Compte-rendu du projet de fin de semestre

Push it DOWN !

LIN Gérald
LOPES MENDES Ailton

Année 2016-2017

Licence mathématiques-informatique
TDe - TP10

Guide d'utilisation

Notre jeu **Push it DOWN !** est fait pour vous !
Un menu simple d'accès s'offrira à vous, vous y trouverez tout ce qu'il faut pour passer un bon moment de réflexion.

Sur ce menu, vous aurez la possibilité de choisir entre deux modes de jeu : le premier mode de jeu contient des niveaux prédéfinis et conçus par nos soins, le deuxième mode de jeu génère des niveaux de manière aléatoire. De plus sur ce deuxième mode vous pourrez choisir le nombre de niveaux.

Une fois en jeu, vous vous déplacerez de sorte à atteindre l'arrivée, qui vous est indiquée en vert à l'opposé de votre point d'apparition.
Vous aurez la possibilité de pousser les blocs qui vous gênent afin de vous frayer un chemin vers la fin du niveau. Attention, vous ne pourrez pousser qu'un seul bloc à la fois.

Vous aurez également accès à différentes touches en jeu :
-*La touche A :* annuler votre dernier déplacement, annuler votre dernière action
-*La touche R :* recommencer le niveau
-*La touche P :* charger le niveau précédent
-*La touche N :* charger le niveau suivant
-*La touche D :* pivoter le niveau à droite
-*La touche G :* pivoter le niveau à gauche
-*La touche Echap :* mettre pause au jeu
-*La touche Q :* quitter le jeu

Maintenant que vous savez quoi faire, le jeu est entre vos mains et à vous de jouer !

Avancée du projet :

Le projet est actuellement terminé, toutes les parties obligatoires ont été complétées : l'affichage du niveau, les déplacements de la bille, les déplacements de blocs, annuler un déplacement, le solveur et son générateur aléatoire…

Depuis la finition du projet, nous n'avons pas rencontrés de bugs particuliers. Tout semble fonctionner correctement et comme il le faut.

Des améliorations possibles :

Une amélioration possible concerne la génération d'un niveau, en effet nos niveaux aléatoires sont si aléatoire que dans certaines cas, le niveau peut être très désagréable à voir, c'est-à-dire qu'on n'y voit pas grand chose mise à part les piles de blocs sur les bords du niveau qui cache une partie de celui-ci.

Une amélioration possible dans le code, est que même si nous avons fait de notre mieux, le code est parfois pas si bien structuré, ce qui amène à devoir chercher un peu pour trouver ce que l'on veut.

D'autres améliorations au niveau du gameplay, mais faute de temps, on n'a pas pu les implémenter au jeu.

Les fonctions et leurs utilités :

(étant donné que les fonctions sont entièrement commentées dans le programme, on ne résume ici que les fonctions principales)

-Récupération de la matrice du niveau : la première chose à faire est de récupérer la matrice écrite dans un fichier texte. On récupère la matrice dans « fabrication_matrice » sous forme de liste de listes.
En même temps, on récupère la hauteur maximale du niveau qui permettra de calculer la proportion des blocs.

-**Affichage du niveau :** grâce à la matrice dans la liste, on peut afficher par l'intermédiaire de la fonction « affiche_bloc » et des indices de la liste tous les blocs du niveau.
De plus, cette fonction « affiche_map » affiche également la bille.

-**Les déplacements :** une fois l'affichage du niveau possible, il faut déplacer la bille et les blocs, pour ce faire on utilise la fonction « directions » qui impose les limites du jeu, c'est-à-dire les bordures, les conditions pour déplacer un bloc, les conditions pour la bille, etc.
Cette fonction renvoie les coordonnées de la bille ou les coordonnées de la matrice modifiées en fonction du déplacement.

-**Annuler un déplacement, une action :** pour créer cette fonctionnalité, on utilise deux fonctions.
La première permet d'ajouter à un dictionnaire de bille et de matrice leurs coordonnées respectives à chaque déplacement grâce à un compteur. Et pour une question de « confort » dans le jeu, elle ne prend pas en compte deux fois de suite les mêmes coordonnées.
La deuxième fonction est l'action d'annuler, une fois que la touche pour annuler est appuyée, la fonction récupère les coordonnées du *compteur-1* pour la bille et la matrice, tout en supprimant les coordonnées du *compteur* actuel.

-**Le solveur :** on commence par la fonction « solveur_optim » qui pour chaque direction à partir du point de départ fait appel à « solveur », ces quatre premières directions sont les premières branches de l'arbre des possibilités qui s'en suit.
La fonction « solveur » est une fonction récursive, elle « simule » les déplacements d'une bille dans le niveau avec comme condition d'arrêt les coordonnées de la bille égale aux coordonnées de la case d'arrivée. Tant que cette condition n'est pas validée, elle vérifie chaque direction créant ainsi de plus en plus de branches dans l'arbre, et à chaque déplacement les coordonnées de la bille et de la matrice sont mémorisées dans un set qui permettra d'éviter les doublons de coordonnées et donc éviter une boucle infinie.
Si la case d'arrivée est atteinte, la fonction renvoie « True », sinon « False ».

-**Générateur de niveau :** grâce au solveur, on peut ainsi créer des niveaux aléatoires, cette fonctionnalité est séparée en trois partie.
La première créée un fichier texte dans lequel va être écrit le niveau, avec une hauteur de 0 à 6.
La deuxième est une fonction qui vérifie grâce au solveur si le niveau est réalisable.
Et la troisième utilise les deux premières, tant que le nombre de niveaux souhaité par le joueur n'est pas atteint, elle fait appel à la création d'un niveau et de son fichier texte, et vérifie ensuite si le niveau est réalisable. Si le niveau est faisable, il est enregistré dans une liste qui sera renvoyée à la fin, sinon on recommence le processus.

-**Changement de niveau :** puisqu'on a une liste des niveaux, pour changer de niveau, il suffit de parcourir cette liste avec un indice.

-**Les boutons de menu :** de simples textes encadrés, on récupère les coordonnées des rectangles et il suffit d'attendre un clic, une fois le clic effectué, on regarde s'il est dans un des rectangles.

La répartition du travail et les difficultés rencontrées :

On a assez bien réparti les tâches, Ailton faisait le corps des fonctions, et (Gérald) j'ajoutais les détails qu'il fallait. De manière globale, Ailton s'est plus occupé de l'aspect technique, et je me suis plus occupé de l'aspect graphique du programme.

Cependant, avant qu'on se soit bien organisé pour créer le meilleur programme qu'on puisse faire et du fait que ce projet est le premier qu'on réalise ensemble, on était assez opposé au niveau du code : l'un était beaucoup plus libre et l'autre était bien plus maniaque, en conséquent on a eu beaucoup de mal accepter la façon de programmer de l'autre.

De manière globale dans le programme, on a rencontré des difficultés dans sa structuration, ce qui nous a fait perdre beaucoup de temps. Et des difficultés également sur les grosses fonctions assez technique.