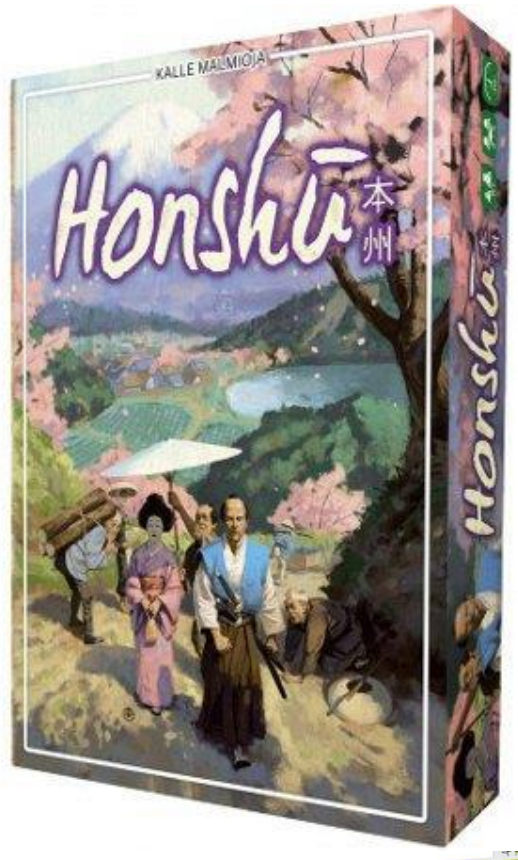
Equipe STRANGER C
Jiahui XU, Kevin XU, Heykel Rajhi, Benoît SCHOLL

Projet Honshu

Rapport LOT B



Sommaire

1 Introduction

1.1 Tâche à effectuer

Pour le lot B, les tâches étaient les suivantes :

Tests de recouvrement :

- Liste des terrains recouverts sur le plateau
- Liste des terrains recouverts par une tuile si on essaie de la poser sur le plateau
- Une tuile ne peut être totalement recouverte à aucun moment de la partie
- Toute nouvelle tuile posée doit recouvrir au moins une case de l'une des tuiles précédemment posée, la précédente case est alors oubliée
- Aucune case Lac ne peut être recouverte.
- Test de la limitation de la zone de jeu
- Fonction permettant le retrait d'une tuile du plateau.

Boucle de jeu :

- Affichage dans le terminal de la grille et des tuiles restantes à placer
- Sélection d'une tuile
- Application possible de rotations à la tuile
- Entrée des coordonnées d'insertion de la tuile dans la grille
- Application des modifications à la grille
- Test de fin de partie (plus de tuiles à placer)

1.2 Répartition des tâches

Kévin XU : Boucle de jeu Principale / Test Fin de Partie
Jiahui XU : Boucle Principale / Fonction Saisie direct Clavier
Benoit SCHOLL : Test de recouvrement / Test unitaire
Heykel RAJHI : Boucle de jeu aléatoire / Affichage des menus

2 Fonctions codées

2.1 Test de recouvrements

2.1.1 VerifHorsTerrain

Dans cette fonction le principe est simple : On veut savoir si l'utilisateur pose sa tuile correctement à l'intérieur de la grille. D'après les coordonnées rentrées par l'utilisateur et taille de la grille, on commence par vérifier si les coordonnées sont bien plausibles donc supérieur à 0. Cela fait on vérifie ensuite si les coordonnées sont bien dans la grille, et que chaque case de la tuile rentrera bien.

2.1.2 VerifLac

On parcourt les 6 cases qui vont être recouverte par la nouvelle tuile, et dès que l'on tombe sur une case de type LAC, on arrête le parcours et on renvoi 0. Si on ne trouve pas de case LAC parmi les 6 cases, on renvoi la valeur 1.

2.1.3 VerifRecouvreUneTuile

On parcourt les 6 cases qui vont être recouverte par la nouvelle tuile, et dès que l'on tombe sur une case différente de « E » (N.B. : E pour Empty symbolise une case vide) on renvoi la valeur 1. Si on ne trouve aucune case différente de E parmi les 6, on renvoi la valeur 0

2.1.4 NeRecouvrePasTotalementUneTuile

Pour résoudre cette problématique, nous sommes partis sur plusieurs piste.
La première idée était de rajouter un Booléen dans la structure case qui indiquait si la case était ou non la dernière représentante de la tuile. Nous avons très vite oublié cette idée car cela impliquait de nombreux parcours complet de grille ; et cela aurait couter chère en temps CPU.

Le temps CPU : Pourquoi nous parait-il si important ?

Nous sommes partis du principe que ce que nous développions dans le lot B allait être utilisé pour le LOT C et l'algorithme du solveur. En effet, un solveur dit « Brute de Force » va essayer toutes les possibilités avec un jeu de départ donnée afin de trouver la combinaison rapportant le plus de point.
Aussi il nous est important de trouver les algorithmes les plus rapide. Nous avons donc décider d'opter pour une nouvelle stratégie : l'histogramme partiel. L'idée est simple : nous créons un tableau d'entier de taille 100. Le nombre 100 a été choisis suite à l'échange avec notre encadrant par Email nous informant qu'il n'y aurait pas plus de 100 tuiles différentes, et donc pas de tuile dont l'indice serait plus grand que 100. Ensuite nous avons délimité la zone de surveillance ainsi :
Etape 1 : On définit la zone où l'on pose une tuile

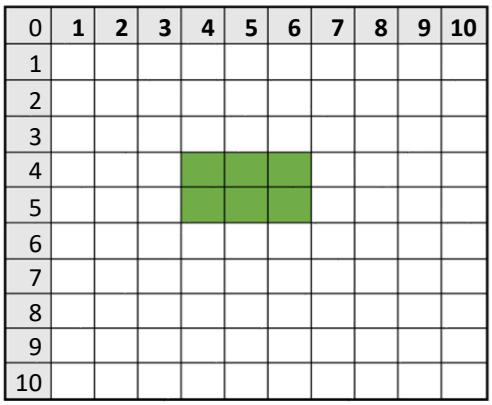


Figure 1 : choix des coordonnées et de l'orientation de la tuile pour sa pose

Ici, la tuile va être posée de façon horizontale avec comme coordonnées (4,4)
Afin de délimiter la zone de parcours, on va identifier chaque zone où une tuile pourrait être posé en amont dont la seule case restante se trouverais dans la zone recouverte par la nouvelle tuile.
Cela nous donne la Figure 2

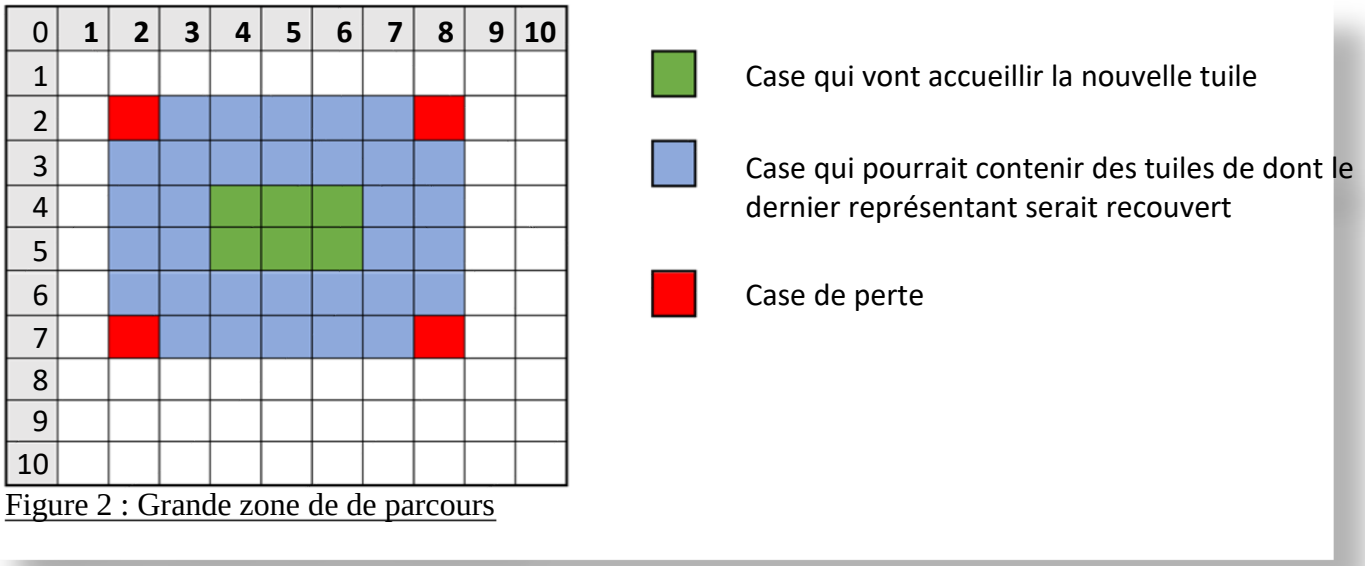


Figure 2 : Grande zone de de parcours

On Parcourt alors une première fois sur la grille dans son état avant la pose de la nouvelle tuile l'ensemble des cases colorées sur ce graphique. On applique ce petit algorithme :

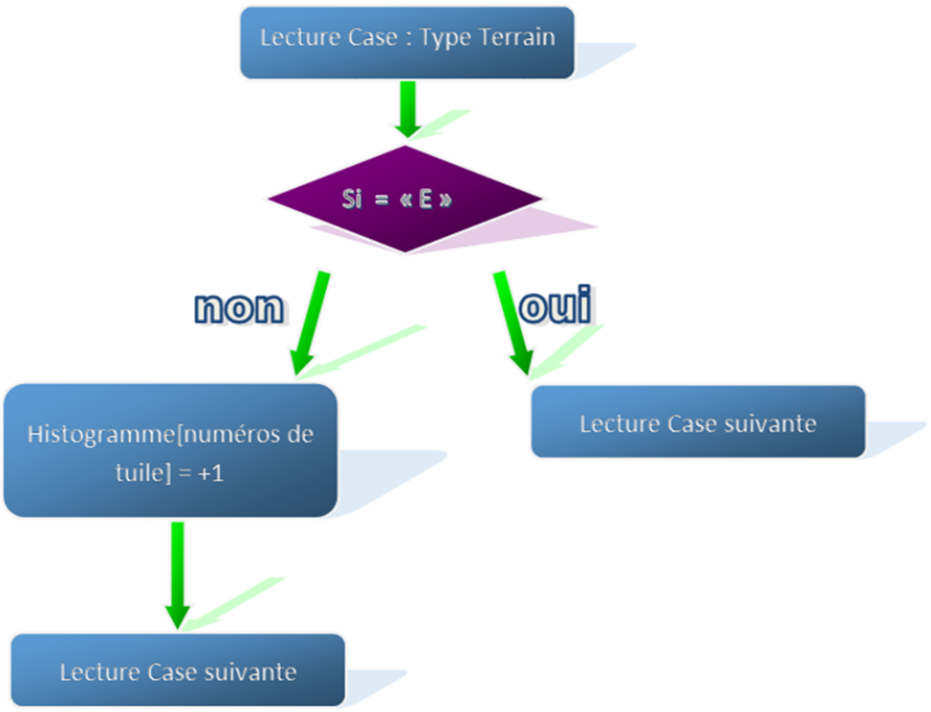


Figure 3 : Premier parcours de grille => construction de l'histogramme

On se retrouve donc avec un tableau contenant le nombre d'occurrence de chaque tuile dans l'ensemble des cases du grand parcours.
Il ne nous reste plus qu'à re-parcourir les 6 case qui vont être recouverte par la dépose de la nouvelle tuile afin de valider la condition : une tuile déjà posé n'est pas entièrement recouverte.
Pour chaque case On applique cette algorithme :

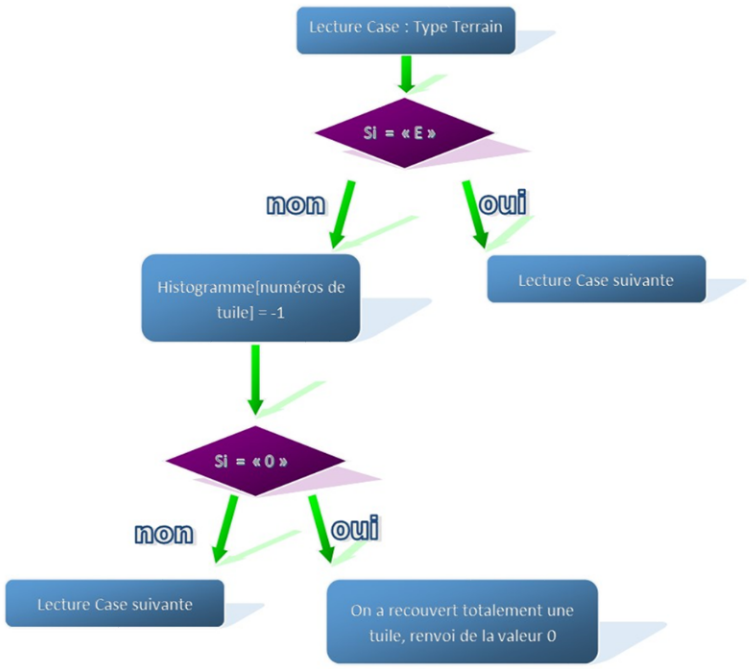


Figure 4 : Deuxième parcours vérification recouvrement totale d'une tuile

Si l'on trouve une case de l'histogramme dont la valeur devient 0, c'est que l'on vient de supprimer la dernière occurrence de la tuile qu'elle représente.

➔ Afin d'éviter tout débordement de tableau, pour chaque lecture de case, on vérifie avant d'y accéder qu'elle fait bien partie de la grille de jeu

➔ Les cases en rouge dans la figure 2 (appelée case perte) représentent des cases inutiles de contrôler dans l'absolue mais nous demanderais de faire des tests supplémentaires afin de les identifier et ainsi conduirait à complexifier le code sans pour autant nous faire gagner du temps CPU.

2.2 Boucle de jeu

2.2.1 boucle_principale

Cette fonction va nous permettre d'afficher l'interface d'une partie de Honshu et permettre au joueur de jouer. Elle prend alors en paramètre un Deck, une grilleDeJeu et un dicoTuiles. Le déroulement d'une partie sera expliqué plus tard.

2.2.2 Jeu_fichier

Cette fonction permet de charger une partie de Honshu à partir d'un fichier présent dans le dossiers sauvegardes. Elle demande à l'utilisateur de choisir 2 fichiers parmi ceux existant qui correspondent à un deck et à un dictionnaire de tuiles. Elle fait ensuite appel à la fonction boucleprincipalepourlancerunepartie.

2.2.3 Jeu_fichier_random

Cette fonction permet au joueur de générer une partie de Honshu avec des tuiles aléatoires grâce à la fonction du lot_A tuile_initrandom.

2.2.4 Test_fin

La fonction test_fin va nous permettre de détecter la fin d'une partie de Honshu. En effet, la partie est finie lorsqu'il ne reste plus de tuile dans le deck, mais elle est également finie lorsque plus aucune tuile ne peut être placer sur le plateau de jeu. Donc même s'il reste encore des tuiles dans le deck, il se peut qu'elles ne peuvent plus être placer. La fonction va tout d'abord vérifier qu'il existe encore des tuiles dans le deck. Puis elle va essayer de les placer. Et elle s'arrête dès qu'une tuile est plaçable. Elle test aussi selon les différentes orientations qu'une tuile possède.

Algorithm 1 – test_fin

```
1:  procedure TEST_FIN(Deck, grilleDeJeu, Dico)
2:      test ← 0
3:      for all T ∈ Deck do
4:          if T ≠ −1 then
5:              test ← 1
6:          end for
7:      end if
8:      end for
9:      if test = 0 then
10:         return 0
11:     end if
12:     l(u) ← 0
13:     for all T ∈ Deck do
14:         if T ≠ −1 then
15:             for Sens ← 0, 3 do
16:                 Rotation de T à l'orientation = Sens
17:                 for X ← 0, taillegrille do
18:                     for Y ← 0, taillegrille do
19:                         if T passent tous les tests de placements en (X, Y) then
20:                             return 1
21:                         end if
22:                     end for
23:                 end for
24:             end for
25:         end if
26:     end for
27:     return 0
28: end procedure
```

3 Jeu

Pour démarrer une partie bien que tout est expliqué dans le Readme il faut entrer la commande "./bin/Honshu"

3.1 Mode de jeu

Notre jeu de Honshu possède deux modes de jeu :

- Avec une initialisation Aléatoire
- A partir de données dans un fichier.

3.2 Déroulement d'une partie

Pour le déroulement d'une partie on l'utilisateur se trouve face à une grille avec des coordonnées (X,Y).Tout d'abord le joueur doit choisir la tuile qu'il compte placer grâce à une numérotations du Deck. Ensuite l'utilisateur peut effectuer une rotation de la tuiles grâce au touche "Q" "D" dès que l'utilisateur est satisfait de l'orientation il peut la confirmer grâce à la touche entrée. Ensuite l'utilisateur aura juste à entrer les coordonnées qu'il souhaite : si la pose n'est pas licite l'utilisateur se verra afficher un message d'erreur et se verra inviter à entrer de nouvelle coordonnées. A tout moment le joueur peut quitter la partie grâce à la touche -1.