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. XIX

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

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 mettre 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

Ce chapitre énumère les objectifs du projet. L'atteinte ou non de ceux-ci devra pouvoir être contrôlée à la fin du projet. Les objectifs pourront éventuellement être revus après l'analyse.

Ces éléments peuvent être repris des spécifications de départ.

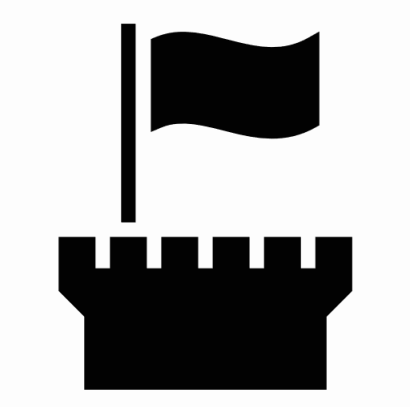
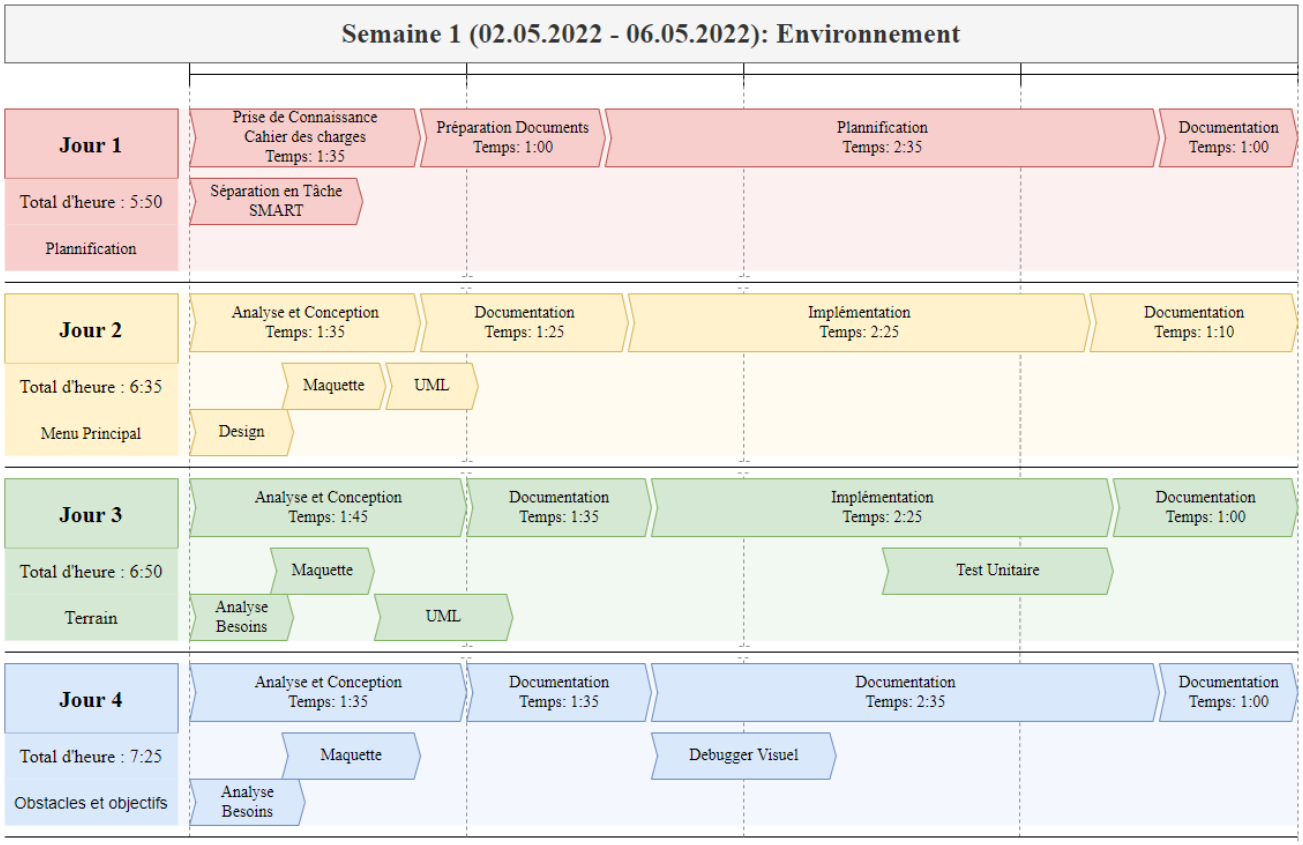
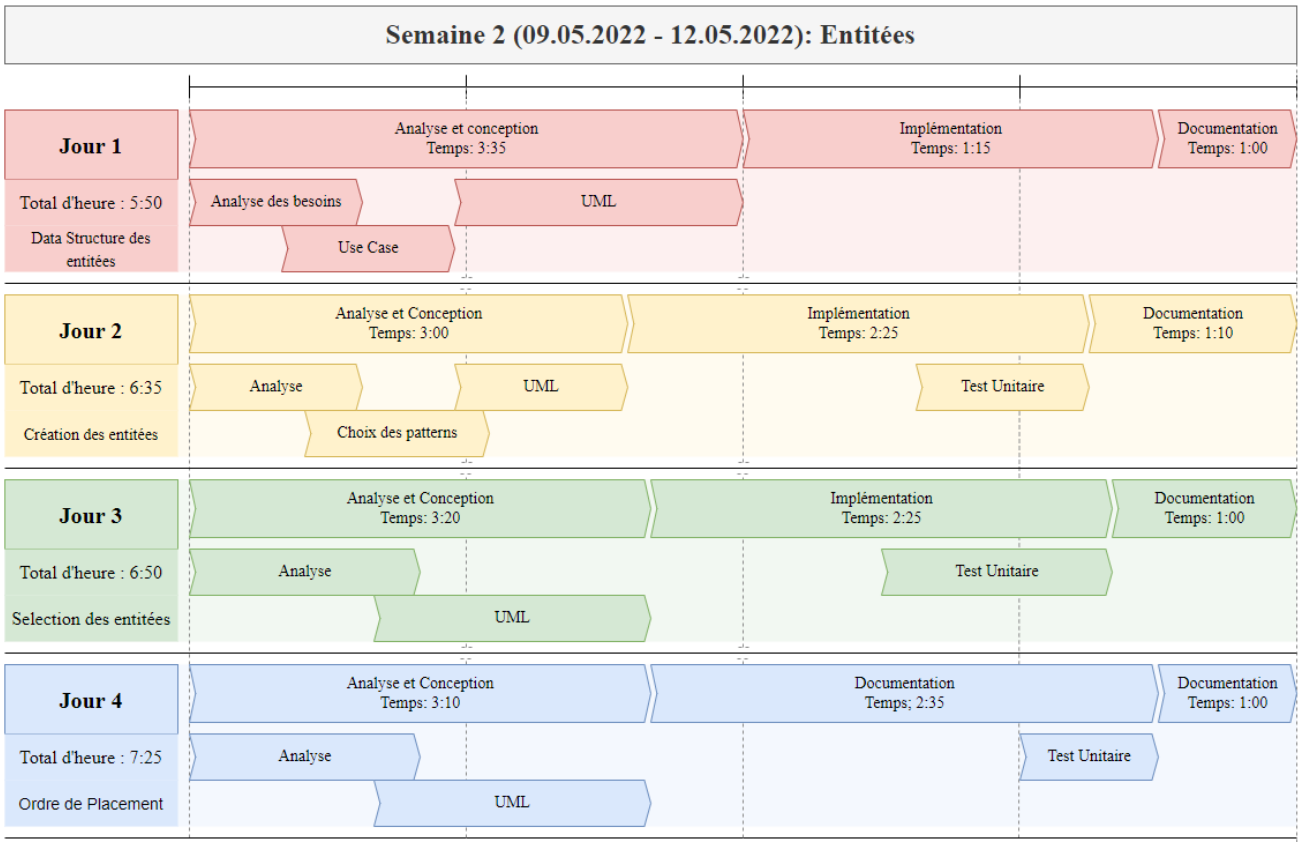
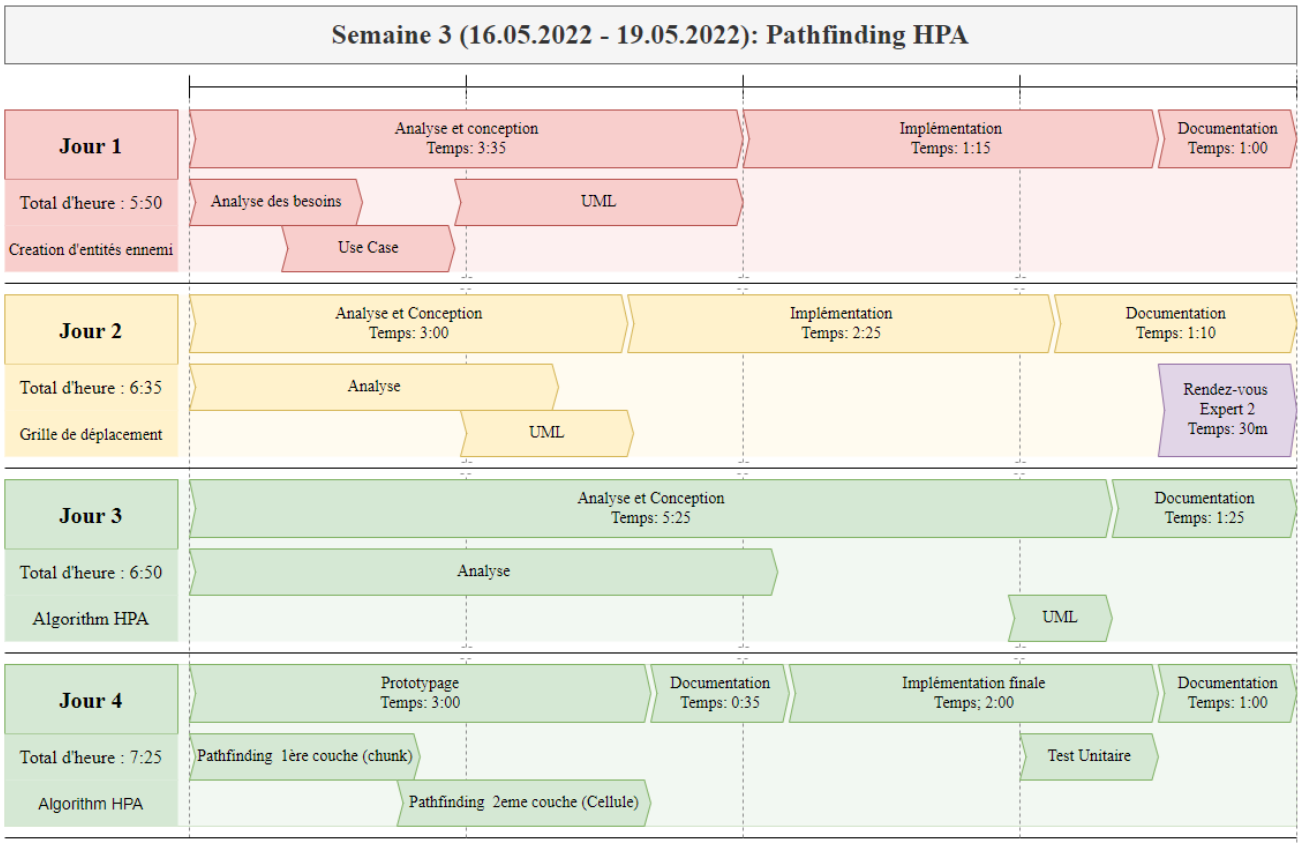


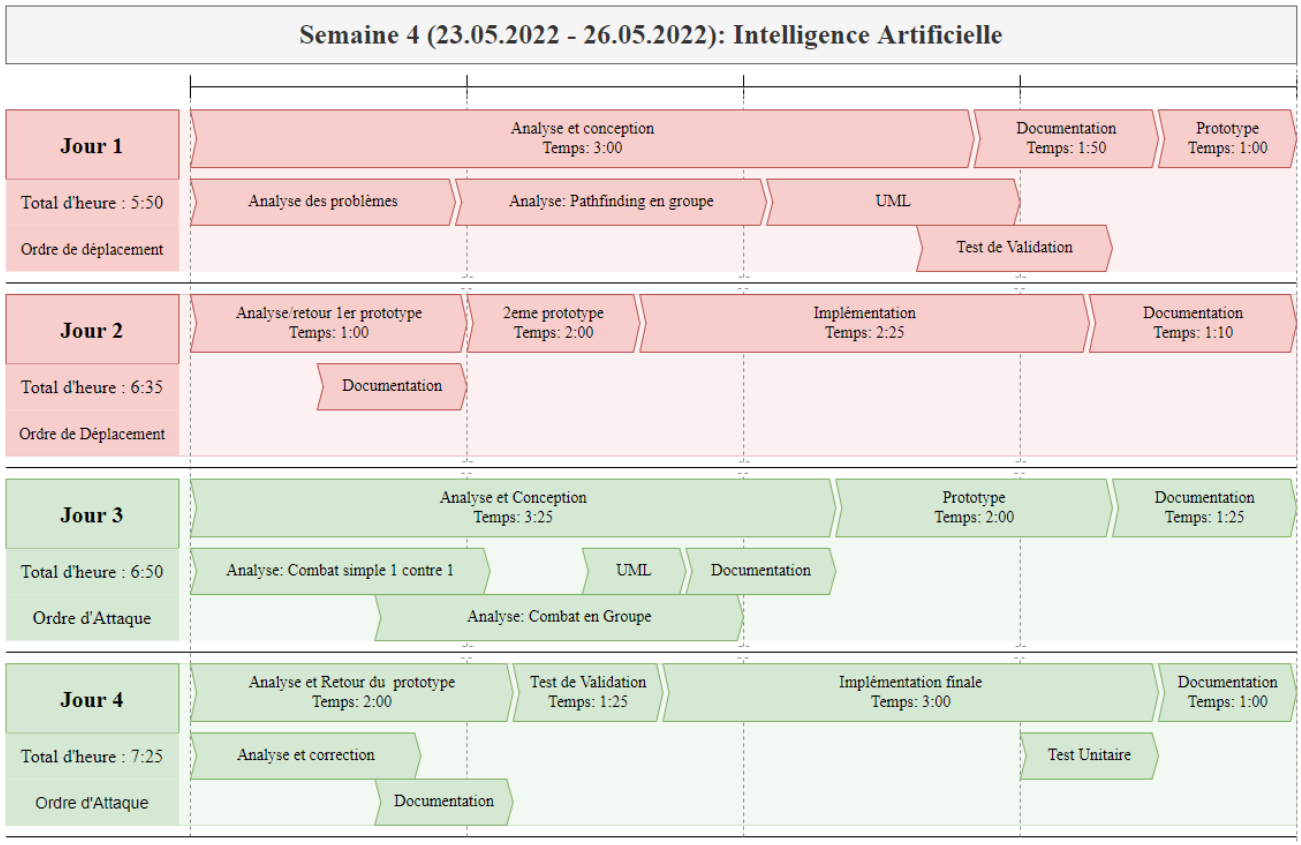
Figure 1.1 – Tower

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

Indentation : A la ligne(ou block) selon les conventions C#[2].

Indentation Inline : Une méthode n'ayant que une ligne doit 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

Langue du code : Anglais.

Langue des commentaire : Français.

Langue des commits github : Français.

Langue 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 qui viendrait selon certaines sources du C et qui à 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 : PascalCase - Sans préfixe.

Interface : PascalCase - préfixe "I".

Variable privée : PascalCase - 1ère Lettre Minuscule - pas de préfixe.

Variable publique : PascalCase - 1ère Lettre Majuscule - pas de préfixe.

Propriété : PascalCase - 1ère Lettre Majuscule - pas de préfixe.

Constante : SnakeCase - Tout en Majuscule - pas de préfixe.

Méthode : PascalCase - Sans préfixe.

Interface : PascalCase - préfixe ”I”.

Variable privée : PascalCase - 1ère Lettre Minuscule - pas de préfixe.

Variable publique : PascalCase - 1ère Lettre Majuscule - pas de préfixe.

Propriété : PascalCase - 1ère Lettre Majuscule - pas de préfixe.

Constante : SnakeCase - Tout en Majuscule - pas de préfixe.

Particularité Unity

Indentation : les événements liés à Monobehaviours(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…
- …

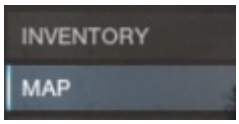
2.2.1 Menu Principal

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 ?

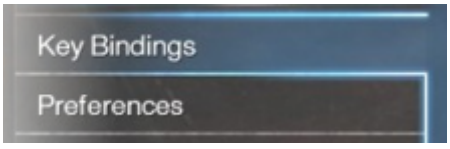
Malgré ne nombreuse recherche il ne semble pas y avoir un convention ou des règles établies, j'ai donc décider de m'inspirer d'un jeux ayant un thème similaire : New World crée par Amazone et dont l'interface graphique avait été très appréciée.
Sans entrée dans le design, prenons quelques points qui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



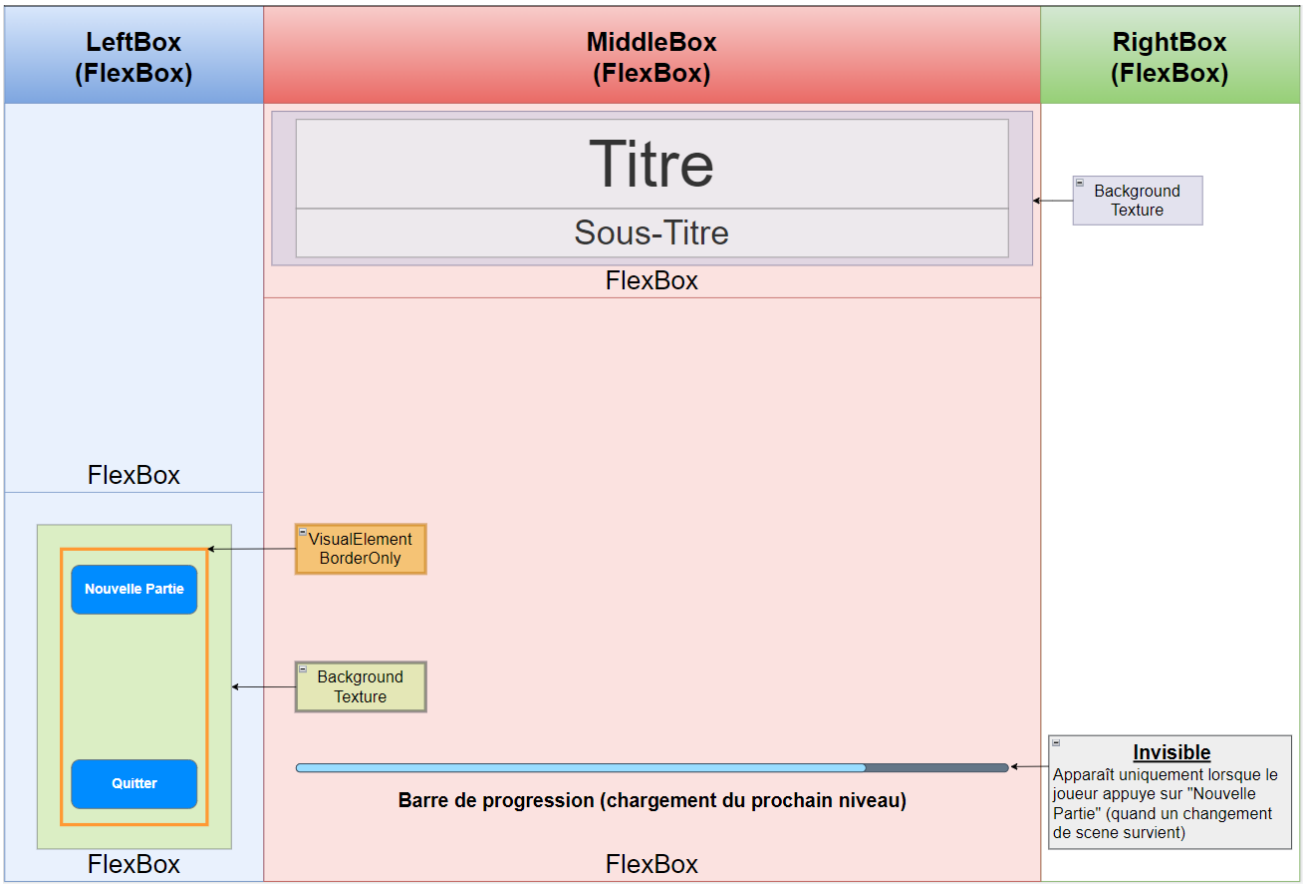
  Aucune couleurs vives même les éléments actifs restent dans des tons ternes.
- **Transition douce** :

  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

Maquette du menu Principale



2.3 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.

2.4 Risques techniques

Énumérez les risques techniques (complexité, manque de compétences,...)
Décrire aussi quelles solutions ont été appliquées pour réduire les risques (priorités, formation, actions, …).

2.5 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.