Projet : CY-FIGHTERS

1. Présentation de l'équipe :

Ce projet a été réalisé par : Kais et Thao.
Il n'y avait pas de répartition stricte des rôles : nous travaillions simultanément sur différentes parties du code, en partageant nos avancées via Discord, à l'écrit ou en appel avec partage d'écran.
Cette méthode nous permettait de réagir rapidement aux besoins ou aux problèmes rencontrés, en avançant étape par étape selon les priorités du moment.

2. Sujet du projet

L'objectif était de développer un jeu de combat au tour par tour en langage C, fonctionnant intégralement dans le terminal.
Le joueur doit composer une équipe de trois personnages pour affronter une autre équipe, contrôlée :

-soit par l'ordinateur (mode Univers),
-soit par un autre joueur (mode Versus).

Chaque personnage dispose de statistiques et compétences uniques (attaque, défense, soins, malus, zone, etc.), et le combat se déroule selon une alternance de tours.

3. Organisation et méthode de travail

Nous avons commencé le projet il y a trois semaines.
Les premiers jours ont été consacrés à la mise en place de notre environnement de travail : installation de VSCode, GCC, Git Bash, etc. configuration du Live Share de VSCode pour coder ensemble en temps réel.
Ensuite, nous avons découpé le travail de façon progressive :

-création des structures (Personnage, Equipe, Compétence, etc.),
-chargement depuis les fichiers textes,
-logique de sélection d'équipe,
-déroulement du combat et effets temporaires,
-gestion de la mémoire, narration et fin de jeu.

Nous avons travaillé de manière intense durant la dernière semaine, souvent 5 à 7 heures d'affilée, pour corriger les bugs, tester les scénarios de combat, et finaliser l'intégration.
Bien que nous travaillions en même temps, il nous arrivait rarement de coder séparément, sauf si un imprévu apparaissait.

4. Problèmes rencontrés et solutions

**Combat et ordre de tour
Au départ, les personnages jouaient en fonction de leur vitesse, ce qui faisait que certains agissaient plusieurs fois d'affilée.
Cela rendait le jeu déséquilibré.

→ Solution : alternance stricte → Équipe 1 personnage 1, Équipe 2 personnage 1, Équipe 1 personnage 2, etc.

La vitesse a ensuite été réutilisée uniquement dans le calcul des chances d’esquive.

**Gestion des compétences et effets
Implémenter les compétences avec recharge et effets temporaires (saignement, régénération, malus...) était complexe.

→Nous avons créé des temps de recharge[] et effets[] pour chaque personnage, mis à jour à chaque tour.

Des messages d'erreur informatifs nous aidaient à détecter les oublis de données.

**Segmentation faults
Plusieurs erreurs de segmentation sont apparues : oublis de libération de mémoire avec free(), tailles de tableaux insuffisantes, accès à des pointeurs non initialisés.

→Elles ont été corrigées par une vérification systématique des allocations et des tailles limites.

**Mode "robot contre robot" abandonné
Nous avions initialement implémenté un mode autonome où deux équipes s’affrontaient automatiquement.

→Par manque de temps, nous avons décidé de l’abandonner pour nous concentrer sur les modes demandés (difficultés du mode joueur vs univers).

**Fatigue et erreurs simples
En fin de projet, la fatigue nous a fait perdre beaucoup de temps à cause d’erreurs simples (variables mal nommées, fautes de frappe, oublis de return, etc.), ce qui a ralenti le débogage.

5. Résultats obtenus

Le jeu est entièrement fonctionnel et respecte le cahier des charges :

-Deux modes de jeu (univers et versus)
-Interface colorée en console (ANSI + ASCII art)
-Chargement dynamique des données depuis des fichiers textes
-Système de tour par tour avec compétences et effets
-Affichage narratif et fin de jeu immersive
-Compilation via Makefile
-Documentation claire via README.md