Rapport du projet CY FIGHTERS

v1.0 Projet CY-Fighters

Par :
GRANDJEAN Yanis
TRENY Hugo
KIALA MPUTU Ronyx

Sommaire

1. Introduction

1.1.Contexte

Dans le cadre de nos études d'informatique nous avons réalisé un projet de fin d'année en groupe.
Parmi tous les sujets proposés notre intérêt s'est porté sur le projet CY IGHTERS, en effet au vu de nos capacités actuelles en programmation il nous a semblé que ce sujet était le plus abordable et qu'il nous permettrait de mettre en pratique toutes les connaissances acquises au cours de l'année.

1.2.Objectif

Dans ce projet il nous est demandé de :

- Créer un jeu qui permet à 1 ou 2 joueurs de faire s'affronter des équipes au tour par tour
- Permettre à l'utilisateur de choisir sa propre équipe de combattant parmi un liste construite à partir d'un fichier
- Un combat dure jusqu'à ce que l'une des 2 équipes soit vaincue.

- Chaque combattant d’une équipe peut attaquer l’adversaire de son choix en utilisant, soit une frappe classique, soit faire appel à une technique spéciale
- Permettre à l’utilisateur d’affronter un autre joueur ou un “bot”
- Mettre au point une interface graphique sur le terminal permettant d’afficher toutes les informations nécessaires aux choix des joueurs

2.Conception

2.1.Conception du programme

Lors de la réalisation du projet, nous nous sommes réparti les tâches afin de pouvoir travailler rapidement et plus efficacement sur le projet. L'un était chargé de la réalisation de la structuration des combattants et fonctionnalités diverses. Un autre de sur la conception des fonctionnalités liées au combat (barre de tour,application des attaques spéciales,etc).
Et le dernier était chargé de la réalisation de l'affichage (esthétisme et ergonomie sur le terminal),ainsi que la conception des fichiers regroupant les statistiques de nos combattants .

Les structures de données utilisées pour représenter les combattants sont définis comme suit :

```c
typedef struct {
    char nom[MAX_NOM];
    int degat;
    char propriete_affectee[MAX_NOM];
    char operation[10];
    char description[MAX_DESC];
    int tours_rechargement;
    int cooldown_actuel;

} TechniqueSpeciale;

typedef struct Combattant {
    char nom[MAX_NOM];
    int pv;
    int pv_max;
    int attaque;
    int buff_attaque;
    int defense;
    int buff_defense;
    int agilite;
    int debuff_agilite;
    int vitesse;
    int action;
    TechniqueSpeciale techniques[MAX_TECHNIQUES];
    int nb_techniques;
    int est_actif;
    int est_KO;
    int position;
    struct Combattant* equipe; // Ajout du pointeur vers l'équipe
    int brulure;
} Combattant;
```

Au début, nous hésitons entre le fait de créer une fonction pour chaque attaque spécial, puis l'associer à un combattant mais n’ayant pas encore aborder ce sujet là en cours nous avons décider de procéder autrement en créant un structure pour les attaques spéciales regroupant chaque caractéristique de cette

dernière.Nous avons limités le nombre d'attaques spéciales à 1 par combattant pour des soucis de manque d'inspiration.

Suite à cela nous avons créer une autre procédure sélection() de personnage qui permet à deux joueur de créer leur propre equipe. Pour ce faire on a dû créer une procédure aperçu() permettant d'afficher les statistiques du personnage selectionné à partir dun fichier.

```c
void apercu(FILE* fichier, int choix) {
    if (fichier == NULL) {
        fprintf(stderr, "Erreur : fichier invalide dans apercu().\n");
        return;
    }

    if (choix < 1 || choix > NBPERSO) {
        fprintf(stderr, "Erreur : choix invalide (%d) dans apercu().\n", choix);
        return;
    }

    char buffer[200];
    char nom[MAX_NOM];
    int numero = 1;
    int stat;

    if (fichier == NULL) {
        fprintf(stderr, "Erreur : fichier invalide dans apercu().\n");
        return;
    }
    rewind(fichier);

    while (numero < choix && fgets(buffer, sizeof(buffer), fichier)) {
        if (strcmp(buffer, "-\n") == 0) {
            numero++;
        }
    }
```

```c
    fgets(nom, sizeof(nom), fichier);
    nom[strcspn(nom, "\n")] = '\0';
    printf("Nom : %s\n", nom);

    fscanf(fichier, "%d\n", &stat);
    printf("PV max : %d\n", stat);
    fscanf(fichier, "%d\n", &stat);
    printf("Attaque : %d\n", stat);
    fscanf(fichier, "%d\n", &stat);
    printf("Défense : %d\n", stat);
    fscanf(fichier, "%d\n", &stat);
    printf("Agilité : %d\n", stat);
    fscanf(fichier, "%d\n", &stat);
    printf("Vitesse : %d\n", stat);

    printf("\n ==== Technique ====\n");

    fgets(buffer, sizeof(buffer), fichier); // Nom technique
    buffer[strcspn(buffer, "\n")] = '\0';
    printf("Technique spéciale : %s\n", buffer);

    fgets(buffer, sizeof(buffer), fichier); // Description
    buffer[strcspn(buffer, "\n")] = '\0';
    printf("Description : %s\n", buffer);

    fscanf(fichier, "%d\n", &stat);
    printf("Tours de rechargement : %d\n", stat);

    fscanf(fichier, "%d\n", &stat);
    /*printf("Cooldown actuel : %d\n", stat); On ne l'affiche pas dans l'aperçu*/
```

Cette fonction sera appeler dans la fonction constructeur_perso() permettant de selectionner et construire le combattant à partir du même fichier à l'aide la fonction fscanf().

```c
void construction_perso(Combattant* perso, FILE* fichier) {
    if (perso == NULL || fichier == NULL) {
        fprintf(stderr, "Erreur : pointeur NULL dans construction_perso().\n");
        return;
    }

    int choix = -1;
    int validation = -1;

    while (validation != 1 || choix < 1 || choix > NBPERSO) { //faire un getInt pour voir si la plage de donnée est correcte
        printf("Entrez le numéro du combattant à sélectionner (1 à %d) : ", NBPERSO);
        choix = getInt(1, NBPERSO);

        if (choix < 1 || choix > NBPERSO) {
            printf("Ce combattant n'existe pas. Réessayez.\n");
        } else {
            rewind(fichier);
            apercu(fichier, choix);
            printf("\nValidez-vous ce choix ? (1 = oui, 0 = non) :\n ");
            validation = getInt(0, 1);
            if (validation != 1) {
                printf("Recommençons la sélection.\n");
            }
        }
    }

    rewind(fichier);

    // Avancer dans le fichier jusqu'au combattant choisi
    char buffer[200];
    int numero = 1;
    while (numero < choix && fgets(buffer, sizeof(buffer), fichier)) {
        if (strcmp(buffer, "-\n") == 0) {
            numero++;

            fgets(perso->nom, MAX_NOM, fichier);
            perso->nom[strcspn(perso->nom, "\n")] = '\0';

            fscanf(fichier, "%d\n", &perso->pv_max);
            perso->pv = perso->pv_max;
            fscanf(fichier, "%d\n", &perso->attaque);
            fscanf(fichier, "%d\n", &perso->defense);
            fscanf(fichier, "%d\n", &perso->agilite);
            fscanf(fichier, "%d\n", &perso->vitesse);

            // Lire la technique spéciale
            fgets(perso->techniques[0].nom, MAX_NOM, fichier);
            perso->techniques[0].nom[strcspn(perso->techniques[0].nom, "\n")] = '\0';

            fgets(perso->techniques[0].description, MAX_DESC, fichier);
            perso->techniques[0].description[strcspn(perso->techniques[0].description, "\n")] = '\0';

            fscanf(fichier, "%d\n", &perso->techniques[0].tours_rechargement);
            fscanf(fichier, "%d\n", &perso->techniques[0].cooldown_actuel);

            perso->techniques[0].propriete_affectee[0] = '\0';
            perso->techniques[0].operation[0] = '\0';
            perso->est_actif = 0;
            perso->nb_techniques = 1;
            perso->action = 0;
            perso->est_KO = 0;
            perso->position = -1;
            perso->brulure = 0;
            perso->debuff_agilite=0;
            perso->buff_defense=0;
            perso->buff_attaque=0;
```

Ensuite nous avons juste appeler cette fonction dans la fonction selection() tant que la taille maximal des equipes n'était pas atteint tout en ajustant la position des personnages.

Pour conception de la parti combat nous avons tout d'abord commencer par créer une fonction procédure tour() permettant de choisir l'action que l'on souhaite effectuer(attaquer un ennemi, lancer une attaque spécial), la cible de cette action et selon l'action choisi on lancera une fonction; attaque() pour les attaque basiques ou attaques spécial().

```c
//Tour du combat
void tour(Combattant* perso, Combattant* equipe) {

    if (!perso || !equipe) {
        fprintf(stderr, "Erreur: paramètre NULL dans tour().\n");
        return;
    }

    if (perso->nb_techniques <= 0 || !perso->techniques) {
        fprintf(stderr, "Erreur: aucune technique définie pour %s\n", perso->nom);
        return;
    }

    if (perso->est_KO == 1) return;

    int choix = -1;
    //choix d'attaque
    while (choix != 1 && choix != 2) {
        printf("%s - Quelle action voulez-vous effectuer ?\n", perso->nom);
        printf("1 = Attaquer\n2 = Capacité spéciale (%s%s)\n",
               perso->techniques[0].nom,
               perso->techniques[0].cooldown_actuel > 0 ? " - en rechargement" : "");
        choix = getInt(1,2);
        if (choix != 1 && choix != 2) {
            printf("Mauvais choix, recommencez.\n");
        }
    }
    TechniqueSpeciale* tech=perso->techniques;
    if (tech->cooldown_actuel > 0) {
        printf("La technique %s est encore en rechargement (%d tours restants).\n",
               tech->nom, tech->cooldown_actuel);
               printf("passage en mode attaque normale\n");
        choix=1;
    if (choix == 1) {
         int choix = -1, validation = 0;
    //choix de cible du combattant
    while (choix < 1 || choix > TAILLE_EQUIPE || validation != 1) {
        printf("Entrez la position du personnage à attaquer : ");
        choix = getInt(1,TAILLE_EQUIPE);
        validation = cible_valide(&equipe[choix - 1]);
    }
        attaque(perso,&equipe[choix - 1]);
        if (!equipe_vivante(equipe)) {
            return; // on quitte tour()
        }
    } else {
        attaque_speciale(perso, equipe);
    }
    if (perso->equipe == NULL) {
        printf("Erreur: impossible de déterminer l'équipe du personnage\n");
    } else {
        Combattant* equipe1 = perso->equipe;
        Combattant* equipe2 = equipe;
        if (perso->equipe == equipe) {
            equipe1 = equipe;
            equipe2 = perso->equipe;
        }
        //réduction du cooldown
        for (int i = 0; i < perso->nb_techniques; i++) {
            if (perso->techniques[i].cooldown_actuel > 0) {
            perso->techniques[i].cooldown_actuel--;
            }
        }
        //Buff et débuff
         if(perso->brulure>0){
             printf("\n");
         //Buff et débuff
          if(perso->brulure>0){
              printf("\n");
              printf("%s souffre de sa brulure\n",perso->nom);
         perso->pv-=10;
         if(perso->pv<0){
             printf("%s succombe a sa brulure\n",perso->nom);
             perso->est_KO=1;
         }
         printf("il reste %d tours de brulure a %s\n",perso->brulure,perso->nom);
         perso->brulure-=1;
     }
     if(perso->buff_attaque>0){
         perso->buff_attaque-=1;
         printf("\n");
         printf("il reste %d tours de buff d attaque a %s\n",perso->buff_attaque,perso->nom);
     }
     if(perso->buff_defense>0){
         perso->buff_defense-=1;
         printf("\n");
         printf("il reste %d tours de buff de defense a %s\n",perso->buff_defense,perso->nom);
     }
     if(perso->debuff_agilite>0){
         perso->debuff_agilite-=1;
         printf("\n");
         printf("il reste %d tours de debuff d agilite a %s\n",perso->debuff_agilite,perso->nom);
     }
         //Met en pause l'écran
         sleep(5);

         afficher_plateau(equipe1, equipe2);
    }
```

Puis nous avons mis en place une fonction phase() permettant de simuler les différentes phases d'un combat; remplissage de la barre d'action, lorsque qu'un personnage remplit sa barre on appelle la fonction tour() lui permettant d'effectuer une action.

Bien sûr à la fin de chaque action on appelle une fonction permettant de vérifier si l'adversaire à toujours au moins 1 combattant vivant à l'aide d'une fonction créer au préalable.

Et finalement pour ce qui est de l'affichage pour résumé synthétiquement ce que fait le fichier affichage.c. Il lit les informations depuis un fichier texte, puis les affiche sous la forme d'un tableau dans le terminal avec un formatage précis(alignement, colonnes, etc.).
Il utilise des structures pour stocker les données et des fonctions pour lire, organiser et afficher les informations de façon lisible.

2.2 Problème rencontrés et solution apportées

Bien que les choses se soit bien passé dans l'ensemble nous avons rencontré certain problème durant la réalisation dont notamment ;

- La réalisation de l'affichage final. Notre approche initiale fonctionnait correctement avec des données statiques, mais elles ne s'adaptaient pas aux données chargées depuis un fichier. Pour résoudre ce problème nous avons dû repenser entièrement la manière dont l'affichage etait généré, afin de le rendre plus flexible et cohérent avec des données dynamiques.

- L'implémentation de buff (ou de débuff) sur la durée. Pour y remédier nous avons tout simplement implémenter des compteurs de tours dans notre structure combattant.

- Trouver une façon d'appliquer une technique spécial. A l'aide de la fonction strcmp() on a comparé le nom de ta technique du combattant puis lancé la fonction qui lui était associé.

3.Conclusion

Ce projet a demandé beaucoup réflexion et investissement de notre part pour pouvoir le mener à bien. Et nous à également permis de mettre en pratique toutes les connaissances acquises au cours de l'année. Ce rapport se veut assez synthétique, n'abordant que notre cheminement et les points les plus essentielles de notre programme.