

Rapport de soutenance finale

FQTM

Mathieu Malabard
Maxence Gay
Baptiste Daumard
Téo Le Vern



SOMMAIRE

1. Introduction

1. Présentation générale
2. Histoire

2. Présentation des membres

1. Mathieu Malabard
2. Maxence Gay
3. Baptiste Daumard
4. Téo Le Vern

3. Bibliographie

4. Les logiciels et outils utilisés

5. Reprise du Cahier des Charges

1. Tâches abandonnées
 1. Système électrique
2. Tâches accomplies
 1. Ennemis

6. Le jeu

1. Menu d'accueil et personnalisation du jeu
2. Lancement de la partie
 1. Différentes manières de commencer à jouer
 2. Génération de la carte
3. Bâtiments
 1. Différents bâtiments
 2. Placement des bâtiments
 3. Interactions avec les bâtiments
4. Système de progression
 1. Inventaire
 2. Arbre de technologies
5. Système d'ennemi et de défense

- 1. Le système d'armement pour le joueur**
- 2. Les différents ennemis**
- 3. Interactions avec les joueurs**
- 4. Système d'apparition et de vagues**
- 5. Les bâtiments défensifs**
- 6. Multijoueur**
 - 1. Mirror**
 - 2. Les joueurs**
 - 3. Les bâtiments**
 - 4. La carte**
 - 5. les technologies**
- 7. Menu Pause**
 - 1. Système de sauvegarde**
 - 8. Fin de jeu**
 - 9. Site Internet**
- 7. Amélioration possible**
- 8. Problèmes rencontrés**
- 9. Erreur commise**
- 10. Bilan du projet**
 - 1. Bilan général**
 - 2. Erreurs commises**
 - 3. Ressentis personnels**
 - 1. Mathieu**
 - 2. Téo**
 - 3. Baptiste**
 - 4. Maxence**
- 11. Conclusion**
- 12. Annexes**
 - 1. Personnages et ennemis**
 - 2. Bâtiments**
 - 3. Items du jeu**
 - 4. Menus et UI**
 - 5. Site web**
- 13. Remerciements**

1 Introduction

Ce rapport traite du projet From Québec to Mars.

Ici est décrit l'aboutissement de cinq mois de travail de groupe.

Ce projet représente une partie de chacun d'entre nous et nous sommes très fiers aujourd'hui de le présenter.

Ce rapport traite le projet dans l'ordre dans lequel un utilisateur découvrira le jeu.

Mais avant, il nous faut nous présenter et présenter le jeu en lui-même, son histoire, son contexte, sa création ainsi que les attentes que l'on avait au début comparées au résultat final.

1.1 Présentation générale

FQTM est un jeu vidéo alliant ingénierie, stratégie et coopération entre les joueurs où l'objectif tout au long de la partie est de coloniser Mars, en effet le but de ce jeu est d'avoir une colonie stable et puissante, résistant aux attaques des martiens, grâce à l'aide d'armes et des nouvelles technologies que nous fabriquerons et qui évolueront tout au long de la partie. La particularité de ce jeu est que chaque partie est totalement unique avec une carte et des minérais totalement aléatoires, ce qui permet de ne pas se lasser ou d'avoir fini le jeu dès qu'on a gagné une partie. Des vagues martiennes viendront tout au long de la partie et le niveau des aliens s'adaptera, il faudra donc s'améliorer rapidement pour ne pas mourir trop rapidement et perdre trop de temps.



1.2 Histoire

En 2242, l'humanité chercha à fuir la planète Terre devenue stérile et dépossédée de la plupart de ses ressources naturelles exploitables. La quasi-totalité de la population fut décimée durant la troisième guerre mondiale qui opposa les états libres dirigé par le Tibet aux états de la coalition de la mongolie extérieure dirigé par le Sri Lanka. Mais un pays résista, Le Canada. Ce pays réussit à ne pas sombrer totalement et continua pendant ce temps à développer le projet top secret ””. Le principe est simple, envoyer des soldats québécois surentraînés faisant partie de l'élite de l'armée québécoise rebelle de kim jong Trois pour coloniser et sécuriser la planète Mars dans le but d'une colonisation massive. Pour cela, une fusée les emmènera avec le strict minimum pour assurer leur survie. L'avenir de tout un pays voir de toute la planète est entre les mains de ces québécois, mais vont-ils réussir à coloniser la planète ? La réponse est entre vos mains!

2 Présentation des membres

2.1 Mathieu Malabard

Depuis le collège, j'ai toujours aimé programmer. Après avoir commencé sur Scratch, puis le python et le langage TI, je n'ai pas eu beaucoup d'occasions de m'améliorer en algorithmie. En effet, je n'avais pas de spécialité SI au lycée et j'ai dû me contenter de l'option ICN en première. Là j'ai découvert la programmation en équipe.

Nous avions décidé à 5 de rénover un site internet pour une entreprise de peinture et plâtrerie. Cependant, l'HTML et le CSS sont très différents des langages plus mathématiques. Cependant, j'ai voulu continuer sur cette lancée et je me suis retrouvé à EPITA.

Ce projet représente pour moi un nouveau challenge bien plus imposant que ce que j'ai pu réaliser jusque là.

2.2 Maxence Gay

Comme de nombreuses personnes dans cette école je suis très intéressé par l'informatique mais plus largement, je suis passionné par toutes les nouvelles technologies et, les personnes jouant un rôle important dans ce milieu comme par exemple Elon Musk qui montre sur chacun de ses projets que l'on peut réussir de réels exploits avec de la détermination et du travail acharné. J'ai déjà commencé à coder des programmes pour des robots en seconde avec un groupe de mon école et j'ai participé au concours IOI France où mon groupe et moi avons fini 1 er du département (Grâce à mon camarade pour être honnête).

J'aime également énormément me documenter sur l'entrepreneuriat pour, à long terme, créer une ou plusieurs entreprises en lien avec l'informatique bien sûr.

J'ai fortement hâte de commencer à coder ce jeu car je pense qu'il va nous apprendre beaucoup tant sur le plan du codage que sur le plan du travail en équipe et la gestion du temps qui est un atout précieux dans ce genre de projet.

2.3 Baptiste Daumard

Depuis de nombreuses années je m'intéresse à l'informatique et à toute autre création informatique. J'ai créé mon premier jeu au collège grâce à Scratch, rien d'incroyable mais c'est tout de même un bon début.

C'est au lycée que j'ai réalisé mes projets les plus complets, grâce à l'option ICN en seconde/première et par la suite la spé ISN en terminal. Durant cette période j'ai appris les langages python, HTML et CSS notamment les deux derniers avec lesquels j'ai réalisé un site internet pour une entreprise. J'ai donc déjà été confronté aux travaux d'équipe et je sais comme

il est difficile de tenir un planning, mais j'en garde une très bonne expérience et j'ai hâte de réitérer avec ce groupe.

J'aime me documenter sur l'informatique car je suis de nature curieux. Pendant le lycée j'avais pour projet de réaliser un jeu sur le moteur Construct 2, j'ai vite arrêté par manque de temps, j'ai malgré tout étudié des notions de game design et je me suis entraîné à créer des sprites.

2.4 Téo Le Vern

Mon engouement pour l'univers informatique est née à mes débuts au collège. Tout au long de ma scolarité suite à divers projets ma curiosité pour ce vaste domaine n'a fait que grandir. Tout d'abord, j'ai découvert la "programmation" avec scratch. J'ai également eu la chance de suivre pendant une semaine, lors de mon stage de 3ème, un développeur.

Puis au lycée, durant ma première année, grâce à un enseignement de découverte, j'ai été initié à l'HTML et le CSS. Durant ma seconde année, j'ai participé à un concours (Course en Cours) qui avait pour but de créer une mini-voiture de course. J'ai modélisé la voiture en 3D grâce au logiciel Catia avant de l'envoyer à l'usine. Ceci m'a permis de me rendre compte que la modélisation 3D n'était pas faite pour moi. Pour ma dernière année, grâce au projet obligatoire de SI, j'ai découvert Arduino.

3 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=WqJ3RIHEB4E>
- 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>

4 Les logiciels et outils utilisés

Pour la partie graphique, nous nous sommes aidés de plusieurs logiciels :

- Piskel et Aseprite pour la création des personnages et des bâtiments.



- Photoshop pour la jaquette du jeu .



Pour la partie codage du jeu vidéo, nous avons utilisé :

- Rider pour la programmation du jeu vidéo.



- Unity pour la partie graphique du jeu vidéo et la mise en relation avec le code.



Pour la gestion et l'organisation du projet nous avons utilisé :

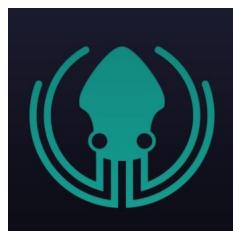
- Google Docs pour mettre en relation nos idées en rapport avec le projet.



- ToDoList pour mettre nos tâches en communs et pouvoir savoir ce qu'il nous reste à faire à chacun.



- GitKraken permettant de gérer le dépôt Git et permettant grâce à son interface simple et intuitive de "push" ou encore "pull" très facilement



5 Reprise du cahier des charges

Au début du projet nous étions très ambitieux. Le but était de créer un jeu similaire à Factorio et surviving Mars, deux jeux de colonisation et de survie sur une autre planète. Nous nous sommes vite rendu compte que ces jeux n'ont pas été créés en 6 mois par des étudiants découvrant unity. Malgré cela nous avons réussi à livrer un jeu de bonne qualité en dépit de certaines conséquences qui n'ont pu être ajoutées. Tel que: Nous voulions implémenter plus de statique afin de gérer notre colonies comme: Un système électriques pour alimenter les bâtiments, jauge de nourritures et d'oxygène et de nourriture pour les joueurs.

Nous voulions créer plus de contenus dans le jeu dont le principe a déjà été créé, comme plus de bâtiments , de ressources ou de crafts.

Beaucoup de ces concepts devait être implémenté en bonus mais suite au révision des partielles nous avons dut faire une croix dessus.

	Mathieu	Maxence	Baptiste	Téo
Mars	Finalisation du multijoueur	Implémentation des outils / MAJ inventaire	Finalisation de la génération procédurale et Implémentation d'un arbre de technologie	Implémentation des outils / MAJ inventaire
	Implémentation des bâtiments	Site Web	Implémentation et design des bâtiments	Site Web
Avril	Implémentation de systèmes électriques	Implémentation des ennemis / Vie du personnage	Implémentation de systèmes électriques	Implémentation des ennemis / Vie du personnage
	Système de fabrication	Finalisation du site Web pour la deuxième soutenance	Développement de plus de bâtiments	Système de gestion de ressources
Mai	Peaufinement de l'arbre de technologie	Intéractions du personnage avec son environnement	Finalisations des graphismes et animations	Intéractions du personnage avec son environnement
	Système de sauvegarde	Musiques, bruitages	Musiques, bruitages	Système de sauvegarde
Juin	Débuggage et perfectionnement du jeu			

6 Le jeu

6.1 Menu d'accueil et personnalisation

(Réalisé par Téo)

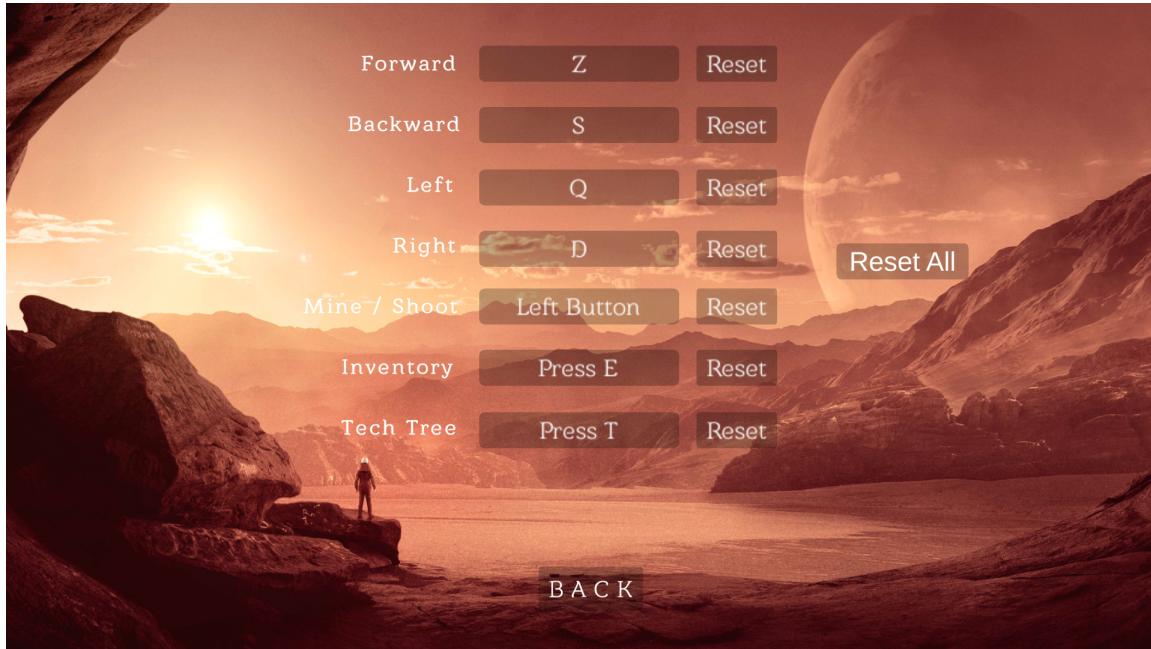
Au lancement de l'application, l'utilisateur va être accueilli sur le menu principal de notre jeu. Il est conseillé de réduire le volume sonore du périphérique de sortie puisque le menu est doté d'une musique de fond ainsi que divers bruitages notamment lors de la sélection de boutons (les réglages par défaut sont faibles pour éviter toutes agressions auditives dès le début). Une fois à ce stade, la navigation à travers les différentes sections du menu peut se faire à l'aide de la souris ou alors grâce aux touches "Flèche du haut" ↑ (arrow up), "Flèche du bas" ↓ (arrow down) et "Entrée" (Enter) du clavier. Tous les boutons possèdent des animations particulières



Menu principal du jeu

Pour pouvoir avoir la meilleure expérience de jeu possible, l'utilisateur va pouvoir effectuer plusieurs réglages se trouvant dans le section Options du menu principal. Ces réglages sont séparés en quatre parties distinctes : les graphismes (graphics), les sons (sounds), l'attribution des touches (Controls) et enfin la section Autre (Other) qui s'adressera à une faible partie des utilisateurs. Tous les réglages sont persistants, c'est à dire que lors du redémarrage du jeu ils sont conservés et restaurés. La sauvegarde de ces données passe par un fichier JavaScript Object Notation (.json).

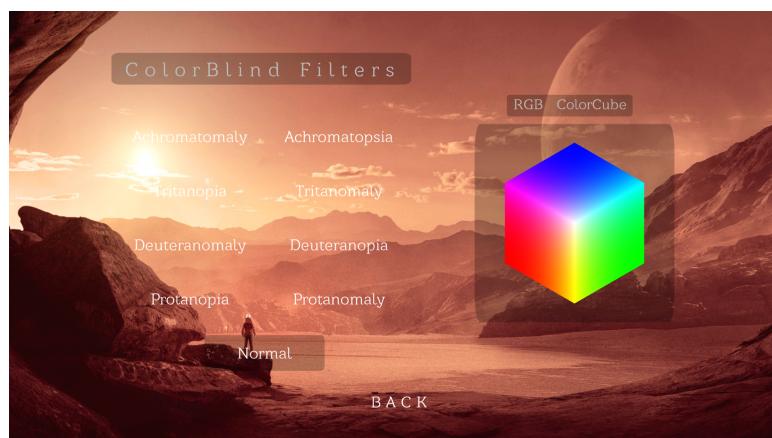
Concernant la section Graphismes, il est possible d'y choisir: la résolution du jeu (par défaut c'est la meilleure résolution de l'écran de l'utilisateur qui est sélectionnée), l'activation ou non du mode plein écran et la qualité des graphismes entre bas (low)/ moyen (medium)/ élevée (high)/ ultra (ultra) étant donné que le jeu est en 2D pixelArt ce dernier a très peu d'impact. Pour la section Sonores, c'est tout simplement plusieurs "sliders" pour changer le volume des divers sons disponibles dans le jeu.



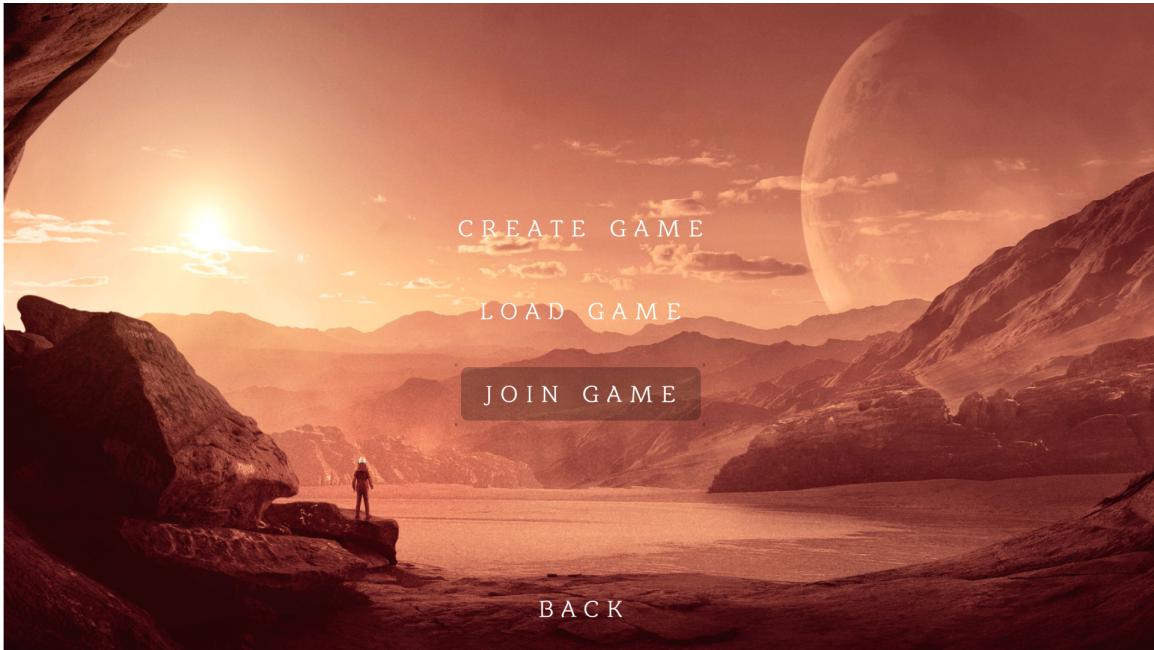
Menu d'attribution des touches

La section "contrôle" offre la possibilité au joueur de modifier toutes les touches du jeu à l'exception de la touche Echap (Escape) qui sert à activer/désactiver le menu pause en jeu, quitter les différentes interfaces des bâtiments. Le système d'attribution des touches possède plusieurs caractéristiques intéressantes. Il s'assure qu'il n'y ait aucun doublons (impossibilité d'attribuer la même touche du clavier à plusieurs actions différentes), possibilité d'annuler le changement de touche en pressant la touche Echap, remettre une touche ou toutes les touches à leurs valeurs par défauts.

Maintenant la plupart des utilisateurs pourra profiter pleinement du jeu et commencer sa folle aventure. Cependant pour une partie d'entre eux, ils (environ 8% des joueurs de jeux vidéos) ne pourront pas avoir une expérience de jeu convenable à cause du daltonisme. C'est pour cela, que nous avons décidé d'implémenter dans la section Other une sélection de filtres pour résoudre ce problème. Une fois le filtre choisi, il sera appliqué de manière permanente en post-traitement de l'image. Le joueur dispose d'un cube RGB pour pouvoir régler son filtre.



6.2 Lancement de la partie



6.2.1 Différentes manières de commencer à jouer

(Réalisé par Téo et Mathieu)

Il existe plusieurs moyens de commencer à profiter du jeu. En premier, il est possible de créer une partie (create game). Cela va lancer une nouvelle scène avec une carte générée de manière aléatoire mais aussi créer un serveur pour qu'un autre joueur puisse rejoindre votre aventure. C'est toujours plus marrant à plusieurs. Il est possible de charger une sauvegarde (à savoir qu'il n'est possible d'avoir qu'une seule sauvegarde de partie) qui va aussi créer un serveur sur la partie chargée. Et enfin, on peut rejoindre une partie en cours uniquement si celle-ci se trouve sur le même réseau que vous. Le menu a été fait par Téo et Mathieu a réalisé les fonctions de sauvegarde (load) ainsi que l'initialisation du serveur en début de partie.

6.2.2 Carte

(Réalisé par Baptiste)

La carte correspond à l'environnement avec lequel le joueur va pouvoir interagir pour évoluer. Il est important dans un jeu bac à sable de pouvoir construire librement dessus. Ici, le joueur va surtout utiliser l'environnement pour avancer dans le jeu.

Nous voulions donner au joueur l'opportunité de pouvoir recommencer des parties sans avoir la même carte. C'est pourquoi nous avons opté pour une génération procédurale de la carte.

Pour la génération procédurale de ressources sur la carte du jeu, nous avons utilisé l'algorithme Perlin Noise. Celui-ci nous permet de générer une carte de dimension (x,y) pseudo-aléatoirement remplie des valeurs entre 0 et 1.

C'est la méthode la plus utilisée dans les jeux vidéos, de plus le Perlin Noise est géré par Unity. Mais le rendu n'est pas très naturel, pour cela on empile plusieurs cartes pour un meilleur rendu.

Chaque couche est appelée "Wave", chaque vague possède les propriétés suivantes: Seed (elle nous permet de compenser le bruit pour ne pas échantillonner la même zone), de Fréquence (le nombre de pics de 1 sur la carte donc plus la fréquence est haute plus la carte est chaotique)

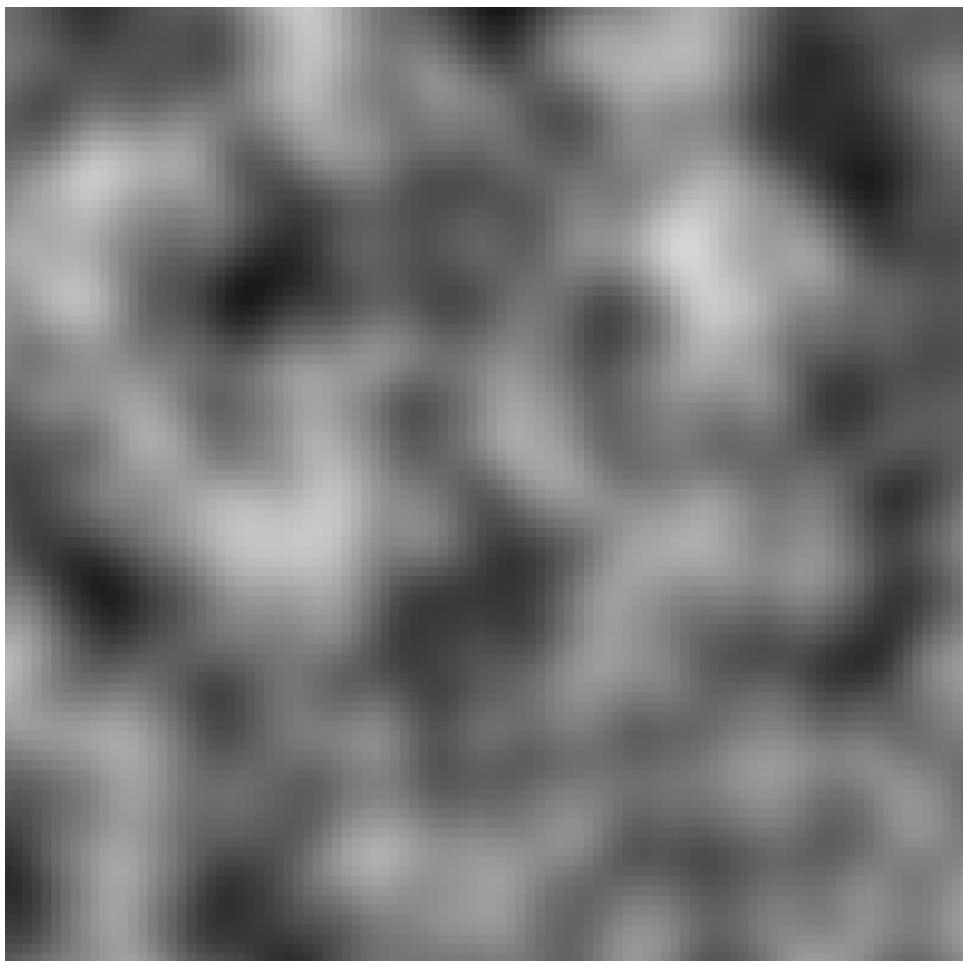
et d'Amplitude (la taille d'une "tache").

Pour cela nous avons créé trois scripts:

- Le premier pour créer les différentes couches de nos cartes , dedans nous créons la nouvelle classe Wave.
- Le second nous permet de créer un objet des Ressources en parcourant le tableau regroupant un tableau des sprites (image) et en regardant si la ressource choisie correspond aux conditions de génération de la NoiseMap.
- Enfin un gros script, héritant des deux premiers. Il génère premièrement trois NoiseMap: une pour l'humidité, une de chaleur et une de hauteur. Leurs noms ont peu d'importance dans la génération actuelle, mais dans le futur développement de l'environnement ils auront de l'importance pour la cohérence.

Ensuite il parcourt chaque coordonnée de notre NoiseMap, à chaque fois, une fonction va parcourir une liste avec toutes les ressources, et elle va renvoyer une autre liste avec toutes les ressources qui peuvent correspondre à la ressource à afficher en fonction des trois NoiseMap générées, elles-même composées de plusieurs Waves. Pour cela elle va regarder si les trois paramètres: la moisissure, la hauteur et la chaleur de la case sont supérieures ou égales aux paramètres de la ressource testée.

Une fois la liste construite avec toutes les ressources susceptibles de correspondre à la ressource à retourner, nous allons parcourir cette même liste pour déterminer celle qui est susceptible de correspondre au mieux à la ressource à afficher, pour cela on réalise une différence entre les paramètres de la case et les paramètres minimums. On additionne ensuite les trois différences et on le garde dans une variable, la ressource qui sera retournée est celle avec la plus petite variable.



Exemple de NoiseMap avec plusieurs 'Wave'

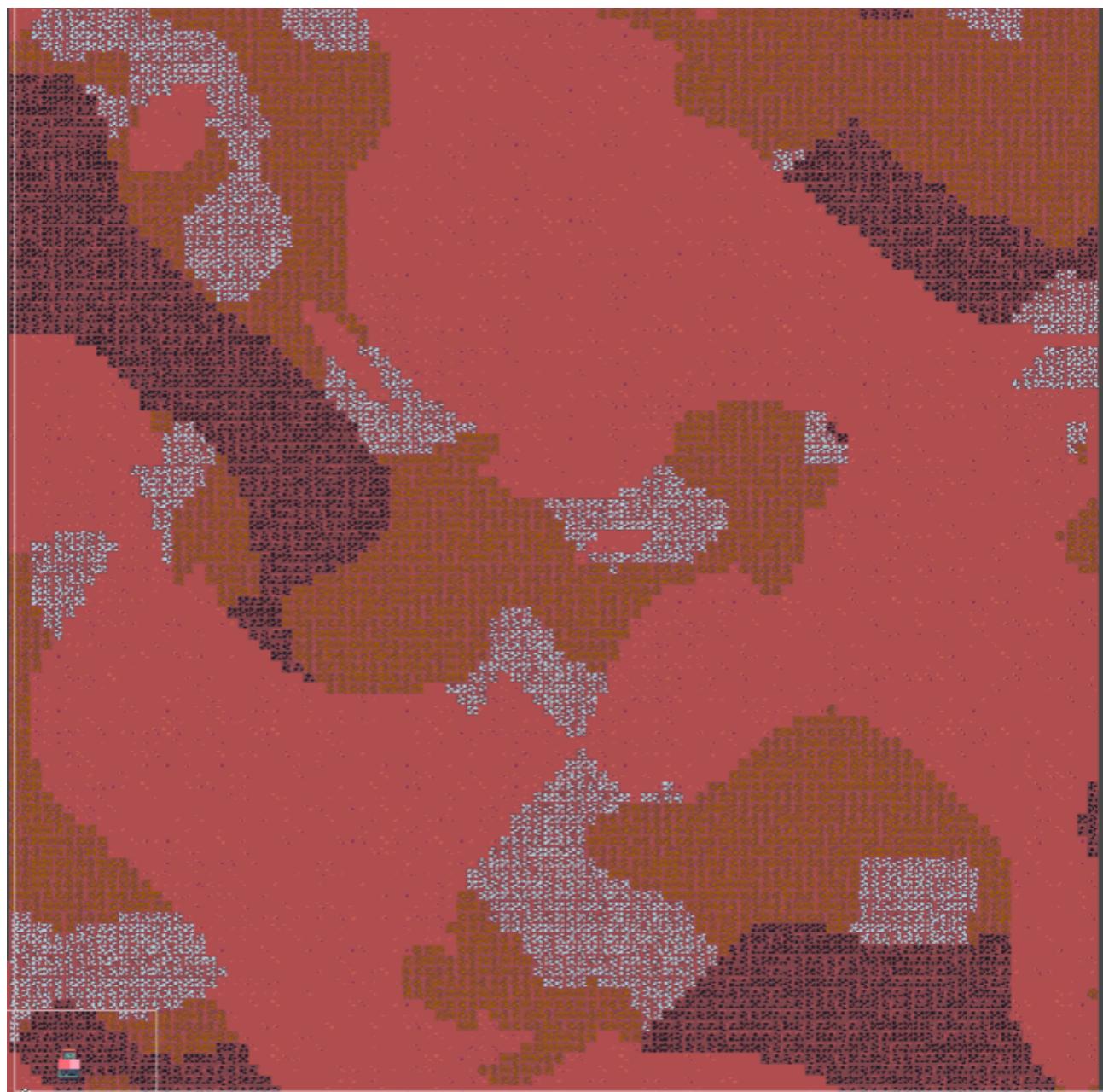
L'environnement de notre jeu comporte différents terrains. Un sol basique ainsi que différentes ressources telles que du charbon, du fer et du cuivre.

Les différentes "tile" (cases) de la carte sont stockées sur une seule image que nous avons découpé sur Unity.

Les différents sprites du sol de base ont été repris sur une bibliothèque libre de droit sur OpenGameArt. Le style graphique correspondait parfaitement au style du jeu, cela nous a permis de gagner du temps.

Chaque sol et ressource possèdent différents sprites pour créer des nuances. Comme vous pouvez le voir sur la figure ci-dessous la génération de la carte présente de grandes taches collées les une aux autres.

La carte est générée aléatoirement lors du lancement d'une nouvelles partie. Pour cela, on utilise le offset une variable qui change l'échantillonnage de la noise du unity.



Carte générée procéduralement de 100x100

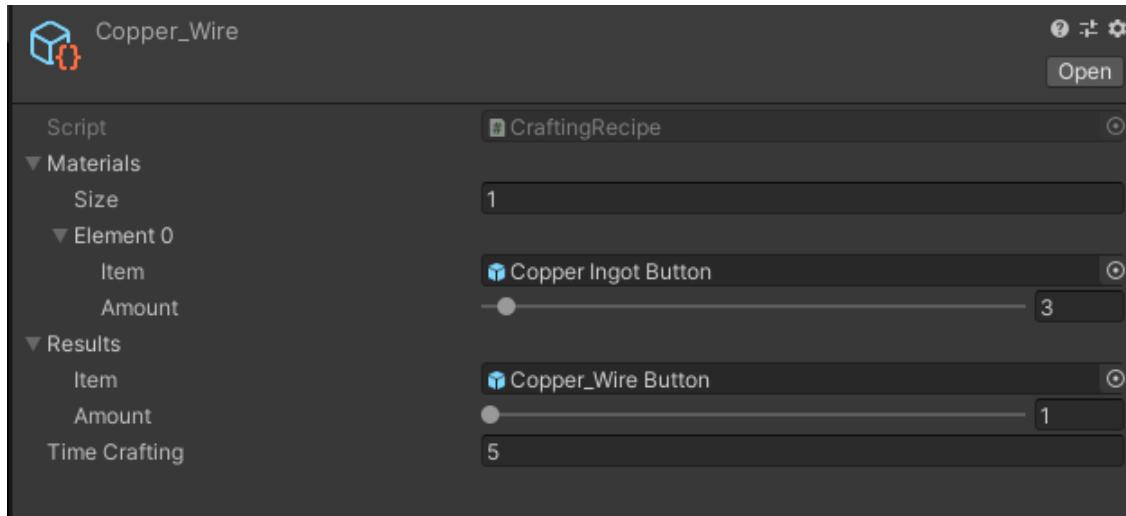
6.3 Bâtiments

6.3.1 Différents 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

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:

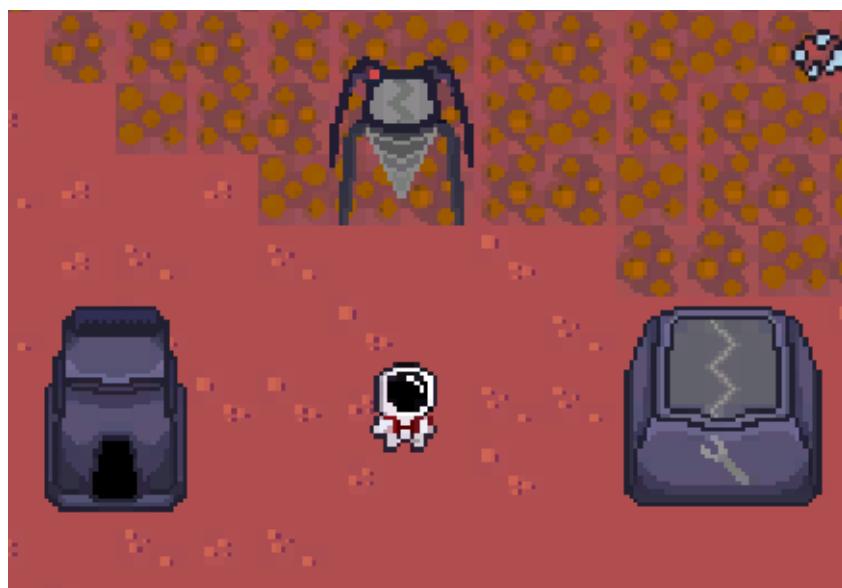
- Le Four (Réalisé par Mathieu Malabard): comme son nom l'indique il permet de cuire des ressources pour en fabriquer des nouvelles. Lorsque l'on interagit avec lui notre inventaire ainsi que le menu du bâtiment apparaissent à l'écran. Le menu est composé de deux cases où le joueur peut placer des objets, 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 (Réalisé par Baptiste Daumard): 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 (Réalisé par Mathieu Malabard): 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 (Réalisé par Baptiste Daumard): 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. Les recettes sont des objets scriptés, c'est à dire que l'on peut choisir les différents paramètres. Les paramètres sont: un tableau d'objets nécessaire pour la création du deuxième paramètre, l'objet que le joueur veut créer, chaque possèdent aussi le nombre requis pour la création, et l et enfin le temps pour fabriquer l'objet.



La fabrication est automatisée, c'est à dire que des objets ou des ressources pourront être amené avec des tapis roulants et la fabrication sera faites en série: la machine va créer des objets tant que les ressources nécessaires sont présentes dans les cases d'entrées. Le bâtiment possède trois cases où les joueurs peuvent placer des ressources qui seront consommé pour créer un nouvelle objet dans la case résultat. Les objets résultant pourront être récupérés et déplacées via les tapis roulant.

(image)

- La Base de départ (Réalisé par Baptiste Daumard): Elle permettra aux joueurs de débloquer des points de compétence pour l'arbre de technologie, pour cela les joueurs devront rassembler des ressources demandées.
- Les Tapis roulants (Réalisé par Téo Le Vern): Ils facilitent les tâches du joueurs en automatisant l'acheminement de ressources entre différents bâtiments. Les objets peuvent être récupéré depuis les cases résultats des bâtiments, déplacé puis ajouté à d'autre bâtiments comme des fours, des coffres ou des machines d'assemblages.
- La Tourelles et les murs (Réalisé par Téo Le Vern): ces bâtiments seront développés dans la parties "Système d'ennemi et défense"



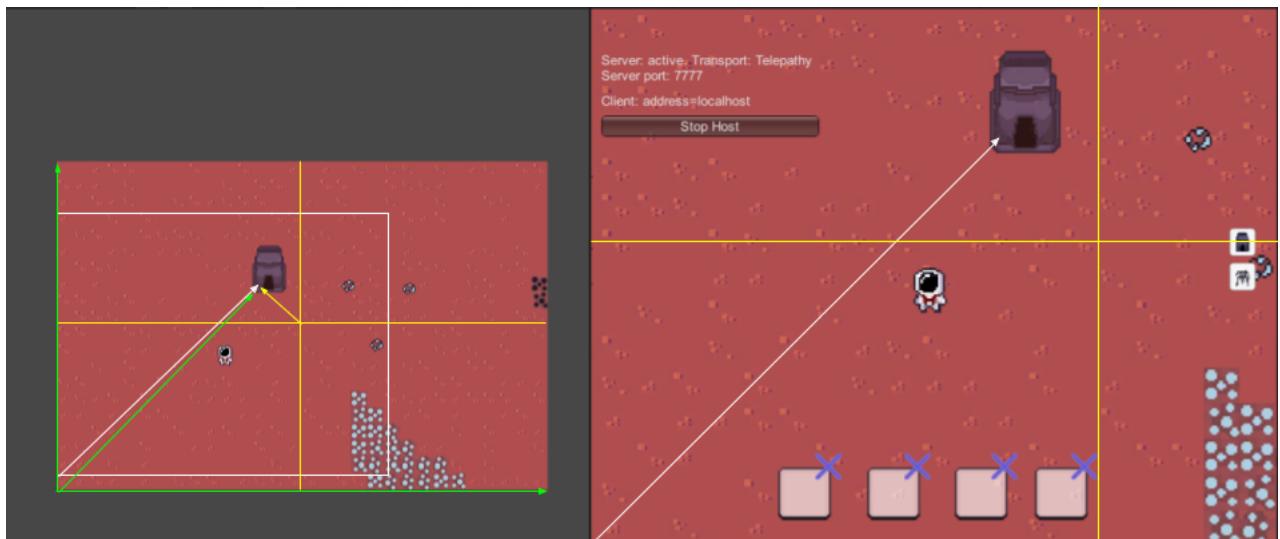
A gauche un four, en haut une foreuse et à droite une machine d'assemblage.

6.3.2 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.

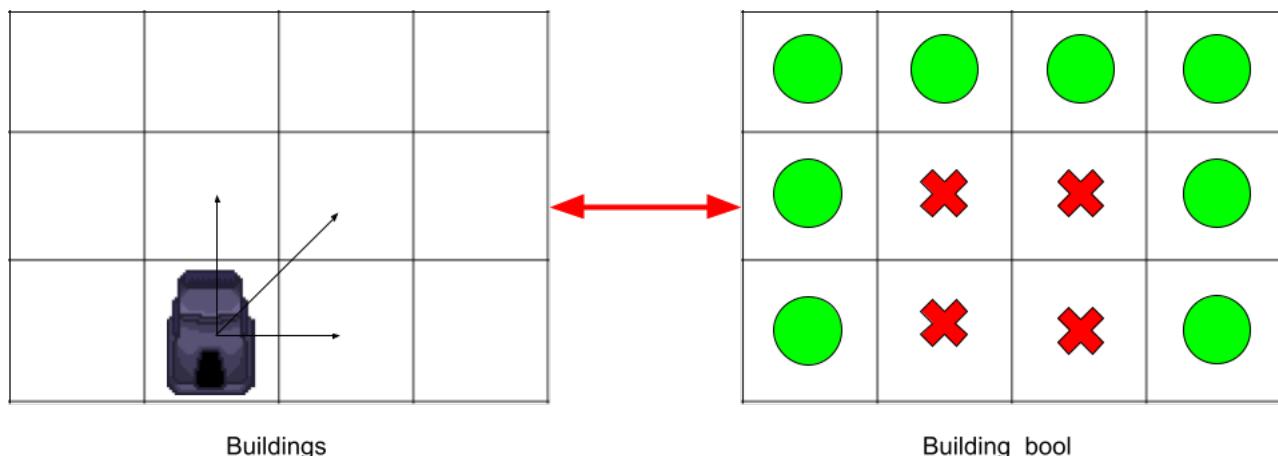
Il faut ensuite convertir ce vecteur en coordonnées réelles. Les axes jaunes correspondent à l'axe des abscisses et l'axe des ordonnées du monde. Le vecteur jaune est déduit par la transformation du vecteur blanc en coordonnées sur le monde.

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 tableau Building ne note la référence du bâtiment sur toutes les cases où il se situe afin de permettre aux tapis roulants de reconnaître qu'un bâtiment se trouve sur cette case.

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.



6.3.3 Interactions avec les bâtiments

(Réalisé par Mathieu)

L'interaction avec les bâtiments a quelque peu changé au long du projet.

Au début, lorsque le joueur entrait dans une zone autour du bâtiment, celui-ci s'ajoutait à une liste de bâtiments. Lorsque le joueur appuyait sur la touche "F", toute la liste de bâtiments était parcourue et le bâtiment le plus proche du joueur était retenu.

Cependant, la position du bâtiment se trouvant en bas à gauche, c'était rarement le bon bâtiment qui était sélectionné.



Ici, c'est le bâtiment en haut à droite qui sera sélectionné

De plus, il était impossible de savoir quel bâtiment était vraiment sélectionné. Nous avons donc décidé d'améliorer cela.

Maintenant, lorsque le joueur place son curseur sur un bâtiment, celui-ci devient jaune, et lorsque le joueur clique sur le bâtiment, l'interface du bâtiment s'ouvre. Ainsi, il est beaucoup plus facile d'interagir avec les bâtiments.



Le curseur se trouve sur le bâtiment de droite

Lors de l'interaction, le menu du bâtiment s'affiche comme ceci.



Le joueur peut alors déplacer des objets depuis son inventaire (à gauche) dans les cases du bâtiment (à droite)

Lorsque le joueur dépose un objet dans un bâtiment, ce bâtiment s'actualise et, pour le four ou la machine d'assemblage, commence à produire des objets en fonction de ce qui est présent dans ses cases.

Pour le four, deux jauge représentent l'avancement de la fonte du produit ainsi que la quantité de carburant restant avant l'utilisation d'un nouveau charbon.

Pour la foreuse, L'interface se présente comme ceci:



Les trois cases au dessus représentent les minéraux que la foreuse peut récupérer et les cases du dessous servent à récupérer les produits récoltés par la foreuse.

Une jauge indique l'avance du forage d'un minerai. Le minerai qui ressort de la foreuse est choisi aléatoirement entre les minerais disponibles sous la foreuse.

Les probabilités sont choisies en fonction des cases sous la foreuse. Si il y a trois minerais de cuivre et un de charbon, chaque fois que la foreuse va sortir un mineraï, il y a trois chances sur quatre pour qu'il s'agisse du cuivre.

Enfin, l'interface de la machine d'assemblage se présente comme cela:



En haut se trouve les cases où poser les composants nécessaires à la fabrication de la recette choisie.

Le sélection des recettes se fait sur la droite de l'écran.

Le résultat de la fabrication se trouve sur la case à droite de la jauge verte. Jauge qui permet de connaître l'avancement de la fabrication.

6.4 Système de progression

6.4.1 Inventaire

(Réalisé par Maxence Gay)

L'inventaire est l'objet central de notre personnage. En effet, sans inventaire, aucun moyen d'accéder aux ressources et donc d'avancer dans le jeu.

Le joueur aura un inventaire sur lui pour disposer d'objets physiquement, chaque joueur aura un inventaire personnel et lorsqu'il mourra, le contenu de son inventaire sera jeté à terre. Le système d'inventaire marche sur la base suivante : Le joueur doit se positionner sur l'item pour que celui-ci soit récolté. (Il servira dans le cas où l'on tuera des ennemis, ou encore dans le cas où des joueurs veulent s'échanger des objets en multijoueur).

Pour cela, lorsqu'un joueur interagit avec un item (objet), il y a une collision entre les deux, dans un premier temps, le programme "Pickup" va lancer la partie "Void OnTriggerEnter2D" du programme qui s'active uniquement dans le cas d'une collision. Ensuite on vérifie si la collision qui opère avec l'item est bien avec un "player" avec "if(other.CompareTag"Player")".

Le programme va par la suite vérifier grâce à une liste prédefinie si l'inventaire n'est pas plein avec une boucle "for" (qui vérifie s'il y a une place dans la liste). S'il est plein, il ne fera rien, sinon il va stocker l'élément item dans la liste, la case deviendra alors pleine et aucun item ne pourra rentrer dedans.

De plus, il faut créer pour chaque item un "button" qui sera un "prefab" (objet préfabriqué utilisé pour instancier des clones en cours de jeu). Puis on assignera ce "button" à chaque item dans son script "Pickup".

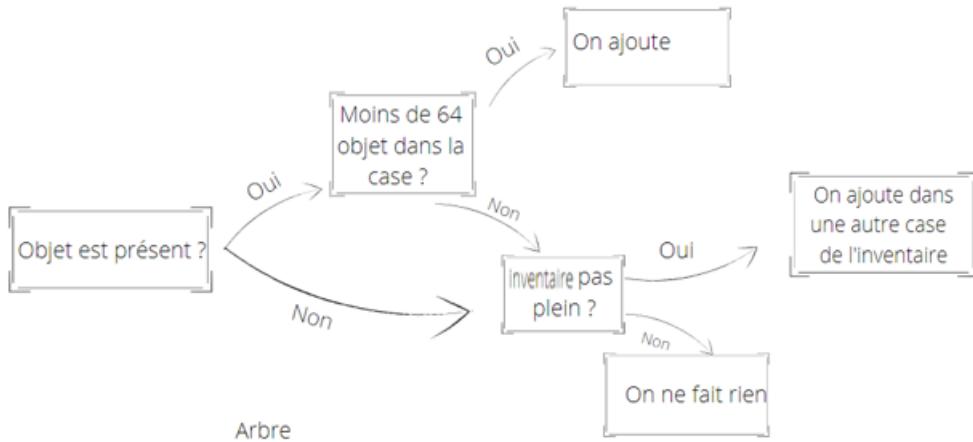
Le joueur quant à lui aura le script "Inventory" qui correspond seulement à la création de deux entrées qui sont les suivantes :

- "IsFull" qui est un booléen qui permet de savoir si le slot (emplacement) est plein ou non.
- "Slot" qui correspond au nombre de cases d'inventaire, nous avons décidé de mettre un grand nombre de cases pour que le joueur puisse avoir de nombreuses ressources dans son inventaire.

Ce script permettra donc d'assigner les slots qui seront soit pleins, soit vides. Pour créer les cases d'inventaire nous avons donc créé des "UI image" qui sont des images qui vont se superposer au-dessus de la caméra (comme pourrait le faire un filtre photo). Ensuite, il faut créer des "UI image" pour les croix qui serviront à lâcher un item.

Pour lâcher un objet à terre, nous avons assigné un bouton sur l'image "croix" de notre UI et lorsque l'utilisateur cliquera dessus, cela détruira l'item dans la liste et créera un clone de l'item sur la carte aux coordonnées du joueur en x et en y+1 pour permettre une meilleure visualisation.

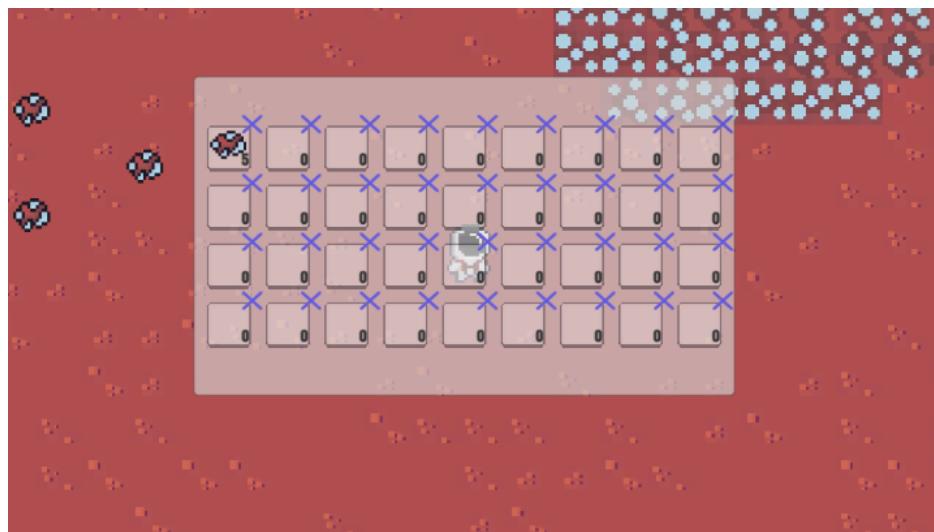
Par la suite, j'ai ajouté le stacking d'objet, cela permettra d'avoir plusieurs objets dans une case et non pas un seul. Pour cela on a ajouté à chaque case une valeur. Cette valeur correspondra au nombre d'itération de l'objet dans l'inventaire. Pour cela le code marchera de la façon suivante :



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.

Ensuite, j’ai également implémenté 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, j’ai rajouté 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’items. Pour y accéder, cela se fera par la touche "E" du clavier. L’inventaire ressemble donc à cela :



6.4.2 Arbre de technologies

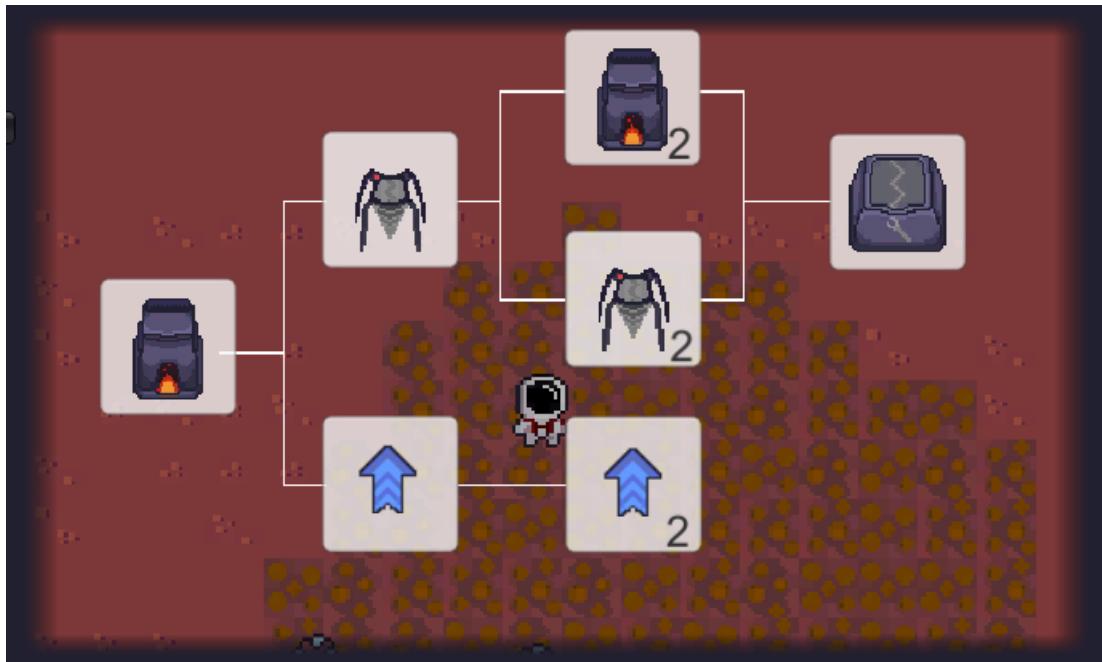
L'arbre de technologie est la façon dont les joueurs progressent en débloquant du nouveau contenu. Dans celui-ci on peut débloquer des bâtiments, des crafts et améliorer les caractéristiques du joueur ou des bâtiments. L'arbre est accessible en pressant la touche "t". Une fois ouvert on peut voir Les 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 privilégié 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 modifie les valeurs correspondant à la statistique voulue.
- Si c'est un craft: Comme pour un bâtiment on passe par une liste de booléen.



Capture d'écran de l'arbre de technologies

Les icônes des technologies qui peuvent être débloqué (Les technologies adjacentes aux dernières débloquées) sont griseses.

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.

La façon dont l'arbre de technologies a été créé nous permet d'en rajouter facilement si l'on souhaitais rajouté du contenu comme dans le cadre d'un DLC, système économique très commun de le monde du jeu vidéo.



Technologie débloquée

6.5 Système d'ennemi et de défense

6.5.1 Système d'armement pour le joueur

(Réalisé par Maxence Gay)

Pour les armes , j'ai commencé par les dessiner sur le logiciel Piskel.

J'ai décidé de faire 4 armes pour ce jeux (que l'on peut retrouver dans annexes) mais par manque de temps, nous n'allons en implémenter qu'une seule, le pistolet. Voici à quoi ressemble le pistolet :



Design des armes FQTM

Par la suite, j'ai créer un script qui permet au sprite "arme" de suivre le mouvement de la souris et de la reproduire, pour cela, on va commencer par créer une variable qui va calculer la position de la souris. Ensuite, on assigne le sprite "arme" à cette position , et cela est fait à chaque update du jeu (soit plusieurs fois par seconde). Ensuite j'ai rajouté un sprite "coup de feu" qui va s'activer lorsque le joueur fera un clic gauche sur sa souris. Cela va donner l'effet que l'arme tire une balle.

Par la suite, j'ai hésité entre deux méthodes pour infliger des dégâts aux aliens :

- La première était l'utilisation de "préfabs". Pour cela lorsque le joueur va cliquer pour tirer cela va instancier un sprite de balle qui va avancer sur la carte jusqu'à avoir une collision avec un ennemis. S'il ne touche aucun ennemi alors rien ne se passera, mais s'il en touche un alors cela va infliger des dégâts à l'entité qui est rentré en collision avec le monstre.
- La deuxième méthode s'appelle le "raycast" . Elle consiste à avoir un laser invisible pour le joueur qui va suivre le mouvement de la souris. Lorsque l'on tirera sur un joueur celui-ci aura le laser sur lui et donc on pourra savoir à quelle entité infliger des dégâts

J'ai décidé de prendre la première méthode car celle ci me paraissait plus intuitive et plus facile à implémenter.

6.5.2 Les différents ennemis

(Réalisé par Téo Le Vern)

Tout d'abord, chaque ennemi est implémenté comme un objet indépendant. Il a des propriétés qui lui sont propres telles que sa vie, sa vitesse, ses dégâts etc ... Voici les différents ennemis disponibles en jeu

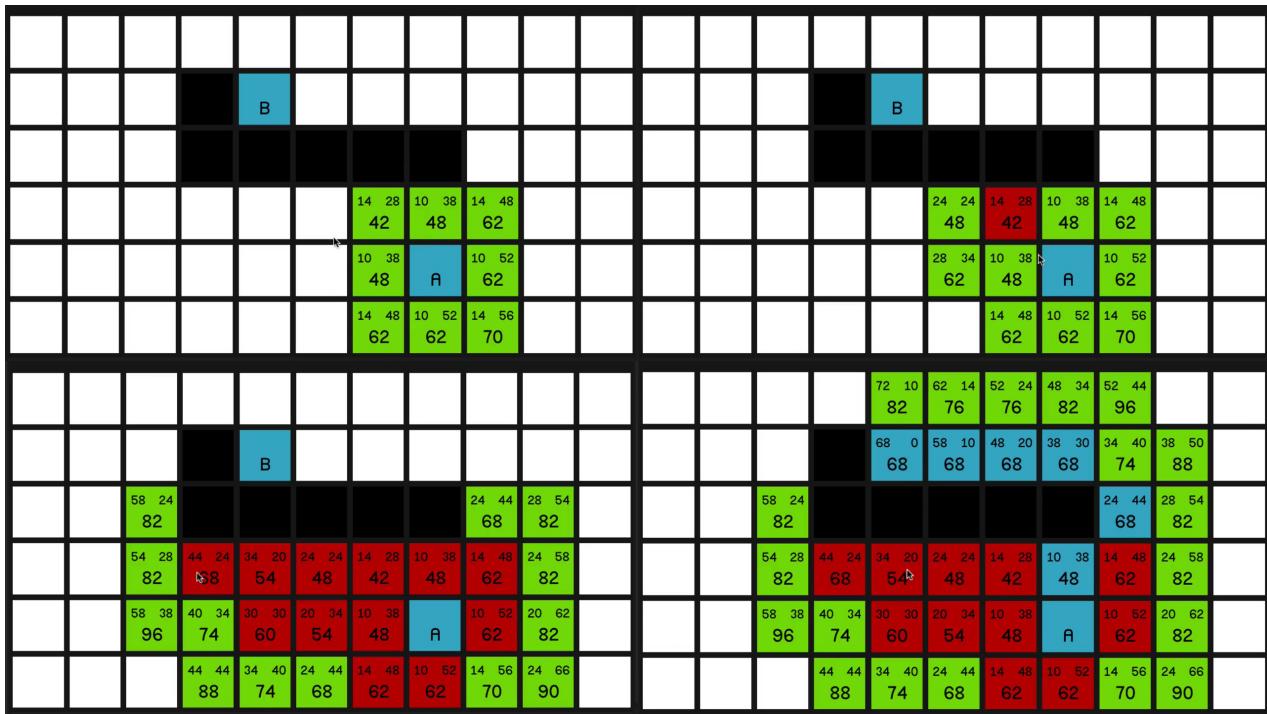


Chaque monstre doit pouvoir se déplacer librement sur la carte afin d'atteindre son objectif. Ils ont pour but de se rapprocher de leur cible pour ensuite interagir avec elle. Pour trouver son chemin dans ce monde, il va s'appuyer sur l'algorithme de "Pathfinding" (Trouver un chemin) visant à trouver 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 noeud (node) ayant chacun une position propre par rapport aux autres et une taille identique.

L'entité se trouve au départ sur un noeud précis (Node A). On va donc ensuite pour chaque voisin de ce noeud vérifier d'abord s'il n'y aucun élément gênant sur le noeud (ex : bâtiments). Dans le cas où le noeud est libre on va calculer trois valeurs (Gcost : distance par rapport au noeud de départ (Node A), Hcost : distance par rapport au noeud d'arrivée (Node B) et FinalCost: Gcost + Hcost). Ces valeurs permettent d'estimer le coût de déplacement vers le noeud voisin.

On va récupérer les noeuds voisins avec le plus faible coût (il est possible qu'il y aient plusieurs noeuds avec le même coût de déplacement) et réitérer ce processus sur chacun des nouveaux voisins admissibles jusqu'à avoir un ou plusieurs chemins atteignant le noeud 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 noeud 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 noeuds en éliminant les noeuds occupés par un obstacle . Cela se traduit dans Unity par le lancement d'un nouveau "scan" de toute la carte pour détecter les modifications.



Chaque ennemie à l'exception du boss final possède une version avancée caractérisée par un changement de couleur (plus foncée). Voici les caractéristiques propres de chaque ennemi:



Le larbin et sa version supérieure sont de couleur vert fluo, ils ont des statistiques moyennes/équilibrées (vie,vitesse,attaque).



Le snarbolax et sa version supérieure sont de couleur bleu claire et bleu foncé, ils ont une grande vitesse avec peu de points de vie et peu de dégâts.



Le golem et sa version supérieure sont de couleur jaune et orange, ils ont énormément de points de vie mais très peu de dégâts et de vitesse.

Les monstres ont des animations et objectifs différents. Mais on peut distinguer deux types de comportement. Pour le larbin et le snarbolax, ils vont chercher à chasser le joueur le plus proches d'eux s'ils se trouvent à leurs portées. Le larbin a une portée de 8 et le snarbolax de 12. Si aucun joueur ne se trouve à proximité, ils vont donc cibler le bâtiment le plus proche. Le monstre "golem" et le boss final du jeu, eux ne cherchent pas à prendre en chasse le joueur, ils vont simplement se diriger vers le bâtiment le plus proche. Le golem attaquera le bâtiments en attaque unique tandis que le boss lui infligera une attaque de zone pouvant ainsi blesser le joueur.



Le boss final a énormément de points de vie, inflige beaucoup de dégât et se déplace lentement

6.5.3 Interaction avec les joueurs

(Réalisé par Téo)

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 un certains nombre de points de vie dépendant du type de monstre. Si le joueur se déplace, le monstre va continuer à suivre le joueur sans lui donner de coup. Pour que le monstre puisse interagir, il faut que le joueur soit 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 sur l'image). Lors de la mort d'un joueur, il relâche son inventaire au sol et va réapparaître près de la base de départ.



6.5.4 Système d'apparition et de vague

(Réalisé par Téo)

Les monstres apparaissent par vague près d'un nid à monstre situé sur la carte. Chaque vague se déclenche à un intervalle de temps qui varie à chaque nouvelle vague (entre 7 à 10 min) sauf pour la première vague qui elle se déclenche après 15 18 minutes de jeu. Le nombre d'ennemi, leurs types dépendent du nombre de bâtiments posés sur la carte. On considère que plus le joueur a des bâtiments plus il est avancé dans le jeu. Il y a aussi un coefficient qui permet d'augmenter les points de vie de chaque monstre selon le nombre de tourelle (bâtiment défensif). Cela permet de garder un certain équilibre tout au long de la partie.



6.5.5 Les bâtiments défensifs

(Réalisé par Téo)

Le joueur aura à sa disposition des murs et des tourelles pour sécuriser sa base.

Les murs auront pour but de ralentir les ennemis. Si ils sont placés de manière stratégiques ils peuvent même influencer le déplacement des monstres. Il y a plusieurs types de mur que l'on peut obtenir en utilisant la recette de base et en faisant varier le minerai utilisé. Bien-sûr plus le matériau est solide, plus le mur aura de points de vie. Voici le mur basique;



Concernant la tourelle, c'est un bâtiment qui s'activera uniquement si un ennemi rentre dans sa zone d'action. Il infligera un certains nombre de dégâts à tous les entités hostiles à portées. Le temps de recharge de la tourelle est de 3 5 secondes.

Comme les autres bâtiments elles pourront être améliorer afin de faire plus de dégâts ou un temps de recharge réduit.



6.6 Multijoueur

(Réalisé par Mathieu)

Le multijoueur est un aspect très compliqué du jeu à gérer. En effet, le transfert de données entre le serveur et les différents clients sont en général limités à des échanges de nombres ou de caractères, ce qui fait que les synchronisations entre les différents clients demandent une charge de travail conséquente.

6.6.1 Mirror

Le système de networking (réseau) inhérent à Unity s'appelle "UNet". Il est très recommandé par les "vétérans" de Unity de commencer à faire du networking sur UNet afin de bien comprendre les bases du réseau. Seulement, par soucis de temps, nous avons préféré implémenter le multijoueur avec un Asset créé par des développeurs de jeu sur Unity.

Cet Asset s'appelle Mirror et se base sur UNet tout en y apportant des modifications facilitant la prise en main du réseau. Il est très apprécié de la communauté Unity étant donné que le passage de UNet à Mirror se fait très facilement.

6.6.2 Les joueurs

Lorsqu'un joueur veut créer une partie, il doit créer un serveur, même s'il est tout seul. Ce joueur est appelé hôte car il joue sur la même instance que le serveur.

Lorsqu'un joueur veut se connecter à un serveur, tous les objets qui ont été détectés comme appartenant au réseau vont apparaître sur l'instance du client (y compris le joueur de l'hôte). De plus, le jouer du client va apparaître sur l'instance de l'hôte.

Un objet appelé "NetworkManager" va s'occuper d'envoyer ou de recevoir les flux de données des autres instances (et donc des autres NetworkManagers)

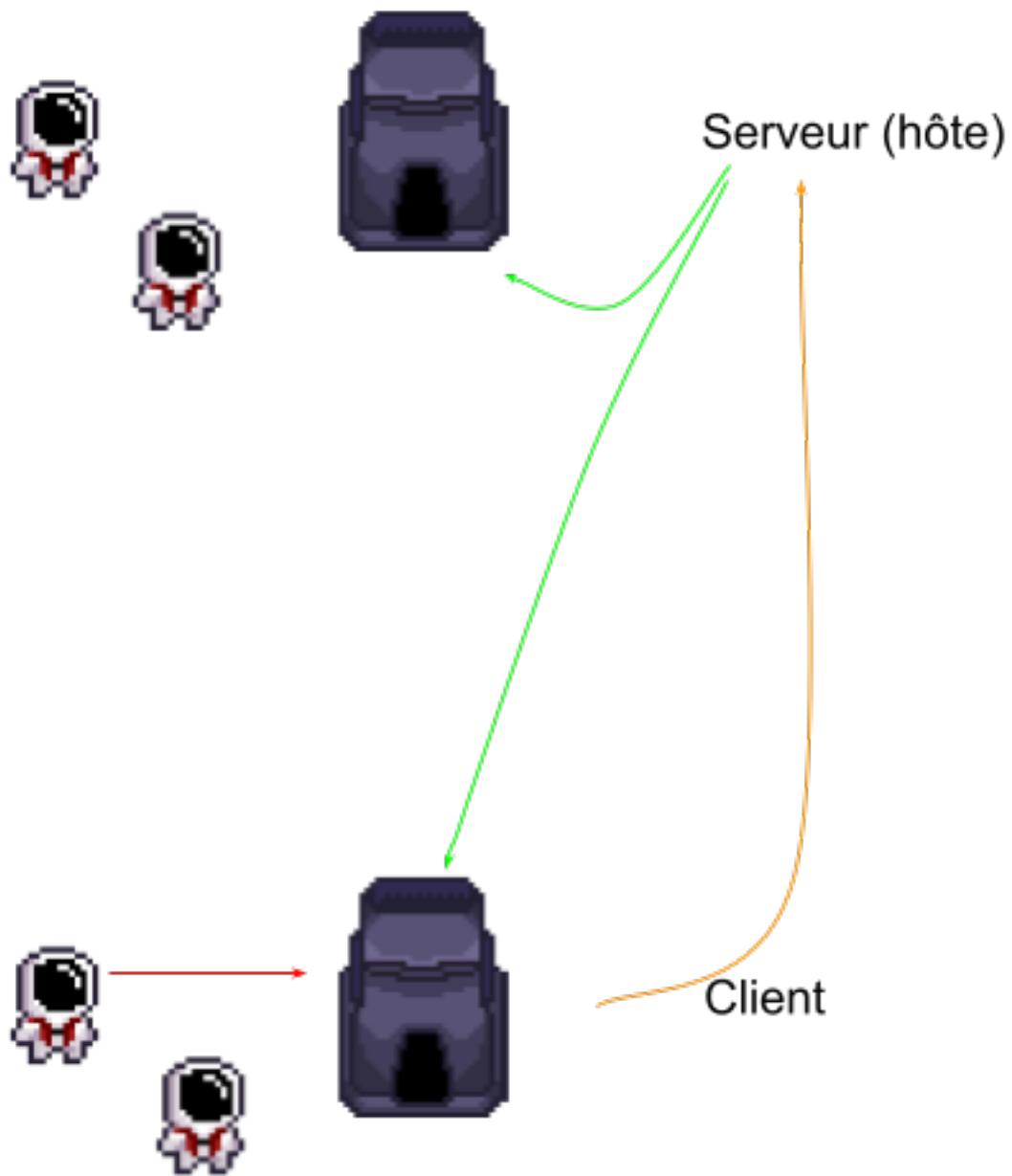
Seulement, les données transférées ne sont que des références à des objets existants dans la base de données du jeu. En effet, quand le NetworkManager reçoit une demande d'instanciation d'objet en jeu, il récupère la référence de l'objet "modèle" à instancier et en créé un nouveau tout en faisant en sorte que le serveur fasse le lien entre l'objet sur l'instance hôte et l'objet nouvellement instancié sur l'instance client.

En effet, fondamentalement, les deux objets sont différents et indépendants car ils sont créés sur deux instances différentes. Il faut donc faire attention à bien notifier le serveur lorsqu'un changement s'opère sur un objet depuis une instance afin de mettre à jour l'objet sur toutes les autres instances.

Le cas typique de modification est le déplacement d'un joueur. Le serveur doit en permanence actualiser le déplacement d'un joueur sur les autres instances. Heureusement, il existe déjà des scripts permettant de ne pas avoir à se soucier de cela.

6.6.3 Les bâtiments

Contrairement aux inventaires des joueurs qui sont personnels et n'ont donc pas besoin d'être actualisés sur tout le réseau, les bâtiments eux doivent en permanence être actualisés.



Par exemple, si le client ajoute du charbon dans un four (schéma au dessus), le Network-Manager doit signaler au serveur que le contenu de la case réservée au charbon dans ce four précisément a changé. Le serveur se charge ensuite de propager l'information aux clients (l'hôte est aussi considéré comme un client). Il en va de même lorsqu'un joueur récupère des objets d'une case d'un bâtiment.

De plus, lorsqu'un joueur place un bâtiment, il faut, comme pour les joueurs, instancier et lier les bâtiments sur les différentes instances clientes.

6.6.4 La carte

Lorsqu'un client se connecte au serveur, il ne récupère pas les informations de la carte, il faut donc les lui apporter. Étant donné que seul le client a besoin de récupérer les données de la carte, (l'hôte génère une nouvelle carte ou récupère les données depuis les fichiers de sauvegarde), il doit le signaler au serveur.

Il est impossible d'instancier une à une chacune des tuiles qui compose la carte sur le serveur car cela ne ferait que créer des copies vides du modèle dans la base de données du jeu sans y apporter les paramètres de chaque tuile.

La méthode la plus simple revient en fait à récupérer les paramètres de génération de la carte (deux nombres décimaux), et de générer la carte à partir de ces données. La génération se faisant suivant un algorithme, si les paramètres sont identiques, la carte est identique.

6.6.5 Les technologies

De la même manière que la carte, le client doit récupérer les technologies déjà débloquées par l'hôte. Cependant, là où la carte est totalement indépendante du joueur et peut-être générées à n'importe quel moment, les technologies sont liées au joueur et doivent donc être récupérées lorsque le joueur a fini de s'instancier.

6.7 Menu pause

(Réalisé par Téo)



Le menu pause s'active avec la touche Echap en jeu, il permet de quitter la partie ou revenir au menu principal pour le client et l'hôte. Pour l'hôte seulement, il permet aussi de sauvegarder les données de la parties en cours.

6.7.1 Système de sauvegarde

(Réalisé par Mathieu)

La sauvegarde d'une partie se fait sur l'ordinateur de l'hôte. Il se fait grâce à une classe inhérente à C#, le BinaryFormatter.

Ce BinaryFormatter transforme les attributs d'une classe (attributs de type simple: entier, chaîne de caractères, tableau...) en chaîne de caractère cryptée en binaire.

La classe BinaryFormatter se charge aussi de transformer une chaîne de caractère cryptée en objet.

Il suffit ensuite de stocker ces chaînes de caractères dans des fichiers sur l'ordinateur de l'hôte pour charger et sauvegarder des parties.

Il y a cependant une difficulté qui est de pouvoir charger le joueur du client depuis l'instance client, qui ne correspond donc pas à l'ordinateur du client. Ces informations doivent donc passer entre les deux instances pour charger correctement le client.

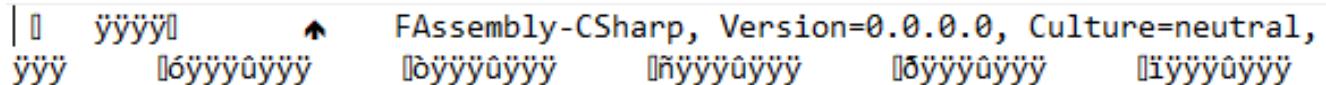
Il en va de même pour la sauvegarde du joueur client qui doit envoyer au serveur ses données de joueur pour que le serveur puisse écrire les fichiers sur l'ordinateur de l'hôte.

Le système de sauvegarde ne permet qu'une seule sauvegarde et est appelé lorsque l'hôte le demande. Il est ensuite appelé lors de la connexion d'un client.

Lorsque le client se déconnecte, il est automatiquement sauvegardé chez l'hôte.

Lorsque l'hôte se déconnecte, tout est sauvegardé, la carte, les technologies, les joueurs...

Si une sauvegarde existait déjà, elle est écrasée.



A screenshot of a Microsoft Notepad window. The code displayed is C# assembly code, specifically:

```
assembly-CSharp, Version=0.0.0.0, Culture=neutral,
Assembly-CSharp
Module1
Method1
System.Void Method1()

    System.Console.WriteLine("Hello World!");

```

Quelques caractères du fichier de sauvegarde de la carte

Le fichier est ouvert sur Bloc-Notes

6.8 Fin de jeu

Le but du jeu est de fabriquer la Poutine dorée, on peut la débloquer grâce à l'arbre des technologies puis la créer dans la machine d'assemblage, c'est en théorie l'objectif principale que les joueurs doivent atteindre. Mais cette objectif n'est pas non plus obligatoire, les joueurs peuvent très bien choisir d'optimiser au maximum l'automatisation des bâtiments ou de tuez un maximum d'aliens pour collecter le maximum d'essence de monstre. L'essence de monstre est un item que relâchera chaque monstre lors de sa mort. (Voir images suivantes) FQTM es un jeu bac à sable, les joueurs sont libres de faire ce qu'ils souhaitent.

6.9 Site Internet

(Réalisé par Maxence)

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'HTML5 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...

Pour l'hébergement du site internet j'ai décidé de le faire grâce à "GitHub", en effet lorsque l'on se crée un compte GitHub, il y a la possibilité d'héberger un site gratuitement. Au début je pensais héberger le site internet sur une plateforme payante tel que OVH ou cela aurait été fait de manière plus professionnel. Mais, le fait d'avoir un hébergement gratuit nous a fait tendre vers cette offre.

Pour finir, voici ci-dessous une image d'une partie de notre site et le lien du site internet, mais nous avons référencé toutes les images en annexes à la fin du rapport.



Lien du site : <https://fqtmt.github.io/FQTM.html>

7 Problèmes rencontrés

Ce projet nous a fait nous arracher de nombreux cheveux à cause de tous les problèmes que nous avons eu. Le problème majeur que nous avons eu a été au niveau de GitKraken. En effet, bien que GitKraken soit une plateforme assez intuitive, elle n'a pas tout de suite apprécié le partage de dossier Unity. En effet, Unity crée une multitude de dossiers dès qu'on lance le logiciel et cela pose un problème au niveau de Git qui se retrouve confrontés à devoir "merge" (mélanger) de nombreux dossiers. Au début du projet nous passions en moyenne sur une échelle de dix heures de travail , deux heures à coder et huit heures à régler tous les problèmes liés à GitKraken et les dossiers qui rentraient en conflit. Mathieu a donc fait un dossier GitIgnore qui a permis de répertorier tous les fichiers que Git ne doit pas traiter. Enfin, au bout de quelques semaines et de nombreux essais, nous pouvions nous concentrer principalement sur le code sans nous soucier de Git.

8 Bilan du projet

8.1 Bilan général

Ce projet nous a appris beaucoup de choses. Premièrement, à nous débrouiller par nous même. En effet lorsque nous n'arrivons pas quelque chose, notre meilleur ami n'est plus le professeur

mais bien internet, et en cherchant sur des dizaines de sites, la réponse arrive très souvent. Aussi, ce projet nous a énormément appris sur Unity et tout ce qu'il permet de faire, en regardant de nombreux tutoriels, en faisant des erreurs et en se documentant, de nombreux automatismes sont arrivés et nous permettent d'être bien plus rapide qu'auparavant. Également, ce projet nous a appris à travailler en équipe, se donner des objectifs à effectuer en un certain temps, et ne pas laisser une personne se débrouiller toute seule lorsqu'elle n'arrive pas une tâche. Enfin, nous avons adopté une méthode de travail en duo plutôt que chacun de notre côté, cela permet

8.2 Erreur commise

Une grosse erreur qui a été commise a été de sous estimer la quantité de travail après la première partie du projet.

En effet, les objectifs que nous nous étions donné en début de projet étaient plus rapides à réaliser que prévu. Nous nous sommes donc donné plus de travail, nous tirant une balle dans le pied.

Nous avons fini par voir trop gros pour un jeu déjà très complexe à réaliser et quelques implémentations n'ont pas été possibles.

De plus, le fait de travailler à plusieurs a fait que, lors des mises en commun des travaux, énormément de problèmes de compatibilités se sont posés, ralentissant grandement notre travail.

Avec tout cela s'ajoutent les nombreux petits bugs qui font planter le jeu.

La charge de travail s'est accentuée avec le temps à cause de tous ces problèmes de compatibilité et nous avons du décider d'abandonner certaines parties du jeu jugées non nécessaires.

8.3 Ressentis personnels

8.3.1 Mathieu

Ce projet a été un véritable challenge pour moi. C'est une charge de travail conséquente que nous avons du fournir en parallèle des examens et des TP. Ce projet n'a pas été constamment une priorité au long de cette année mais c'est un projet qui m'a tenu à cœur.

Pouvoir enfin réaliser quelque chose de vraiment concret a été une de mes motivations principales lors de ce projet. Et même si tout ce que nous avions imaginé n'a pas pu être implémenté, je garde une fierté par rapport au résultat final qui s'avère pour nous tous, être à la hauteur de nos espérances.

En tant que chef de projet, j'ai ressenti le besoin de superviser chaque aspect du projet, chaque fonctionnalité du jeu.

J'ai essayé d'être le plus présent possible pour mes collègues afin d'être le plus efficace possible, tout en n'oubliant pas le travail que je devais fournir moi-même.

J'aurai découvert au long de ce projet les joies de Unity (et surtout les peines), ainsi que le fonctionnement de Git.

Malgré tout les problèmes que nous avons pu rencontré, tant du point de vue du travail que celui de la motivation, je garde un très bon souvenir de ce projet.

8.3.2 Téo

FQTM fût un réel plaisir. Les jeux vidéos représentent une partie importante des mes loisirs. Je ne pensais pas que c'était possible de réaliser un tel projet avec si peu d'expérience. Cela a bien sûr représenté une charge de travail conséquente et n'a pas était facile. Mais j'ai toujours eu du plaisir à réaliser les différentes tâches qui m'ont été confiées.

Tout d'abord ce projet à affiner mes compétences de recherches d'informations déjà existantes. Puis, j'ai découvert Unity, un univers rempli de nouvelles fonctionnalités puissantes mais avec aussi avec ses limites. J'ai aussi appris énormément sur la gestion des dépôts Git et aussi que GitKraken était un outil très pratique.

8.3.3 Baptiste

FQTM était le second gros projet que j'ai eu à faire en groupe dans ma vie, l'expérience de mon premier m'a permis de ne pas reproduire les mêmes erreurs, comme : respecter le cahier de charge, communiquer sur l'avancement avec les autres personnes du groupe et garder un répertoire Git propre.

Ce projet m'a aussi permis de m'améliorer, au début du projet j'utilisais beaucoup de tutoriels et j'avais beaucoup de mal à corriger les bugs, contrairement à maintenant où j'ai pu me décrocher des tutoriels pour coder avec ce que j'ai appris et j'ai pu aider à résoudre certains bugs. Je suis très content et fier de ce que nous avons fait, même si nous avons eut de grandes ambitions, que nous n'avons pas pu toujours combler, le jeu est complet et original.

De plus l'un de mes rôles dans le projet était de réaliser les graphismes du projet, j'ai aimé remplir cette tâche car j'ai pu exprimer ma fibre artistique et j'ai pu initier mes camarades sur des logiciels de graphisme afin de m'aider pour réaliser de simples designs.

8.3.4 Maxence

Pour ma part, ce projet m'a énormément appris à chercher par moi-même ce qui est l'un de mes défauts principaux. J'avais auparavant toujours l'impression que j'avais besoin d'avoir une piste pour commencer ou que je devais être guidée tout du long pour pouvoir mener à bien un projet. Mais, au final, nous avons réussi à bien s'organiser et à faire un jeu dont nous sommes fiers (même si nous aurions aimé faire un jeu plus complet).

Aussi, ce projet m'a appris à savoir s'organiser et prévoir un planning sur des mois, tout cela peut paraître anodin mais ce sont les bases de l'entrepreneuriat et, j'ai pour objectif de monter mon entreprise d'ici quelques années si j'ai la possibilité. Donc j'ai adoré pouvoir créer les plannings, diviser les tâches à faire pour chacun etc...

9 Conclusion

Pour conclure le projet FQTM a connu des débuts difficiles car aucun de nous n'avait une quelconque expérience dans le développement de jeu vidéo. Mais, nous avons réussi à tous nous motiver et atteindre des objectifs assez hauts bien que ce ne soit pas ceux que l'on s'était fixés. Pour chaque membre du groupe ce projet fut une expérience très enrichissante. Ce projet aura été l'occasion d'apprendre de nombreuses notions que nous n'avions jamais vu tel que la gestion de projet avec GitKraken , le développement de jeu avec Unity ou encore le design avec Aseprite. Nous sommes très heureux de pouvoir présenter notre jeu et nous avons hâtes de recevoir les retours des différents joueurs. En espérant que FQTM réussira à les divertir.



10 Annexes

10.1 Personnages et ennemis



Figure 1: Larbin



Figure 2: Larbin Supérieur



Figure 3: Snarbolax



Figure 4: Snarbolax Supérieur



Figure 5: Golem



Figure 6: Golem Supérieur



Figure 7: Boss du jeu



Figure 8: Personnage principal

10.2 Bâtiments



Figure 9: Machine d'assemblage



Figure 10: Foreuse



Figure 11: Four



Figure 12: Nid d'alien



Figure 13: Bobine tesla



Figure 14: Base de départ



Figure 15: Coffre



Figure 16: Mur

10.3 Items du jeu



Figure 17: Armes du jeu



Figure 18: Circuit électrique



Figure 19: Lingot de cuivre



Figure 20: Bobine de cuivre



Figure 21: Minerai de fer



Figure 22: Minerai de charbon



Figure 23: Minerai de cuivre



Figure 24: Lingot de fer



Figure 25: Organe de monstre 1



Figure 26: Organe de monstre 2

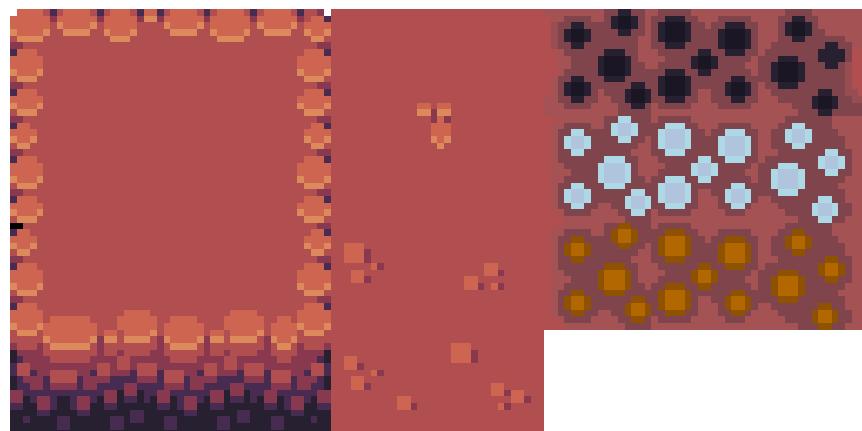


Figure 27: Organe de monstre 3



Figure 28: Poutine dorée

10.4 Terrain



10.5 Menus et interfaces



: Menu Bâtiments

10.6 Site internet



Figure 29: Page principale

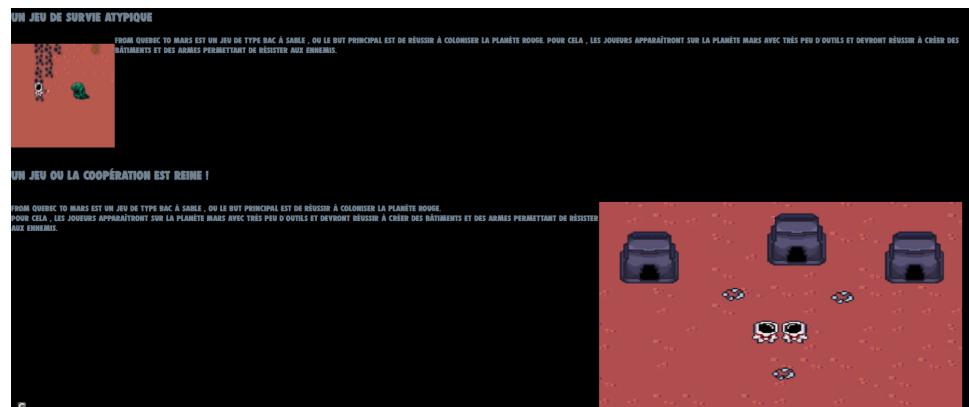


Figure 30: Bas de la page principale

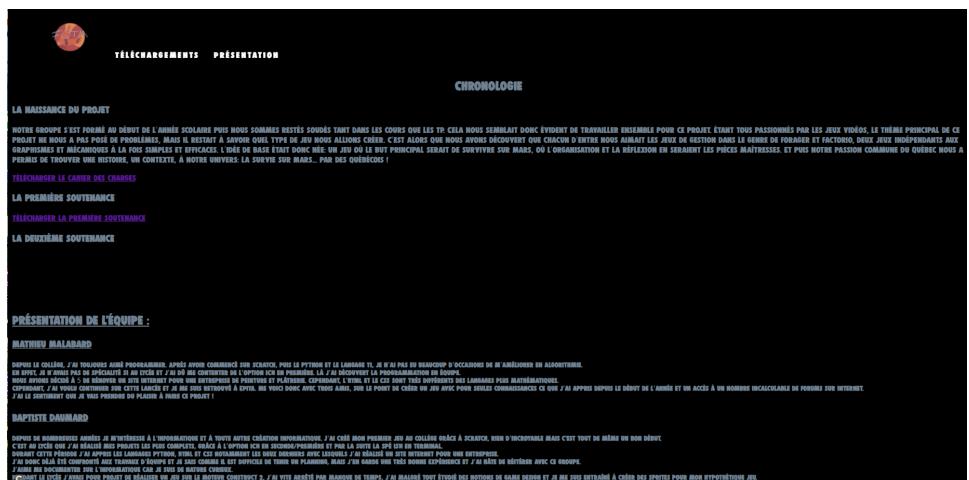


Figure 31: Page de présentation

11 Remerciements

Pour ce projet, nous aimerais tout d'abord remercier tous ces sites ou vidéos de personnes qui donnent accès à un panel d'information gigantesque totalement gratuitement. On pourrait par exemple citer les chaînes Youtube de "Brackeys" ou encore "Code Monkey".

Ensuite nous voulons remercier toutes nos familles qui nous ont motivé en nous faisait croire qu'ils aimaient vraiment notre jeu et qu'ils le trouvaient "marrant".

Enfin, nous finirons par remercier les profs qui nous ont aidés à nous améliorer à chaque soutenance et nous ont dit les erreurs que nous avions commises, comment nous pourrions nous améliorer et comment en éviter.

