Rapport du projet des petits chevaux

Clément Gindrier

Le but du projet était de faire un programme portable au maximum, c'est pourquoi je n'ai pas coloré le jeu, ni appelé des fonctions systèmes comme system(« pause »).

1. Comment jouer :

Si une sauvegarde existe (sauvegarde.txt), rentrer 1 pour la charger, 0 sinon.

Sinon rentrer le nombre de joueurs entre 2 et 4, puis pour chaque joueur s'il est joueur ou non (rentrer 0 pour joueur et 1 pour PNJ).

Le jeu commence :

Si aucun cheval n'est jouable ou qu'un PNJ joue, vous devez simplement appuyer sur « entrer » pour continuer. Vous pouvez aussi rentrer « 0 » pour quitter ou sauvegarder.

Pour jouer vous devez rentrer le numéro d'un cheval. Les chevaux jouables sont marqués entre accolades. Par exemple: Quel cheval voulez-vous jouer ? {1, 2, 3, 4} :

Vous pouvez là aussi rentrer « 0 » pour quitter ou sauvegarder.

2. Règles choisies :

En plus de la consigne, j'ai rajouté qu'on ne peut pas être 2 sur la même case (règle officielle, non précisée dans la consigne). Ainsi, un cheval ne peut pas être sorti ou joué s'il finit sur une case occupée par un allié. Par contre il peut le doubler.

3. Comportement du personnage non joueur :

Le rôle du PNJ va être de choisir intelligemment quel cheval jouer. Il va appliquer les priorités suivantes :

1 : gagner, monter l'escalier ou arriver pile en dessous.

2 : manger (même en reculant avec la fin de la boucle).

3 : sortir un cheval (si possible).

4 : avancer un joueur dans le jeu qui ne recule pas.

Si plusieurs chevaux ont la même priorité, il choisit le cheval le plus avancé dans le je

4. Structure :

Ma structure `Joueur` est allouée dynamiquement en un tableau dynamique de type Joueur qui dépend du nombre de joueurs. On aura donc joueur[0] qui correspond à la structure du premier joueur (Jaune) etc.

```
Joueur* joueur = (Joueur*)malloc(nb_joueurs * sizeof(Joueur));
```

La structure est composée des éléments suivants :

- `char couleur` : contient J, R, V ou B selon la couleur du joueur (utile pour l'affichage)

- `int pos[4]` : contient la position, c'est-à-dire la case où se situent les 4 chevaux du joueur (0 pour l'écurie jusqu'à 63 la case d'arrivée). Utile pour savoir où se situe chaque cheval dans le jeu.

`-int x[4];`

`-int y[4]` : coordonnées des chevaux par rapport au plateau. Utile pour afficher les chevaux mais aussi pour les collisions avec les chevaux adverses (pour les manger).

- `Bool pnj; Bool est une énumération « OUI » ou « NON ».`

5. Demander un nombre :

Même en vérifiant le nombre, scanf_s validait « 3a » ou « 3.5 » en le comptant comme « 3 ». J'avais donc le choix entre demander une chaîne de caractères puis la convertir ou vérifier que le buffer après « 3 » est vide. J'ai choisi la 2nd option en utilisant `fgetc(stdin).` L'avantage de cette méthode est qu'elle valide « 02 » comme « 2 ».

La fonction vérifie que le nombre est dans un tableau donné en entrée, elle est donc réutilisable pour chaque demande.

6. Sauvegarde :

Lorsque l'on quitte pendant le jeu en rentrant « 0 », le jeu nous propose de sauvegarder la partie en cours en rentrant « 1 ».

La sauvegarde écrase le fichier texte ou le crée.

Au lancement du programme, la reprise de la partie sera alors proposée (laissant la sauvegarde intacte, pouvant donc être rejouée encore et encore).