
KaizerWald
Castle Defense

Florian Duruz

UN TRAVAIL PRÉSENTÉ DANS LE CADRE D'UNE FORMATION
CFC



FORMATION PROFESSIONNELLE ACCÉLÉRÉE
CENTRE PROFESSIONNEL NORD VAUDOIS
SUISSE
MAI 2022

RÉSUMÉ

Situation de départ

L'objectif de ce projet est de créer un jeu type castle defense, où le but est de défendre un château contre des vagues successives d'ennemis qui tenteront d'accéder l'intérieur du château en détruisant les portes et les unités contrôlés par le joueur.

Contrairement à un tower defense, la tactique joue un rôle important car l'IA ennemie s'arrêtera pour se défendre contre les troupes du joueur, les mécaniques de jeux devront inciter le joueur à utiliser la tactique pour remporter la victoire face à un ennemi supérieur en nombre.

Mise en oeuvre

Pour Mener à bien ce projet, les éléments suivants ont été mis en place :

Intéractions des régiments

Comprends : Présélection, Sélection, Placement, Déplacement, Ordre d'attaque

Pathfinding HPA*

Il s'agit d'un système de pathfinding plus sophistiqué utilisé dans les grosses industries afin d'améliorer les performances qui deviennent problématiques très rapidement à mesure que la taille du terrain augmente ainsi que le nombre d'entités ; le principe est de découper le terrain en partitions afin de précalculer des chemins à l'intérieur des dites partitions.

Système de combat

les régiments auront un système de tir automatique dès qu'une entité est à porté, il faudra qu'une entité puisse jauger le risque de tir ami et choisir de ne pas tirer le cas échéant.

En mêlé il faudra que les troupes tiennent la formation.

Résultat

La plupart des objectifs ont une base solide mais rien qui permet de faire un jeu complet, les objectifs étaient bien trop ambitieux pour un temps aussi court, malheureusement cet état de fait n'a pu être constaté qu'après analyses complètes des besoins et des cas limites.

Le Système de combat et la partie la plus incomplète car intrinsèquement dépendant d'une implémentation complète et finie de la structure du régiment qui bien que fonctionnelle actuellement manque d'éléments indispensables au système de combat, comme la destruction contrôlée des entités.

TABLE DES MATIÈRES

Table des figures	4
Glossaire	5
1 Analyse préliminaire	6
1.1 Introduction	6
1.2 Objectifs	7
1.3 Planification initiale	8
2 Analyse / Conception	10
2.1 Convention de nommage	10
2.2 Menu Principal	12
2.2.1 Tests de validations	14
2.3 Régiments	14
2.3.1 Interactions	14
2.3.2 Régiments : Présélection	17
2.3.3 Régiments : Sélection	19
2.3.4 Régiments : Placement	20
2.3.5 Régiments : Réarrangement	29
2.3.6 Tests de validations	30
2.4 Pathfinding HPA	31
2.4.1 Introduction	31
2.4.2 Décomposition des tâches	32
2.4.2.1 Cluster	33
2.4.2.2 Les Portes	34
2.4.2.3 Connexion entre les Portes	35
2.5 Stratégie de test	36
2.6 Risques techniques	37
3 Réalisation	38
3.1 Général	38
3.2 Planification finale	39
3.3 Éléments repris du pré-tpi	40
3.4 Régiment	41
3.5 Pathfinding : Hierarchical Pathfinding A*	42
3.6 Description des tests effectués	43

3.7	Erreurs restantes	48
4	Conclusion	50
4.1	Objectifs atteints / non-atteints	50
4.2	Points positifs / négatifs	50
4.3	Suites possibles pour le projet	51
4.4	Bilan Personnel	51
A	Appendix	53
A.1	Journal de Travail	54
	Bibliographie	58

TABLE DES FIGURES

1.1	Total War : Tir en formation	7
2.1	Feuilles de Style	13
2.2	Les Fusiliers en blanc sont présélectionnés (cercles bleus à leurs pieds)	17
2.3	Le carré dessiné par le joueur présélectionne chaque régiment qu'il enveloppe . .	18
2.4	Les marqueurs jaunes (cercles au pieds des unités) identifient les régiments sélectionnés	19
2.5	Les marqueurs blancs indiquent où doivent se déplacer les unités	20
2.6	Note : pour simplifier la compréhension, l'espacement entre les unités est de 0 .	22
2.7	Réarrangement automatique d'une collection type List	29
2.8	Réarrangement suivant l'algorithme	29
3.1	asset de l'unity assez store[6](modifié) utilisé dans le projet	38
3.2	Prototype : les points noirs représentent les portes que va emprunter le régiment .	42
3.3	Ordinateur : CPNV	44
3.4	Ordinateur : Gaming	45
3.5	Ordinateur : Gaming	46
3.6	Ordinateur : CPNV	47
3.7	Une fois au sol, le programme interdit à ces unités de tirer	48
3.8	Napoléon lui-même n'aurait jamais accepté cela..	48

GLOSSAIRE

API (Application Programming Interface) : Collection de fonctionnalités permettant aux services d'une applications de communiquer entre eux. 29

Intelligence artificielle : l'intelligence artificielle (IA) fait référence à des systèmes ou des machines qui imitent l'intelligence humaine pour effectuer des tâches et qui peuvent s'améliorer en fonction des informations collectées grâce à l'itération.. 52

Mod : Un mod (abréviation de modification) est une modification par une personne tierce d'un jeu vidéo existant, se présentant sous la forme d'un greffon qui s'ajoute à l'original, pour ajouter une fonctionnalité ou modifier les fonctionnalités existantes. . 6

Multithreading : Un processeur est dit multithread s'il est capable d'exécuter efficacement plusieurs threads simultanément. Contrairement aux systèmes multiprocesseurs (tels les systèmes multi cœur), les threads doivent partager les ressources d'un unique cœur¹ : les unités de traitement, le cache processeur et le translation lookaside buffer ; certaines parties sont néanmoins dupliquées : chaque thread dispose de ses propres registres et de son propre pointeur d'instruction. Là où les systèmes multiprocesseurs incluent plusieurs unités de traitement complètes, le multithreading a pour but d'augmenter l'utilisation d'un seul cœur en tirant profit des propriétés des threads et du parallélisme au niveau des instructions. Comme les deux techniques sont complémentaires, elles sont parfois combinées dans des systèmes comprenant de multiples processeurs multithreads ou des processeurs avec de multiples cœurs multithreads. . 38

Real Time Strategy Game : Jeu de stratégie en temps réel. 19, 38

Separation of Concerns : Séparations des responsabilités, chaque système doit avoir une unique responsabilité. 15

CHAPITRE 1

ANALYSE PRÉLIMINAIRE

1.1 Introduction

Le Projet consiste à la création d'un jeu de stratégie en temps réel sous la forme d'un Castle Defense(défense de château) un genre surtout représenté dans le monde du Mod et qui était très populaire dans le RTS¹ Warcraft 3 ; ce projet est réalisé dans le cadre d'un travail personnel individuel(TPI) pour l'établissement du CPNV.

L'objectif de ce projet est avant tout de travailler sur l'intelligence artificielle plus précisément sur l'algorithme lié à la recherche de chemin qui est le centre du projet, le combat sera aussi abordé mais restera dans un forme plus basique et se concentrera sur l'analyse du comportement des entités dans le cadre d'un combat au sein d'un groupe/régiment. Ce projet s'inscrit dans la continuité du Pré-TPI qui avait pour but de me familiariser et de poser les bases des algorithmes utilisés dans ce projet.

Les algorithmes suivants ont été mis en place :

- ☒ **A*** : probablement l'algorithme le plus populaire qui a l'avantage de toujours trouver le chemin le plus court et est aussi très performant.
- ☒ **FlowField** : Algorithme développer pour répondre un à besoin bien spécifique aux RTS, permet à tout un groupe d'avoir la direction a prendre via un champs de vecteurs placé sur le terrain, chaque Vecteur Indiquant le chemin à prendre.

Il y a cependant un problème millénaire qui frappe les jeux utilisant ces algorithmes, plus le terrain est grand plus le processeur peine à calculer les chemins, ce phénomène est à multiplier par le nombre d'entités devant calculer ces chemins.

Un algorithme plus sophistiqué à vu le jour pour répondre à ce problème, le HPA ou hierarchical Pathfinding A* qui sera au coeur du projet et qui régira les mouvements des entités dans le projet.

Une attention particulière sera aussi apporté à l'architecture et à la structure du code, ce projet étant destiné à avoir une continuité, il est impératif que le code soit lisible afin d'assurer un maintien continu.

1. Real Time Strategy Game (Jeu de stratégie en temps réel)

1.2 Objectifs

Gestion de groupe des entités

Cette partie comprendra l'interaction du joueur avec des entités fonctionnant en groupe les principaux objectifs comprendront :

☒ **Sélection par groupe** : Une entité ne sera jamais sélectionnable individuellement, quand le joueur sélectionne une unité tous sont régiment sera sélectionné.

☒ **Déplacement en groupe** : Sûrement la partie la plus complexe, ce sujet à fait coulé beaucoup d'encre et ne semble pas avoir trouvé de solution uniforme, chaque solution à ce jour correspondant à des besoins spécifiques.

L'objectif sera de développer un algorithme en conjonction avec l'algorithme Pathfinding² (décris plus bas) permettant aux entités de se déplacer comme une seule unité et de donner un semblant de cohésion lors des déplacements.

☒ **Combat en formation** : Autant pour les attaques à distance que au corps à corps, les régiments devront se battre comme une unité et ne pas briser le rang, dans le cadre du corps à corps, la formation pourra être déformée mais jusqu'à une certaine limite fixée arbitrairement.

Ce point sera un élément central en terme de jeu, car la formation aura un impact direct sur la capacité du régiment en combat.

Combat à Distance : Les troupes ne pourront tirer que sur une rangée(voir image : 1.1) inspiré du système de tir des anciens jeux total war, une formation allongée offrira donc une plus grande capacité de feu, cependant cela se fera au détriment de la capacité de défense(décrise au point suivant)

Combat en Mêlée : Une formation plus compacte offrira une meilleure capacité de défense quand le régiment est engagé au corps à corps.



Figure 1.1 – Total War : Tir en formation

Algorithme Pathfinding HPA³

Sur la base des travaux effectués lors du pré-tpi, il faudra mettre en place un système de hierarchical Pathfinding A* en apportant un changement afin d'adapter l'algorithme au besoin de projet, à savoir l'utilisation d'un Flowfield (ou champs de vecteurs).

Pour ce faire, les éléments suivant devront être implémenté :

☒ **Génération d'une grille partitionnée** : Il faudra en premier lieu générer un grille sur tout le terrain en utilisant l'outil mis en place précédemment. Cette dernière devra être partitionnée afin de pouvoir cibler les futurs calculs à une région spécifique.

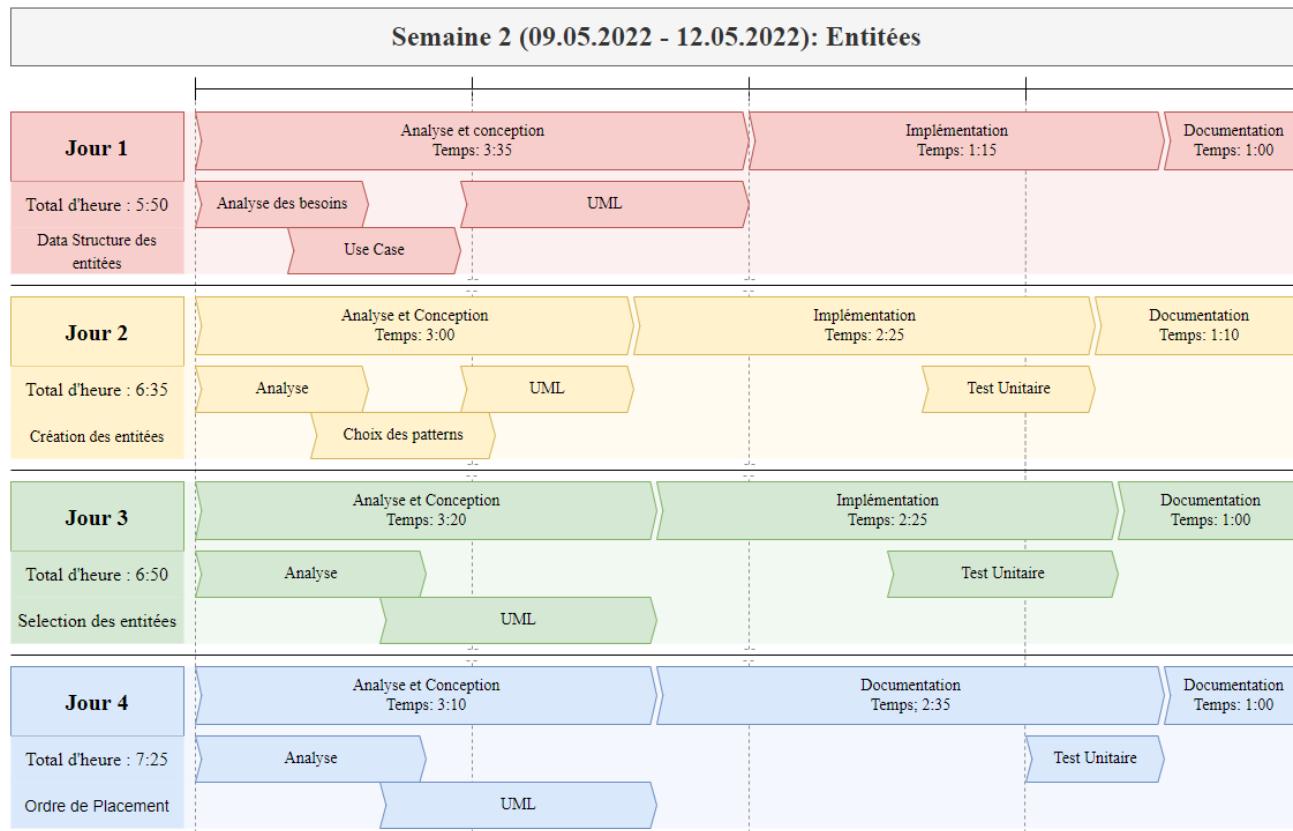
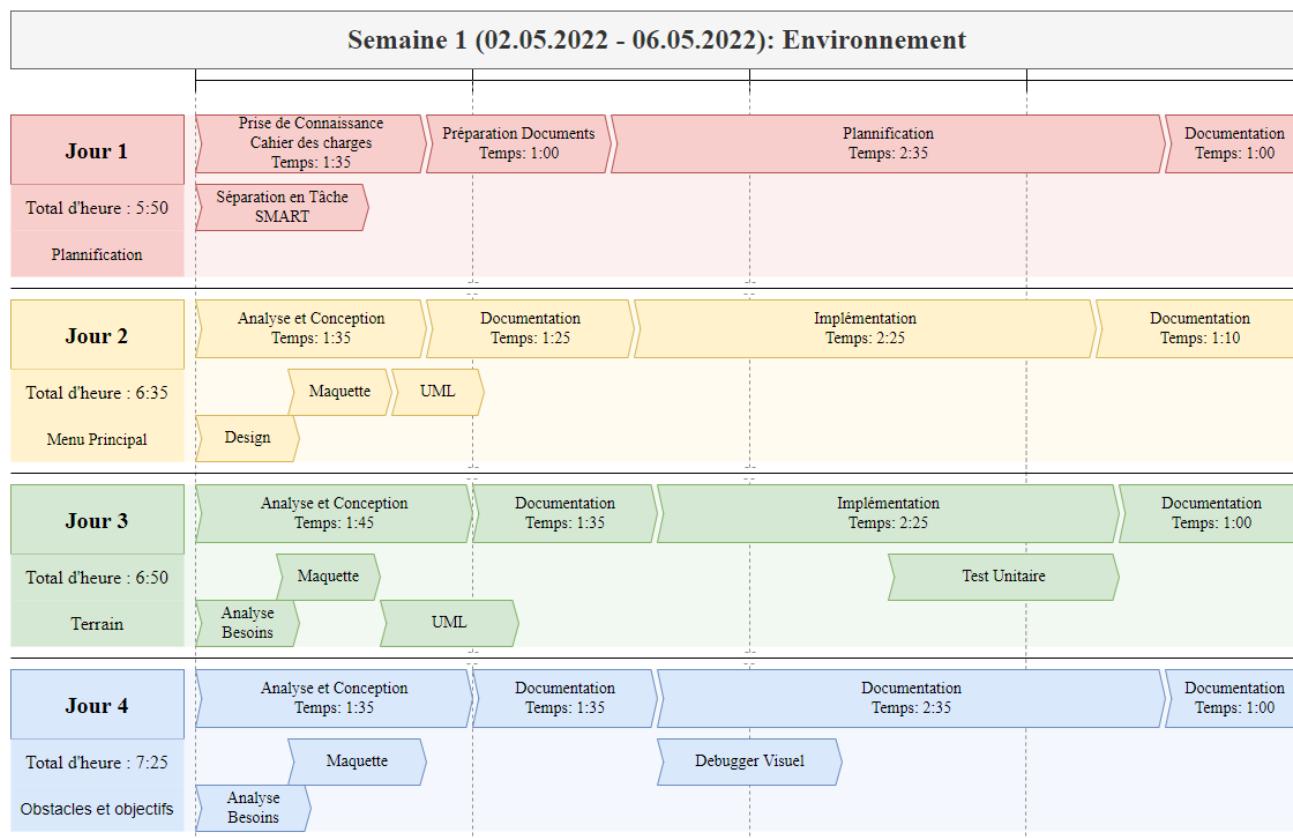
☒ **Structure de donnée** : Chaque partition devra contenir un certains nombre d'informations afin de permettre le bon fonctionnement de l'algorithme(détaillés dans la partie analyse de ce document)

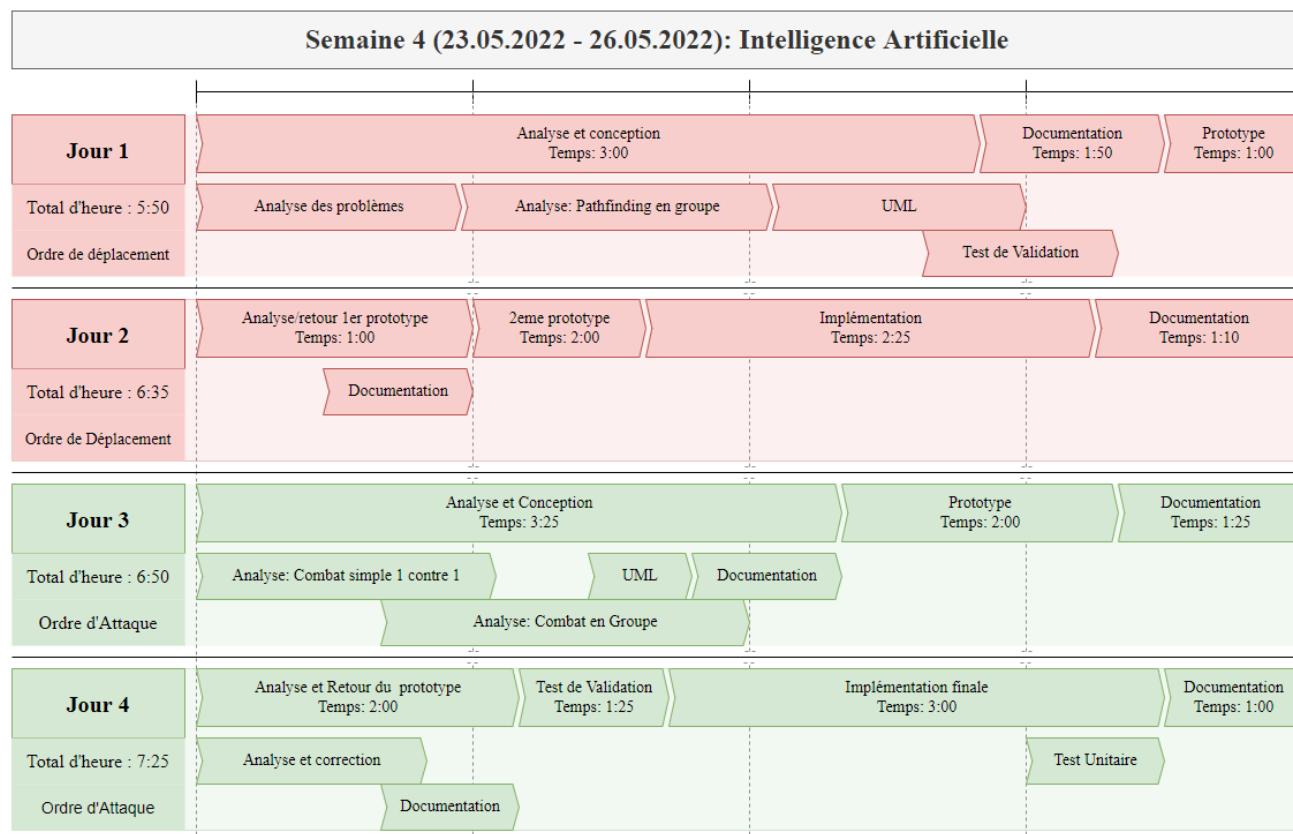
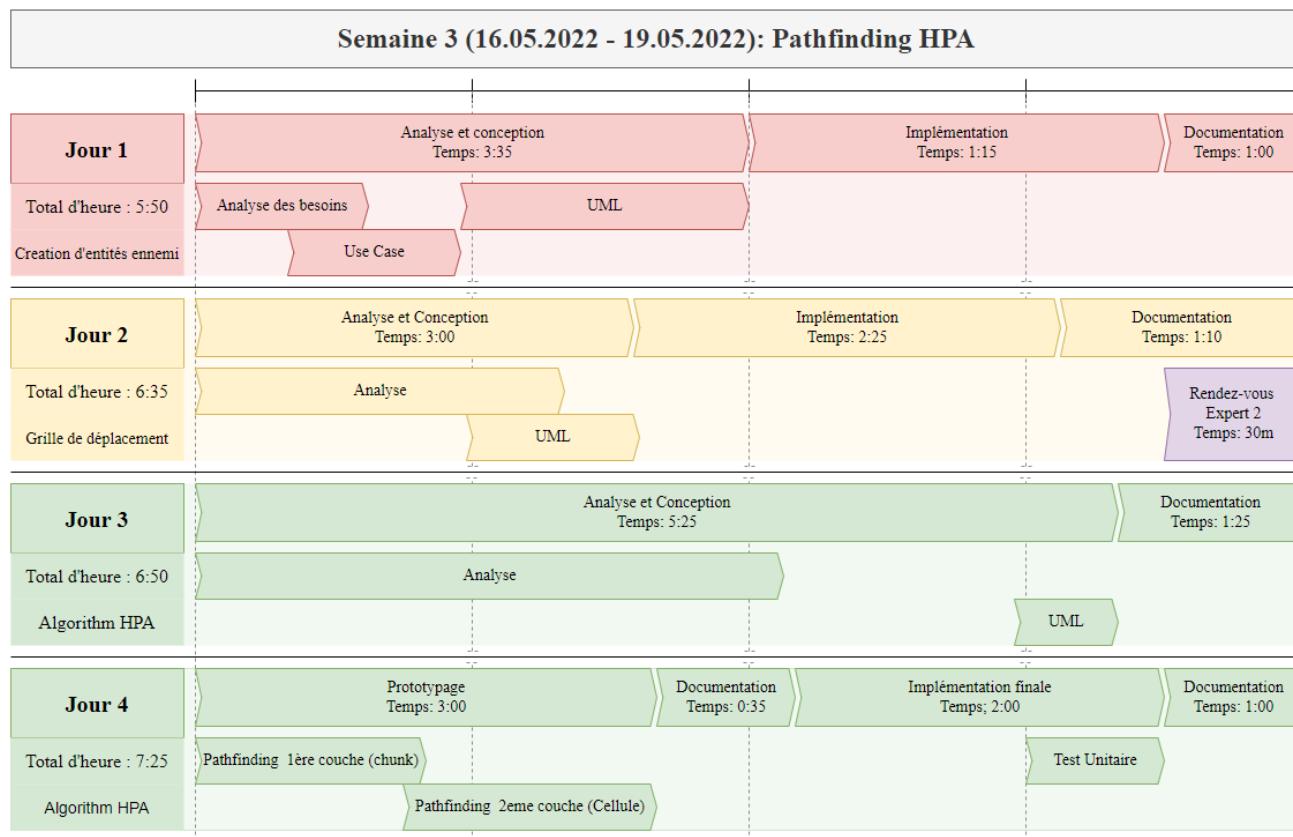
☒ **Adaptation de l'algorithme HPA au Flowfield** : La version traditionnelle du HPA utilise uniquement le A*(il s'agit en réalité d'une abstraction en plusieurs couches de cet algorithme), il est malgré tout possible d'y inclure des algorithmes différents dont le flowfield qui sera utilisé.

2. Pathfinding : Recherche de Chemin

3. Hierachical Pathfinding A*

1.3 Planification initiale





CHAPITRE 2

ANALYSE / CONCEPTION

2.1 Convention de nommage

Général

S'aligne en général avec les conventions de C#[4] sauf pour le typage qui doit être explicite. La raison est que les algorithmes utilisent beaucoup de mathématique et les types sont sujet à beaucoup de conversions, afin de faciliter la lecture, le typage est explicite afin que le lecteur n'ai pas à traquer les conversions de type. Faisant aussi gagner du temps au lecteur qui n'aura pas à aller regarder à droite du "=" pour connaître le type.

- ⌘ **Indentation** : A la ligne(ou block) selon les conventions C#.
- ⌘ **Indentation Inline** : Les méthodes de type "Setter" de champs doivent utiliser le body-expression de C#(=>) au lieu des crochets({ }).
- ⌘ **Typages** : Explicite uniquement.

Langue du projet

S'alignant sur les conventions principalement utilisées en Suisse romande

- ⌘ **Code** : Anglais.
- ⌘ **Commentaire** : Français.
- ⌘ **Commits github** : Français.
- ⌘ **Documents externes** : Français.

Code

Suis globalement les conventions Usuelles de C# à l'exception des variables privés qui normalement a le préfixe "_", ce dernier a été abandonner car il s'agit d'une vieille pratique qui viendrait selon certaines sources du C et qui a été reprise mais qui ne fait pas sens car contrairement au C nous n'avons pas besoin de préciser si la variable est globale ou locale, ce préfixe n'ajoute donc rien.

- ⌘ **Méthode** : CamelCase - 1^{ère} Lettre Majuscule - Sans préfixe.
- ⌘ **Interface** : CamelCase - 1^{ère} Lettre Majuscule - préfixe "I".

- ☒ **Champs privée** : pascalCase - 1^{ère} Lettre Minuscule - pas de préfixe.
- ☒ **Champs publique** : CamelCase - 1^{ère} Lettre Majuscule - pas de préfixe.
- ☒ **Propriété** : CamelCase - 1^{ère} Lettre Majuscule - pas de préfixe.
- ☒ **Variable privée** : pascalCase - 1^{ère} Lettre Minuscule - pas de préfixe.
- ☒ **Constante** : SNAKE_CASE - Tout en Majuscule - pas de préfixe.
- ☒ **Sealed** : Tout héritage comprenant des composantes virtuels ou abstracts devra préciser préciser un sealed sur la classe parente.

Particularité Unity

- ☒ **SerializedField** : Les champs ou propriétés ayant l'attribut [SerializedField] prennent une majuscule (même si le champs est privé).
- ☒ **Indentation** : les événements liés à Monobehaviour(Awake, Start, Update,etc...) ne doivent jamais utiliser le Body-expression de C#, Même si il n'y a que une ligne.
- ☒ **Job System** : les structs comprenant une interface liée au job system de Unity prendra en préfixe "J".

2.2 Menu Principal

Le Menu principal peut sembler être un sujet trivial, par définition le joueur n'est pas sensé y passer beaucoup de temps, cependant s'agissant de la première image du jeu présenté au joueur et donc la première impression donnée au joueur, il est important de soigner son aspect ; le menu principal sera en effet le premier élément jugé par le joueur.

Design : Qu'est-ce qu'un bon menu principal ?

Malgré de nombreuses recherches il ne semble pas y avoir une convention ou des règles établies, j'ai donc décidé de m'inspirer d'un jeu ayant un thème similaire : New World créé par Amazone et dont l'interface graphique avait été très appréciée.

Sans entrée dans le design, prenons quelques points qui de mon point de vue ont un rôle dans le succès de l'interface.

⌘ Fondu :

la fenêtre n'est pas un carré régulier, les bords sont irrégulier donnant une impression de fondu adoucissant l'effet de rupture avec le jeu,

⌘ Sobriété :



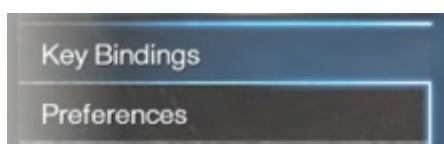
Aucune couleurs vives même les éléments actifs restent dans des tons ternes.

⌘ Transition douce :



Un bouton ne change jamais entièrement de couleur, la transition est comme une lumière qui serait sous le bouton nous laissant suggérer la forme qu'avait le bouton, ne donnant pas l'impression désagréable que le bouton a été entièrement changé.

⌘ Le visuel suggère la transition qui va suivre :



Dans le cadre d'un choix multiples, la lumière est présente sur le côté où va apparaître la réponse

Unity a depuis un moment effectué une transition majeure dans l'élaboration des interfaces utilisateurs ; via le package UI Toolkit, la nouvelle interface utilise maintenant un système proche de ce qui se fait en développement web au delà de quelques nomenclatures, le système est en tout point identique au html fonctionnant via un fichier UXML d'où il est possible d'écrire manuellement l'interface via le code ; Unity laisse aussi la possibilité de créer l'interface en plaçant les éléments manuellement directement sur le canvas.

La Mise en forme elle est faite via un code USS (équivalent de CSS), là encore Unity propose un raccourci sur l'éditeur via une fenêtre d'inspection donnant un accès direct aux paramètres modifiables d'un élément sélectionné.

Il est cependant conseillé dans le cadre de la mise en forme de créer des feuilles de style USS qui pourront être reprises dans l'interface graphique (Voir Figure : 2.1) afin de garder une constance dans le design des éléments et éviter de recréer plusieurs fois le même objet et de pouvoir les réutiliser dans d'autres projets, en effet toutes modifications css via l'interface graphique créées une nouvelle propriété dans le fichier UXML.

Maquette du menu Principale

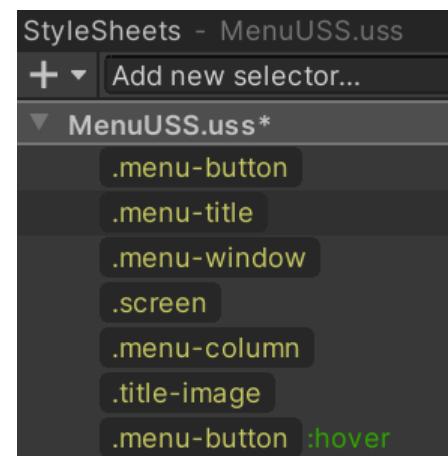
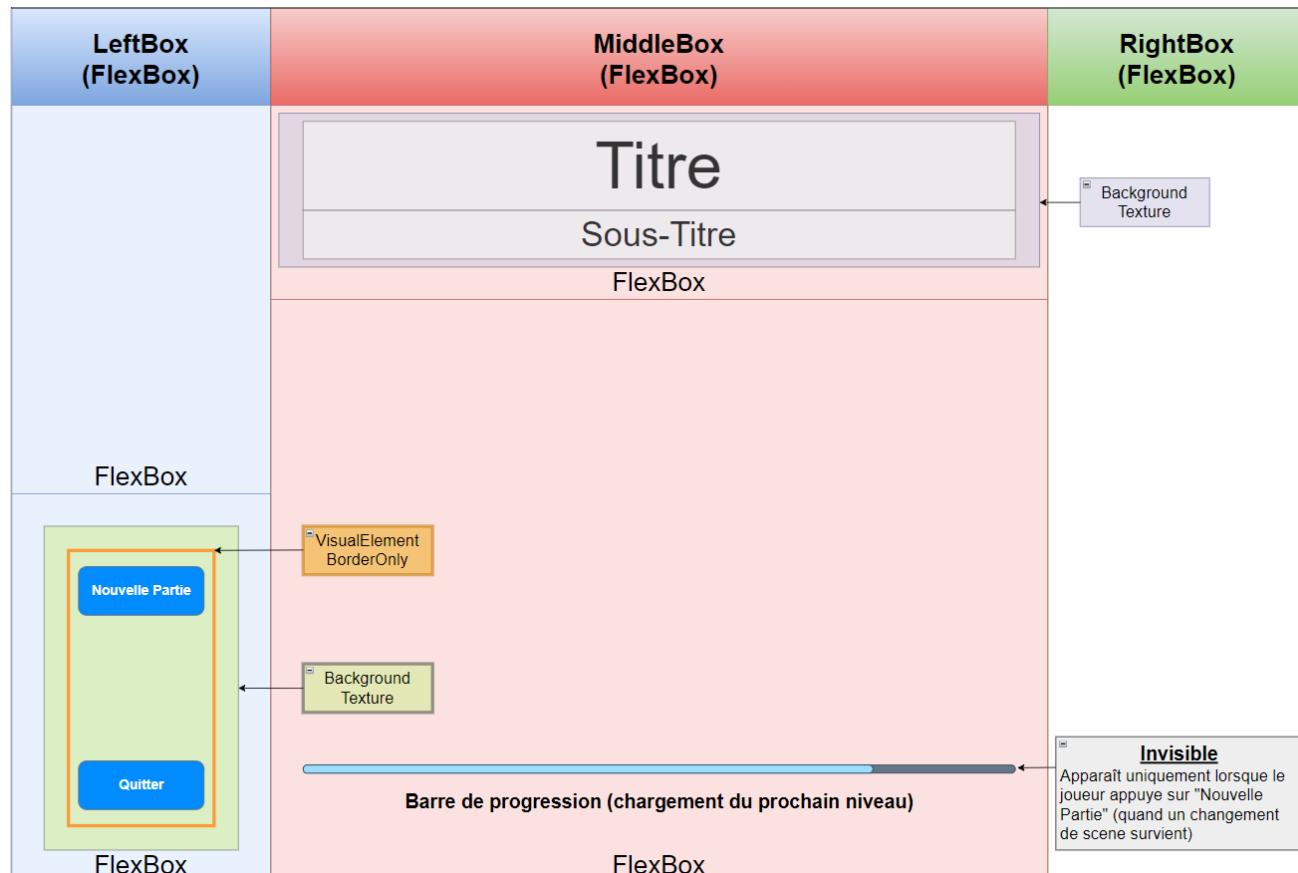


Figure 2.1 – Feuilles de Style

2.2.1 Tests de validations

Tests d'Acceptations : Menu Principal				
Ordre	Contexte	Condition	Resultat	Validation (Oui/Non)
1	J'ouvre le Jeu		Le Menu et ses composantes apparaissent	
2	J'appuye sur le bouton "Nouvelle Partie"		La barre de chargement apparaît	
3	La barre de chargement est complète		la scène de jeu est chargée	
4	J'appuye sur "Quitter"		Le jeu se ferme	

2.3 Régiments

La structure des entités en tant que régiment/groupe demande de mettre au clair quelque aspect avant l'élaboration de la conception :

2.3.1 Interactions

Retenant le cahier des charges, il nous faudra dans le cadre des interactions, concevoir :

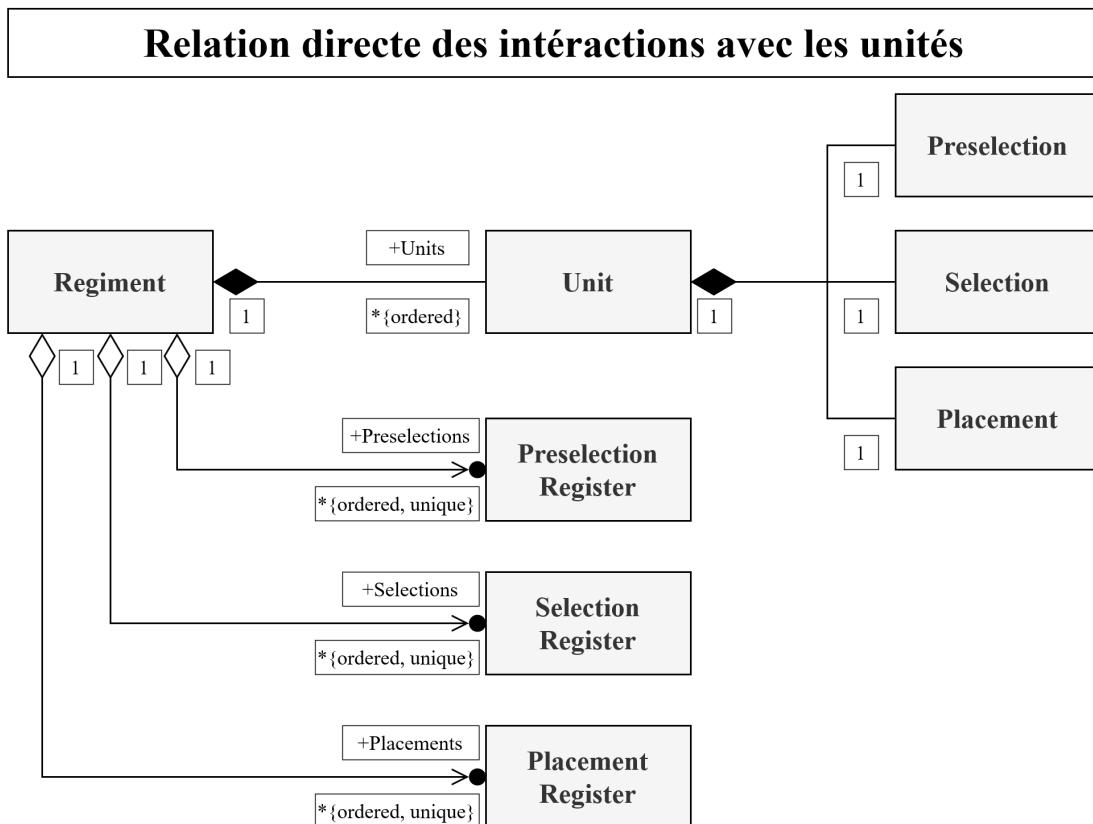
- ☒ La Présélection des régiments.
- ☒ La Sélection des régiments.
- ☒ Le Placement des régiments.

Chacun de ces éléments correspondra à un objet qui pourra être activé ou désactivé, mais à qui revient cette responsabilité ?

Registre : Correspond à la liste des éléments contenu dans les systèmes suivants :

- Présélection
 - Sélection
 - Placement

deux schémas peuvent être proposés chacun ayant des implications profondes dans la structure du programme :



Cette forme prévoit un lien direct entre les unités et les différents indicateurs dans une relation parent-enfant où l'unité serait le parent.

Avantages

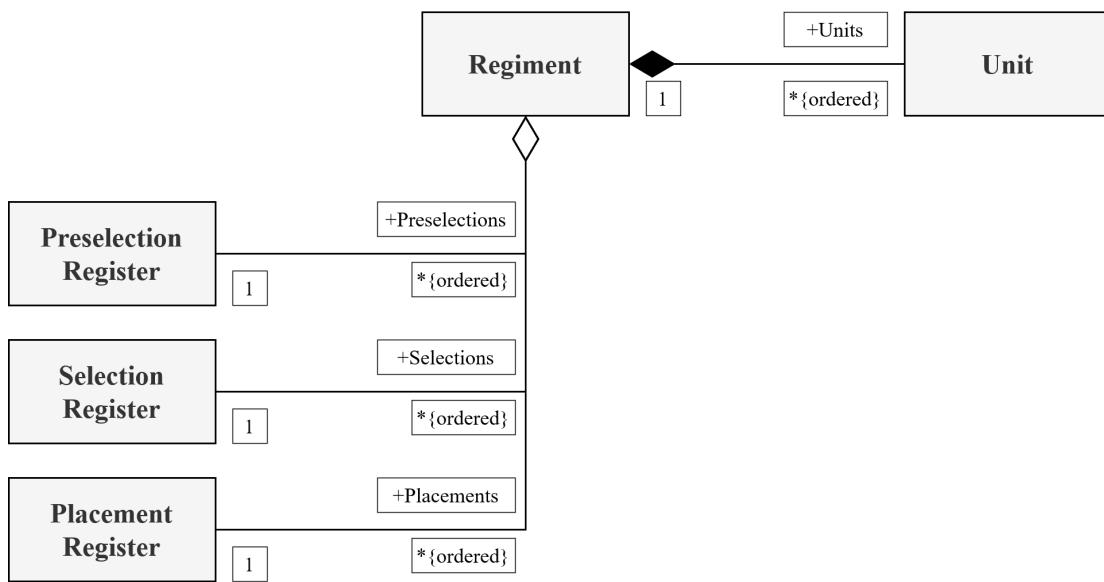
- ☒ Gestion des déplacements gérée par Unity.
- ☒ Gestion de la destruction des marqueurs simplifiée.
- ☒ Besoin réduit de passer par un système complexe d'interfaces/abstracts.

Inconvénients

- ☒ Architecture ne respectant pas le principe Separation of Concerns¹; il y a une longue chaîne de dépendances entre le système en charge de activer/désactiver les marqueurs ce qui rend le programme sensible aux changements (un changement dans la classe qui gère le régiment peu casser le système de sélection par exemple).
- ☒ Accès aux ressources d'un système à un autre potentiellement problématique.

1. Separation of Concerns : Séparations des responsabilités, chaque système doit avoir une unique responsabilité

Relation indépendante des interactions avec les unités



L'autre solution est de rendre totalement indépendant chaque marqueur des unités du régiment dans le sens où un marqueur ne serait pas lié à une référence spécifique d'une unité en jeu.

Avantages

- ☒ Gestion des registres à la destruction d'une unité, moins complexe.
- ☒ Gestion de la destruction des marqueurs simplifiée
- ☒ Séparation propre et claire des responsabilités

Après consultation avec le chef de projet, la deuxième solution est préférée, pour la raison suivante :

Souhaitons-nous pouvoir sélectionner une unité individuellement (sans sélectionner tout son régiment) ? Non, une autre raison est la séparation de chaque fonctionnalité en système distinct permettant de respecter l'exigence SOC.

Le partage des ressources ce fera via un coordinateur qui référencera les trois systèmes et sera en charge de faire transiter les ressources d'un système à un autre.

Inconvénients

- ☒ Gestion manuelle des mouvements de chaque marqueur.
- ☒ Implémentation d'un système d'interfaces/abstracts complexe pour obtenir un code lisible.
- ☒ Gestion de cas particuliers liée à la gestion manuelle du mouvement des marqueurs.

2.3.2 Régiments : Présélection

Pourquoi ? Les entités fonctionnant en groupe, il est important que le joueur puisse déterminer quels régiments seront sélectionnés avant d'engager l'action, notamment dans les cas où plusieurs régiments seraient positionnés au même endroit, leurs unités s'entremêlant les une aux autres (voir image ci-dessous).

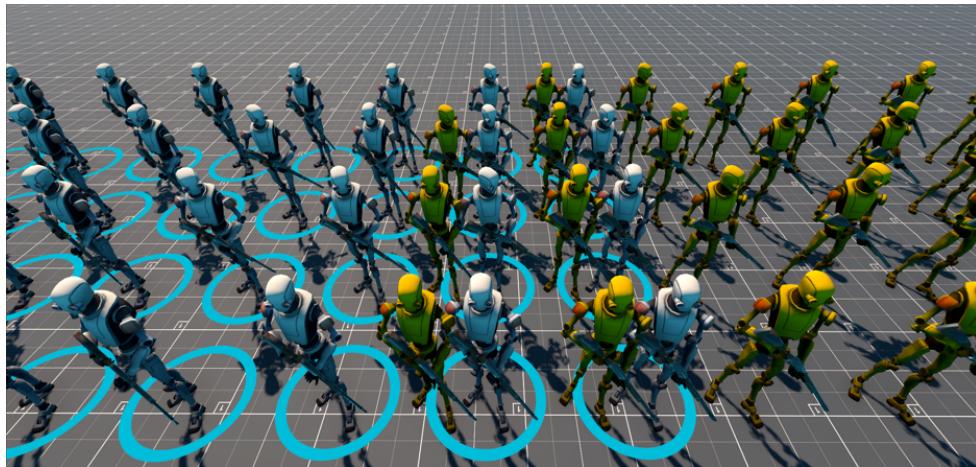


Figure 2.2 – Les Fusiliers en blanc sont présélectionnés (cercles bleus à leurs pieds)

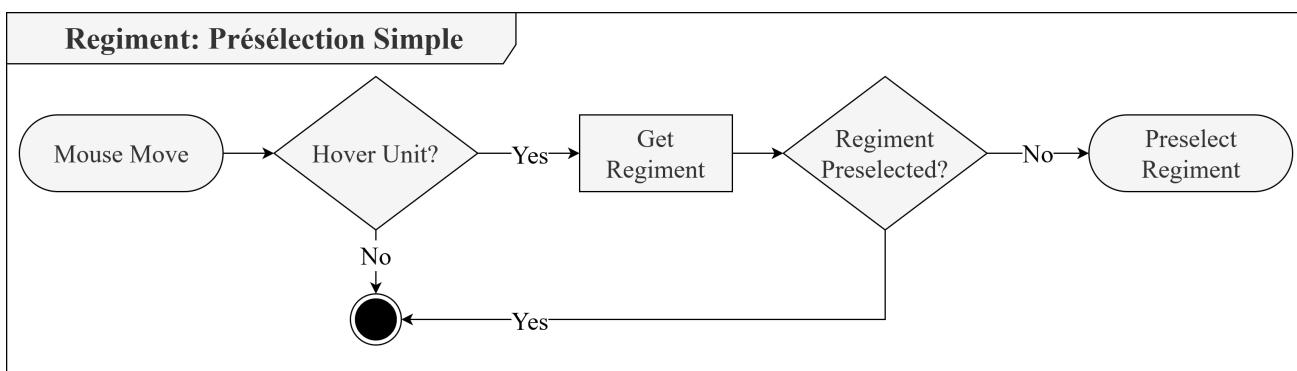
La Présélection fonctionnera de pair avec la sélection(voir chapitre suivant : Sélection d'un régiment) en effet ce système informant à l'avance quels régiments seront sélectionnés, il lui suffira d'envoyer un message à la classe concernée pour effectuer la sélection.

Il y aura deux types de présélection distincts :

Présélection Simple

Cette dernière ce fera lorsque le curseur passe par dessus une unité, dès lors le système va récupérer le régiment auquel appartient la cible, puis va provoquer la présélection de tous ses membres.

Schématique la structure du programme donnera quelque chose de ce type :



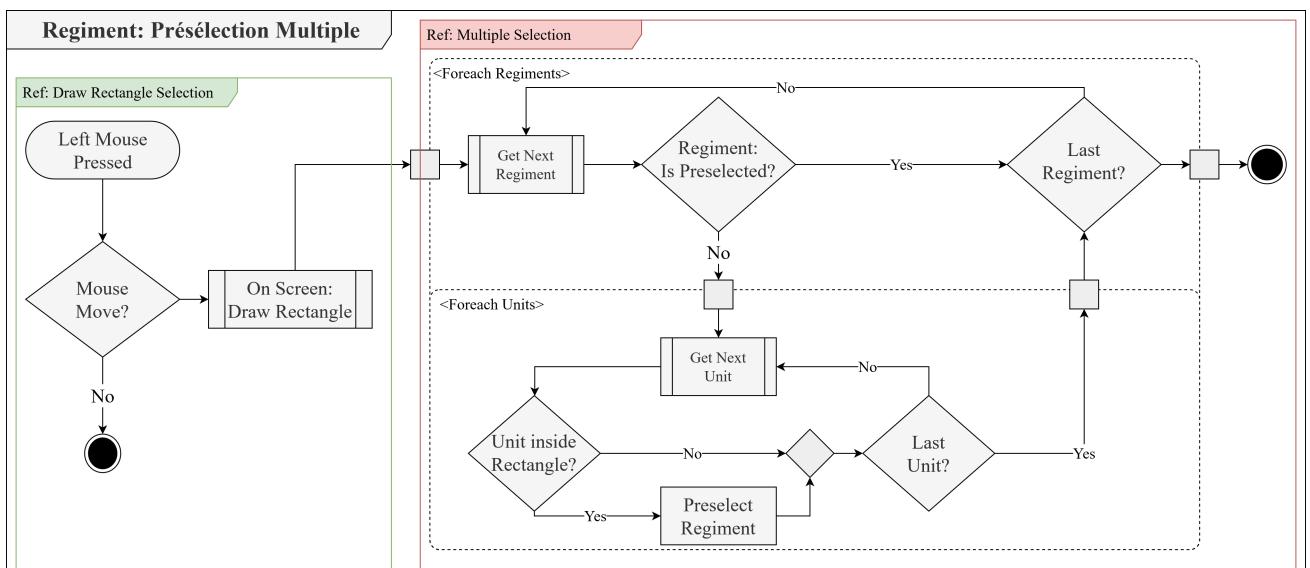
Présélection Multiple

La Présélection devra permettre au joueur de sélectionner plusieurs entités en maintenant le clique gauche de la souris enfoncé ; en bougeant ensuite la souris, un rectangle se dessinera, ainsi toutes les unités sous le rectangle verront leur régiment être sélectionné.



Figure 2.3 – Le carré dessiné par le joueur présélectionne chaque régiment qu'il enveloppe

Il y aura donc deux séquences : le "dessin" du rectangle et la détection des régiments présélectionnés.



2.3.3 Régiments : Sélection

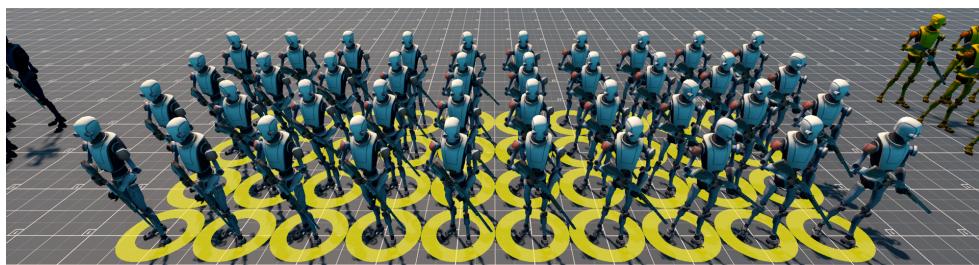


Figure 2.4 – Les marqueurs jaunes (cercles au pieds des unités) identifient les régiments sélectionnés

La sélection sera un système très court car reprenant bon nombre du travail qui a déjà été fait par la présélection(voir : chapitre présélection), avec cependant une fonctionnalité supplémentaire : "Ajouter uniquement" ; qu'entend-on par là ?

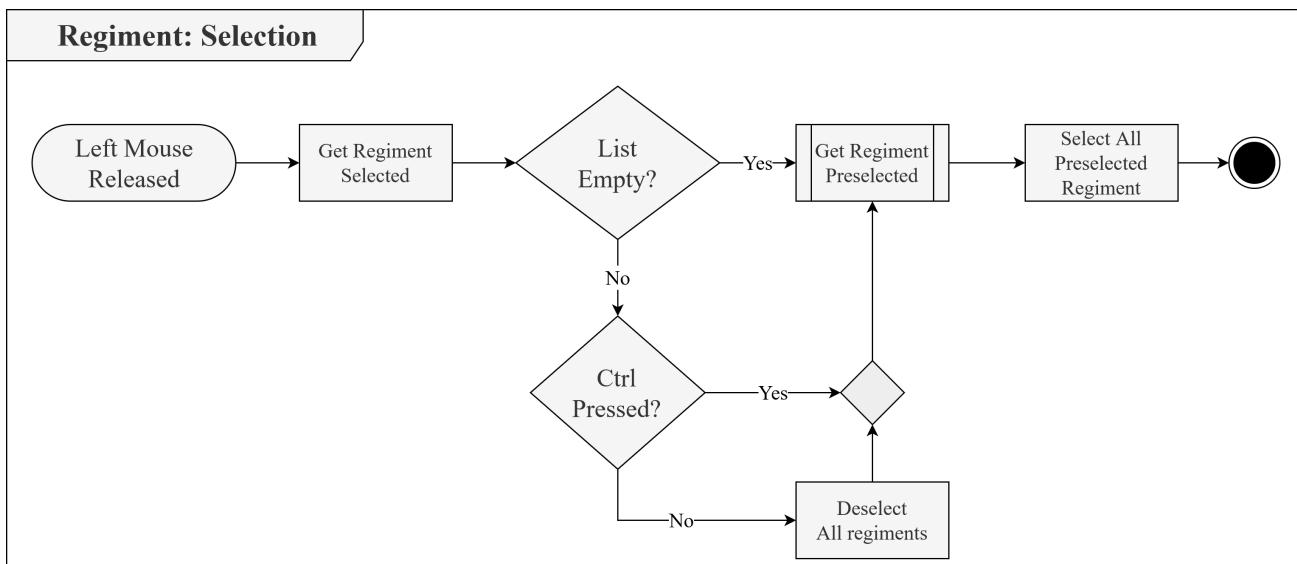
La fonctionnalité de base est la suivante :

"Quand je relâche le clique gauche de ma souris, désélectionne tous les régiments, et sélectionne tous les régiments préselectionnés".

Note Importante

l'Accès aux régiments préselectionnés qui est une ressource externe au système de sélection se fera via un coordinateur mentionné précédemment.

Mais le joueur voudra peut-être ajouter des régiments à la sélection actuelle sans avoir à les présélectionner à nouveau ceux déjà sélectionnés ; une mécanique connue dans les Real Time Strategy Game et de désactiver la possibilité de désélectionner sitôt que une touche est maintenue enfoncee (généralement : Ctrl Gauche du clavier).



Attention : Remarquez que la sélection survient après la présélection ; ce qui implique que la déprésélection des régiments survient avant vidant la liste des présélections avant que le système de sélection ne puisse la consulter.

2.3.4 Régiments : Placement

La partie la plus technique dans l'interaction des régiments tant le nombre de cas particuliers sont légions et les adaptations nécessaires pour parer à toutes éventualités augmentent la complexité de l'algorithme.

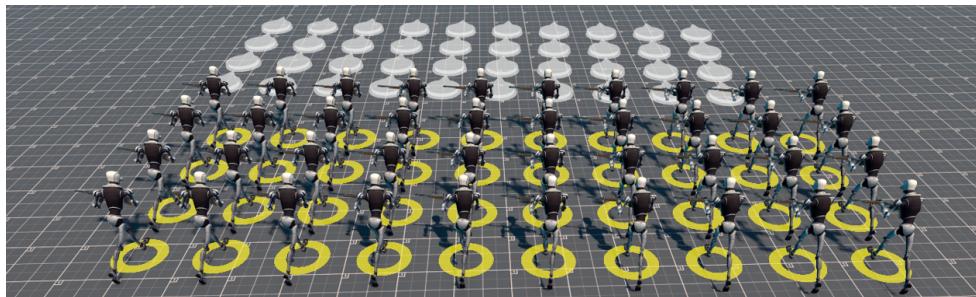


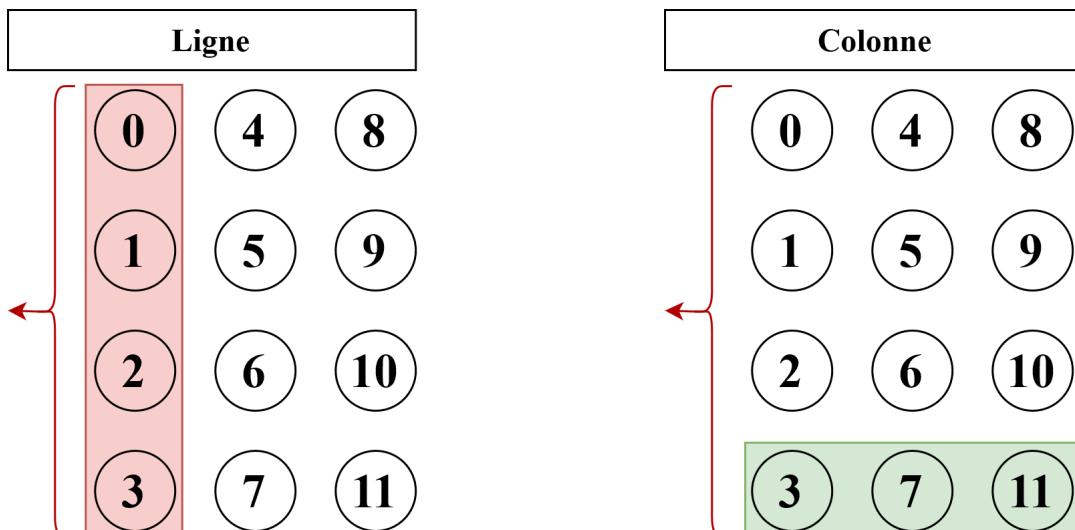
Figure 2.5 – Les marqueurs blancs indiquent où doivent se déplacer les unités

Afin de faciliter la compréhension nous irons étape par étape ajoutant les complexités une à une afin de faciliter l'analyse et l'implémentation, et dans le pire des cas de permettre de proposer une implémentation même imparfaite qui permette de tester les autres fonctionnalités du projet.

Nomenclature

- ☒ **Distance du curseur** : distance parcourue par le curseur entre le début du clique gauche et sa position actuelle :
- ☒ **Taille minimale d'un régiment(ligne ou colonne)** : cette dernière comprend le nombre d'unités autorisées multiplié par la taille d'une des unités en question en prenant en compte l'espacement entre les unités.
- ☒ **Configuration** : On parle dans le cadre du régiment des variables dynamiques suivantes : Nombre actuelle : d'unité par ligne, d'unité par colonne, d'unité sur la dernière colonne.

Les termes "Colonnes" et "Lignes" seront souvent utilisés, au vu du caractère dynamique des formations il est important de poser les définitions suivantes :

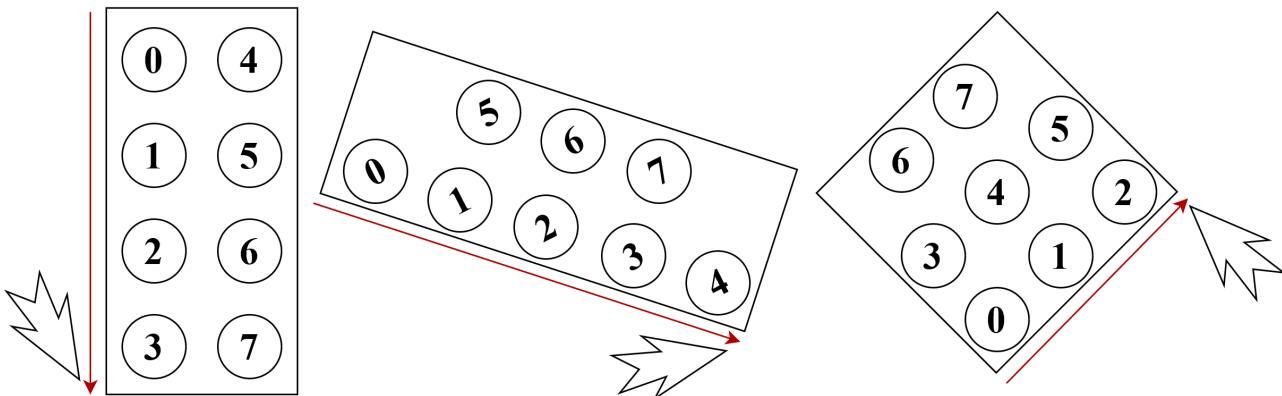


Les Lignes correspondent aux rangées perpendiculaire à la direction où regarde le régiment.

Les Colonnes correspondent aux rangées parallèles à la direction où regarde le régiment.

Implémentation de base

Pour cette partie nous allons nous concentrer sur le cas d'une seule sélection, le résultat souhaité et de positionner dynamiquement le régiment sélectionné en effectuant la séquence suivante : Bouton droit de la souris pressé + bouger la souris.



Les paramètres suivants seront à prendre en compte :

- ⌘ La direction du défilement de la souris
- ⌘ Le minimum d'unité par rangée qui est autorisé pour le régiment sélectionné
- ⌘ Le maximum d'unité par rangée qui est autorisé pour le régiment sélectionné
- ⌘ Le nombre d'unités sur la dernière rangée.

Note Importante

Le nombre de colonne et de ligne d'un régiment changeant dynamiquement, l'utilisation d'un array multidimensionnelle est proscrit !

Pour récupérer les valeurs x (Index dans la ligne) et y (N° ligne) il faudra appliquer la formule suivante :

Soit :

int i : L'indexe de l'unité dans le régiment.

int width : Nombre d'unités par ligne dans la configuration actuelle du régiment.

int y = i / width;

int x = i - (y * width);

Avant d'atteindre le point de déformation

La première phase consiste à vérifier la distance parcourue par le curseur, les régiments ayant, quelque soit leur type une taille de ligne minimale, aussi lorsque la distance parcourue par le curseur n'est pas supérieure d'au moins une taille d'unité (unité qui compose le régiment) le régiment conservera sa configuration.

Ce qui implique que lorsque la "distance du curseur" est inférieur à la taille minimale du régiment cette dernière devra tout de même apparaître, cependant, seul une rotation sera effectuée.

Note : bien que le terme rotation est utilisé, le positionnement sera recalculé pour des raisons de temps, la rotation en question étant un calcul complètement différent.

Algorithme

Quelques variables utilisées seront

- ☒ **Position de départ(Start)** : Position du curseur au moment du clique droit (cette position sera celle de la première unité du régiment).
- ☒ **Position Actuelle(End)** : Position actuelle du curseur.
- ☒ **Distance parcourue par le curseur** : Distance(End-Start).
- ☒ **Direction du placement** : Direction normalisé du vecteur formée par End-Start.
- ☒ **Espace entre les unités** : Correspond à l'espacement fixe entre deux unités d'un régiment et qui est définie par la classe du régiment.
- ☒ **Taille d'unité** : Correspond à la taille d'une unité d'un régiment et est défini dans la classe du régiment.

Nombre d'unités par ligne

La première variable à mettre à jour est le nombre d'unité par ligne, il nous faudra pour cela calculer le nombre d'unités pouvant être placé sur la longueur tracée par le curseur.

Pour obtenir le nombre d'unités par ligne il nous faut faire le calcul suivant :

$$\text{Nombre d'unités par ligne} = \frac{\text{Distance parcourue par le curseur}}{\text{Taille d'unité} + \text{Espace entre les unités}} + 1$$

Pourquoi ajouter 1 ?

La première unité n'est pas comprise, dans les faits, le calcul exprime plutôt le nombre de séparations entre les unités, en partant du point "start" au point "end". Que nous pouvons interpréter comme le nombre de coordonnées valides après la première unité (impliquant qu'elle n'est pas incluse).

Dans les deux cas, il nous faut ajouter la position "Start" correspondant à la première unité.

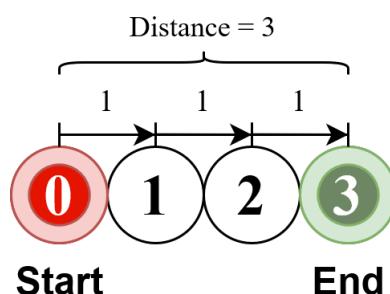
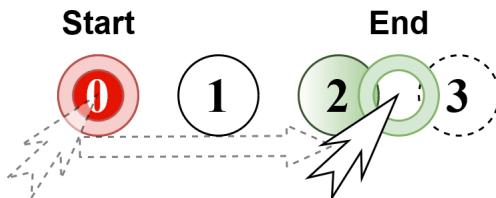
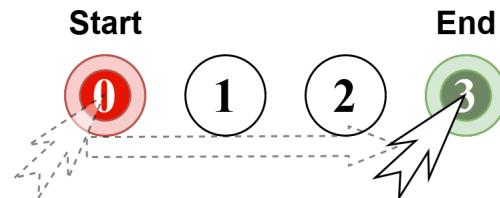


Figure 2.6 – Note : pour simplifier la compréhension, l'espacement entre les unités est de 0

Il faudra aussi arrondir à l'inférieur la valeur obtenue, Attention cependant, la longueur étant un type float, il faudra utiliser un comparateur prévu pour ce type dans notre cas, Unity fourni : Mathf.Approximate(float a, float b).



la ligne comprendra les coordonnées :
 $\{ 0, 1, 2 \}$



la ligne comprendra les coordonnées :
 $\{ 0, 1, 2, 3 \}$

Nombre de lignes complètes

Il nous faut obtenir combien de lignes sont complètes, par cela nous entendons que le nombre de troupe est égal au nombre d'unité par ligne obtenu au calcul précédent :

$$\text{Nombre de lignes complètes} = \frac{\text{Nombre total d'unités}}{\text{Nombre d'unités par ligne}}$$

Note : Il est conseillé d'utiliser un type non décimal pour la formule, autrement il faudra arrondir à l'inférieur le résultat obtenu.

Direction des colonnes arrières

Dans cette partie il nous faut déterminer la direction du régiment afin de placer les lignes d'unités dans le sens souhaité ; le comportement souhaité du point de vu du joueur est le suivant :

Ce qu'il faut retenir

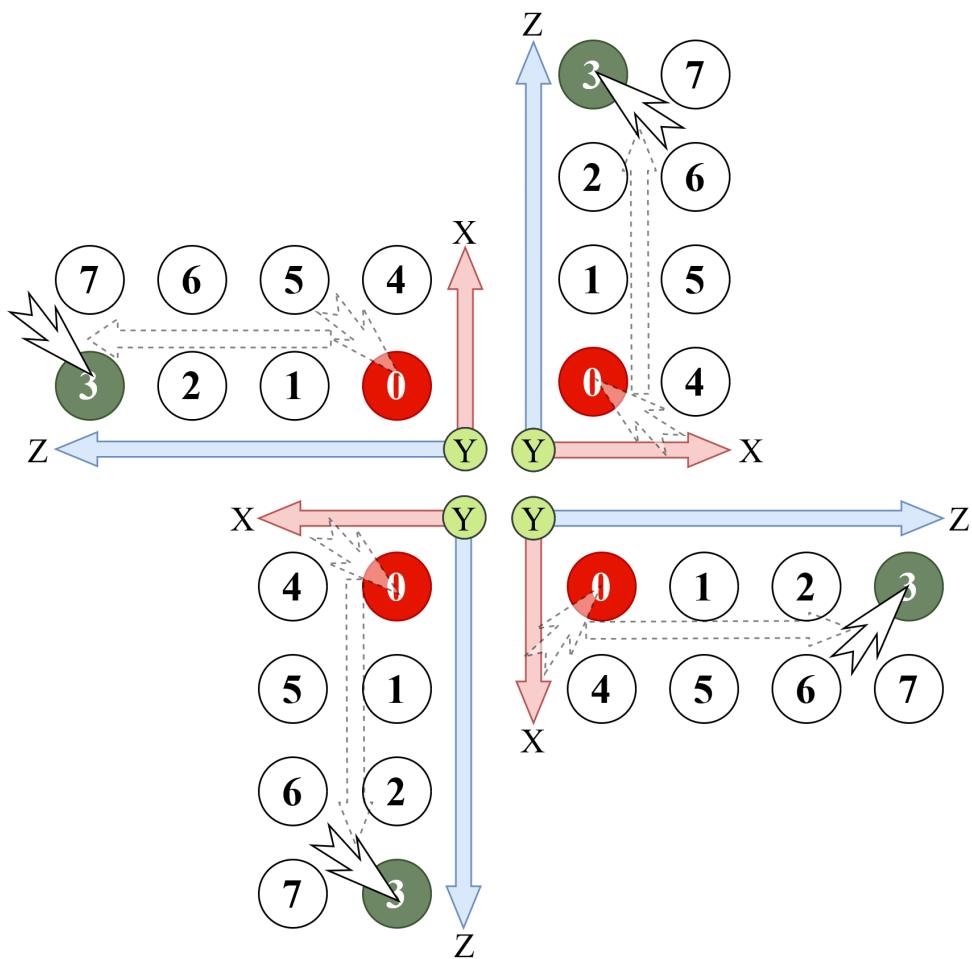
- 1) Le déplacement du curseur est représenté par l'axe Z dans le calcul.
- 2) Le vecteur recherché correspond vecteur perpendiculaire négatif à Z.

Il est important de noter qu'étant dans un environnement en trois dimensions, le calcul du vecteur perpendiculaire comporte une particularité, en effet il nous faudra utiliser l'axe Y. Nous n'irons pas dans les détails, cependant cette information nous permet d'utiliser le vecteur négatif de Y où la valeur de la coordonnée Y = (0, -1, 0).

Pour le calcul nous utiliserons la formule proposée par le moteur Unity : Vector3.Cross(Vector3 lhs, Vector3 rhs) où lhs = direction du curseur, et rhs = vecteur Y ou (0, -1, 0). Le calcul prendra ainsi la forme suivante :

$$\text{Direction colonne} = \text{Vector3.Cross}(\text{Direction du curseur}, \text{Vector3}(0, -1, 0))$$

Important : la direction ainsi obtenue doit être normalisée pour que les calculs suivants soient corrects.



Nombre d'unités sur la dernière ligne du régiment

Enfin nous pouvons obtenir le nombre d'unités sur la dernière ligne du régiment par la formule suivante :

Nombre d'unités dernière ligne =

Nombre total d'unités – Nombre de lignes complètes * Nombre d'unités par ligne

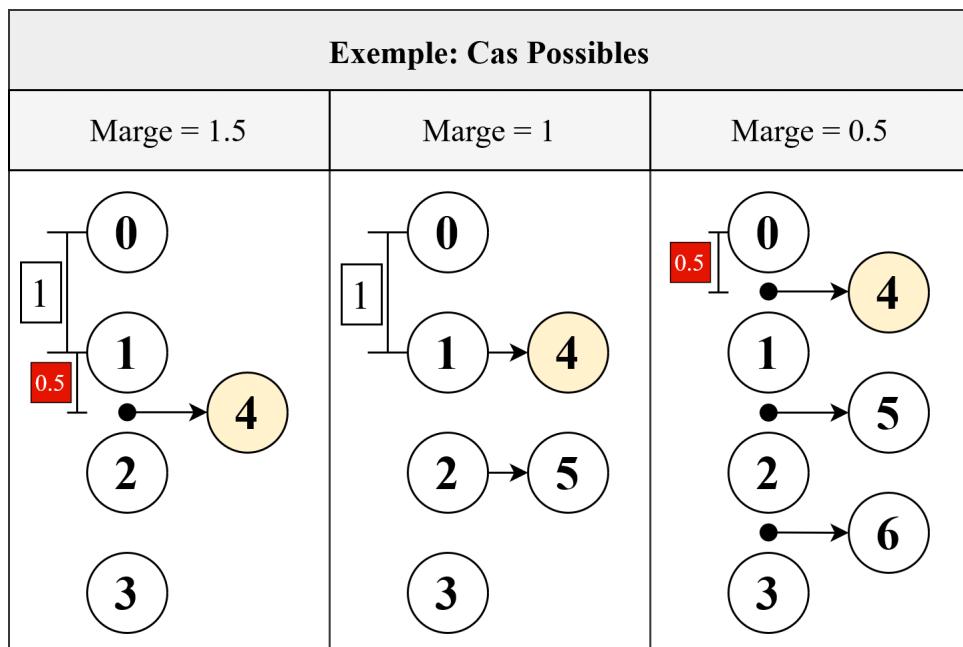
Note : Il est conseillé d'utiliser un type non décimal pour la formule, autrement il faudra arrondir à l'inférieur le résultat obtenu.

L'importance de la dernière ligne

La dernière rangée est assez technique, en effet cette dernière peut ne pas être pleine pouvant donner des cas où les unités seront décalées d'une demi taille (taille d'une unité du régiment + espacement dans le régiment) afin que la dernière ligne soit bien centrée comme le montre les exemples ci-dessous.

Les Valeurs ci-dessous sont à prendre comme une marge à ajouter avant chaque placement d'unités sur la dernière ligne(voir l'algorithme complet plus loin).

Le calcul pour obtenir cette valeur est le suivant :



$$\text{Marge} = \frac{\text{Nombre d'unité par ligne} - \text{Nombre d'unités dernière ligne}}{2}$$

Algorithme complet

```

1  Vector3 [] GetRegimentFormation(Regiment regiment)
2  {
3      float UnitSize = regiment.UnitSize;
4      float UnitOffset = regiment.UnitsOffsetInRegiment;
5
6      float totalUnitSize = UnitSize + UnitOffset;
7
8      //Position de la souris sur le terrain au moment du clique droit
9      Vector3 StartMouse;
10     //Position actuelle de la souris sur le terrain
11     Vector3 EndMouse;
12
13     float MouseDistance = Vector3.Distance(EndMouse - StartMouse);
14
15     //Guard Clause
16     if(MouseDistance < regiment.minRowDistance) return;
17
18     Vector3 [] formationPositions = new Vector3[regiment.TotalUnits];
19
20     int NumberUnitPerLine = 1 + MouseDistance/(UnitSize+UnitsOffset);
21
22     float numRows = regiment.TotalUnits / (float)NumberUnitPerLine;
23     //nombre de lignes
24     int totalNumRow = Mathf.CeilToInt(numRows);
25     //nombre de lignes complètes
26     int numCompleteRow = Mathf.FloorToInt(numRows);
27
28     int NumberUnitLastRow = regiment.TotalUnits - numCompleteRow*
NumberUnitPerLine;

```

```

29
30
31
32     for ( int i = 0; i < regiment.TotalUnits; i++)
33     {
34         // Coordonnées dans le régiment et non dans l'espace
35         int y = i / NumberUnitPerLine;
36         int x = i - (y * NumberUnitPerLine)
37
38         //Position sur la ligne ( coordonnée x)
39         Vector3 linePosition;
40         Vector3 lineDirection = (EndMouse - StartMouse).normalized;
41         linePosition = StartMouse + x*(lineDirection*totalUnitSize);
42
43         //Dernière Ligne ?
44         if(y == totalNumRow && totalNumRow != numCompleteRow)
45         {
46             float offset = (NumberUnitPerLine - NumberUnitLastRow) / 2f;
47             linePosition += offset;
48         }
49
50         //Position sur la colonne ( coordonnée y)
51         Vector3 columnPosition;
52         Vector3 yAxis = new Vector3(0, -1, 0); // Vector3.down
53         Vector3 columnDirection = Vector3.Cross(lineDirection, yAxis);
54         columnPosition = linePosition+y*(columnDirection*totalUnitSize);
55
56         formationPositions[ i ] = linePosition + columnPosition;
57     }
58 }
```

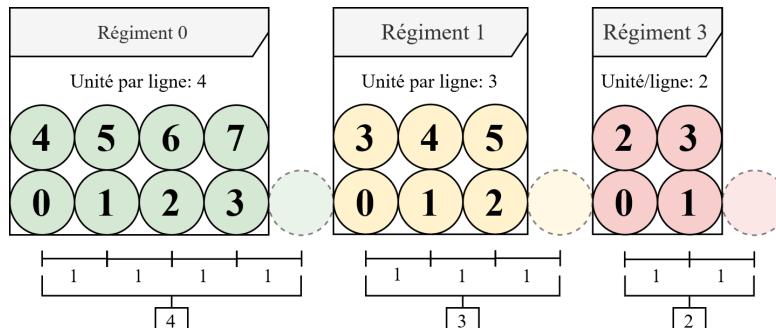
Placement à plusieurs régiments

Lorsque plusieurs régiments seront sélectionnés, il faudra ajouter les éléments suivants à l'algorithme :

- ☒ Il faudra que la distance parcourue permette d'augmenter la taille de chaque régiment.
- ☒ L'emplacement des régiments devra être au plus proche de leur position actuelle.
- ☒ Il y aura une marge à ajouter pour l'unité de départ des régiments.
- ☒ La taille actuelle et après placement sera à prendre en compte.

Distance parcourue : Nombre d'unité à ajouter/réduire par ligne

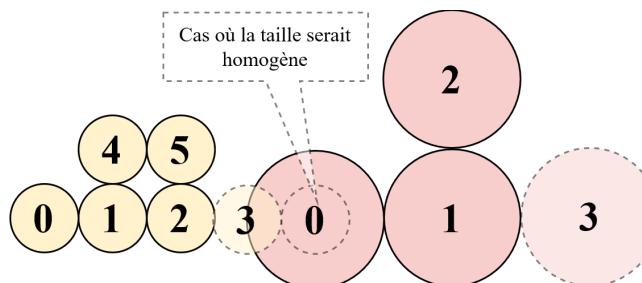
La première Adaptation consiste à redéfinir la règle qui autorise le lancement de l'algorithme, nous avons en effet plusieurs régiments sélectionnés et le comportement souhaité est une "déformation" homogène, autrement dit, il faut que chaque régiment augmente ou réduit leur nombre d'unités par ligne de manière égal.



Attention : à ne pas confondre Nombre d'unités et nombre d'espace entre unités. nous allons utiliserons exclusivement l'espacement entre les unités pour les calculs suivant.

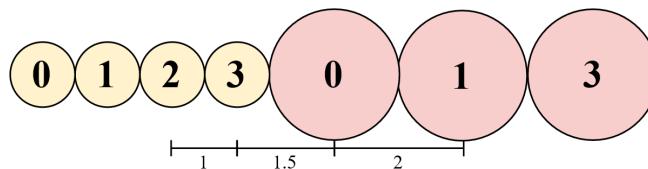
Note Concernant l'espacement entre régiments

Une question à se poser est l'espacement entre les régiments, en effet si dans le cadre du prototype, les unités ont toutes la même taille qu'arrivera-t-il si l'on devait ajouter des unités avec des tailles différentes ?



Il y aurait un risque que les unités se retrouvent les une sur les autres au placement, Il faudra donc effectuer le calcul suivant :

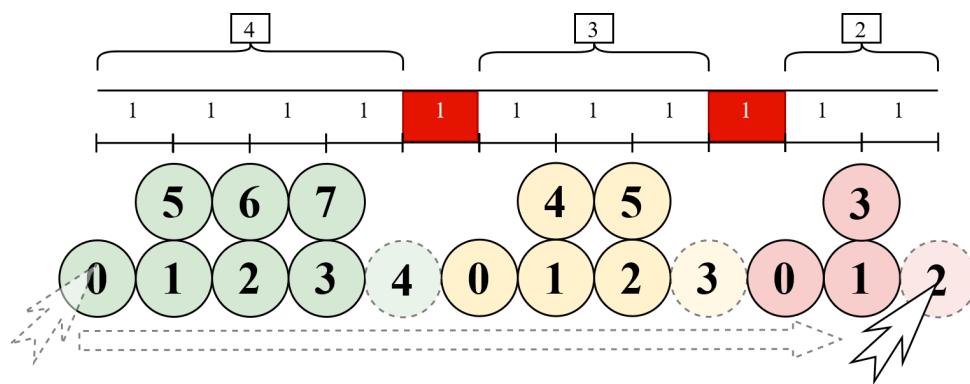
$$\text{Espace entre régiment} = \frac{\text{Espace entre Unité N} + \text{Espace entre Unité N-1}}{2}$$



Le résultat devra être ajouté à la vérification de la distance parcourue par le curseur.

Implication pour la distance parcourue par le curseur :

Important : Dans le cadre du placement multiple, il s'agit davantage de calculer le nombre d'unité à ajouter à la valeur "*Nombre d'unités par ligne*", aussi il faudra que les régiments conservent en mémoire la valeur correspondant au nombre d'unités par ligne actuelle.



le calcul est en deux étapes :

1. Calculer la distance minimale des régiments sélectionnés :

Il s'agit là de prendre la longueur totale des régiments sélectionnés (ou taille de ligne). Attention il s'agit de la taille allant du centre de la première unité jusqu'au centre de la dernière unité (voir partie précédente : Implémentation de base, partie "Nombre d'unités par ligne").

Il faudra ensuite ajouter l'espacement entre les régiments (voir page précédente : "Note Concernant l'espacement entre régiments"). Il faudra très certainement passer au travers d'une boucle utilisant le même ordre que pour le placement des troupes.

2. Diviser le tout par le nombre de régiments sélectionnés.

Enfin le calcul déterminant le nombre d'unités ajouté sera le suivant :

$$\text{Nombre d'unités ajouté*} = \frac{\text{Distance curseur} - \text{Taille minimale des régiments}^{**}}{\text{Nombre de régiments sélectionnés}}$$

* Si le type choisi est à décimal, il faudra arrondir à l'inférieur.

** Comprend aussi l'espacement entre les régiments

Finalement il faudra à chaque itération (sur les régiment) ajouter à la position start (position de départ du curseur) un décalage égal à la taille de la ligne du régiment précédemment placé (+ l'espacement entre régiment calculé précédemment) qu'il faudra multiplier par la direction du vecteur formé par la position de départ et final du curseur.

2.3.5 Régiments : Réarrangement

Un autre aspect à prendre en compte est le réarrangement dynamique d'un régiment quand une de ses unités est tué. Le choix de la collection est de ce fait primordial, il y a un comportement souhaité précis, suivant un algorithme demandant un contrôle précis dans l'agencement des unités au sein d'un régiment.

Il est aussi important que le réarrangement soit effectif et ne doit en aucun cas être simulé via des valeurs temporaires ou en déplaçant uniquement les unités dans la configuration voulue, car cela provoquera des problèmes quand le joueur repositionnera le régiment ; les unités n'ayant pas changées effectivement leur indexe dans le régiment, tenteront de reprendre leur position lors du prochain ordre de repositionnement pour se conformer à la position qu'il leur est assigné dans le régiment, exposant au joueur une vision de chaos et de désordre lui indiquant un manque flagrant d'attention apporté au produit ; bien que fonctionnellement le résultat soit le même, d'autres fonctionnalités pourraient être impactées négativement comme le combats, les unités mettant plus de temps à se reformer sur de courtes distances et donc mettant potentiellement plus de temps se mettre en position de tir.

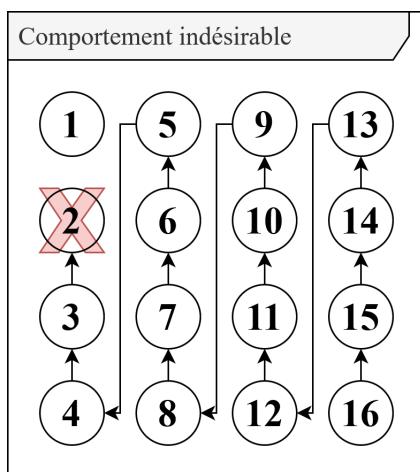


Figure 2.7 – Réarrangement automatique d'une collection type List

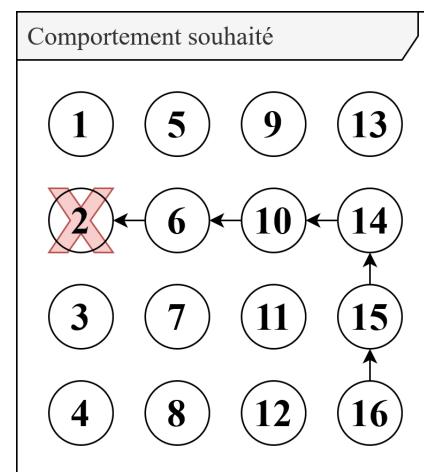


Figure 2.8 – Réarrangement suivant l'algorithme

Sur toutes les collections à disposition en C# c'est l'array qui est choisi pour les raisons suivantes :

- ☒ Flexibilité
- ☒ Manuel : Pas de réarrangement automatique
- ☒ Compatibilité avec l'API du moteur de jeu Unity
- ☒ Performance

Algorithme de réarrangement

L'algorithme utilisé devra prendre en compte les paramètres suivants :

- ☒ Largeur(Ligne) du régiment en nombre d'unité
- ☒ Longueur(Colonne) du régiment en nombre d'unité
- ☒ Index(dans le régiment) de l'unité détruite

Note : Les régiments changeant dynamiquement certaines de leurs propriétés comme la largeur et la longueur, les paramètres seront fixes et égals à l'instant où l'ordre de réarrangement est lancé.

2.3.6 Tests de validations

Présélection

Tests d'Acceptations : Présélection Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	Le curseur passe au dessus d'une unité	unité non préselectionnée	Préselectionne tout son régiment	
2	Le curseur n'est plus au dessus d'une unité	unité préselectionnée	Dépéslectionne le régiment	
3	Je maintiens le clique gauche de la souris enfoncé + je déplace ma souris		Dessine un rectangle à l'écran	
4	Je maintiens le clique gauche de la souris enfoncé + Je déplace ma souris	Passe au dessus d'une unité	Préselectionne tout son régiment	
5	Je maintiens le clique gauche de la souris enfoncé + Je déplace ma souris	Passe au dessus de plusieurs unités ayant appartenant à des régiments différents	Préselectionne le régiment de chaque unité	

Sélection

Tests d'Acceptations : Sélection Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	Je relâche le clique gauche de la souris	curseur sur unité + régiment préselectionné	Sélectionne le régiment	
2	Je relâche le clique gauche de la souris + Ctrl non maintenu	- Autres régiment(s) sélectionné(s) - Curseur sur unité + régiment préselectionné	- Désélectionne les autres régiments - Sélectionne le régiment préselectionné	
3	Je relâche le clique gauche de la souris + Ctrl maintenu	- Autres régiment(s) sélectionné(s) - Curseur sur unité + régiment préselectionné	Sélectionne le régiment préselectionné	
4	Après avoir fait un rectangle (préselection), je relâche le clique gauche de la souris	Un ou plusieurs régiments sont sélectionnées + Touche Ctrl non maintenu	- Désélectionne les autres régiments - Sélectionne les régiments préselectionnés	
5	Après avoir fait un rectangle (préselection), je relâche le clique gauche de la souris	Un ou plusieurs régiments sont sélectionnées + Touche Ctrl maintenu	Sélectionne les régiments préselectionnés	

Placement/Mouvement

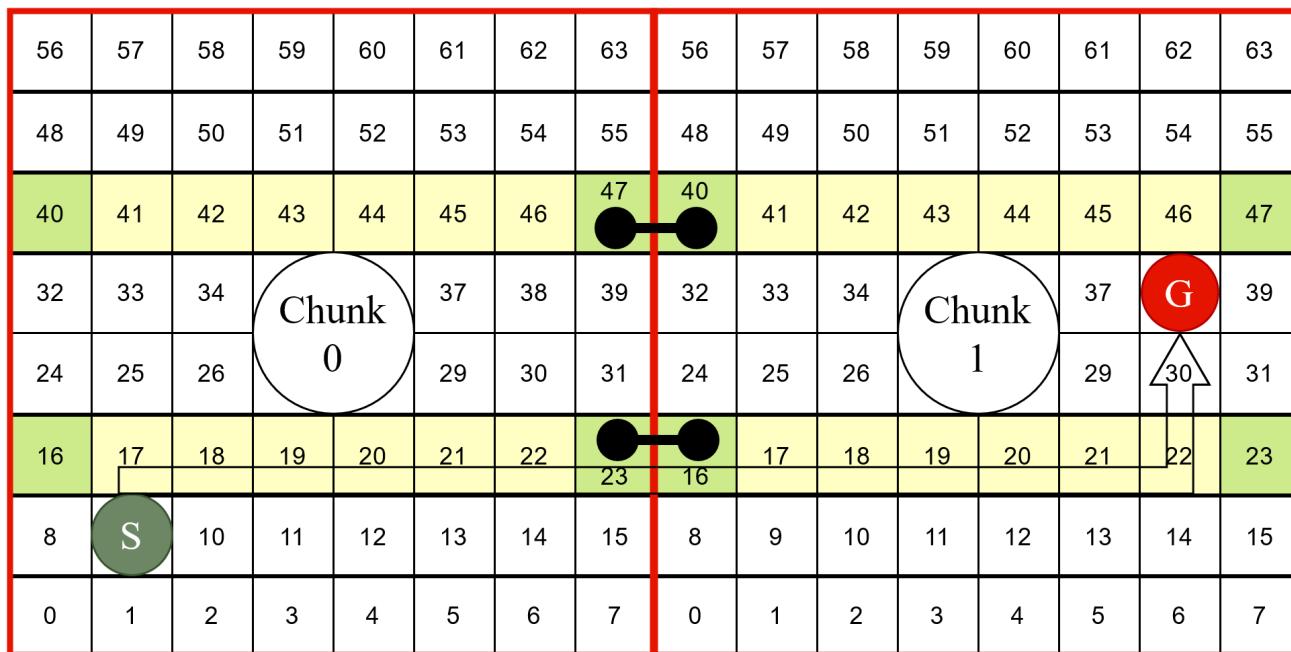
Tests d'Acceptations : Placement Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	J'appuie sur la touche "Espace" du clavier		les marqueurs des unités apparaissent	
2	je maintiens le clique droit de la souris + Je bouge la souris	Régiment(s) sélectionné(s)	les marqueurs de placements apparaissent	
3	je maintiens le clique droit de la souris + Je bouge la souris	Régiment(s) sélectionné(s)	les marqueurs suivent la souris	
4	je maintiens le clique droit de la souris + Je bouge la souris loin du centre de rotation	Régiment(s) sélectionné(s)	le nombre d'unités par ligne augmente (jusqu'au maximum prévu)	
5	je maintiens le clique droit de la souris + Je bouge la souris proche du centre de rotation	Régiment(s) sélectionné(s)	le nombre d'unités par ligne diminue (jusqu'au minimum prévu)	
6	je maintiens le clique droit de la souris + Je bouge la souris + je relâche le clique droit	Régiment(s) sélectionné(s)	les unités avancent jusqu'aux marqueurs	

2.4 Pathfinding HPA

2.4.1 Introduction

Qu'est-ce que le Hierarchical Pathfinding A* ou HPA ?

Il s'agit d'une abstraction du système de pathfinding A* et qui a pour but de réduire le temps de calcul du processeur sur de grande surface. Ce travail est fait sur la base du travail de recherche effectué par Adi Botea, Martin Müller et Jonathan Schaeffer [1]
Exemple prenons un terrain de 16x8.



Le principe est de découper la grille en chunks (ci-dessus : 8x8 en rouge), et de précalculer les chemins entre deux portes définies (ci-dessus : cellules vertes) ; ainsi lorsque une unité doit aller du point "S" au point "G", ce dernier n'aura pas à calculer les chemin dans les chunks.

Le calcul dynamique se fera sur les chunk eux-mêmes, réduisant considérablement le temps d'itérations du calcul de l'A*.

Avantages :

- ☒ Chemins précalculés à la compilation
- ☒ Introduction d'obstacles dynamiques facilité et extrêmement performant.
- ☒ Recalcul de chemin très performant.
- ☒ Calcul de chemin impossible performant.

Désavantages :

- ☒ Mise en place d'une structure de données complexe.
- ☒ Coût en Mémoire RAM non négligeable selon la structure mise en place.
- ☒ Identification et Gestion des cas limites.
- ☒ Qualité du chemin réduit (le chemin choisi ne sera pas forcément le plus court).

Note : les implémentations du HPA sont nombreuses, chaque implémentation devant s'adapter au besoin du projet. Dans le cadre de ce projet et au besoin en performance de ce dernier, au vu du temps disponible, nous nous concentrerons sur l'approche la plus directe possible.

2.4.2 Décomposition des tâches

La complexité du HPA vient principalement de la structure de données complexe qu'elle demande en effet, l'algorithme demande de stocker un grand nombre de données pour être efficaces et si l'implémentation de cette structure n'est pas solide des problèmes apparaîtront dans le pire des cas vers la fin de l'implémentation (quand il est trop tard).

Afin de ne pas nous perdre dans la conception, tout comme la gestion du régiment, nous allons implémenter pas à pas l'algorithme ajoutant les complexité une à une.
Les découpages seront les suivants :

Pour un Cluster :

- ☒ Composition d'un cluster
- ☒ Placements des portes (servira à connecter les clusters entre eux).
- ☒ Calcul des chemins dans le cluster.
- ☒ Calcul des Distances entre les portes.

Connexion entre deux Cluster :

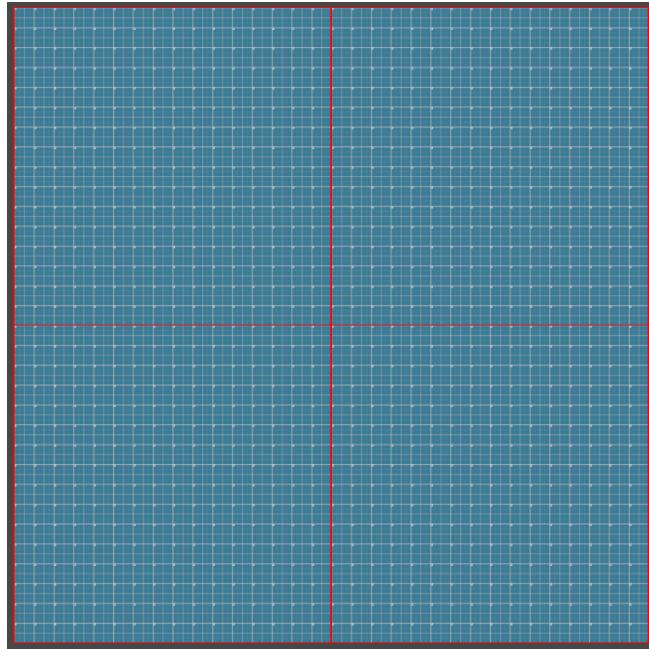
- ☒ Partition adjacente à une porte
- ☒ Calcul de chemin entre deux partitions différentes
- ☒ Trouver la porte la plus proche du pathfinder.
- ☒ Trouver la porte la plus proche de la destination.
- ☒ Calculer les connexions entre les autres portes via un A* classique.

Après Analyse, il ne fait pas sens d'aller plus loin en effet, chaque algorithme de pathfinding ne peut se concevoir si la manière de se déplacer des entités n'est pas fixée, d'autant plus que la structure particulière liée au régiment pose des problématiques très différentes des structures canoniques propre au HPA ou au A* en général.

Le mouvement des régiments est malheureusement une difficultés qui a été négligé et n'est pas en l'état représentatif du résultat souhaité surtout dans sa mécanique posant un frein au pathfinding, si l'implémentation de base permet de bricoler quelque chose, cela ne sera pas représentatif du comportement voulu et ne sera pas productif dans le sens où la forme finale sera dans tous les cas très différentes, il est d'autant plus vrai que l'implémentation actuelle du HPA fait abstractions d'un grand nombre de contraintes comme les obstacles, nous aborderons plus loin la problématique dans la partie réalisation.

2.4.2.1 Cluster

Le Cluster est la représentation des données dans une partitions de la grille dont nous pouvons voir les limites en rouge sur l'image ci-dessous :

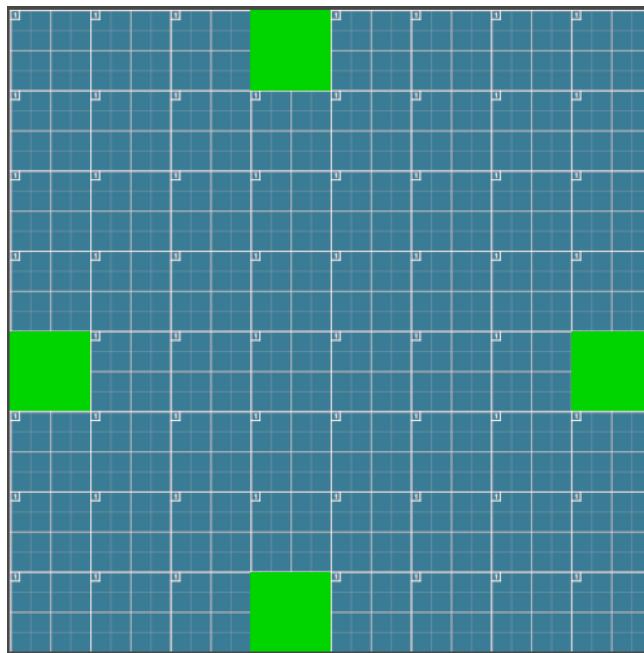


Ci-dessus : Une représentation de la grille en jeu (32x32), les lignes rouges montrent la séparation entre les partitions de taille (16x16).

Nous allons dans un premier temps établir la structure de données interne à un cluster :

Nous savons qu'un cluster doit avoir 4 portes, permettant d'établir les liens avec ses voisins, la position des portes doit normalement suivre un certains nombre de règles.

Nous n'aborderons pas ces règles ici et allons aller au plus simple et simplement les poser sur les cellules au centre des bords de la grille (arrondie à l'inférieur si il n'y a pas de chiffre rond possible).



Ci-dessus : les portes sont représentées en vert.

Il nous faut maintenant nous poser les questions suivantes :

- ☒ Y'a-t-il des situations où il pourrait y avoir plusieurs portes par bord ?

Oui, nous en verrons les détails plus loin, plusieurs articles[2] décrivent des situations où plusieurs portes sont nécessaires notamment quand des obstacles sont présents.

- ☒ Doit-on pouvoir accéder à une porte en identifiant le bord auquel elle est rattachée ?

Il est trop tôt pour le dire, cependant cela serait bénéfique dans une optique de debug, nous allons donc utiliser une collection permettant cela.

Synthèse

Suivant les besoins établis précédemment il nous faut les éléments suivants :

Un enumérateur pour définir les bords du cluster (Nord, Est, Sud, Ouest).

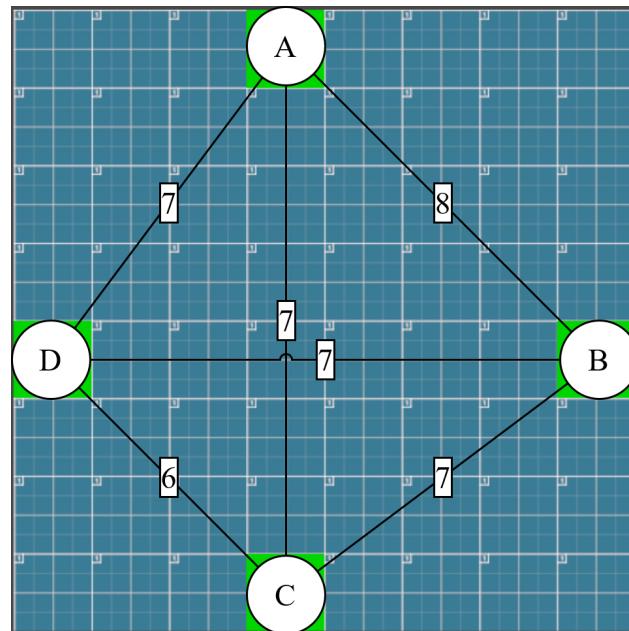
une liste pour chaque des côtés du cluster contenant les portes attachées à chacun de ses bords.

2.4.2.2 Les Portes

Les portes sont l'équivalent des noeuds dans les graphiques de l'Algorithm de Dijkstra[7], cependant contrairement aux algorithmes conventionnels du même type, il nous faudra stocker un grand nombre d'informations et surtout, il faudra pouvoir les récupérer.

En se basant sur l'illustration ci-dessous nous pouvons établir les besoins suivants :

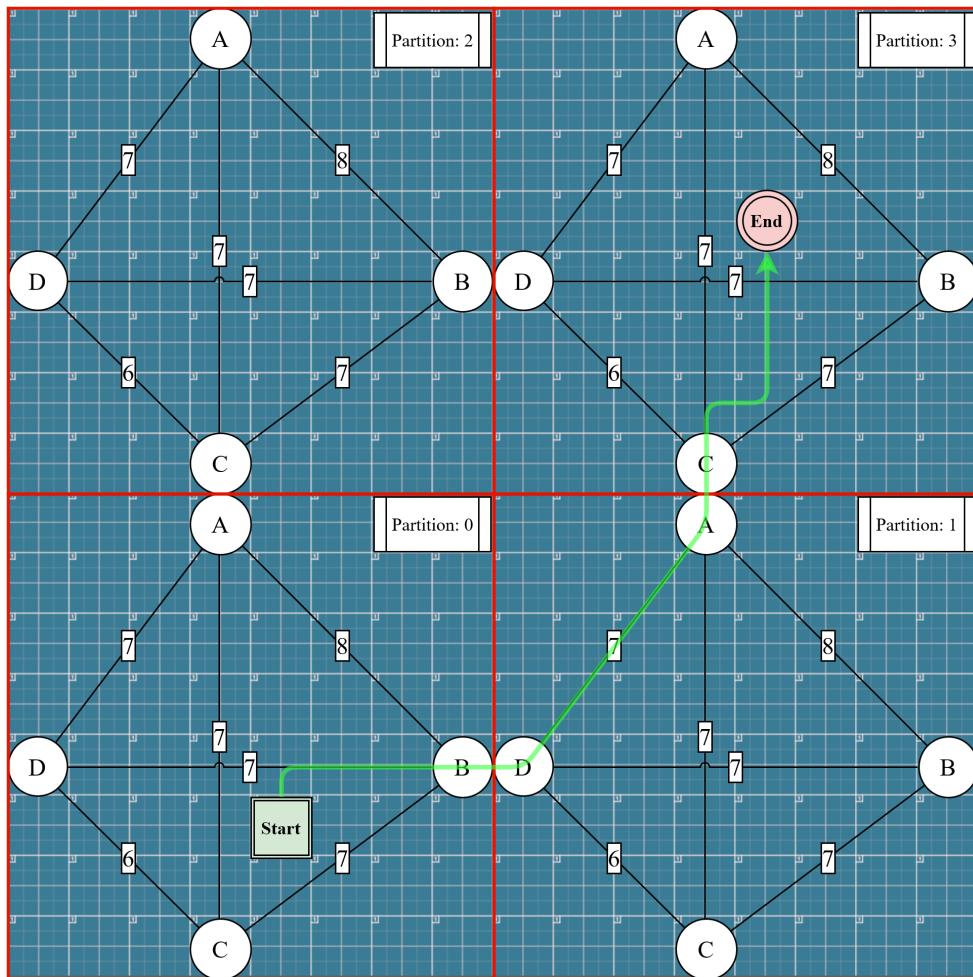
- ☒ L'indexe de la cellule
- ☒ Les chemins possibles vers les autres portes (du même cluster).
- ☒ La distance entre les différents chemins.



2.4.2.3 Connexion entre les Portes

Maintenant que nous avons nos portes et les chemins qui les relie dans la même partition, comment les lier avec une autre partition ?

Prenons le cas suivant :



la porte de départ(B) et d'arrivée(C)

Nous avons en effet poser les portes mais les entités ne commenceront pas forcément sur un porte, tout comme la destination souhaitée elles ne sont pas prédictibles.

Note : Nous faisons toujours abstractions de tous obstacles

La stratégie est la suivante :

1. Calculer le coût de chaque porte des partitions contenant le départ et la destination, représentant la distance heuristique sous la forme nombre de cellule x et y d'une porte de la partition de départ jusqu'à la cellule de destination (et inversement).
2. Les distances de chaque porte calculé, il faudra dans l'ordre croissant tester via un algorithme A* si la cellule "start" et "end" peuvent atteindre la porte la plus proche et à défaut de pouvoir l'atteindre testera les autres par ordre de distance.
3. Appliquer un A* classique sur les portes en partant de End (il faudra faire le lien entre les portes).

2.5 Stratégie de test

Le projet sera testé de deux manières :

1. **Tests Unitaires** : qui seront directement fait sur le programme Unity.
2. **Tests de validations** : qui eux seront fait avec des personnes externes au projet ; ces derniers se verront données une listes de tests à effectuer dans l'ordre donné et devront noter si oui ou non le résultat attendu a été obtenu.

Note : La liste des test est présente dans les sections notées "Tests de Validations" dans le chapitre dédié à l'analyse et conception.

Tests de Validations

Les tests de validation se feront en deux phases, afin de pouvoir refactoriser à la fois le programme en cas de bug mais aussi revoir la forme des tests si ces derniers n'ont pas été assez clairs ou n'était pas assez précis.

Phase 1

- ✉ Antoine Dubois : Ordinateur du CPNV
- ✉ Gaetan Epars : Ordinateur personnel (Ordinateur Gaming)

Phase 2

- ✉ Thirusan Rajadurai : Ordinateur du CPNV
- ✉ Azad Safai : Ordinateur personnel (Ordinateur Gaming)

2.6 Risques techniques

Algorithme de Pathfinding HPA

Bien que fonctionnant de la même manière que le A*, le node graph utilisé par le HPA est différent dans sa modélisation(voir le chapitre : 2.4)

Réduction du risque :

- ☒ Commencer par une implémentation uniquement A* afin d'avoir une fondation solide.
- ☒ Il faudra très probablement implémenter un debugger visuel permettant de marquer le terrain, permettant ainsi de traquer visuellement ce que fait l'algorithme.

Mouvement de groupe des entités

Il s'agit là d'un secret bien gardé, en effet si beaucoup se sont penché sur les mouvement de groupes d'entités, le maintien d'une formation semblent être une implémentations marginale, les articles traitant de ce sujet sont rares[5] (car peut-être trop spécifique jeux-vidéos ?).

Réduction du risque :

Fixer les objectifs à atteindre de l'algorithme afin d'en déterminer les cas limites et les exécuter un par un afin de ne faire face qu'à un bug à la fois.

Temps

Les sujets traités sont complexes et sont surtout mis dans le contexte d'un projet demandant, de les faire interagir avec des systèmes extérieurs, bien que les ayant étudier dans des formes plus simples, cela c'est toujours limités à des projets d'expérimentation/d'étude. Il est clair que le projet ne sera pas un produit fini mais un prototype qui pourra lui déboucher sur un produit fini.

Réduction du risque :

Implémenter dans un premier temps chaque système dans des projets séparés, puis les assembler un à un afin de cibler les conflits entre les systèmes.

CHAPITRE 3

RÉALISATION

3.1 Général

Dès la panification il était clair que des sacrifices allaient devoir être fait, les priorités devant être mises sur les implémentations clés du projets comme la structure des entités en régiment et le pathfinding. Un problème qui était à prévoir était aussi la limitation matériel à disposition, il a fallut très vite se soucier des performances en jeu, comme mesure préventive j'ai utilisé des assets demandant peu de ressources issue ma bibliothèque personnelle (Asset gratuit sur l'asset store de Unity que j'ai modifié afin de réduire le nombre de vertex, compressé les textures et assembler en une seul entité afin de réduire le travail de la carte graphique) que j'utilise régulièrement dans mes projets personnels afin de faciliter le prototypage.

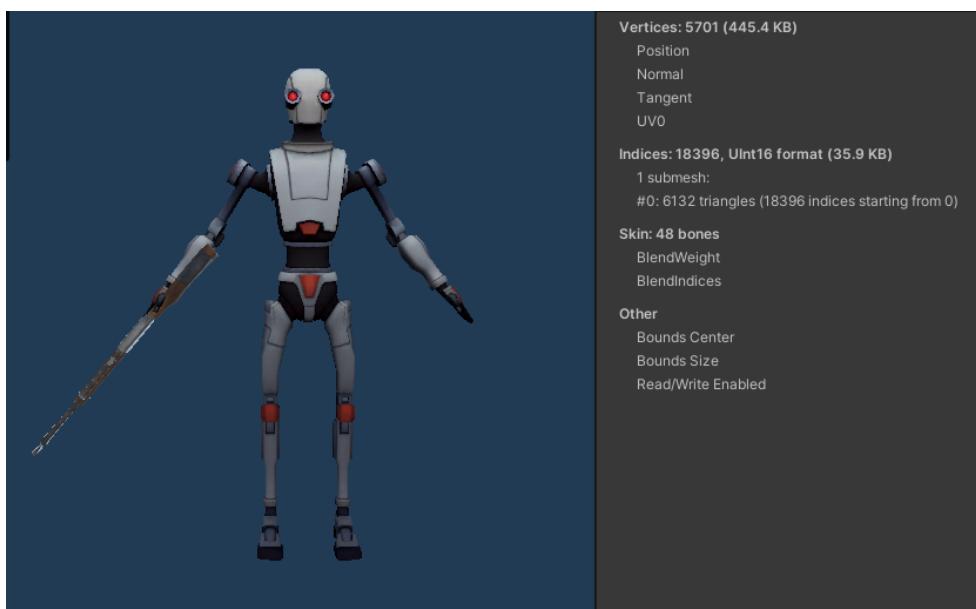
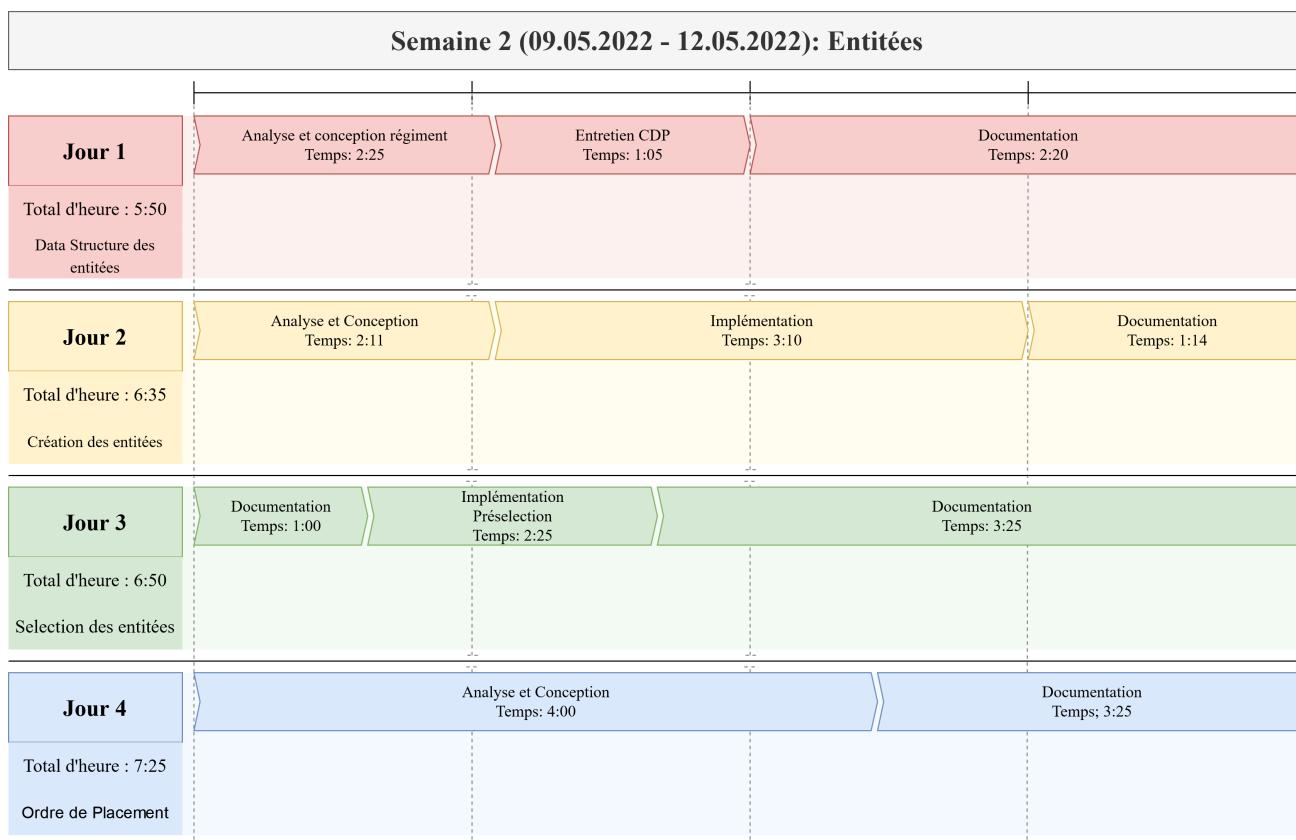
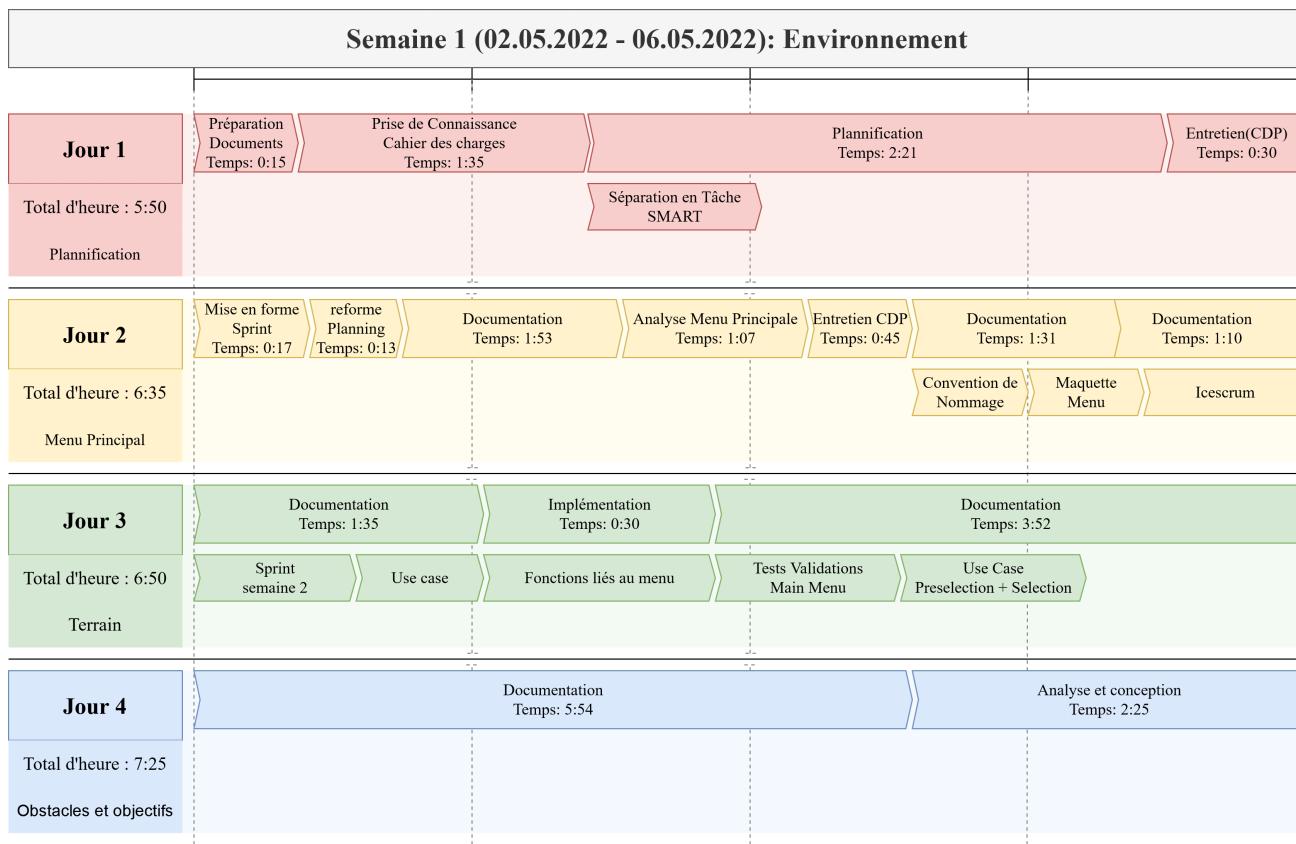
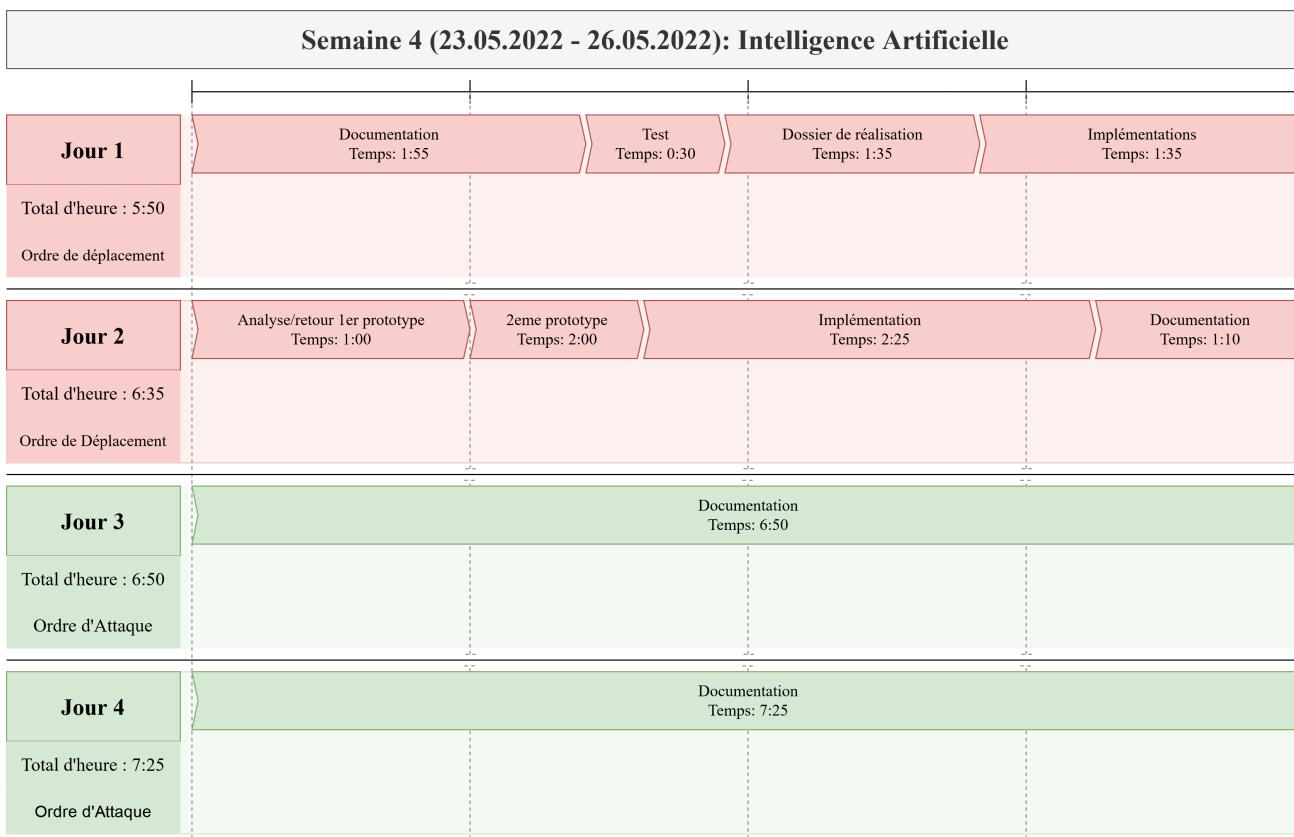
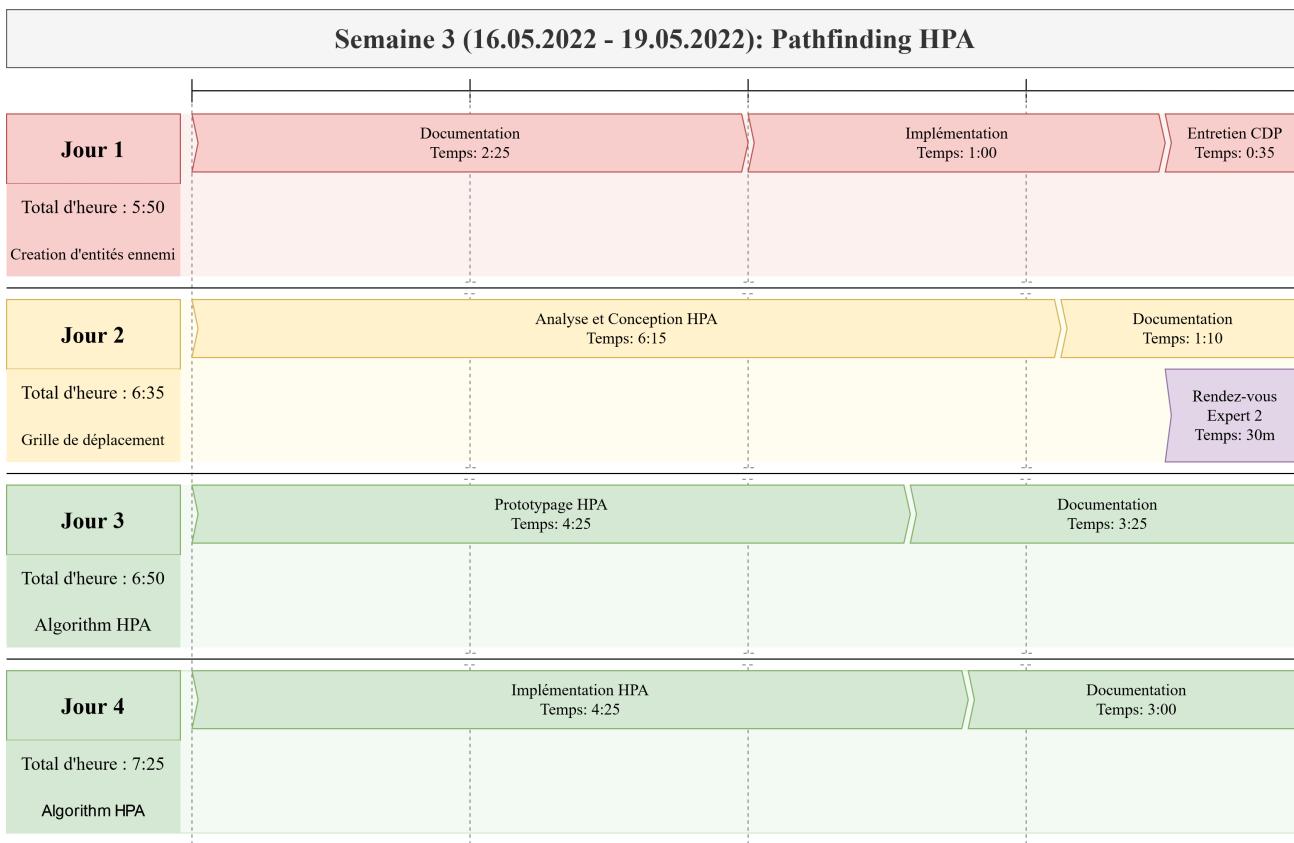


Figure 3.1 – asset de l'unity asset store[6](modifié) utilisé dans le projet

Malgé ces mesures, l'éditeurs à crasher plus d'une fois, le Real Time Strategy Game est malheureusement connu pour être une genre très gourmand en terme de CPU de part le nombre important d'entités à l'écran, pour remédier à cela il a fallu recourir au Multithreading (avec package : Job System de Unity) permettant d'utiliser toute la puissance de calcul du processeur de la machine du CPNV.

3.2 Planification finale





3.3 Éléments repris du pré-tpi

Certains éléments essentiels au bon fonctionnement du jeu ont été repris du pré-tpi :

- ☒ Caméra en vue du dessus (ou vue RTS)

- ☒ Outil de création de grilles : Utilisée dans le cadre du pathfinding

3.4 Régiment

Le Régiment a été la partie qui a été sous-estimée en terme de complexité, la structure particulière et le comportement qui ont découlé de l'implémentation, ont demandé beaucoup de préparations et d'anticipations lors de la conception.

Note : Étant étroitement lié aux autres implémentations, la complexité de la structure du régiment va se répercuter sur tous les autres éléments du projet.

les principales complexités sont les suivantes :

- ☒ Le Code peut très vite devenir incontrôlable si il n'est pas travaillé.
- ☒ Chaque interaction doit se répercuter sur chaque membre du régiment.
- ☒ Toute restructuration (placement des unités dans le régiment) du régiment doit être fait exactement de la même manière sur chaque composantes externes (exemple : marqueurs de sélections).
- ☒ Les actions des unités doivent se coordonner avec les autres membres du régiment afin de conserver la cohésion du groupe.
- ☒ Dans certains cas, des changements dans l'attribution des positions des unités au sein d'un régiment doivent s'opérer afin de garantir le mouvement le plus court possible, car un déplacement plus long empêchera le régiment de tirer sur un période plus longue (un régiment ne peut pas tirer en courant) pouvant impacter négativement l'expérience de jeu.
- ☒ La lisibilité du code n'est plus une option mais une obligation, l'ordre des unités au sein d'un régiment beaucoup trop important et demande donc de contrôler parfaitement ce qui s'y passe afin de pouvoir transmettre les bonnes informations au moment exacte où un changement survient.

Le problème majeur reste la gestion de la destruction des unités d'un régiment, la référence étant utilisée à d'autres endroits, il faut s'assurer que chaque référence soit retirée, mais plus important encore il faut que la destruction se fasse à la fin de toutes les mises à jour utilisant une des références à détruire afin que les processus asynchrones aient le temps de terminer leur calcul sans encombre (et éviter un crash du programme).

Changement par rapport au planning initial et Conception

L'implémentation à pris plus de temps que prévu notamment dû aux abstractions qu'il a fallu mettre en place afin d'assurer le maintien du code tout au long du projet.

Les quelques erreurs liées au code ne sont pas d'ordre structurel et ont vite été corrigées

- ☒ dans l'itération la dernière ligne est égal à "total de ligne - 1" et non "total de ligne".
- ☒ oubli de changer la rotation des marqueurs

Possibilité d'amélioration

Le coût de l'algorithme augmente proportionnellement au nombre d'unités contenu dans le régiment, aussi il serait intéressant d'utiliser le multithreading proposé par Units, avec la package Job System.

3.5 Pathfinding : Hierarchical Pathfinding A*

Cette partie a fait le frais de l'implémentation du régiment, en effet pour une implémentation correcte il aurait fallu une implémentation à la fois complète et fixe afin de pouvoir établir clairement les difficultés et proposer des solutions adaptées à la structure particulière des régiments.

En effet, tout pathfinding doit s'adapter à la manière de naviguer des entités, si il y a des algorithmes de base, tous ont des faiblesses et aucun n'a de réponses universelles ; il était donc impératifs de clairement définir la façon de se déplacer des entités/régiments ce qui n'a pas pu être fait, aussi l'implémentation actuelle bien que fonctionnelle ne respecte pas le principe de cohésion voulu dans un régiment, chaque unité faisant bande à part pour rejoindre le point de ralliement et ce même si cela implique de rompre le rang.

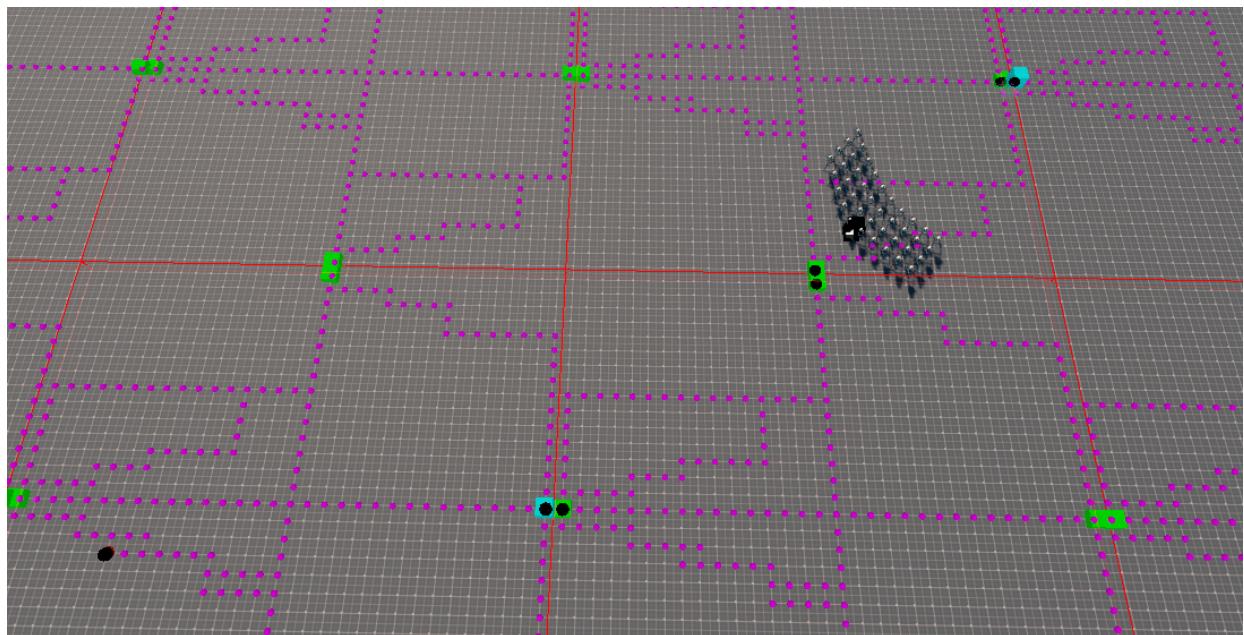


Figure 3.2 – Prototype : les points noirs représentent les portes que va emprunter le régiment

Note Importante : Les points colorés ci-dessus sont des éléments du debugger visuel de Unity et ne sont donc pas visibles en jeu.

Dans la version actuelle chaque régiment suit une route individuelle il n'y a donc pas encore de cohésion entre les différents régiments lors des déplacements.

3.6 Description des tests effectués

Dans les deux phases, un des participant effectuait le test sur un ordinateur du CPNV. et un le faisait sur un ordinateur fait pour le jeu-vidéo.

Conditions des tests

Les participants avaient uniquement la feuille avec l'énoncé des tests à effectuer, il a fallu les informer des contrôle de caméra(ne fait pas parti des tests) car certains n'avaient jamais joué à un jeu type RTS et n'était donc pas familiers avec les contrôles de base.

Problème sur les machines modestes

Les tests ont montré que des problèmes apparaissent uniquement chez les utilisateurs utilisant les machines du CPNV (non prévu pour le jeu), notamment pour l'action permettant la sélection des entités ; ce problème survient si le joueur tente une sélection "rapide" le jeu n'arrive pas à enregistrer à temps la préselection empêchant ainsi l'action de sélection. Un des utilisateur à cependant remarqué qu'en effectuant les actions lentement, le problème ne survient jamais. Il est à noter que ce problème n'a pas été rencontré sur les machines plus puissantes.

Problème dans la formulation des tests

Les problèmes rencontrés lors de la phase 1, sont notamment dû à une mauvaise description des tests à effectuer. La Phase 2 c'est mieux passé une fois reformulée.

Note sur les performances

Les performances reste acceptable en terme de fluidité d'image, bien que des ralentissement se font ressentir sur les machines plus modeste.

Solutions envisagées

Comme pour le mouvement des régiments, utiliser le Multithreading pour la présélection peut régler le problème.

Travail TPI : Castle Defense

Par Florian Duruz

2.2.2.6 Tests d'Acceptations

Testeur : Antoine Dubois
Antoine Dubois

Présélection

Tests d'Acceptations : Présélection Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	Le curseur passe au dessus d'une unité	unité non préselectionnée	Préselectionne tout son régiment	Oui
2	Le curseur n'est plus au dessus d'une unité	unité préselectionnée	Dépéslectionne l'unité	✓
3	Je maintiens le clique droit de la souris enfoncé + je déplace ma souris		Dessine un carré à l'écran	✓
4	Je maintiens le clique droit de la souris enfoncé + Je déplace ma souris	Passe au dessus d'une unité	Préselectionne tout son régiment	✓
5	Je maintiens le clique droit de la souris enfoncé + Je déplace ma souris	Passe au dessus de plusieurs unités ayant appartenant à des régiments différents	Préselectionne le régiment de chaque unité	✓

Sélection

Tests d'Acceptations : Sélection Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	Je relâche le clique gauche de la souris	Un ou plusieurs régiments sont sélectionnées + Touche Ctrl non enfoncé	Désélectionne tous les régiments précédemment sélectionnés	✓
2	Je relâche le clique gauche de la souris	Un ou plusieurs régiments sont sélectionnées + Touche Ctrl enfoncé	Sélectionne + dépréselectionne tous les régiments	✗

Placement/Mouvement

Tests d'Acceptations : Placement Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	J'appuie sur la touche "Espace" du clavier		les marqueurs des unités apparaissent	✓
2	je maintiens le clique droit de la souris + Je bouge la souris	Régiment(s) sélectionné(s)	les marqueurs de placements apparaissent	✓
3	je maintiens le clique droit de la souris + Je bouge la souris	Régiment(s) sélectionné(s)	les marqueurs suivent la souris	✓
4	je maintiens le clique droit de la souris + Je bouge la souris loin du centre de rotation	Régiment(s) sélectionné(s)	le nombre d'unités par ligne augmente	✓
5	je maintiens le clique droit de la souris + Je bouge la souris proche du centre de rotation	Régiment(s) sélectionné(s)	le nombre d'unités par ligne diminue	✓
6	je maintiens le clique droit de la souris + Je bouge la souris + je relâche le clique droit	Régiment(s) sélectionné(s)	les unités avancent jusqu'aux marqueurs	✓

Figure 3.3 – Ordinateur : CPNV

Travail TPI : Castle Defense

Par Florian Duruz

2.2.2.6 Tests d'Acceptations

2.2

Testeur : 
Gaetan Epers

Présélection

Tests d'Acceptations : Présélection Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	Le curseur passe au dessus d'une unité	unité non préselectionnée	Préselectionne tout son régiment	✓
2	Le curseur n'est plus au dessus d'une unité	unité préselectionnée	Dépéslectionne l'unité	✓
3	Je maintiens le clique droit de la souris enfoncé + je déplace ma souris		Dessine un carré à l'écran	✓
4	Je maintiens le clique droit de la souris enfoncé + Je déplace ma souris	Passe au dessus d'une unité	Préselectionne tout son régiment	✓
5	Je maintiens le clique droit de la souris enfoncé + Je déplace ma souris	Passe au dessus de plusieurs unités ayant appartenant à des régiments différents	Préselectionne le régiment de chaque unité	✓

Sélection

Tests d'Acceptations : Sélection Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	Je relâche le clique gauche de la souris	Un ou plusieurs régiments sont sélectionnées + Touche Ctrl non enfoncé	Désélectionne tous les régiments précédemment sélectionnés	✓
2	Je relâche le clique gauche de la souris	Un ou plusieurs régiments sont sélectionnées + Touche Ctrl enfoncé	Sélectionne + dépréselectionne tous les régiments	✓

Placement/Mouvement

Tests d'Acceptations : Placement Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	J'appuie sur la touche "Espace" du clavier		les marqueurs des unités apparaissent	✓
2	je maintiens le clique droit de la souris + Je bouge la souris	Régiment(s) sélectionné(s)	les marqueurs de placements apparaissent	✓
3	je maintiens le clique droit de la souris + Je bouge la souris	Régiment(s) sélectionné(s)	les marqueurs suivent la souris	✓
4	je maintiens le clique droit de la souris + Je bouge la souris loin du centre de rotation	Régiment(s) sélectionné(s)	le nombre d'unités par ligne augmente	✓
5	je maintiens le clique droit de la souris + Je bouge la souris proche du centre de rotation	Régiment(s) sélectionné(s)	le nombre d'unités par ligne diminue	✓
6	je maintiens le clique droite de la souris + Je bouge la souris + je relâche le clique droit	Régiment(s) sélectionné(s)	les unités avancent jusqu'aux marquageurs	✓

Figure 3.4 – Ordinateur : Gaming

Travail TPI : Castle Defense

Par Florian Duruz

2.2.2.6 Tests d'Acceptations

Présélection

ARAD Safi *S. Duruz*

Tests d'Acceptations : Présélection Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	Le curseur passe au dessus d'une unité	unité non préselectionnée	Préselectionne tout son régiment	✓
2	Le curseur n'est plus au dessus d'une unité	unité préselectionnée	Dépéslectionne le régiment	✓
3	Je maintiens le clique gauche de la souris enfoncé + je déplace ma souris		Dessine un rectangle à l'écran	✓
4	Je maintiens le clique gauche de la souris enfoncé + Je déplace ma souris	Passer au dessus d'une unité	Préselectionne tout son régiment	✓
5	Je maintiens le clique gauche de la souris enfoncé + Je déplace ma souris	Passer au dessus de plusieurs unités ayant appartenant à des régiments différents	Préselectionne le régiment de chaque unité	✓

Sélection

Tests d'Acceptations : Sélection Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	Je relâche le clique gauche de la souris	curseur sur unité + régiment préselectionné	Sélectionne le régiment	✓
2	Je relâche le clique gauche de la souris + Ctrl non maintenu	- Autres régiment(s) sélectionné(s) - Curseur sur unité + régiment préselectionné	- Déselectionne les autres régiments - Sélectionne le régiment préselectionné	✓
3	Je relâche le clique gauche de la souris + Ctrl maintenu	- Autres régiment(s) sélectionné(s) - Curseur sur unité + régiment préselectionné	Sélectionne le régiment préselectionné	✓
4	Après avoir fait un rectangle (préselection), je relâche le clique gauche de la souris	Un ou plusieurs régiments sont sélectionnées + Touche Ctrl non maintenu	- Déselectionne les autres régiments - Sélectionne les régiments préselectionnés	✓
5	Après avoir fait un rectangle (préselection), je relâche le clique gauche de la souris	Un ou plusieurs régiments sont sélectionnées + Touche Ctrl maintenu	Sélectionne les régiments préselectionnés	✓

Placement/Mouvement

Tests d'Acceptations : Placement Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	J'appuie sur la touche "Espace" du clavier		les marqueurs des unités apparaissent	✓
2	je maintiens le clique droit de la souris + Je bouge la souris	Régiment(s) sélectionné(s)	les marqueurs de placements apparaissent	✓
3	je maintiens le clique droit de la souris + Je bouge la souris	Régiment(s) sélectionné(s)	les marqueurs suivent la souris	✓
4	je maintiens le clique droit de la souris + Je bouge la souris loin du centre de rotation	Régiment(s) sélectionné(s)	le nombre d'unités par ligne augmente (jusqu'au maximum prévu)	✓
5	je maintiens le clique droit de la souris + Je bouge la souris proche du centre de rotation	Régiment(s) sélectionné(s)	le nombre d'unités par ligne diminue (jusqu'au minimum prévu)	✓
6	je maintiens le clique droit de la souris + Je bouge la souris + je relâche le clique droit	Régiment(s) sélectionné(s)	les unités avancent jusqu'aux marqueurs	✓

Figure 3.5 – Ordinateur : Gaming

2.2.2.6 Tests d'Acceptations

Présélection

Thirusan Rajadurai
Thirusan

Tests d'Acceptations : Présélection Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	Le curseur passe au dessus d'une unité	unité non préselectionnée	Préselectionne tout son régiment	✓
2	Le curseur n'est plus au dessus d'une unité	unité préselectionnée	Dépéslectionne le régiment	✓
3	Je maintiens le clique gauche de la souris enfoncé + je déplace ma souris		Dessine un rectangle à l'écran	✓
4	Je maintiens le clique gauche de la souris enfoncé + Je déplace ma souris	Passe au dessus d'une unité	Préselectionne tout son régiment	✓
5	Je maintiens le clique gauche de la souris enfoncé + Je déplace ma souris	Passe au dessus de plusieurs unités ayant appartenant à des régiments différents	Préselectionne le régiment de chaque unité	✓

Sélection

Tests d'Acceptations : Sélection Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	Je relâche le clique gauche de la souris	curseur sur unité + régiment préselectionné	Sélectionne le régiment	NON
2	Je relâche le clique gauche de la souris + Ctrl non maintenu	- Autres régiment(s) sélectionné(s) - Curseur sur unité + régiment préselectionné	- Déselectionne les autres régiments - Sélectionne le régiment préselectionné	✓
3	Je relâche le clique gauche de la souris + Ctrl maintenu	- Autres régiment(s) sélectionné(s) - Curseur sur unité + régiment préselectionné	Sélectionne le régiment préselectionné	✓
4	Après avoir fait un rectangle (préselection), je relâche le clique gauche de la souris	Un ou plusieurs régiments sont sélectionnées + Touche Ctrl non maintenu	- Déselectionne les autres régiments - Sélectionne les régiments préselectionnés	✓
5	Après avoir fait un rectangle (préselection), je relâche le clique gauche de la souris	Un ou plusieurs régiments sont sélectionnées + Touche Ctrl maintenu	Sélectionne les régiments préselectionnés	✓

Placement/Mouvement

Tests d'Acceptations : Placement Régiment				
Ordre	Contexte	Condition(s)	Réultat	Validation (Oui/Non)
1	J'appuie sur la touche "Espace" du clavier		les marqueurs des unités apparaissent	✓
2	je maintiens le clique droit de la souris + Je bouge la souris	Régiment(s) sélectionné(s)	les marqueurs de placements apparaissent	✓
3	je maintiens le clique droit de la souris + Je bouge la souris	Régiment(s) sélectionné(s)	les marqueurs suivent la souris	✓
4	je maintiens le clique droit de la souris + Je bouge la souris loin du centre de rotation	Régiment(s) sélectionné(s)	le nombre d'unités par ligne augmente (jusqu'au maximum prévu)	✓
5	je maintiens le clique droit de la souris + Je bouge la souris proche du centre de rotation	Régiment(s) sélectionné(s)	le nombre d'unités par ligne diminue (jusqu'au minimum prévu)	✓
6	je maintiens le clique droit de la souris + Je bouge la souris + je relâche le clique droit	Régiment(s) sélectionné(s)	les unités avancent jusqu'aux marqueurs	✓

Figure 3.6 – Ordinateur : CPNV

3.7 Erreurs restantes

Gestion des destructions des entités

La Destruction des entités est uniquement simulée pour le moment, l'algorithme n'a pas pu être mise en place par manque de temps ; détruire une entité est pour le moment dangereux, du aux nombreuses références et surtout au Multithreading, le système à mettre en place n'est pas compliqué en soit mais demande de finalisé l'algorithme de réorganisation des régiments (voir chapitre : 3.3), la structure permettant d'envoyer l'information à tout les systèmes demandant de se mettre à jour est lui implémenté et fonctionnelle, cependant il faut assurer le déterminisme de l'algorithme au risque de modifier les mauvaises références et de supprimer des références encore existantes.

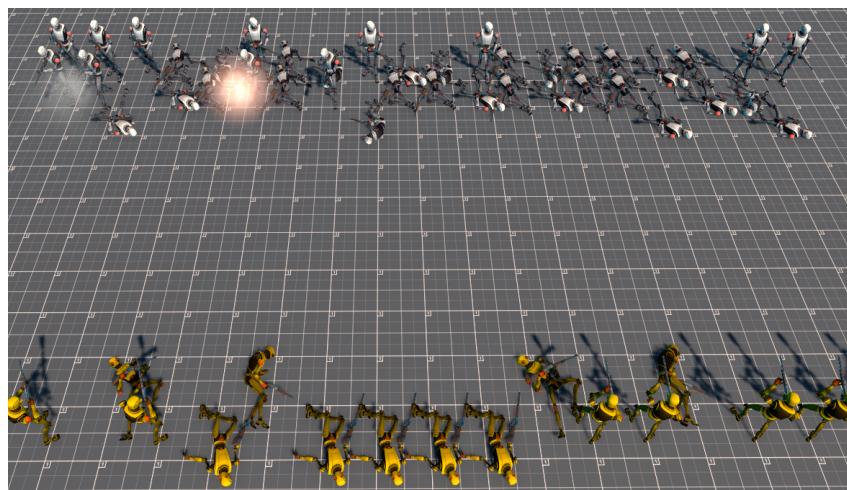


Figure 3.7 – Une fois au sol, le programme interdit à ces unités de tirer

Cela pose un problème car les entités étant encore existante, les unités ennemis continuent à tirer sur les morts (ce qui est discutable au niveau de la convention de Genève).

Cohésion des unités au sein d'un régiment

La cohésion de groupe des unités d'un régiment n'est pas complète, si les ordres sont bien donnés à tout le groupe, une fois lancé, c'est chacun pour soi ; sans parler de l'aspect visuel, cela a un impact sur l'expérience de jeu, les troupes mettant plus de temps effectuer une manœuvre, le régiment tout entier mettra plus de temps à pouvoir se mettre en position de tir, ce qui couplé au visuel chaotique peut frustrer le joueur (à raison).

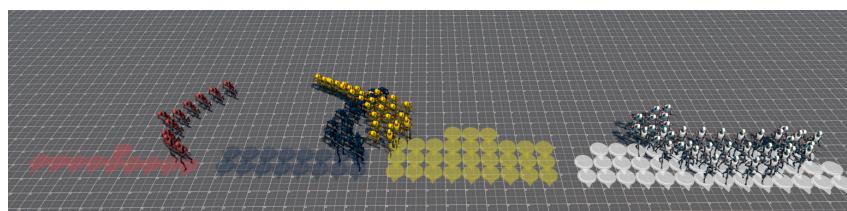


Figure 3.8 – Napoléon lui-même n'aurait jamais accepté cela..

Il n'y a malheureusement pas de solution rapide à cela, ce projet m'aura fait comprendre pourquoi peu de jeux utilisent cette structure d'entités, les opérations demandent beaucoup d'analyses et beaucoup de calculs pour des mouvements/actions qui sont d'ordinaire simples

(quand elles sont faites pour des entités individuelles) mais qui appliqués à un groupe prend une tout autre dimension.

Pathfinding

Bien que fonctionnelle cette partie fait abstraction d'un tel nombre d'éléments (obstacles, manière de se déplacer des entités, élévation, etc..) que le résultat est plus proche du prototype que d'un produit final, pour preuve, la gestion des chemins courts n'a pas été implémentés, ce qui se traduit par un énorme détour si l'unité est placé au bord d'une partition et souhaite faire un pas en avant (dans la partition adjacente) car l'implémentation actuelle force les entités à passer par les portes (Voir chapitre : Pathfinding HPA) des partitions de terrain.

CHAPITRE 4

CONCLUSION

4.1 Objectifs atteints / non-atteints

Atteints :

- ☒ Intéractions : Présélection, sélection, placement,
- ☒ Tir Automatique
- ☒ Différenciation entre les camps (joueur-ennemi)
- ☒ Pathfinding HPA

Non-Atteints :

- ☒ Système de manche (vagues d'ennemis successives)
- ☒ Condition de victoire et de défaite
- ☒ Combat en mêlée (corps à corps)
- ☒ Intéractions des régiments incomplètes sur certains points (notamment pour la cohésion de groupe)
- ☒ Pathfinding incomplet et très "résumé" et n'exploite que très peu les recherches dont s'inspire l'implémentations [1]
- ☒ La partie "FlowField" du pathfinding n'a pas pu être implémentée, cette dernière devait s'inspirer des travaux de jdxdev[3]

4.2 Points positifs / négatifs

Positifs :

- ☒ La base de l'implémentation des régiments est solide, permettant une continuité du projet
- ☒ Les étapes détaillés dans la conception on permit une implémentation rapide et efficace.
- ☒ bien qu'imcomplètes, les features présentes en jeux permettent de voir le potentiel du jeu et me donne envie de continuer ce projet.
- ☒ L'analyse qui a été faite durant ce projet et les implémentations qui en découlent m'ont permis de gagner en discipline notamment sur l'importance de documenter, la documentation bien que laborieuse, est un gain de temps considérable sur le long terme que j'ai pu expérimenter et constater tout au long de ce projet.

Négatifs :

- ☒ Perte de temps sur des aspects "secondaires" comme le menu principal (qui n'était pas dans les objectifs)
- ☒ Mauvaise gestion du temps, le planning initial était beaucoup trop ambitieux et les objectifs dans la manière de les implémenter et surtout dans leur forme n'étaient pas réalistes au vu du temps donné, il aurait fallu proposer des formes plus simples.
- ☒ Sous-estimations de la complexité de certaines implantations, c'est ce qui a sans doute été le plus gros point faible du projet, en instaurant dans la base même du projet une structure complexe, tout les éléments qui ont suivi devaient suivre la même ligne directrice d'implémentation au risque vite perdre le contrôle et ne plus savoir où vont les ressources et qui les modifient, cela à eu un impact important pour le prototypage qui est devenu beaucoup plus compliqué et qui, pour correspondre au comportement voulu devait impérativement utiliser la structure des régiments.
- ☒ Volonté d'aller plus loin, bien que cela ait ralenti le projet, je considère que travailler à être médiocre est plus dommageable sur le long terme (voir : conclusion).

4.3 Suites possibles pour le projet

Évolutions & Améliorations

Il y a encore beaucoup à faire dans ce projet et les premiers résultats sont prometteurs, il y a cependant un gros travail d'analyse et de recherche à faire, je compte bien profiter de mon temps libres et des vacances pour effectuer ces travaux, et affiner ma manière de documenter ce projet car c'est bien la documentation qui permet cette continuité.

Les principaux travaux à faire sont les suivants :

Régiments

- ☒ Destructions des entités
- ☒ Système de tirs et de visée
- ☒ Synchronisation des actions (comme le tir ou la reformation lors des mouvements)
- ☒ Retravailler le système de mouvement

Pathfinding

- ☒ Revoir la structure de données
- ☒ Revoir le type de déplacement.
- ☒ Prendre en compte les obstacles
- ☒ Obstacles dynamiques

4.4 Bilan Personnel

Le Projet bien que ambitieux m'a permis de comprendre l'importance d'une bonne documentation dans le gain de temps qu'elle apporte à long terme à un projet et surtout dans une optique de continuité de projet, chaque implémentations (qui dans ce projet utilise beaucoup de mathématiques) peut être revisités, chaque étape étant documentée pas à pas.

Dans une autre mesure, les complexités qui ont été un choix personnels ont ralenti considérablement le projet, d'aucun diront qu'il s'agit d'une faiblesse hors, comme énoncé dans les points négatifs, je considère qu'il est important de voir plus haut.

Ce projet à pour but finale de développer une intelligence artificielle. Il s'agit là d'un but personnel qui me tient à cœur, car il s'agit d'un point que j'ai toujours souhaité voir mis en

avant dans les jeux et qui malheureusement semble être le point stagnant de l'industrie, pire encore cet aspect est de plus en plus marginalisé dans l'industrie car considéré comme risqué, de nombreux articles parlent du "Sorry State" de l'Intelligence artificielle dans les jeux et cela n'est pas prêt de s'améliorer car la majeure partie de l'industrie contourne le problème en utilisant le multijoueur afin de ne pas prendre le risque d'innover. Il y a cependant une opportunité réel car l'IA n'a jamais été utilisée à son plein potentiel, les possibilités sont immenses et les concurrents peu nombreux, pour quiconque sera prêt à y mettre le temps et l'énergie nécessaire.

Si je devais résumé, l'erreur est d'avoir traité de projet comme un travail de recherche, hors le temps ne permettait pas d'effectuer un tel travail.

Je reste cependant sur une note positive car l'objectif premier qui est de pouvoir continuer le projet est sur ce point réussi, il y a une séparation claire entre les implémentations terminées et celle en cours de prototypage, permettant de facilement les disloquer pour les retravailler chacune de leur côtés.

Remerciement

Je tiens tout particulièrement à remercier Mr. Loic Viret pour m'avoir permis de faire ce projet, pour son soutien et sa confiance en mes capacités à réaliser un projet de cette envergure. Merci encore de ne pas m'avoir fait faire un énième Ecommerce.

ANNEXE A

APPENDIX

A.1 Journal de Travail

Date	Début	Fin	Heure	Tâche	Description	Solutions	Sources
02/05/2022	10:05	10:20	00:15	Organisation des documents de bases	Mise en place du dossier de projet: journal de travail Rapport Annexe au rapport Projet Unity Création d'un nouveau projet Unity		
02/05/2022	10:20	10:54	00:34	Prise de connaissance du cahier des charges	Prise de connaissance du cahier des charges - Analyses des points importants - Segmentation en feature		
02/05/2022	10:54	12:15	01:21	Planification	Planification semaine 1 et 2		
02/05/2022	13:30	14:30	01:00	Planification	Planification semaine 3 et 4		
02/05/2022	14:30	15:05	00:35	Entretien avec le Chef de projet	Modalité sur la planification et la gestion du projet	A Faire Mardi: Analyse et conception avant les sprint Etablissement des sprints à faire Etablissement des tâches (dans les sprints)	
03/05/2022	08:00	08:17	00:17	Sprint Mise en Forme	Envoyer une proposition de mise en forme au chef de projet		
03/05/2022	08:17	08:30	00:13	Remise en forme du planning	Reception du rendez-vous de l'expert 2 demandant d'ajuster la planification en conséquence		
03/05/2022	08:30	10:23	01:53	Documentation	Rémpissage de la partie Analyse préliminaire partie beaucoup plus longue et laborieuse qu'envisagée de par les termes à mettre dans le glossaire et les acronymes à mettre en bas de page	le week end sera LONG!	
03/05/2022	10:23	11:30	01:07	Analyse Menu Principale	Analyse préliminaire	étude de l'interface du jeu New World	
03/05/2022	11:30	12:15	00:45	Entretien avec le Chef de projet	Question sur l'organisation des sprint Convention de nommage Analyse et structure du Canvas	Utilisation de Icescrum Chaque élément du code doit être fixé Ne pas aller trop loin et ne mettre que des images ciblant précisément le sujet qui support l'illustration	
03/05/2022	13:30	14:29	00:59	Convention de Nommage	fait		
03/05/2022	14:30	15:02	00:32	Maquette du Menu	Utilisation de Draw IO	Terminé	
03/05/2022	15:02	16:05	01:03	Migration sur Icescrum	migration des sprint de github à icescrum		

Sprint semaine1				
04/05/2022	08:50	09:30	00:40	Migration sur Icescrum
04/05/2022	09:50	10:31	00:41	Sprint Semaine 2
04/05/2022	10:31	11:08	00:37	Use Case Régiment et Unité: définition des concernes
04/05/2022	11:08	11:46	00:38	
04/05/2022	11:46	12:05	00:19	Implémentation des la partie graphique selon maquette Implémentation du Menu
04/05/2022	13:29	13:52	00:23	Implémentation des fonctions liées au menu Menu : Fonction
04/05/2022	13:52	14:49	00:57	Documentation de la partie Tests de validation Menu principale Interaction régiment
04/05/2022	14:49	15:47	00:58	Documentation
04/05/2022	14:57	16:07	01:10	UML : Use Case Préselection Selection
04/05/2022	16:08	16:55	00:47	UML Diagramme d'activité Menu Principal
05/05/2022	08:00	08:57	00:57	Mise en forme final de la partie documentation Menu principale
05/05/2022	08:57	10:37	01:40	Documentation : Objectifs
05/05/2022	10:37	11:40	01:03	Documentation: Risque Technique
05/05/2022	11:40	12:15	00:35	Mise en forme Documentation
05/05/2022	13:32	15:01	01:29	Remise en forme du dossier de projet
05/05/2022	15:02	16:55	01:53	Analyse de la partie Pathfinding Documentation sur le sujet afin de construire une explication brève concise et simple
09/05/2022	09:50	10:30	00:40	Conception Régiment Partie Introduction Analyse des possibilité
09/05/2022	10:30	11:30	01:00	Algorithmie de rearrangement
09/05/2022	11:30	12:15	00:45	Conception Régiment Analyse des possibilité
09/05/2022	13:30	14:35	01:05	Entretien avec le Chef de projet choix de design des régiments (unités-marqueurs) mise au points concernant la forme des tests
09/05/2022	14:35	14:46	00:11	Mise au propre du document Latex
09/05/2022	14:46	15:15	00:29	Documentation sur le choix de la structure des régiment selon directive
10/05/2022	08:00	09:41	01:41	Analyse des besoins pour le système régiment Définition des besoins en terme de comportements comportement similaires entre les composantes potentiel candidat pour les interfaces ou abstrats

10/05/2022	10:00	10:30	00:30	Préparation de base de l'environnement Unity	Préparation des prefabs notamment les factory précréeer afin de faciliter le prototypage
10/05/2022	10:30	12:15	01:45	Prototypage rudimentaire sur base de l'analyse	Gros Soucis de conception, il faut reprendre depuis le début et bien cerner les responsabilités de chaque classe
10/05/2022	13:30	15:28	01:58	Suite Prototypage	redéfinition des interface selon les observation précédentes
10/05/2022	15:28	15:55	00:27	Soucis liées aux interface sur Unity	L'implémentation a pris plus de temps, car il a fallu adapter plus d'élément que prévu Cependant l'architecture semble être solide en l'état
11/05/2022	08:00	08:50	00:50	Relecture de la documentation	vérification de l'orthographe et de la grammaire réimportation d'images qui étaient de mauvaises qualités
11/05/2022	08:50	10:48	01:58	System de Preselection fonctionnelle	
11/05/2022	10:48	11:34	00:46	Lecture sur les conventions d'écritures UML	Lecture sur la convention concernant les abstract et les interfaces et leur représentation dans un schéma
11/05/2022	11:34	12:15	00:41	Diagramme de classe Régiment	
11/05/2022	13:30	13:40	00:10	BUG avec certains régiments	L'interface Repassait dans tout le registre, ce qui réinitialisait les valeur des autres régiments, précédemment enregistré
11/05/2022	13:40	15:12	01:32	Documentation: Preselection Diagram de flux	
11/05/2022	15:12	16:55	01:43	Documentation Selection	
12/05/2022	08:00	09:35	01:35	Documentation Selection + UML	
12/05/2022	09:50	12:15	02:25	Analyse de la partie Placement	
12/05/2022	13:30	16:55	03:25	Explication de chaque phase du placement + algorithme	Expliquer un algorithme est plus long qu'imaginé, des sacrifice vont devoir être fait
16/05/2022	09:50	10:30	00:40	Fin de la doc concernant le placement de 1 régiment	
16/05/2022	10:30	12:15	01:45	Algorithme avec plusieurs régiment(Placement)	
16/05/2022	13:30	14:30	01:00	Implémentation Placement multiple	
16/05/2022	14:30	15:05	00:35	Entretien avec chef de projet	
17/05/2022	08:00	09:30	01:30	Analyse HPA	
17/05/2022	09:30	10:27	00:57	Introduction et présentation du HPA	
17/05/2022	10:27	12:15	01:48	documentation	
17/05/2022	13:30	14:50	01:20	HPA: Conception	
17/05/2022	14:50	15:30	00:40	HPA: Conception	

17/05/2022	15:30	16:05	00:35	Entretien avec le second expert	
18/05/2022	08:00	10:30	02:30	Prototypage rudimentaire pour vérifier la compatibilité de certain container	Il y a un risque lié à la limitation des container Nativ utilisée par Unity pour le jobSystem
18/05/2022	10:30	12:15	01:45	Prototypage: trouver les portes de chaque chunk	
18/05/2022	13:30	16:06	02:36	Documentation HPA	début Grille Structure de données des portes
18/05/2022	16:06	16:55	00:49	Documentation Datastructure/Architecture du programme	
19/05/2022	08:00	09:26	01:26	Changement de stratégie	L'algorithme est très différent dans son fonctionnement que les algorithmes vus précédemment Il faudra faire l'algorithme pas à pas ajoutant les complexités une à une afin de: 1) être plus efficace dans son implémentation 2) proposer une documentation plus structurée et améliorer la compréhension
19/05/2022	09:26	12:15	02:49	HPA: Conception D'un chunk	But: Monter la logique des emplacements des portes et les chemins qui les relient entre elles sans se préoccuper des partitions adjacentes pour le moment
19/05/2022	13:30	15:42	02:12	Analyse et documentation des besoins pour l'algorithme	Quelques soucis dans les containers utilisés
19/05/2022	15:42	16:55	01:13	implémentations	
23/05/2022	09:50	11:45	01:55	Documentation	Révision de la structure et orthographe, mise en page et partie manquante/remise à plus tard
23/05/2022	11:45	12:15	00:30	Test par des tier	Mise en place de l'environnement + Build Testeurs: Antoine Dubois Gaëtan Epers
23/05/2022	13:30	13:50	00:20	Dossier de réalisation	Partie régiment
23/05/2022	13:50	14:15	00:25	Entretien avec chef de projet	Il faut lier le pathfinding avec le système de mouvement actuel! Et faire un système de tir!
23/05/2022	14:15	15:05	00:50	Dossier de réalisation	
24/05/2022	08:00	08:55	00:55	implémentation système de tir	Difficulté liée à la structure particulière des régiments
24/05/2022	08:55	09:35	00:40	Implémentation du système de HPA au projet	
24/05/2022	09:35	12:15	02:40	Documentation: Réalisation	
24/05/2022	13:30	16:05	02:35	Documentation	
25/05/2022	08:50	09:30	00:40	Documentation: Stratégie de test + Réalisation	
25/05/2022	09:50	10:00	00:10	Scéance Tests	Testeurs: Azad Saffai Thirusan Rajadurai
25/05/2022	10:00	12:15	02:15	Documentation Fin réalisation + tests effectué	
25/05/2022	13:30	16:55	03:25	Documentation: Conclusion + résumé	

BIBLIOGRAPHIE

- [1] Jonathan Schaeffer Adi Botea, Martin Müller. Near optimal hierarchical path-finding, 2004. Last accessed 17.05.2022, <https://webdocs.cs.ualberta.ca/~mmueller/ps/hpastar.pdf>.
- [2] Narendra S Chaudhari Amandeep Singh Sidhu. Hierarchical pathfinding and ai-based learning approach in strategy game design, 2008. Last accessed 19.05.2022, <https://www.hindawi.com/journals/ijcgt/2008/873913/>.
- [3] jdxdev. Rts pathfinding 1 – flowfields, 2020. Last accessed 25.05.2022, <https://www.jdx-dev.com/blog/2020/05/03/flowfields/>.
- [4] Microsoft. C# coding conventions, 2021. Last accessed 13.04.2022, <https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions>.
- [5] Dave Pottinger. Implementing coordinated movement, 2021. Last accessed 05.05.2022, https://www.gamasutra.com/view/feature/3314/coordinated_unit_movement.php?print=1.
- [6] Unity. Unity asset store, 2019. Last accessed 19.05.2022, <https://assetstore.unity.com/packages/3d/characters/robots/space-robot-kyle-4696>.
- [7] Wikipedia. Algorithme de dijkstra, 2022. Last accessed 19.05.2022, https://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra.