PROJET 1010 L2 INFO G-7

Thibault Odorico – Tristan Ramirez de Ribas – Antoine Gouyon – Alexy Lefevre

I. La grille de jeu avec la pièce déplaçable

I.1) La classe Matrice :

Au début du projet on comptait modéliser la grille du 1010 sous forme de Matrice avec un tableau à 2 dimensions tout simple. ex: int tab[10][10];

Cependant on s'est vite rendu compte qu'il fallait aussi modéliser les pièces du jeu avec des matrices et comme le format des pièces est variant, on a décidé de créer notre propre classe Matrice qui crée une matrice d'entiers de n'importe quel format et peut les enregistrer et les charger facilement dans des fichiers.
(Methodes save et load)

I.2) La classe Pièce :

Pour modéliser nos Pièces on s'est dit qu'une pièce n'était rien d'autre qu'une matrice avec des coordonnées qui varient, ainsi pour créer notre classe Pièce, on lui à fait un héritage des attributs et des méthodes de la classe Matrice. Cela en y rajoutant des attributs de positions avec des méthodes pour les changer.

I.3) La classe Grille :

La classe Grille va modéliser la partie centrale du 1010, qui est la partie de la grille où on bouge la pièce.
Pour nous grâce à nos 2 classes précédentes il était facile de modéliser cette partie car la grille de jeu n'est rien d'autre qu'une matrice avec une pièce à l'intérieur.

On a donc fait hérité la classe Grille des attributs et méthodes de la classe Matrice en y rajoutant un attribut de type Pièce. Des méthodes pour bouger la pièce dans la grille et contrôler son bon comportement ont ensuite étaient rajoutés.

II. Le set de trois pièces qui se réactualise aléatoirement

II.1) La classe SetDePiece :

On a choisi de modéliser cette partie par une classe qui possède en attribut un tableau de 3 Pièces, un indice (entre 0 et 2), le nom du fichier où les pièces de jeux sont stockées. En ce qui concerne les méthodes, une qui permet de récupérer la pièce vers laquelle pointe le SetDePiece. une qui permet de changer la pièce pointée, une autre qui permet de marquer les pièces comme "placées", une dernière qui recharge les pièces à partir du nom du fichier passé en paramètre.

III. Les scores

III.1) La classe Score

Nous avons créé la classe Score pour qu'elle puisse calculer les points accumulés pendant le jeu au fur et à mesure que les pièces sont posées. Pour cela, une méthode calcule les points apportés par le multiplicateur. Deux autres qui vérifient chacune si une ligne ou une colonne est pleine. On s'est rendu compte qu'il fallait modifier la classe Matrice, on a ajouté deux méthodes qui permettent de modifier entièrement les valeurs d'une ligne ou d'une colonne, ici pour les remettre à zéro une fois « pleine ». Au final, tout ceci à été regroupé dans une méthode de la classe Score qui appelle toutes les fonctions décritent.

III.2) Sauvegarde des scores dans un fichiers

Pour mettre à jour le fichier de sauvegarde des scores, on a découpé le fichier en deux parties. La première partie, constituée des 5 premières lignes du fichier qui représentent le classement des 5 meilleurs score, et la seconde partie qui est tout le reste des scores. Après chaque partie il fallait donc vérifier si le joueur avait battu le score d'un des 5 premiers. Après la vérification qui compare le score du joueur à ceux des 5 meilleurs scores, s'il avait battu un des 5, il fallait alors modifier le classement. Ce n'est qu'après plusieurs heures d'essais et de recherches, qu'on s'est rendu compte qu'on ne pouvait pas écrire "au milieu" d'un fichier. En effet, on peut écrire à la fin sans problème, ou alors "au milieu", mais en écrasant des données existantes. Notre idée de départ, qui était d'insérer le score du joueur juste avant celui qu'il venait de battre dans le classement, n'était donc pas la bonne. Ce qu'il a été nécessaire de faire, c'est de :
- recopier les lignes des scores du classement jusqu'à la position (1 à 5) du nouveau score du joueur dans un fichier temporaire
- écrire le nouveau score à sa place dans le fichier temporaire
- recopier le reste du fichier dans le fichier temporaire à la suite du classement
- renommer le fichier temporaire pour qu'il devienne le nouveau fichier de sauvegarde des scores
C'est donc grâce à cette technique, qui consiste à recopier ce qui a avant dans un nouveau fichier, insérer notre ligne, et recopier ce qu'il y a après, qu'on a réussi à garder ce fichier de sauvegarde à jour.

III.3) Affichage des scores

Pour pouvoir afficher les score, nous avons créé une fonction qui va lire les 5 premières lignes du fichier des scores, et mettre chacune des lignes dans un tableau de string. Puis, au final, pour afficher les scores, il suffisait de faire un print de chacune des cases du tableau de string et de déterminer les coordonnées où on voulait les afficher dans la matrice. Cette fonction print était donné dans la classe Window.

IV. Le traitement des options pris en paramètre par le jeu

IV.1) La fonction lireOptions

Pour gérer les différentes options demandées on a d'abord codé une fonction lireOption qui permettait de reconnaître les différents paramètres possibles et d'y répondre spécifiquement :
- Soit par de simple sortie standard pour l'affichage des noms et de la version.
- Soit pour l'affichage de l'aide, étant un texte plus conséquent, récupérer le contenu d'un fichier et l'afficher dans le terminal afin de faciliter la rédaction et les modifications de l'aide du jeu au cours du développement.

Mais on n'avait pas encore prévu que l'option pour inclure un nouveau set de pièce nécessitait un paramètre. C'est pourquoi nous l'avons défini en dehors de la fonction lireOption avec des conditions spécifiques.

V. La classe TenTen qui reprend tout

La classe TenTen est la classe qui va modéliser le jeu entier, elle reprend donc tout notre travail et possède en attribut, une Grille, un setDePiece, un Score, le nom du joueur demandé en début de partie, et enfin le nombre d'action effectuées pour pouvoir revenir en arrière.

V.1) Fonction d'affichage des classes

Des fonctions d'affichage on étaient développées pour chacune de ces classes pour pouvoir tout afficher dans des fenêtres facilement.

Remarque : On a fait en sorte d'afficher tout sous forme carrée, on affiche à chaque fois sous le format : 2 cellules de terminal en largeur pour 1 en hauteur. A plus ou moins grande échelle selon les paramètres passés dans les fonctions.

V.2) La boucle de jeux

La boucle principale de jeux se trouve dans la méthode run() et fait appel à toutes les fonctions principales développées pour le 1010 :
- Contrôle de la Pièce dans la Grille.
- Actualisation du SetDePiece.
- Retour en arrière grâce aux fonctions de sauvegarde.
- Actualisation du score.
- Etc..

La méthode play() permet quant à elle de lancer la boucle principale en mode Ncurse ainsi que de charger et sauvegarder des parties.

V.3) Sauvegarder et reprendre une partie

Depuis le début de notre projet on a développé des méthodes de sauvegarde et de chargement pour toutes nos classes. Ainsi pour sauvegarder notre jeu de 1010 on a simplement développé une nouvelle méthode qui fait un appel à toutes les méthodes de sauvegardes des objets et on a appliqué le même principe pour le chargement.