Rapport de Projet : CY FIGHTERS

Partie Développement - Hachem

Contributions Principales

J'ai conçu la boucle de jeu principale (*gameloop*) et l'intégralité du système de combat. Ce système gère :

- **Les points de vie et l'énergie des personnages**
- **Les mécaniques d'attaque et de dégâts**
- **Le système d'esquive basé sur la vitesse**
- **Les techniques spéciales et leurs effets**

Principales Innovations

- **Système de régénération d'énergie** inspiré de *Red Shadow* (régénération automatique en fin de tour)
- **Mécanique d'esquive probabiliste** liée à la vitesse du personnage
- **Gestion dynamique des effets de combat** tour par tour

Partie Interface - Amine

Développement de l'Interface Utilisateur

J'ai développé l'interface utilisateur en mode terminal après l'abandon de SDL2 en raison de problèmes techniques. L'interface comprend :

- **Affichage clair des équipes** avec barres de vie (#) et d'énergie (>)
- **Section "ACTION ORDER"** pour l'ordre des tours
- **Boîte de dialogue centrale** pour les messages du jeu
- **Système HID modulaire** pour les interactions clavier

Défis Majeurs

- **Reconception complète pour le terminal**
- **Résolution des problèmes de compatibilité multiplateforme**
- **Optimisation de l'affichage des données**

Organisation et Méthodologie

Nous avons adopté une workflow Git rigoureux :

- **Développement en branches dédiées**
- **Revue de code avant merge**
- **Tests réguliers sur configurations LINUX**
- **Documentation partagée des modules**

Résultats et Bilan

Le jeu final offre :

- **Un système de combat équilibré et dynamique**
- **Une interface terminal claire et réactive**
- **Une bonne modularité** (pour d'éventuelles extensions ou ré utilisation pour d’autres projet)
- **Une base de code maintenable et documentée**

Compétences Acquises

Cette expérience nous a permis d'acquérir des compétences en :

- **Gestion de projet collaboratif**
- **Résolution de problèmes techniques**
- **Adaptation aux contraintes**
- **Assurance qualité logicielle**

Partie Architecture et Intégration - Julien

Contributions Principales

J’ai pris en charge la création et l'organisation initiale du projet en :

- **Mettant en place la structure GitHub (dépôt, branches, TODO list)**
- **Concevant l'architecture système de base**
- **Développant le système de fichiers et de données modulable**

Système Clé Implementé

J'ai conçu un framework central comprenant :

Des fichiers .h organisés pour les structures fondamentales :

- **Personnages et leurs attributs**
- **Système d'équipes**
- **Gestion des effets et talents**

Un système de données externalisées (fichiers texte) permettant :

- **Ajout de personnages sans modification du code**
- **Création de talents via simple configuration**

Innovations Techniques

Particularités du système :

- **Modularité poussée pour une extensibilité maximale**
- **Mécanisme d'import des données depuis fichiers externes**
- **Architecture découplée pour intégration aisée des autres modules**

Défis Rencontrés

Principales difficultés surmontées :

Gestion complexe de la mémoire :

- **Problèmes récurrents avec strcat() et allocations**
- **Segmentation faults difficiles à tracer**

Import des données :

- **Lecture/écriture sécurisée depuis fichiers**
- **Conversion texte → structures C**

Maintenance de la cohérence système :

- **Garantir la stabilité malgré la flexibilité**
- **Documentation exhaustive pour les collaborateurs**

Processus d'Intégration

J'ai assuré l'unification des composants :

Connexion des structures de données avec :

- **La gameloop de Hachem**
- **Le système de combat**

Adaptation de l'interface d'Amine pour :

- **Afficher les données dynamiques**
- **Interagir avec le framework**

Bilan Technique

Points forts du résultat :

- **Code hautement modulable et documenté**

- **Base solide pour d'éventuelles extensions**
- **Intégration fluide des différentes parties**

Retour d'Expérience

Acquis majeurs :

- **Maîtrise avancée de la gestion mémoire en C**
- **Compétences en architecture logicielle modulaire**
- **Expérience précieuse en intégration système**
- **Gestion des contraintes inter-modules**

Perspectives d'amélioration :

- **Interface de configuration plus intuitive**
- **Meilleure gestion des erreurs I/O**
- **Système de logs plus complet**

Évaluation Globale

Bien que le C ait présenté des défis particuliers (gestion mémoire, pointeurs), l'architecture finale offre une base solide et flexible. Le temps limité n'a pas permis d'exploiter pleinement le potentiel créatif du système.

PARTIE BONUS : Les idées non gardés

Initialement prévus pour une version graphique du jeu, ces sprites n'ont pas été intégrés suite à notre basculement vers une interface terminal. Voici les éléments visuels qui n'ont pas été exploités :

Amine - Sprites non utilisés (Aseprite)

Initialement, j'avais prévu de créer tous les assets graphiques sous **Aseprite**, avec :

- **Des sprites pixel-art pour les personnages**
- **Des animations d'attaque/effets spéciaux**
- **Une interface utilisateur et un menu stylisé**

Julien - Système de progression par niveaux
J'avais conçu un système où chaque personnage :

- **Gagnait de l'XP après les combats**
- **Montait en niveau (avec cap au level 50)**
- **Augmentait ses stats (attaque, défense, vitesse)**
- **Débloquait de nouveaux talents à certains paliers**

Hachem - Éditeur de personnages/talents
Un système complet était prévu pour :

- **Créer des personnages custom via fichier JSON**
- **Designer des talents uniques avec :**
  - **Effets spéciaux (empoisonnement, soins...)**
  - **Conditions d'activation**
  - **Modificateurs de stats**
- **Générer des équipes thématiques**