans le jeu.
Ils sont primordiaux à l'automatisation de l'usine et essentiels à la découverte de nouvelles technologies

4.1.1 Différents Bâtiments

Les Bâtiments sont des éléments centraux du gameplay de notre jeu. Nous pouvons les débloquer via l'arbre de technologie, puis ils peuvent être créés dans les machines d'assemblage et finalement devenir accessibles pour la construction dans un menu de l'inventaire. Ceux que nous avons implémentés pour l'instant sont: le four, le coffre, la foreuse, la machine d'assemblage et la base de départ. Nous allons décrire dans cette partie leurs différents fonctionnalités:

- Le Four: comme son nom l'indique il permet de cuire des ressources pour en fabriquer des nouvelles. Lorsque l'on interagit avec lui notre inventaire ainsi que le menu du bâtiment apparaissent à l'écran. Le menu est composé de deux cases ou le joueur peux placer des objets, dans la première on peut placer la ressource à cuire et le combustible dans la deuxième. Les ressources brûlées sont détruites et une nouvelle est créée dans une troisième case.
- Le Coffre: également explicites, il permet un prolongement de l'inventaire en stockant des ressources que le joueur n'a pas besoin. Chaque coffre possède une mémoire qui se charge à chaque ouverture. Car si on garde les objets comment dans l'inventaire, tous les inventaires des coffres seront communs et ce n'est pas ce que nous voulons.
- La Foreuse: uniquement plaçable sur des filons de ressources. Le bâtiment générera la ressource primaire sur là qu'elle il est placé. La vitesse de production des minerais peut être amélioré. Les minerais produit seront déposés directement sur la carte à un point de sortie défini. Ils pourront ensuite être récupéré par un tapis roulant.
- La Machine d'assemblage: Ce bâtiment va fabriquer en continu et tant qu'il est approvisionné un objet que le joueur choisit. Les différentes recettes pourront être débloqué via l'arbre de technologie. La fabrication est automatisée, c'est à dire que des objets ou des ressources pourront être amené avec des tapis roulants.
- La Base de départ: Elle permettra aux joueurs de débloqué des points de compétence pour l'arbre de technologie, pour cela les joueurs devront ressembler des ressources demandées.

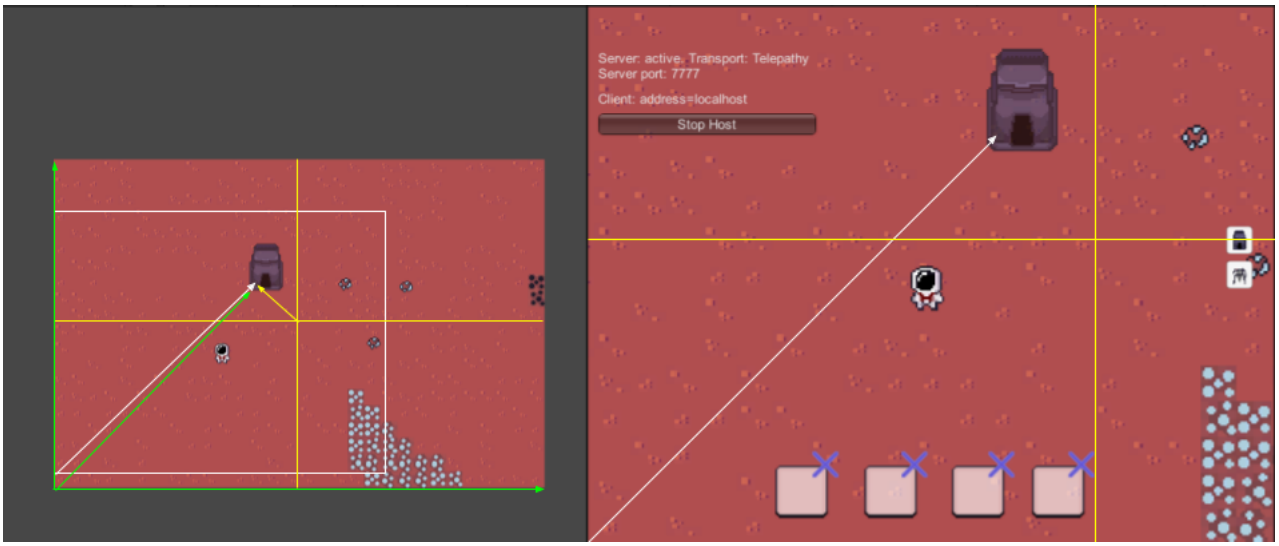
A gauche un four, en haut une foreuse et à droite une machine d'assemblage.

4.1.2 Placement des Bâtiments

Le placement des bâtiments se fait selon une grille. Il a donc fallu créer cette grille. Cela se traduit par une classe nommée "gridMap". Cette classe est composée de plusieurs fonctions qui mettent en relation les différentes coordonnées possibles :

- Les coordonnées réelles, les coordonnées dans le monde.
- Les coordonnées sur la grille, uniquement des entiers et avec un possible décalage par rapport aux coordonnées réelles.
- Les coordonnées de la souris sur l'écran, qui sont différentes des coordonnées réelles.

Ces fonctions permettent donc de connaître, par rapport à la position de la souris, les coordonnées du bâtiment à la fois sur la grille et dans le monde.



Placement d'un bâtiment

A gauche de l'image se trouve la scène sur laquelle le joueur est, et à droite l'écran du joueur. D'abord, il faut calculer les coordonnées de la souris sur l'écran. Le rectangle blanc correspond à l'écran du joueur et le vecteur blanc à la position de la souris.

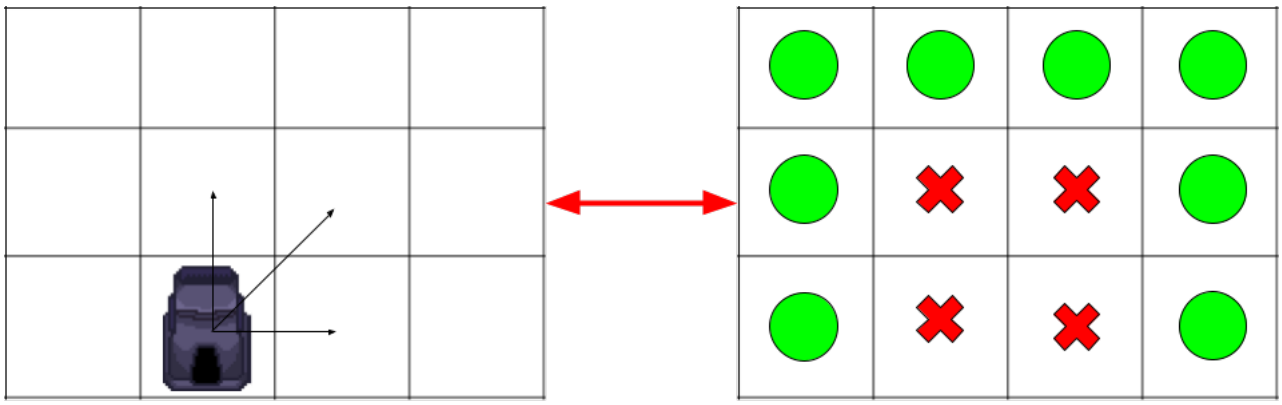
Il faut ensuite convertir ce vecteur en coordonnées réelles. Les axes jaunes correspondent à l'axe des abscisses et l'axe des ordonnées du monde. Le vecteur jaune est déduit par la transformation du vecteur blanc en coordonnées sur le monde.

Maintenant, il faut transformer ce vecteur en coordonnées sur la grille. Les axes verts sont les abscisses et ordonnées de la grille. On remarque que l'origine de la grille n'est pas au même endroit que l'origine du monde, il y a un décalage pour centrer la grille sur le monde. On obtient le vecteur vert à partir du vecteur jaune et on peut remarquer qu'il ne pointe pas tout à fait au même endroit, il y a eu un arrondissement des coordonnées à l'entier inférieur.

Enfin, il faut retransformer ces coordonnées grille en vecteur du monde. Cela peut sembler redondant, mais il faut passer par les coordonnées de la grille pour pouvoir empêcher le joueur de construire des bâtiments les uns sur les autres.

Pour cela, la grille possède deux tableaux de même dimension que la carte:

- Buildings, qui contient le bâtiment posé sur chaque case de la grille
- Building_bool, qui détermine si une case est libre ou non avec des booléens



Sur un petit exemple comme au-dessus, lorsque l'on veut placer un bâtiment, il faut connaître sa hauteur et sa largeur (2-2 dans ce cas).
Le tableaux Building ne note que la case inférieure gauche sur laquelle le bâtiment a été posé et le tableaux Building_bool lui, récupère la hauteur et la largeur du bâtiment pour déterminer les cases que le bâtiment occupe.

Ainsi, lors du placement d'un bâtiment, le jeu va vérifier grâce au tableau de booléens, si toutes les cases que le bâtiment occuperait sont vides.
Le tableau Building, lui, servira pour charger une partie.

Certains bâtiments sont spéciaux. Par exemple, la foreuse ne peut se placer que sur des cases ressources pour des raisons évidentes. Il a donc fallu créer un troisième tableau qui récupère le type de sol sur chaque case. Une fonction va ensuite vérifier si, parmi toutes les cases que

la foreuse occuperait, il se trouve au moins un minerai.

La sélection des bâtiments se fait par l'intermédiaire de boutons. Chaque bouton appelle une fonction avec, comme argument, un entier qui correspond à l'identité du bâtiment. La fonction va chercher dans un tableau de référence, le bâtiment qui correspond. Le bâtiment d'identité n est placé à la $n^{ième}$ place du tableau

Chaque bâtiment "existe" en deux versions:

- Un bâtiment "Final", celui qui va être posé sur toutes les scènes et avec lequel les joueurs vont pouvoir interagir.
- Un bâtiment "Transparent". Ce bâtiment n'apparaît que sur la scène du joueur qui est en train de construire. Il suit le curseur de la souris et se teint en rouge si ce n'est pas possible de construire de bâtiment sur la case courante.

4.1.3 Interaction avec les bâtiments

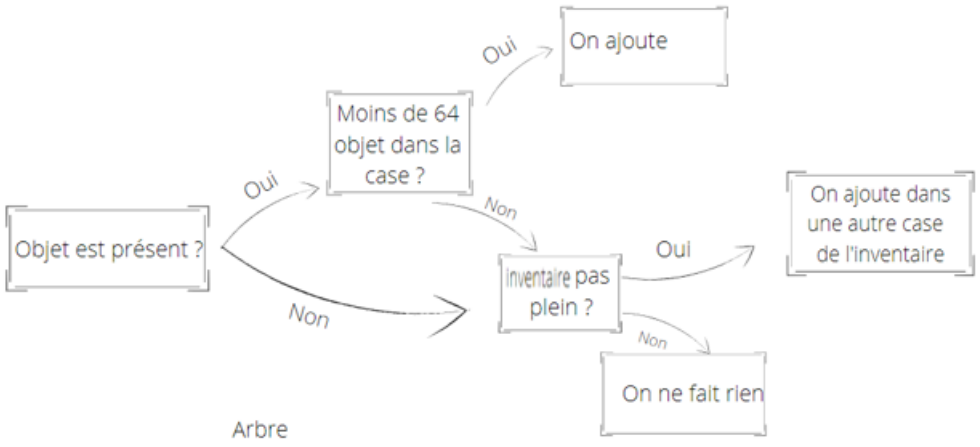
Pour qu'un joueur puisse interagir avec un bâtiment il doit être à une distance minimale de celui-ci. On vérifie cette distance grâce un cercle de collision et des fonctions qui réagissent à l'entrée ou sortie du joueur dans le cercle. Si le joueur est entré dans le cercle, il peut presser la touche "F" pour lancer l'interaction avec le bâtiment, Une fois l'interaction lancée, le menu du bâtiment s'affiche et le joueur ne peut plus se déplacer. Chaque bâtiment va, lors de l'interaction, faire apparaître une interface différente en fonction de son type.
Pour afficher le menu, on utilise la fonction "Display" que nous avons codé, elle change l'alpha (soit l'opacité) et le blocksRaycast (un booléen qui permet ou non d'interagir avec l'UI).

4.2 Menu en jeu

Les menus en jeu correspondent aux différentes interfaces auxquelles les joueurs auront accès. Elles doivent être intuitives et compréhensibles et permettre au joueur d'interagir facilement avec l'environnement de jeu.

4.2.1 Inventaire

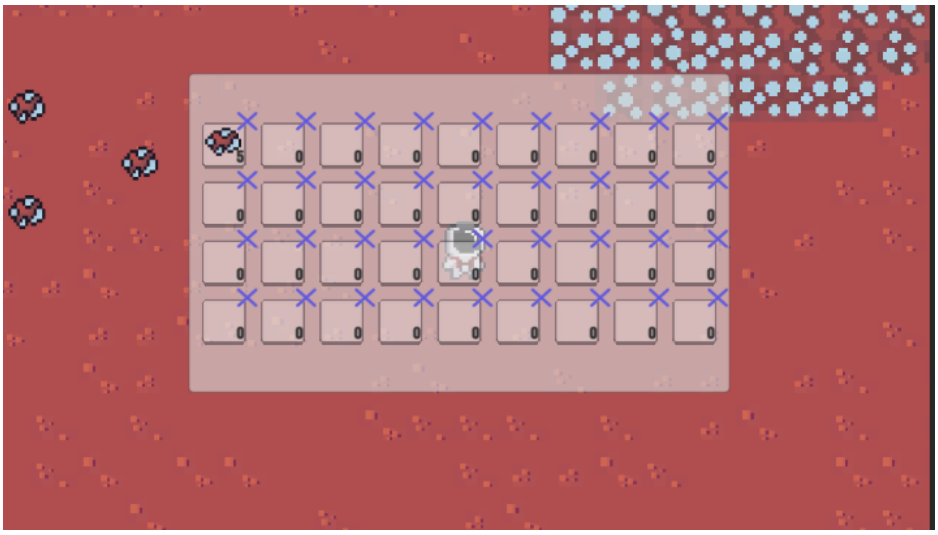
Pour la partie inventaire , nous avons essayé dans un premier temps d'ajouter le stacking d'objet, cela permettra d'avoir plusieurs objets dans une case et non pas un seul. Pour cela on a ajouté à chaque case une valeur. Cette valeur correspondra au nombre d'itération de l'objet dans l'inventaire. Pour cela le code marchera de la façon suivante :



Arbre

Nous avons choisi 64 pour le nombre maximum d'itération de l'objet dans l'inventaire car cela est utilisé sur Minecraft, et notre inventaire se voulait inspirer de son inventaire qui est très intuitif.
Puis ensuite nous avons également mit de quoi jeter l'item à terre. Cela vérifie dans un premier temps que l'objet à plus de 1 item et dans ce cas cela va seulement réduire le nombre d'item de 1, et dans le cas où il n'y a qu'un seul item, cela va détruire l'item de la case de l'inventaire.

Dans un second temps, nous avons rajouter un panel pour l'inventaire, cela permet donc de ne plus avoir quatre cases comme avant mais un inventaire entier pour stocker bien plus d'item. Pour y accéder, cela se fera par la touche "E" du clavier. L'inventaire ressemblera donc maintenant à cela :



Difficultées rencontrées :
Pour cet inventaire nous avons surtout rencontré des difficultés pour le stacking (empilement) d'objet . Premièrement, affecter une variable à une case de l'inventaire peut sembler simple mais nous avons dû créer de nombreuses variables pour seulement ajouter un item dans l'inventaire.
Deuxièmement, un problème a été rencontré lorsque l'on jetait un item à terre. Lorsque nous arrivions à 1 item restant celui-ci ne se détruisait pas et pouvait même aller dans les négatifs, et donc créer un bug de duplication. Pour résoudre cela, nous avons créé des cas spéciaux lorsqu'il n'y a plus d'items dans la case, cela détruit le "GameObject" de l'item.

4.2.2 Menu de paramètres graphiques

De nouveaux réglages ont été ajouté au menu:

- La sélection ou non du mode plein écran
- Changer la résolution
- Changer la qualité des graphismes
- Mode daltonien

4.2.3 Arbre de technologies

L'arbre de technologie est la façon dont les joueurs progressent en débloquant du nouveau contenu. Dans celui-ci on peut débloquer des bâtiments, des crafts et améliorer les caractéristiques du joueur ou des bâtiments. L'arbre est accessible en pressant la touche "t". Une fois ouvert on peut voir Les différents boutons pour débloquer des technologies ainsi que le nombre de points de technologies disponible.(Voir Capture ci-dessous)
Pour débloquer des points de technologie les joueurs devront réunir dans la base de départ un certain nombre de ressources, si le nombre de points est insuffisant les joueurs ne pourront pas débloquer de nouvelles technologies.

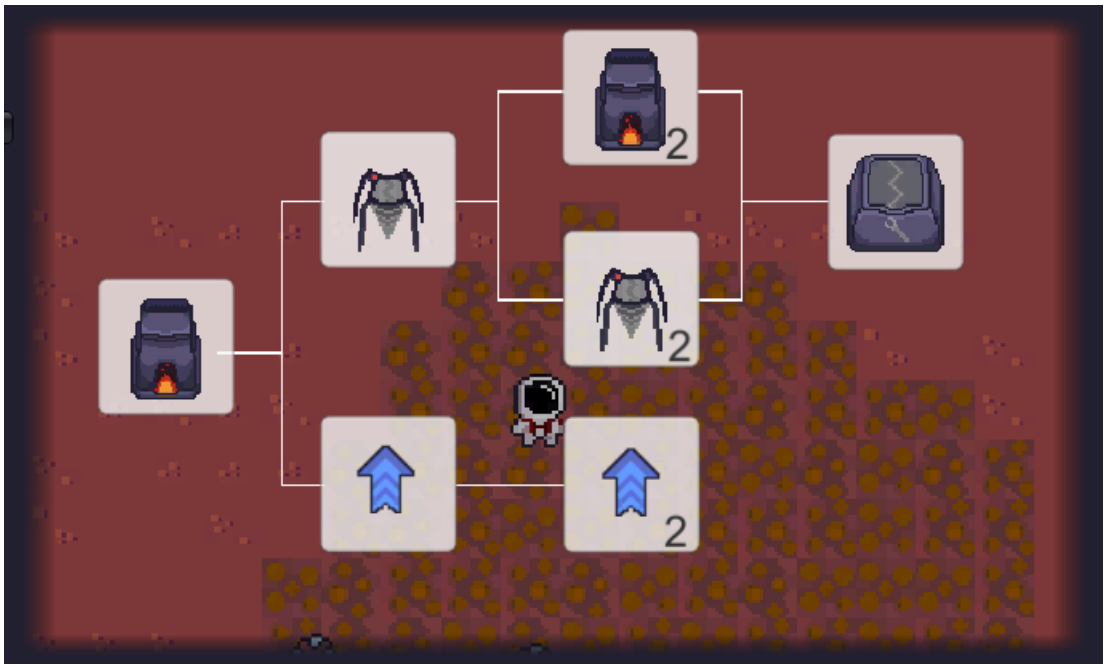
Le paragraphe qui suit vous expliquera comment marche le processus pour débloquer une technologie. Tout d'abord, l'ensemble des technologies sont listée dans une énumération(enum). les technologies débloquées sont quant à elles contenu dans une liste, cette structure de donnée a été privilégier car il est très facile de savoir si un élément est présent et donc si un joueur peut utiliser une technologie.
Chaque technologie possède un bouton pour être débloqué, lorsque l'on clique sur un bouton on regarde si la technologie est débloquée: car on ne veut pas ajouter plusieurs fois une technologie à la liste des technologies débloquées, si la technologie nécessite une ou des technologies prérequis, on regarde si elles sont débloquées. Une fois tous les tests passés, suivants le type de technologie débloquer l'action change:

- Si c'est un bâtiment: le joueur possède un tableau de booléen où les index sont corrélés avec la liste des bâtiments. Donc lorsqu'on débloque un bâtiment, la case du tableau a l'index correspondant au bâtiment débloqué passe à vrai. Quand on veut construire un bâtiment on regarde si la variable à l'index correspondant.
- Si c'est une amélioration de statistique: on appelle simplement une fonction qui modifient les valeurs correspondant à la statistique voulue.
- Si c'est un craft: Comme pour un bâtiment on passe par une liste de booléen.

Les icônes des technologies qui peuvent être débloqué (Les technologies adjacentes aux dernières débloquées) sont grisées.
Les icônes des technologies non déblocables sont quant à elles toutes foncé.
Les technologies déjà débloquées ont une petite marque pour être différencier. De plus l'image des liens entre les technologies change lorsque deux technologies qui sont débloquées change et devient plus visible. Lorsque le joueur débloque une technologie, il, faut donc modifier tout l'affichage.

Dans son état actuel l'arbre de technologie est complètement fonctionnel, mais il n'est absolument pas dans sa forme finale. Nous devrons rajouter d'autres technologies, peut-être des modifications graphiques.

Nous réfléchissons à modifier l'affichage: au lieu d'afficher avec L'UI, nous pensons afficher l'arbre de technologie sur une scène indépendantes. Ce changement rendra plus facile l'affichage en permettant de défiler dans le menu.
Ces Petits changements seront rapides car la base de l'arbre de technologie est faite, et implémenter de nouvelles technologies sera très simple.



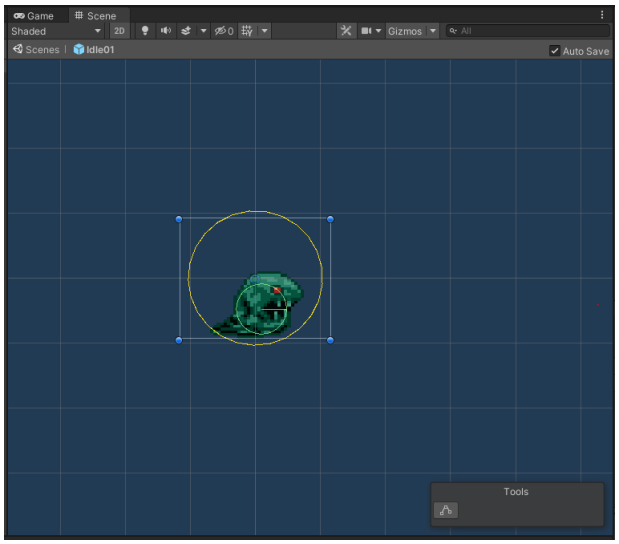
Capture d'écran de l'arbre de technologies

4.3 Intelligence Artificielle

Concernant l'intelligence artificielle, cela va concerner notamment les monstres, extraterrestres hostiles envers le joueur et ses diverses constructions. Le principe est d'implémenter une entité capable de choisir l'action à faire selon la situation et de se déplacer (trouver le chemin le plus court d'un point A à un point B) d'elle même en tenant compte des divers changements qui peuvent intervenir sur la carte (exemple : construction/destruction d'un bâtiment).

4.3.1 Ennemi

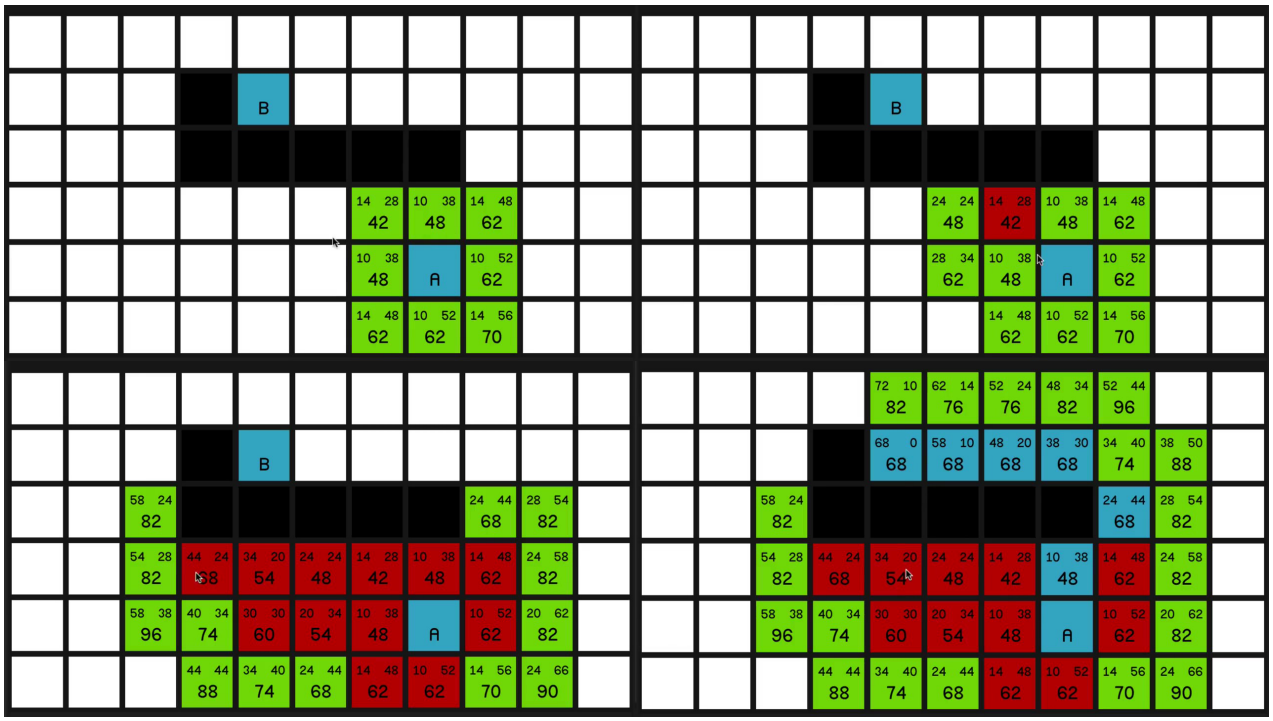
Tout d'abord, l'ennemi doit être implémenté comme un objet indépendant. Il a des propriétés qui lui son propre telle que sa WayPointDistance (Distance de l'objectif) qui permet de définir quand le monstre s'est suffisamment rapproché de sa cible (joueur) pour pouvoir interagir avec celui-ci.



Ensuite, il faut qu'il puisse se déplacer librement sur la carte afin d'atteindre son objectif. Il a pour but de se rapprocher de sa cible pour ensuite interagir avec elle. Pour trouver son chemin dans ce monde, il va s'appuyer sur l'algorithme de "Pathfinding" (Trouver un chemin) visant à trouver le chemin le plus court entre un point A et un point B.

Dans un premier temps, il faut créer une grille (grid) composée entièrement de nœud (node) ayant chacun une position propre par rapport aux autres et une taille identique.
L'entité se trouve au départ sur un nœud précis (Node A). On va donc ensuite pour chaque voisin de ce nœud vérifier d'abord s'il n'y aucun élément gênant sur le nœud (ex : bâtiments). Dans le cas où le nœud est libre on va calculer trois valeurs (Gcost : distance par rapport au nœud de départ (Node A), Hcost : distance par rapport au nœud d'arrivée (Node B) et FinalCost: Gcost + Hcost). Ces valeurs permettent d'estimer le coût de déplacement vers le nœud voisin.

On va récupérer les nœuds voisins avec le plus faible coût (il est possible qu'il y aient plusieurs nœuds avec le même coût de déplacement) et réitérer ce processus sur chacun des nouveaux voisins admissibles jusqu'à avoir un ou plusieurs chemin atteignant le nœud d'arrivée (Node B)



Maintenant, il faut prendre en compte les divers changements sur la carte. La position de la cible (target) du monstre va s'actualiser à chaque déplacement effectué par celle-ci (chaque déplacement du joueur) ainsi le nœud d'arrivée va se mettre à jour en recalculant la trajectoire. Mais aussi à chaque placement d'un bâtiment, il faut changer la grille de nœuds en éliminant les nœuds occupés par un obstacle . Cela se traduit dans Unity par le lancement d'un nouveau "scan" de toute la carte pour détecter les modifications.

Nous souhaitons implémenter différents types d'ennemis qui auront chacun un comportement spécifique:

- Basique: Monstre avec des attributs moyens qui ciblera le joueur
- Rapide: Monstre avec peu de points de vie mais avec une vitesse de déplacement élevée
- Mastodonte : Peu de dégât, beaucoup de points de vie, lent, cible les défenses
- Kamikaze : Très fragile, gros dégâts (explose), cible les bâtiments défensifs
- Boss : Monstre final qui permettra de mettre fin à la partie

4.3.2 Interaction avec les joueurs

Une fois que l'entité a atteint sa cible. Elle va pouvoir interagir avec elle. Dans notre cas, le monstre va frapper le joueur, lui infligeant ainsi des dégâts. Le joueur possède au départ cent points de vie, à chaque coup donné par le monstre, il va perdre 10 points de vie. Si le joueur se déplace, le monstre va continuer à suivre le joueur sans lui donner de coups. Pour que le monstre puisse interagir, il faut que le joueur soit immobile à côté du monstre pendant une courte période de temps. Si le joueur perd tous ses points de vie, il va mourir (représenté par le changement de l'apparence du joueur de blanc à rouge).

4.4 Multijoueur

Le multijoueur est sûrement la partie qui nous a posé le plus de problème. Elle n'est en réalité pas terminée étant donné que toute implémentation doit se faire en multijoueur.

Nous procédons de la manière suivante:

1. Nous implémentons, sur une branche à part, le système (par exemple les bâtiments) en essayant au maximum de prendre en compte le multijoueur
2. Nous fusionnons la branche séparée avec la branche principale et nous réglons les conflits
3. Nous implémentons le multijoueur en réglant les différents bugs rencontrés dans les tests

Seulement, étant nouveau au réseau en général, il a fallu beaucoup de temps pour comprendre les principes du multijoueur.

4.4.1 Principe

Le réseau multijoueur sur Unity est en théorie très simple. Chacun des joueurs possède sa propre scène et le serveur possède une scène à lui. Lorsque l'on veut créer un objet, le déplacer ou changer des variables chez tous les clients, il faut le faire depuis le serveur qui va se charger de transmettre l'information chez tous les clients.

Pour tous les objets qui ne nécessitent pas d'apparaître chez tous les clients comme les objets d'interface du joueur (par exemple le bâtiment transparent), il est possible de les créer uniquement chez le client afin de ne pas surcharger le serveur d'informations inutiles

4.4.2 Inventaire

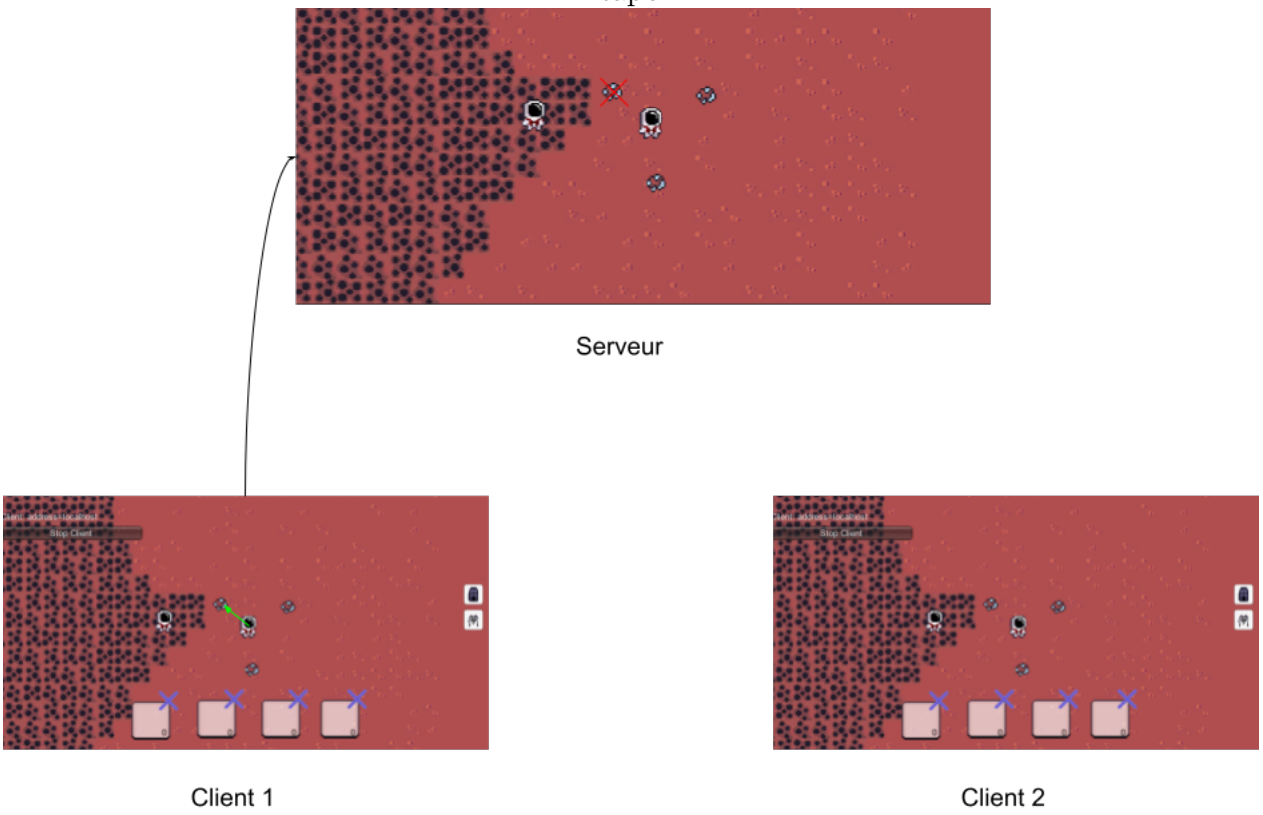
L'inventaire correspond à la première vraie implémentation multijoueur. En effet, le serveur se chargeant seul de coordonner les mouvements et apparitions des joueurs, il n'y avait pas beaucoup de choses à faire avant l'inventaire.

Cet inventaire a posé énormément de problèmes du fait de notre incompréhension du système de réseau. Mathieu s'en est occupé et il lui a fallu environ un mois et demi pour arriver à quelque chose de pourtant très simple.

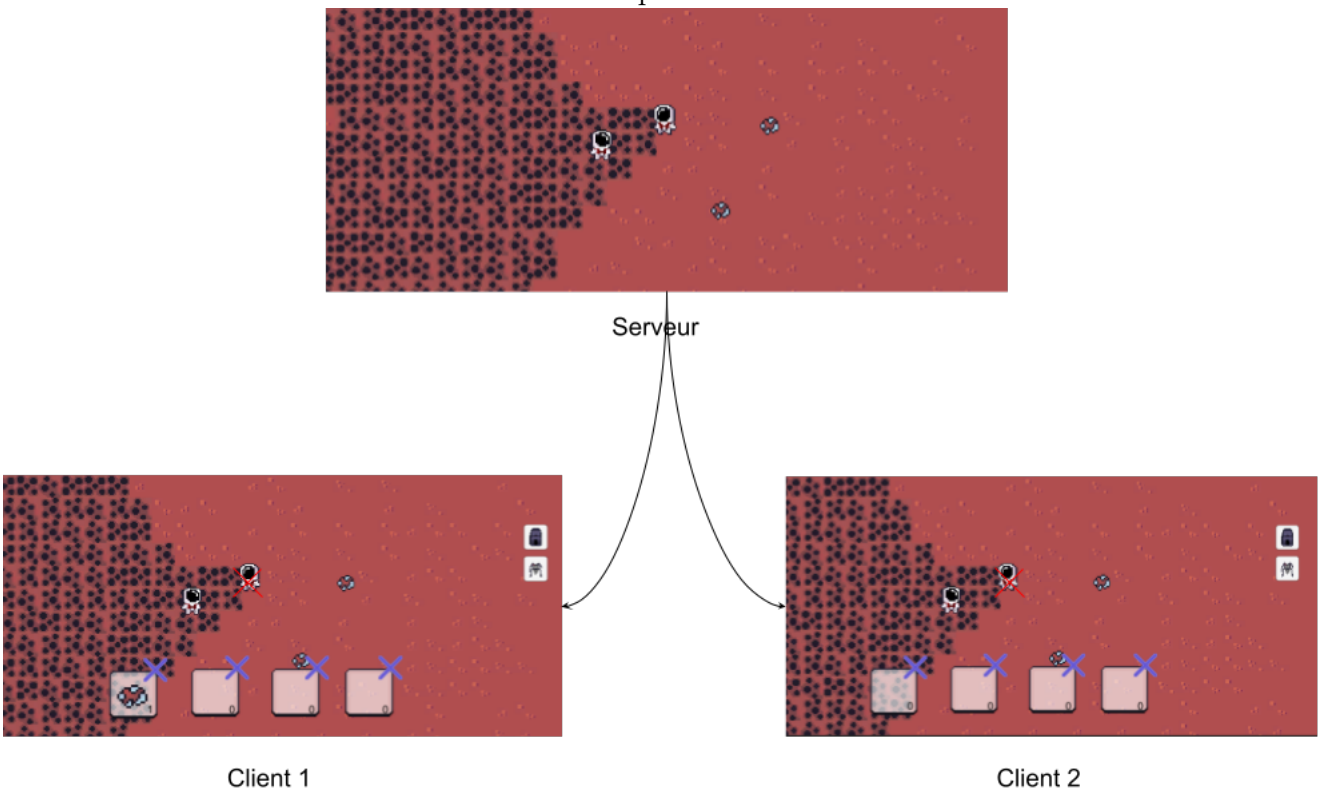
Le problème majeur était le fait que nous ne comprenions pas comment s'articulait la scène serveur par rapport aux scènes clients. Du moment que le joueur était host (à la fois serveur et client) tout marchait mais dès qu'il était client, certaines fonctions ne se lançaient pas et aucune erreur ne s'affichait.

La solution que nous avons trouvée est la même que pour les bâtiments: créer un tableau de référence pour les ressources et leur donner une identité.

Étape 1



Étape 2



La récupération d'un objet s'effectue de la manière suivante:

1. Le joueur détecte un objet "Ressource" au sol
2. Un objet est ajouté dans l'inventaire du joueur (objet interface ou compteur)
3. Le joueur envoie au serveur l'objet ramassé qui doit être détruit
4. Le serveur détruit l'objet sur sa scène et sur chacune des scènes client

Et lorsque le joueur lâche un objet

1. Le jeu cherche l'objet correspondant à l'objet "interface" que le joueur veut lâcher et l'envoie au serveur
2. Le serveur instancie sur sa scène l'objet et l'instancie sur toutes les scènes clients
3. Un objet est supprimé de son inventaire (compteur ou objet interface)

4.4.3 Bâtiments

Le placement de bâtiment s'effectue de la manière suivante:

1. Le joueur choisit grâce aux boutons le bâtiment à construire
2. Le jeu calcule les coordonnées où le bâtiment sera placé, récupère le numéro d'identité du bâtiment et envoie ces informations au serveur
3. Le serveur instancie le bâtiment sur sa propre scène et change les différents tableaux des bâtiments
4. Le serveur renvoie les informations à chaque client: il instancie le bâtiment sur chaque scène et change les tableaux sur les scènes client.

4.5 Ennemis

Les ennemis sont uniquement gérés par le serveur, tant leur apparition que leurs déplacements ou leurs interactions avec le joueur.
Cela permet au serveur de n'actualiser que leur position sur chacun des clients, limitant ainsi les transferts de données.

Les objets qui permettront de faire apparaître des ennemis ne le feront que sur le serveur. Pour cela, ces objets en question présents sur toutes les scènes clients et sur la scène serveur, doivent vérifier si oui ou non ils se situent sur le serveur. Si oui, ils peuvent faire apparaître un ennemi, si non, ils ne font rien.
Lors de l'apparition d'un ennemi sur le serveur, l'objet va demander au serveur de le faire apparaître sur tous les clients.

Les interactions avec le joueur sont aussi gérées par le serveur. Seul l'ennemi sur le serveur peut attaquer le joueur. Le serveur envoie au joueur l'actualisation de sa barre de vie.

Quant aux attaques du joueur vers l'ennemi, elles ne peuvent être gérées que par le client. Il faut donc signaler au serveur un changement d'état de la barre de vie de l'ennemi qui se chargera de l'actualiser sur tous les clients à son tour.

4.6 Site Internet

Concernant le site internet, nous avons décidé de le développer nous-même c'est à dire de ne pas passer par des plateformes permettant la création d'un site internet à partir de 'templates' existants tel que "wix.com" ou autres. Nous avons fait ce choix pour découvrir le fonctionnement de l'html 5 et les bases de CSS mais aussi pour pouvoir avoir une plus grande liberté quant à la personnalisation de notre site internet. Nous espérons également pouvoir découvrir un peu le java script pour animer notre site mais cela dépendra de l'avancement de notre projet.

Pour la structure du site internet, nous avons choisis de le séparer en trois pages distinctes.

Tout d'abord, il y a page principale contenant un menu tout en haut permettant de naviguer à travers les différents pages. Sur le reste de la page , se trouve notre premier bloc avec le logo de notre équipe ainsi que le trailer de notre jeu. Pour l'instant nous avons inséré le trailer d'un jeu (Surviving mars) qui est une source d'inspiration pour notre jeu. Cela nous a permis de commencer à travailler sur l'affichage du trailer et son rendu sur le site. Nous n'avons plus qu'à modifier la source de la vidéo pour la remplacer par notre trailer qui sera réalisé à la fin du projet quand le jeu sera fini. Puis, grâce à une ancre (bouton explorer) l'utilisateur peut naviguer sur les différents bloques présents sur la page principale afin de découvrir les principes et bases de notre mais aussi c'est points forts (mise en avant de la coopération ...).

La seconde page permettra tout simplement de télécharger notre jeu grâce à un lien.

Enfin, la dernière page est une description de l'équipe avec le rôle de chacun dans ce projet. De plus, on décrit rapidement l'origine de ce projet et partageons ,notre cahier des charges ainsi que le rapport de la première soutenance...

A ce stade, nous n'avons pas entièrement fini le site internet. En effet, le corps principal avec l'emplacement des divers boutons et la structure est en place. Le contenu est presque "définitif". Le travail restant est la mise en page grâce au "CSS" et comme évoqué précédemment la mise en mouvement et un effet 3D grâce au Java Script si le temps nous le permet.

Pour finir, voici ci-dessous une image d'une partie de notre site

5 Réalisation

5.1 Problèmes rencontrés

Plus nous avançons dans le projet, plus nos implémentations deviennent précises et plus il est difficile de trouver de l'aide sur internet. Il nous faut donc redoubler d'ingéniosité pour répondre à nos demandes.

Par exemple, la façon dont nous voulons implémenter notre arbre de technologie demande dans un premier temps de faire fonctionner les technologies en réseau. Il faut de plus que l'interface de l'arbre soit adaptée à un arbre plus grand. Nous voudrions ajouter une barre de défilement pour que notre interface soit plus agréable à regarder.
Nous voudrions, de plus, ajouter un code couleur aux différentes cases de notre arbre en fonction de l'état de notre technologie (bloquée/débloquée/déblocable).

Ce genre de problème a beaucoup de solutions. Nous pourrions par exemple, déplacer l'interface de l'arbre sur une autre scène, créer une barre de défilement dans l'interface directement...
Chaque solution a ses avantages et inconvénient et nous devons tous les prendre en compte afin d'optimiser au mieux le jeu.

Cependant, nous comprenons de mieux en mieux le fonctionnement de Unity et, plus nous programmons, plus nous savons ce que nous devons faire.
Cela est particulièrement vrai pour le système réseau.

Tous ces problèmes ont fortement impacté notre motivation. Après la première soutenance, nous voulions tous faire une courte pause sur le projet. Mais lorsque nous nous sommes remis au travail, les problèmes n'ont pas aidé à retrouver cette motivation.
Nous avons cependant surmonté ces problèmes et notre jeu est maintenant beaucoup plus avancé.

5.2 Ressenti global à ce stade

Pour l'instant ce projet reste motivant, même si certains problèmes nous ont parfois fait perdre notre patience et nous ont fait arrêter de travailler pendant quelques jours. Malgré cela nous restons tous les quatre dans une bonne dynamique et nous sommes prêt à tout donner pour cette dernière ligne droite.

Le manque de temps commence à se faire ressentir de plus en plus et il nous faut accélérer la cadence pour pouvoir, à la fin du projet, peaufiner les derniers détails afin d'avoir au final, un jeu agréable à jouer.

Nous sommes conscients que cette dernière ligne droite représente une charge de travail conséquente et nous ne voulons surtout pas nous laisser déborder par cette charge. Nous ne comptons pas reproduire l'erreur de cette deuxième partie de projet, à savoir prendre trop de repos vis à vis du projet.

6 Conclusion

Ces deux derniers mois nous ont permis d'implémenter les bases des systèmes principaux de notre jeu:

- Le placement de bâtiments
- Les interactions joueurs/bâtiments
- Un inventaire plus optimisé
- Un arbre de technologie
- Des ennemis
- Le tout en réseau

Même si beaucoup de difficultés ont été rencontrées durant cette deuxième partie de projet, nous avons réalisé un grand bond en avant.
En effet, que ce soit au niveau du multijoueur ou de l'arbre des technologies, beaucoup de systèmes ont été implémentés et il ne reste qu'à les compléter.
Tous ces ajouts se feront dans les mois à venir et ne représenteront qu'une petite charge de travail en plus.

Les prochains mois consisteront en l'implémentation des systèmes suivants:

- Système de fabrication
- Systèmes électriques
- Fonctions des bâtiments (e.g la foreuse qui creuse)
- Systèmes de sauvegarde
- Musiques et bruitages

Nous nous approchons de plus en plus de la version finale du jeu qui, nous l'espérons, aura toutes les fonctionnalités que nous voulions lui implémenter.