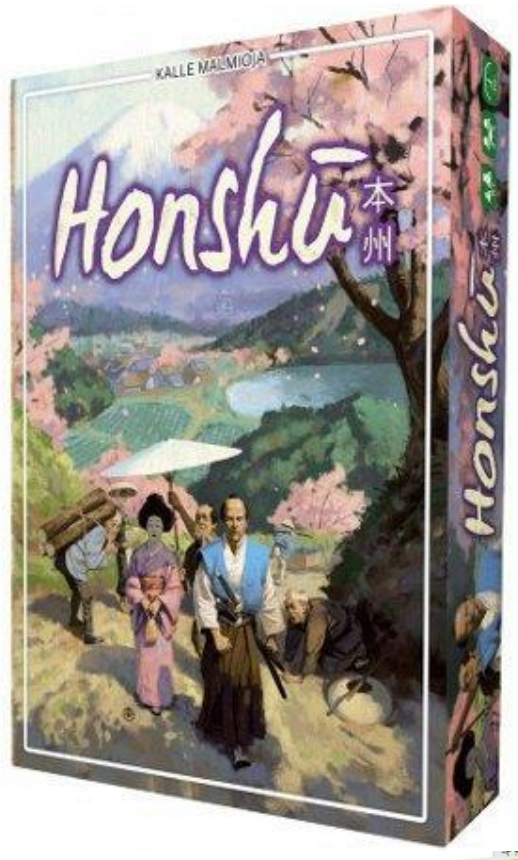
Equipe STRANGER C
Jiahui XU, Kevin XU, Heykel, Benoît SCHOLL

Projet Honshu

Rapport LOT A



Sommaire

Partie 1 : Structure et ensemble de données utilisées

Le jeu du honshu repose sur des cases composant des tuiles que l'on pose sur une grille de jeu. Chaque case ayant sa nature (Village, Forêt, Usine….). Nous avons donc commencé par créer un dictionnaire, **terrain**, de type de case représenté par un caractère dont voici le tableau de correspondance :

Char	Correspondance
E	Case Vide (E pour empty)
P	Plaine
F	Forêt
L	Lac
V	Village
R	Ressource
U	Usine

Structure CASE

On a ensuite créé une structure Case qui sera utilisée sur la grille.

laCase	
Int	NumeroTuile
terrain	typeCase

➡ L'entier NumeroTuile permettra de connaitre le numéro de tuile auquel appartient la case. Par défaut, si la case est vide, NumeroTuile =-1. Grâce à cette information, on va pouvoir mettre en application la règle : une tuile posée ne doit
➡ pas être entièrement recouverte.
Le champ terrain est lui le type de terrain actuellement assigné à la case. Par défaut, il vaut 'E' (pour vide).

Structure Grille de Jeu

La structure Grille de Jeu contient elle la grille (sous forme de tableau de pointeur) ainsi que quelques informations sur le jeu.

grilleDeJeu	
int	taille
laCase	**Grille
int	nbTuilePosées
int	*tuilesDejaPosees

➡ L'entier Taille nous permet de connaître la taille de la grille du jeu.
➡ Le tableau **Grille est un tableau de structure Case. Son allocation est dynamique (tableau de pointeur).
➡ L'entier nbTuilePosées nous sert à connaitre le nombre de tuile déjà posés.
➡ Le tableau dynamique tuilesDejaPosees nous sert d'historique pour rejouer notre partie (et donc recalculer la grille en cas d'enlèvement de tuile déjà posée).

Grâce au tableau tuilesDejaPosees, nous somme capable d'identifier la nature du terrain recouvert par une tuile en cas de suppression de cette dernière. Il nous est donc inutile de rajouter un tableau par case contenant la liste des terrains recouvert.

Structure Tuile

Tuile	
int	Id
Terrain	contenu[3][3]
int	orientation

➡ L'entier ID nous sert à identifier la tuile.
➡ Le tableau Terrain de taille [3][3] contient la tuile. La colonne supplémentaire va nous servir à gérer la rotation (cf Algo Rotation).
➡ L'entier Orientation nous permet de connaître la rotation qui a été appliqué à la tuile.

Structure DECK

DECK	
Int	Taille
tuile*	deckTuile

➡ L'entier Taille nous permet de connaître le nombre de Tuile restant dans notre main.
➡ deckTuile est un tableau dynamique de tuile(s).

Structure dicoTuiles

dicoTuiles	
int	taille
tuile	*leDico

➡ Taille est un entier qui représente la taille du dictionnaire de tuile
➡ leDico est un tableau dynamique de tuiles qui contient toutes les tuiles du jeu (y compris celles qui ne font pas partie du deck).

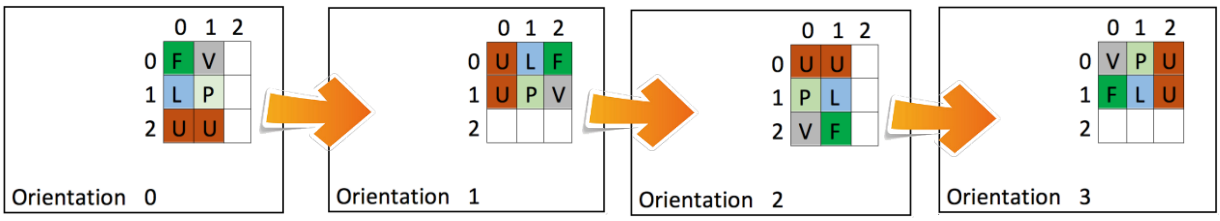
Cette dernière structure va nous servire à contenir toute les tuiles du jeu. Elles sont ranger dans l'ordre d'Id, de sorte que la tuile dont l'identification IdTuile est 5 se trouve en case 5 du tableau de tuile.

Algorithme des principales fonctions

Fonction Rotation Tuiles

Pour cet algorithme, nous allons tirer parti de notre tableau de taille [3][3].

Voici avec un petit schéma comment se comporte notre fonction Rotation.



Pour ce faire, on utilise un tableau tampon de char à une dimension [6], contenant l'ensemble des cases de la tuile, et l'on réécrit le tableau de taille [3][3] de la structure tuile avec les nouvelles données.

Fonction Poser Tuile

Pour poser une tuile, seulement 2 cas s'offre à nous : si la tuile est horizontale ou verticale.
On ne recopie pas les cases vide du tableau (soit la dernière colonne dans le cas d'une tuile verticale, soit la dernière ligne dans l'autre cas).
Afin de ne pas avoir de problème de dépassement, on vérifie en préambule si on ne dépasse pas de la grille de Jeu, puis on recopie le type de terrains de chaque case ainsi que le numéro de la tuile que l'on est en train de poser.

Test Unitaire

Voici la liste des tests Unitaires que l'on a fait :

- Rotation
- Initialisation d'une tuile
- Initialisation d'une tuile à partir d'un fichier
- Création d'une tuile aléatoire
- Lire la case
- Lire le numéro de tuile
- Lire le type de terrain
- Créer une grille
- Charger un jeu à partir d'un fichier

Pour l'ensemble de ces tests, nous avons utilisé la librairie Cunit qui à l'avantage de ne pas s'arrêter à la première erreur mais de bien réaliser l'ensemble des tests demandé, puis de générer un rapport.

Organisation

Pour nous organiser, nous avons dut faire face à un problème de distance. Benoit SCHOLL habitant à saint Etienne, nous avons essentiellement travailler en télétravail. Pour se faire nous avons utilisé plusieurs outils :

Trello

URL : https://trello.com/b/5MPin1dm/projet-1a
Nom : Stranger C / Projet 1A

GitLab

URL : https://gitlab.com/strangerC/lot_A
Nom : Stranger C / Lot A

Facebook Messenger

URL : https://www.facebook.com

Enfin, nous nous sommes réunis plusieurs fois tous ensemble dans les salles de cours de l'ENSIIE.