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

TABLE DES MATIÈRES

TABLE DES FIGURES

GLOSSAIRE

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

Framework : Collection de librairies ayant des fonctionnalités associées via une API unifiée. XXVI

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. . V

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 jour surtout représenté dans le monde du modding et un Mod très populaire dans le RTS[1] Warcraft 3 ; ce projet est réalisé dans le cadre d'un travail d'un travail personnel individuelle(TPI) pour l'établissement du CPNV.

L'objectif de ce travail 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 familiarisé 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 qui sera au coeur du projet et qui régira les mouvement 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 jouer sélectionne une unité tout 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 bien spécifiques.

  L'objectif sera de développer un algorithme en conjonction avec l'algorithme Pathfinding [2] (décrit 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 impacte directe 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(décrite au point suivant)
  Combat en Mêlée : Une formation plus compacte offrira une meilleure capacité de défense quand le régiment est engagée au corps à corps.

Figure 1.1 – Total War : Tir en formation

Algorithme Pathfinding HPA [3]

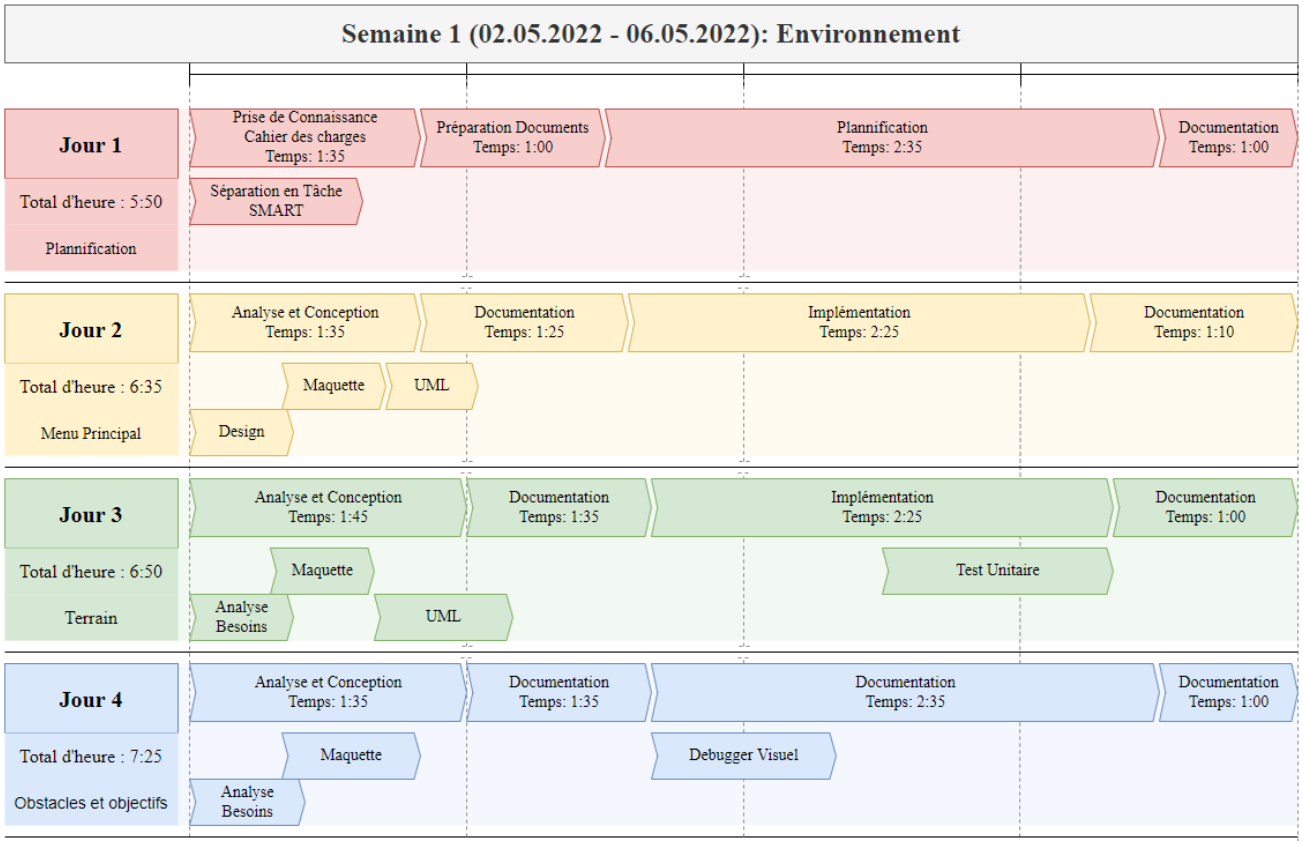
Sur la base des travaux effectués lors du pré-tpi, il faudra mettre en place un système de hierarchical Pathfinding A* en apportant un changement afin d'adapter l'algorithme au besoin de projet, à savoir l'utilisation d'un Flowfield (ou champs de vecteurs).

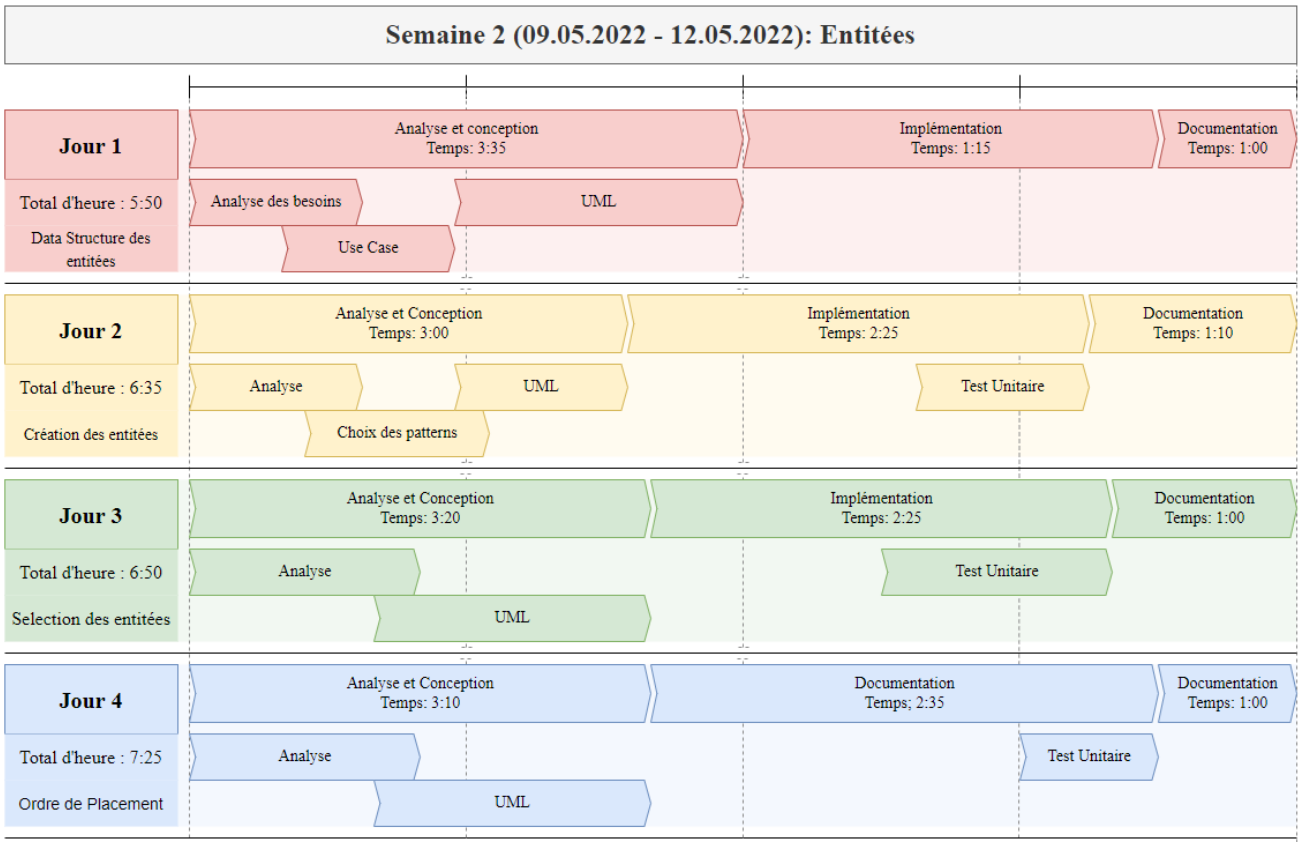
Pour ce faire, les éléments suivant devront être implémenté :

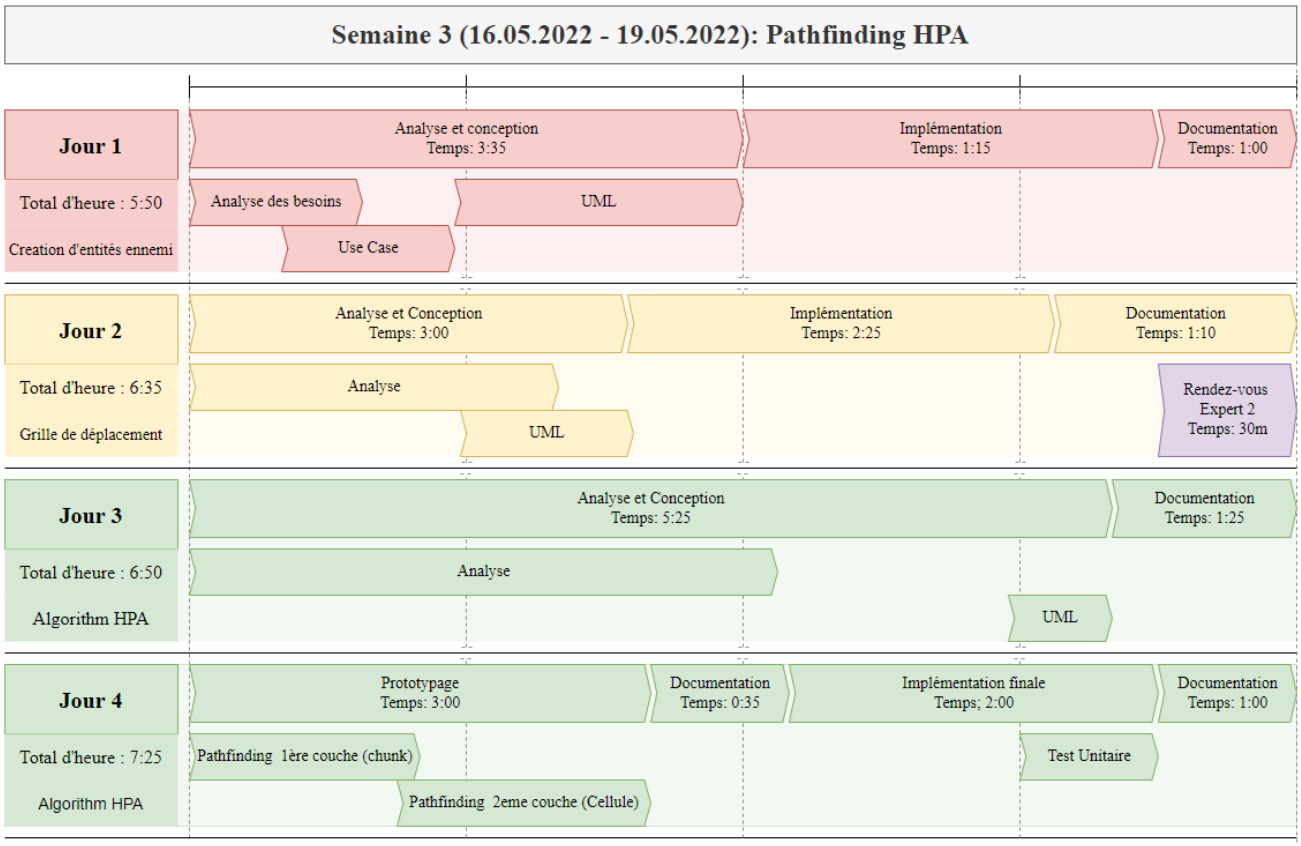
- **Génération d'une grille partitionnée** : Il faudra en premier lieu générer un grille sur tout le terrain en utilisant l'outil mis en place précédemment. Cette dernière devra être partitionnée afin de pouvoir cibler les futures calculs à une région spécifique.
- **Structure de donnée** : Chaque partition 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'algoritme HPA au Flowfield** : La version traditionnelle du HPA utilise uniquement le A*(il s'agit en réalité d'une abstraction en ou plusieurs couche de cet algorithme), il est malgré tout possible d'y inclure des algorithmes différents dont le flowfield qui sera utilisé.

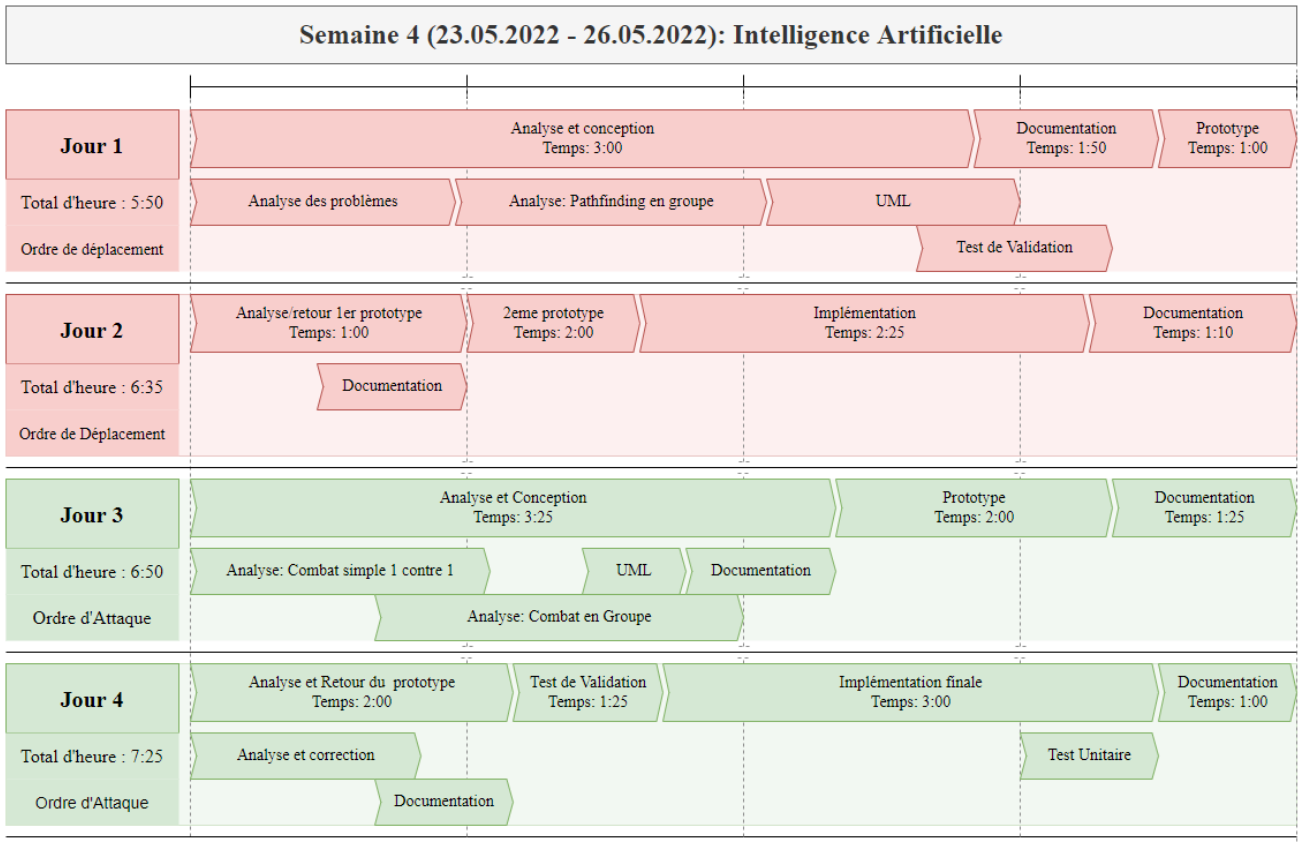
2. Pathfinding : Recherche de Chemin
3. Hierachical Pathfinding A*

1.3 Planification initiale






Semaine 3 (16.05.2022 - 19.05.2022): Pathfinding HPA
Jour 1
Total d'heure : 5:50
Creation d'entités ennemi
Analyse et conception
Temps: 3:35
Implémentation
Temps: 1:15
Documentation
Temps: 1:00
Analyse des besoins
UML
Use Case
Jour 2
Total d'heure : 6:35
Grille de déplacement
Analyse et Conception
Temps: 3:00
Implémentation
Temps: 2:25
Documentation
Temps: 1:10
Analyse
UML
Rendez-vous
Expert 2
Temps: 30m
Jour 3
Total d'heure : 6:50
Algorithm HPA
Analyse et Conception
Temps: 5:25
Documentation
Temps: 1:25
Analyse
UML
Jour 4
Total d'heure : 7:25
Algorithm HPA
Prototypage
Temps: 3:00
Documentation
Temps: 0:35
Implémentation finale
Temps; 2:00
Documentation
Temps: 1:00
Pathfinding 1ère couche (chunk)
Pathfinding 2eme couche (Cellule)
Test Unitaire


Semaine 4 (23.05.2022 - 26.05.2022): Intelligence Artificielle
Jour 1
Total d'heure : 5:50
Ordre de déplacement
Analyse et conception
Temps: 3:00
Documentation
Temps: 1:50
Prototype
Temps: 1:00
Analyse des problèmes
Analyse: Pathfinding en groupe
UML
Test de Validation
Jour 2
Total d'heure : 6:35
Ordre de Déplacement
Analyse/retour 1er prototype
Temps: 1:00
2eme prototype
Temps: 2:00
Implémentation
Temps: 2:25
Documentation
Temps: 1:10
Documentation
Jour 3
Total d'heure : 6:50
Ordre d'Attaque
Analyse et Conception
Temps: 3:25
Prototype
Temps: 2:00
Documentation
Temps: 1:25
Analyse: Combat simple 1 contre 1
UML
Documentation
Analyse: Combat en Groupe
Jour 4
Total d'heure : 7:25
Ordre d'Attaque
Analyse et Retour du prototype
Temps: 2:00
Test de Validation
Temps: 1:25
Implémentation finale
Temps: 3:00
Documentation
Temps: 1:00
Analyse et correction
Documentation
Test Unitaire

CHAPITRE 2

ANALYSE / CONCEPTION

2.1 Convention de nommage

Général

S'aligne en général avec les conventions de C#[2] 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 facilité la lecture, le typage est explicite afin que le lecteur n'ai pas à traquer les conversions de type. De plus on fait gagner du temps au lecteur en affichant directement au lecteur, ce dernier n'ayant pas besoin de regarder à droite du "=" pour trouver 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 et globale ou locale, ce préfixe n'ajoute donc rien.

- **Méthode** : CamelCase - 1ère Lettre Majuscule - Sans préfixe.
- **Interface** : CamelCase - 1ère Lettre Majuscule - préfixe "I".

- **Champs privée** : pascalCase - 1ère Lettre Minuscule - pas de préfixe.
- **Champs publique** : CamelCase - 1ère Lettre Majuscule - pas de préfixe.
- **Propriété** : CamelCase - 1ère Lettre Majuscule - pas de préfixe.
- **Variable privée** : pascalCase - 1ère Lettre Minuscule - pas de préfixe.
- **Constante** : SNAKE_CASE - Tout en Majuscule - pas de préfixe.

Particularité Unity

- **SerializedField** : Les champs ou proriétés 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.

2.2 Concept

Le concept complet avec toutes ses annexes : Par exemple :

- Multimédia : carte de site, maquettes papier, story board préliminaire, …
- Bases de données : interfaces graphiques, modèle conceptuel.
- Programmation : interfaces graphiques, maquettes, analyse fonctionnelle…
- …

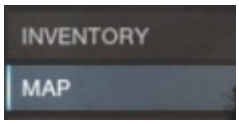
2.2.1 Menu Principal

Le Menu principale peut sembler être un sujet triviale, 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 ?

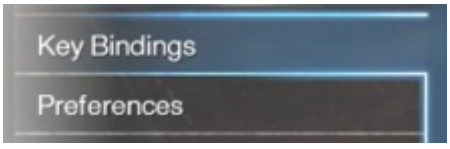
Malgré ne nombreuse recherche il ne semble pas y avoir un convention ou des règles établies, j'ai donc décider de m'inspirer d'un jeux ayant un thème similaire : New World crée par Amazone et dont l'interface graphique avait été très appréciée.
Sans entrée dans le design, prenons quelques points qui de mon point de vu 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 ou va apparaître la réponse

Unity a depuis un moment effectué une transition majeure dans l'élaboration des interfaces utilisateur ; 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.

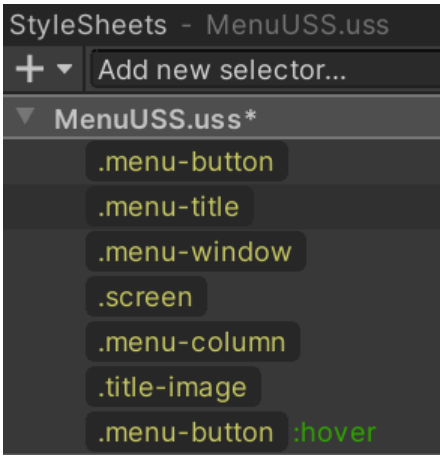
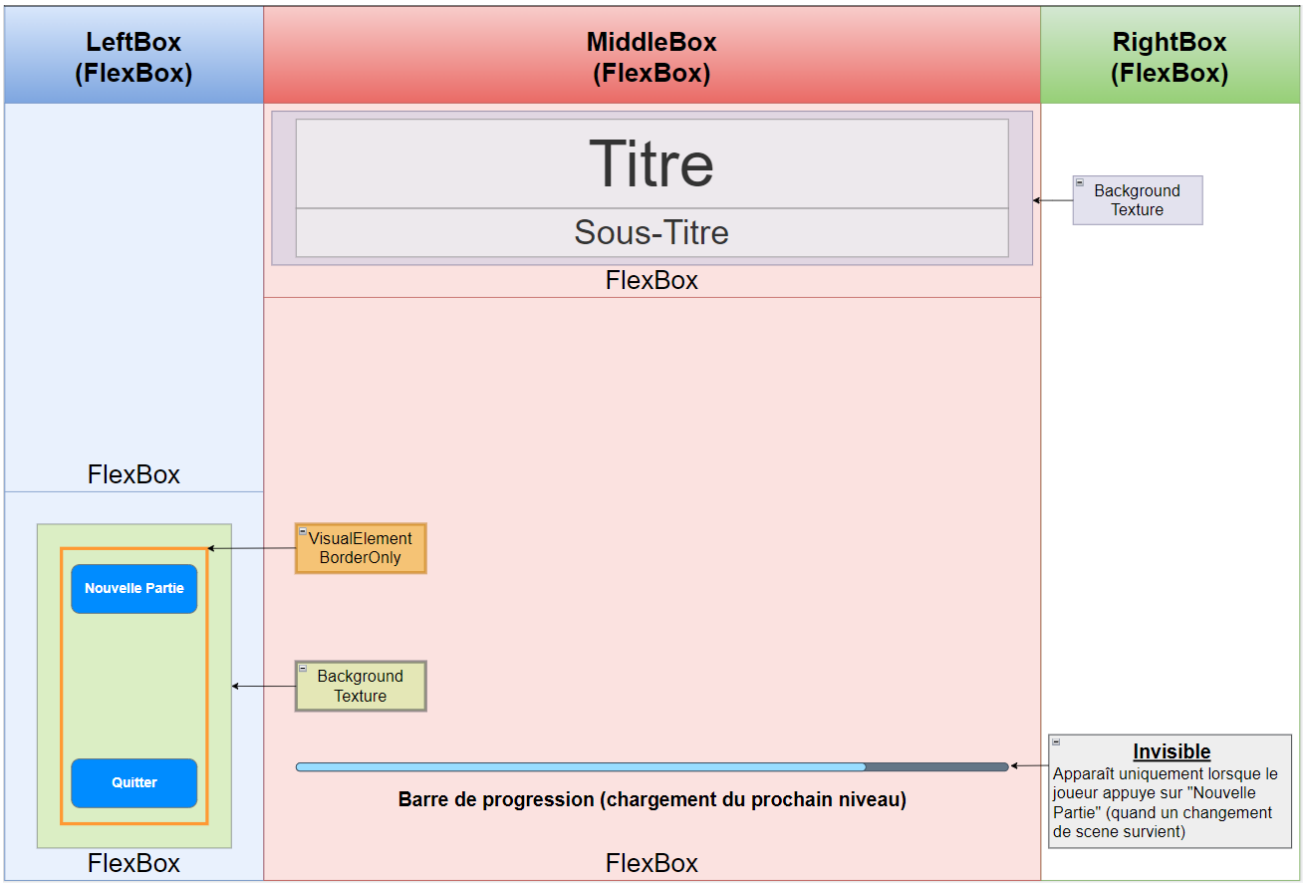


Figure 2.1 – Feuilles de Style

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 directe aux paramètre modifiable 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 reprise et utiliser aussi 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 toute modification css via l'interface graphique crée une nouvelle propriété dans le fichier UXML.

Maquette du menu Principale



2.2.2 Régiments

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

Interactions

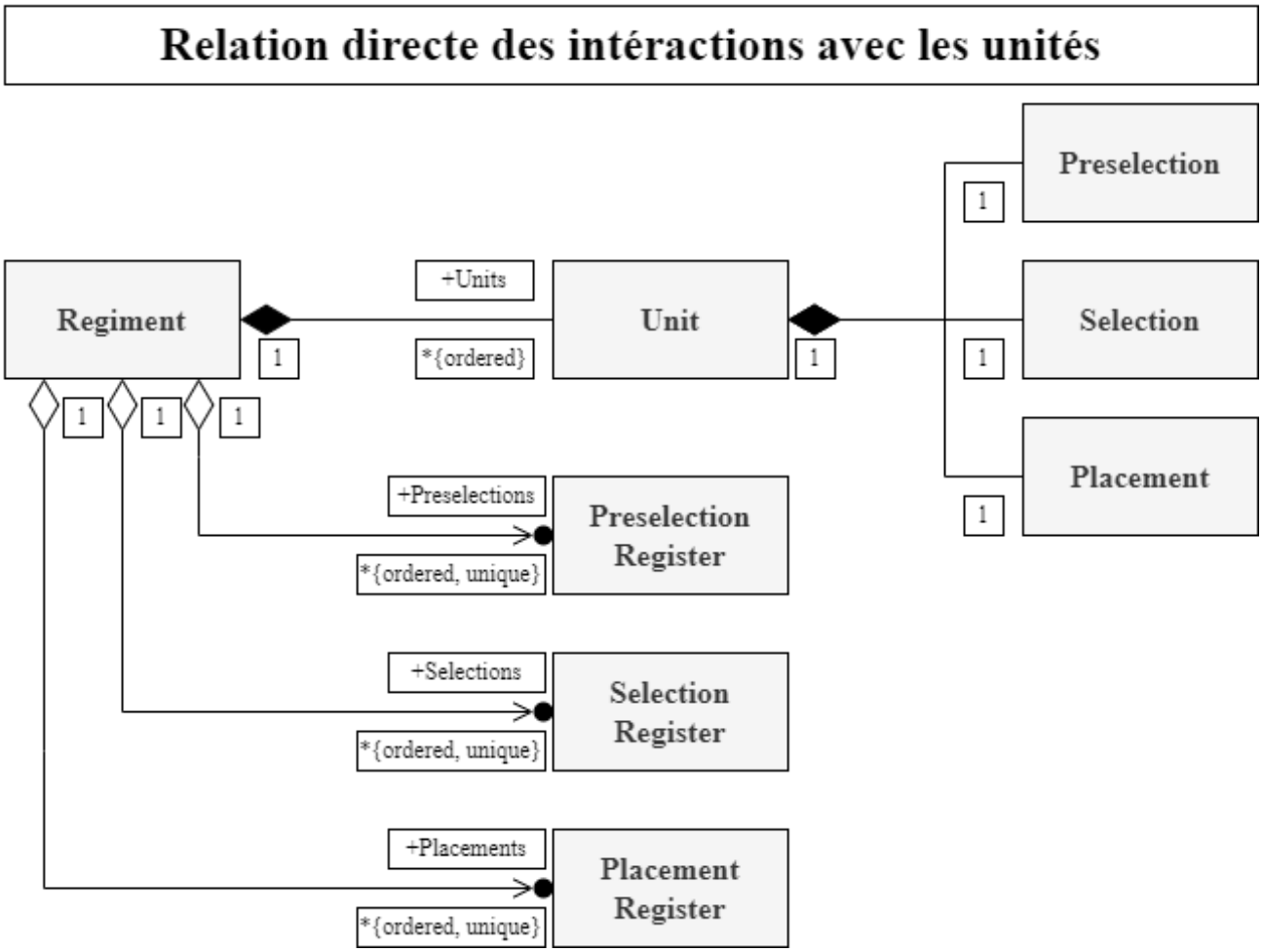
Reprenant le cahier des charges, il nous faudra dans le cadre des interactions, concevoir :

- 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 proposé chacun ayant des implications profondes dans la structure du programme :



Cette forme prévoit un lien directe entre les unités et les différents indicateur dans une relation parent-enfant ou l'unité serait le parent.

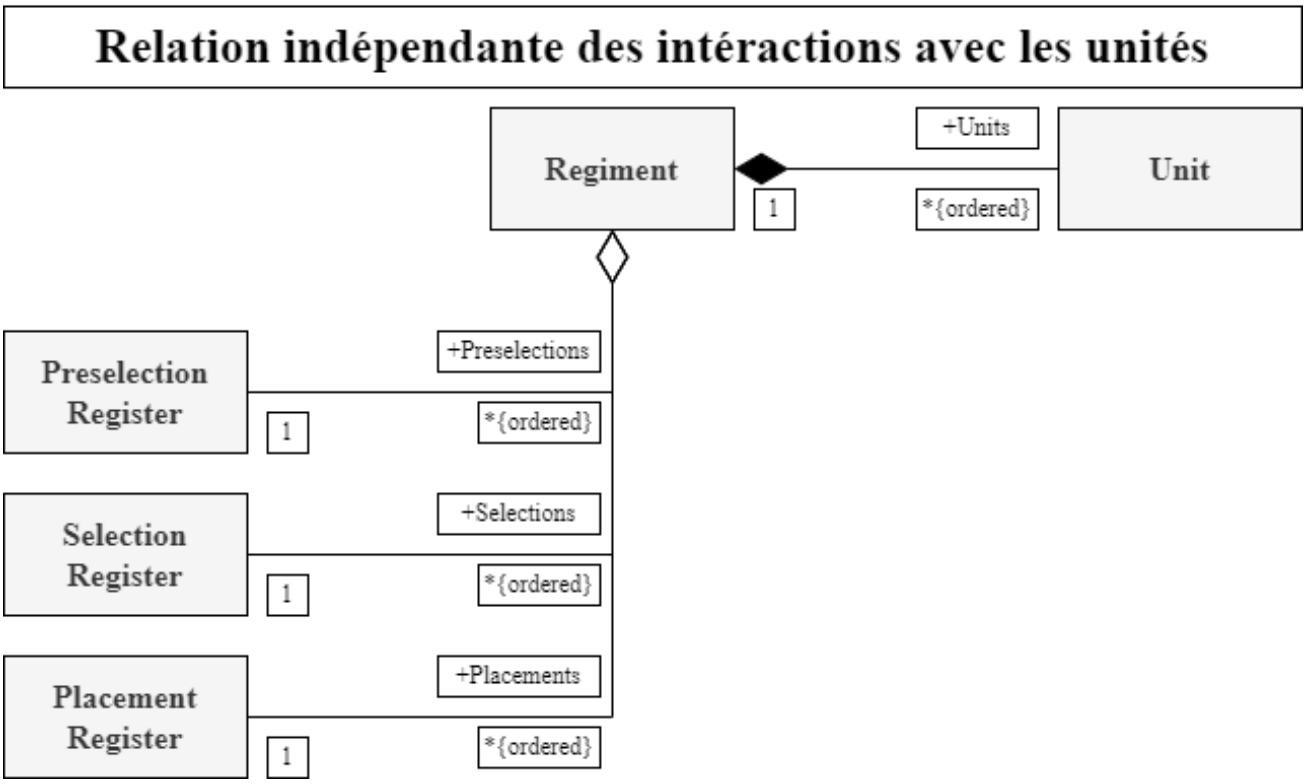
Avantages

- Gestion des déplacements gérés par Unity.
- Gestion de la destruction des marqueurs simplifiés
- Besoin réduit de passer par un système complexe d'interfaces/abstract

Inconvénients

- Architecture ne respectant le le SOC [1] 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 séléction par exemple).
- Accès au ressources d'un système à un autre potentiellement problématique.

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



L'autre solution est de rendre totalement indépendant chaque marqueur des unités du régiment dans le sens ou 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 simplifiés
- Séparation propre et claire des responsabilités

Inconvénients

- Gestion des mouvements du marqueurs manuels
- Implémentation d'un système d'interface/abstract complexe pour obtenir un code lisible.
- Gestion de cas particuliers liés à la gestion manuel du mouvement des marqueur.

Un autre aspect a prendre en compte est le réarrangement dynamique d'un régiment quand une de ses unités est tué. Les choix de la collection est de ce primordiale, il y a un comportement précis suivant un algorithme demandant un contrôle précis dans l'agencement des unités dans la collection choisie.

Il est aussi important que le réarrangement soit effectif et ne doit en aucun cas être simulé via un container temporaire ou en déplaçant uniquement les unités dans la configuration voulu, car cela va de causer des problèmes quand le joueur repositionnera le régiment ; les unités n'ayant pas changé effectivement leur index dans le régiment,tenteront de reprendre leur position lors du prochaine ordre de repositionnement tel qu'il est indiqué 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, il se pourrait que d'autres fonctionnalitées comme le combats puissent être impacté, les unités mettant plus de temps à se reformer sur de courtes distances.

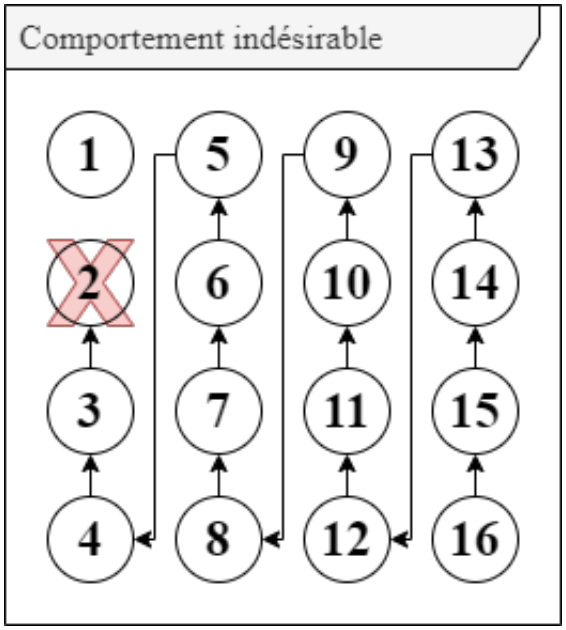


Figure 2.2 – Réarrangement automatique d'une collection type List

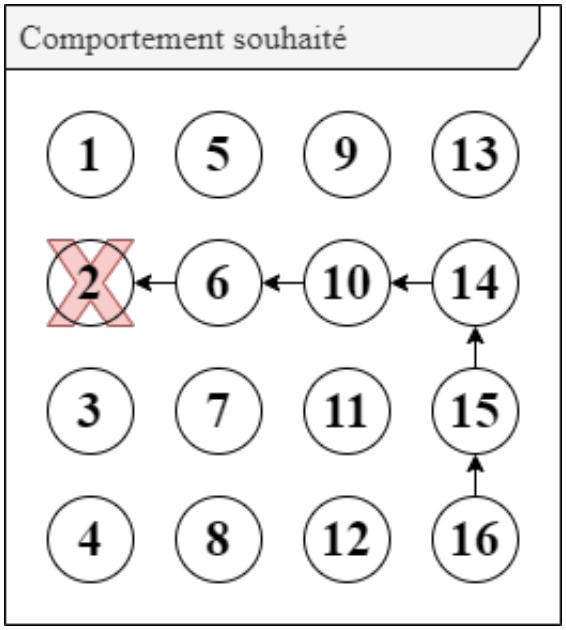


Figure 2.3 – Réarrangement suivant l'algorithme

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 leur propriétés comme la largeur et la longueur, les paramètres seront fixes et égal à l'instant où l'ordre de réarrangement est lancé.

2.3 Pathfinding HPA

Tout d'abord comment fonctionne le pathfinding A* ?

Variables importantes

H : Distance entre la Destination et la position actuelle
G : Distance entre le Début et la position actuelle
F : Coût Total (H + G)

2.4 Stratégie de test

Décrire la stratégie globale de test :

- types de des tests et ordre dans lequel ils seront effectués.
- les moyens à mettre en œuvre.
- couverture des tests (tests exhaustifs ou non, si non, pourquoi ?).
- données de test à prévoir (données réelles ?).
- les testeurs extérieurs éventuels.

Tests d'Acceptations : Menu Principal

Ordre	Contexte	Condition	Resultat
1	J'ouvre le Jeu		Le Menu et 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

Tests d'Acceptations : Présélection Régiment

Ordre	Contexte	Condition(s)	Resultat
1	Le curseur passe au dessus d'une unité	unité non présélectionnée	Préselectionne tout son régiment
2	Le curseur n'est plus au dessus d'une unité	unité préselectionnée	Dépéselectionne 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

Tests d'Acceptations : Sélection Régiment

Ordre	Contexte	Condition(s)	Resultat
1	Je relâche le clique gauche de la souris	Un ou plusieurs régiments sont séléctionnées + Touche Ctrl non enfoncé	Désélectionne tous les régiments précédemment sélectionnés
2	Je relâche le clique gauche de la souris	Un ou plusieurs régiments sont séléctionnées + Touche Ctrl enfoncé	Sélectionne + dépréselectionne tous les régiments

2.5 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 : 2.3)

Réduction du risque :

- Commencer par une implémentation uniquement A* afin d'avoir une fondation solide.
- Il faudra très probablement implémenter un débugger 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 implémentions marginale, les articles traitant de ce sujet sont rares[3] (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 traîtés 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.

2.6 Dossier de conception

Fournir tous les document de conception :

- Le choix des outils logiciels pour la réalisation et l'utilisation
- Réaliser les maquettes avec un logiciel.
- Organigramme.
- Architecture du programme.
- Pseudo-code / structogramme.

CHAPITRE 3

RÉALISATION

3.1 Dossier de réalisation

Cette partie comprendra le déroulement du projet, la façon dont les implémentations ont été réalisé, leur fonctionnement, les difficultés rencontrés et la résolution de ces difficultés. Devront aussi apparaître :

1. Les compromis fait.
2. Les changements par rapport aux plans initiaux et pourquoi.
3. Présenter les alternative à une implémentation.
4. Motiver le choix d'une alternative par rapport à une autre.

3.2 Description des tests effectués

Pour chaque partie testée de votre projet, il faut décrire :

1. les conditions exactes de chaque test.
2. les preuves de test (papier ou fichier).
3. tests sans preuve : fournir au moins une description .

3.3 Erreurs restantes

S'il reste encore des erreurs :

1. Description détaillée.
2. Conséquences sur l'utilisation du produit.
3. Actions envisagées ou possibles.

CHAPITRE 4

CONCLUSION

4.1 Objectifs atteints / non-atteints

Atteints :

- Objectif1.
- Objectif2.
- Objectif3.
- Objectif4.

Non-Atteints :

- Objectif5.
- Objectif6.
- Objectif7.
- Objectif8.

4.2 Points positifs / négatifs

Positifs :

- Positif1.
- Positif2.
- Positif3.
- Positif4.

Négatifs :

- Negatif5.
- Negatif6.
- Negatif7.
- Negatif8.

4.3 Difficultés particulières

4.4 Suites possibles pour le projet

Évolutions & Améliorations

4.5 Bilan Personnel

I Will reference someone[1]

ANNEXE A

APPENDIX

A.1 Journal de Travail

Framework test frame API test api

BIBLIOGRAPHIE

[1] Donald E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

[2] Microsoft. C# coding conventions, 2021. Last accessed 13.04.2022, https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions.

[3] Dave Pottinger. Implementing coordinated movement, 2021. Last accessed 05.05.2022, https://www.gamasutra.com/view/feature/3314/coordinated_unit_movement.php ?print=1.