

Projet informatique: Honshu (Lot B)

Romain PEREIRA
Douha OURIMI
Afizullah RAHMANY
Guangyue CHEN

08/04/2018

Sommaire

1	Introduction	2
2	Rapport lot B	3
2.1	Interfaces 'affichage.h' et 'entree.h'	3
2.2	1ère implémentation	3
2.2.1	Partie affichage	3
2.2.2	Partie entrée	3
2.3	2ème implémentation avec ncurses	3
2.3.1	Partie affichage	4
2.3.2	Partie entrée	6
3	Conclusion	6

Préambule

Ce projet est réalisé dans le cadre de nos études à l'ENSIIE. L'objectif est de prendre en main des outils de 'programmation agile', en développant un jeu de carte : le Honshu.



Figure 1: *Plateau de jeu*

1 Introduction

Nous voici dans la deuxième partie du projet. Pour ce lot nous étions en charge de la réalisation d'un jeu avec affichage en mode terminale. Comme une partie du lot B a déjà été réalisé lors du lot précédant nous avons décidé de scinder le groupe en deux parties et de faire deux affichages différents. L'un répondant stricto sensu au demande du lot B et un autre visuellement plus agréable et user-friendly. Le travail a été donc réparti de la manière suivante :

- Guangyue et Romain ont été en charge d'un affichage avec la bibliothèque ncurses en mode terminale. Cette bibliothèque permet un rendu visuellement plus esthétique de manière plus aisée.
- Douha et Afizullah ont été en charge d'un affichage avec en mode terminale avec toutes les fonctions écrites ex nihilo :

- Afizullah implémenta l'affichage de la gille et des cartes en main.
- Douha était en charge des entrées. Il s'agit d'implémenter l'interaction qu'aura le joueur avec l'état du jeu.

Les deux modes de rendu sont accessibles en recompilant le jeu avec les commandes:

```
# make MODE=ncurses  
ou  
# make MODE=standart
```

Nous étions également en charge des tests de recouvrement et de la limitation de la zone de jeu. Il convient de préciser que les tests de recouvrement ont été réalisé de manière intrasèque lors de l'implémentation du lot précédant

2 Rapport lot B

2.1 Interfaces 'affichage.h' et 'entree.h'

Définissent les structure et les fonctions devant être implémenter pour un type d'affichage ou de rendu. Les 2 modes (standart et ncurses) ont chacune une implémentation différente ('src/standart/', et 'src/ncurses'), mais tous le reste du code reste inchangé: le Makefile s'occupe de compiler avec la bonne implémentation.

2.2 1ère implémentation

2.2.1 Partie affichage

affichage.h Il s'agit d'afficher la grille, et les tuiles dans la main du joueur. En effet, chaque case de la grille possède un attribut type qu'il convient de convertir en un caractère (à l'aide la fonction `case_char`) qu'on affichera ensuite dans l'emplacement ad hoc dans la grille affiché. La connaissance des tuiles sur la grille n'est pas nécessaire, en revanche, c'est le cas pour l'affichage de la main du joueur. Pour discriminer une tuile dans la main et une tuile posée sur la grille, on se sert du flag présent de la structure tuile indiquant si celle-ci est sur la grille ou non.

```

* Honshu git:(master) # ./bin/honshu.out -r 8.1
Le jeu est initialisé
Le jeu boucle

Grille
.....
.....
.....
....LF..
....PR..
....RP..
.....

Main du joueur
-----
0:  L R
   L V
   U U
-----
1:  U L
   F U
   R V
-----
2:  R L
   F P
   U R

Tapez:
1. si vous voulez insérer une tuile
2. si vous voulez tourner une tuile
3. si vous voulez supprimer une tuile
> 1
```

2.2.2 Partie entrée

entree.h Il s'agit d'implémenter la manière dont le joueur pourra interagir avec la partie. l'input du joueur sera interpréter pour que celle-ci modifie la grille et les tuiles de la partie donc les structure ad hoc. On offre au joueur la possibilité d'insérer, faire pivoter ou supprimer une tuile. Ces possibilités sont offertes par les fonctions `grille_insert` et `grille_can_insert` qui assure les tests de recouvrement et de limitation de la grille. En effet une tuile est décrit par son orientation et par sa coordonnée qui celle de sa case en haut à gauche. On vérifie qu'elle ne recouvre ni une case ni entièrement une autre tuile ni ne dépasse de la grille et recouvre au moins une case. Une tuile ne peut être insérer à travers la fonction `grille_insert` que si elle a été passé en argument de la fonction `grille_can_insert` et a retourné 1. On peut alors de rafraîchir l'affichage à travers la boucle de jeu.

2.3 2ème implémentation avec ncurses

Affiche le jeu: Celle-ci étant plus technique, elle a été décrite par les algorithmes suivantes.

2.3.1 Partie affichage

Affiche le jeu: Affiche la grille, les tuiles dans la main ainsi la tuile sélectionnée et présentement sur la grille. Les tuiles ayant des états différents auront des couleurs différentes (celle qui est sélectionnée sera soit vert si l'insertion est possible, rouge sinon..

Algorithm 1: Affiche le jeu

Affiche le titre
Affiche les tuiles
Affiche la grille
Affiche la tuile selectionnée dans la grille
Affiche les commandes de jeu.

Affiche toutes les tuiles Affiche les tuiles dans des couleurs différentes suivant leur état et si elles sont sélectionnées par le joueur.

Algorithm 2: Affiche toutes les tuiles

Require: une structure de *honshu* ;
Ensure: renvoie Void, et affiche les tuile dans le terminal
affiche le titre
 $x \leftarrow \text{Index_de_tuile} \% 2 * 7$
 $y \leftarrow \text{Index_de_tuile} / 2 * 7$
for Chaque *tuile* de *Honshu* **do**
 tuile_à_affiche \leftarrow *Honshu.tuile*
 if *tuile_à_affiche* est SUR_CARTE **then**
 change la couleur en rouge
 end if
 Rappelle la fonction *afficher_une_tuile* et affiche *tuile_à_afficher* en (x,y)
end for

Affiche une tuile: Affiche une tuile dans le terminale, si elle est déjà selectionnée, sa couleur change.

Algorithm 3: Affiche une tuile

Require: une *tuile* et deux coordonnées (*x* , *y*) ;
Ensure: Renvoie Void, et affiche une tuile en (*x*,*y*)
for Chaque *case* de *tuile* **do**
 Par la *rotation* de la *tuile*, déduire la place de ces *cases*
 Rappelle la fonction *case_char* et affiche le caractère correspondant au type de la case
end for

Affiche le grille: Affiche la grille dans le terminale.

Algorithm 4: Affiche la grille

Require: une structure de *honshu* ;
Ensure: renvoie Void, et affiche la grille dans le terminale
grille \leftarrow *Honshu.grille*
 affiche le titre
for Chaque *case* de la *grille* **do**
 cases \leftarrow *grille.case* (par rappel de la fonction *grille_get*)
 if *case* \neq NULL **then**
 if *case* est dans une tuile affichée **then**
 la tuile prend la couleur rouge
 end if
 end if
 Affiche les *cases*
end for



2.3.2 Partie entrée

L'implémentation avec 'ncurses' a été rapide, grâce au travail effectué au lot A. Ncurses fourni une fonction

getch()

qui lit un caractère sur l'entrée standard. Une action a été attribué à chaque touche, voir *ncurses/entree.c* pour plus de détails.

3 Conclusion

Pour ce lot, nous avons mis en place un système d'entrée/sortie modulaire, qui nous a permis de tous travailler en parallèle efficacement. Grâce à cette séparation du travail efficace, nous n'avons eu que peu (voir pas) de 'fusion' (merge) de code à effectuer. De plus, ce système d'interface entrée/sortie nous a permis d'implémenter rapidement 2 interfaces graphiques différentes, et pourra être amplement ré-utilisé pour le lot D (affichage avec SDL). Pour la suite nous préférons garder l'implémentation avec ncurses car elle est plus performante et aboutie. Il nous restera à réfléchir à l'implémentation du solveur sachant que l'algorithme proposé (testant toutes les possibilités d'insertion) a été imaginé lors du choix des structures pendant le lot A.