58434 - Lukasz Matusewicz (D212)
58183 - Ayoub Nedjar (D112)

Rapport Projet DEV4 -Tetris

Diagramme de classes

La classe qui contrôlera notre partie sera la classe Game . Lorsque celle- ci est créée , la partie est lancée . Dans cette classe nous retrouvons la brique qui est actuellement jouée ainsi que sa position sur le tableau , l'état de la partie et le score actuel de la personne qui joue .
Dans cette classe nous allons aussi trouvé différentes méthodes qui nous permettront de vérifier que la brique actuellement jouée ne sors pas de la dimension du tableau .
Nous pourrons aussi manipuler cette brique en la déplaçant sur le tableau ou la tourner dans les sens horloger ou anti-horloger .
La position de cette pièce est en fait le centre de la brique que l'on voudra manipuler . En fonction de sa forme , l'on regardera dans les directions (bas , gauche , droite) s'il n'y a pas d'obstacle qui pourrait bloquer cette pièce .
On peut aussi retrouver les getter du board pour que l'on puisse savoir quel endroit de ce tableau est occupé pour savoir quand une brique serra posée et quand est-ce qu'on doit passer à la prochaine brique .
Sachant que le jeu se déroule jusqu'à un certain moment de la partie soit le joueur perd , soit il atteint l'objectif nécessaire pour gagner . Une méthode serra implémentée pour qu'elle puisse gérer cette fin de partie .
Pour mettre à jour le score du joueur , une méthode ferra le calcul nécessaire pour modifier ce score .

Différentes classes du model :

Brique :

La classe Brique est de type abstract car l'on aura plusieurs sous-classe qui représenteront les briques jouables . Cette classe comportera un vector<vector<bool>> qui représente la forme que la brique aura ainsi qu'une Position centre qui nous donnera un point de repère pour la classe Game pour savoir si une position dans le board est occupée ou pas.
Cette classe aura aussi une méthode virtuelle rotate(Direction d) qui nous permettra de tourner la matrice de notre vector<vector<bool>> . Cette méthode serra pareille pour toutes les briques .
On a décidé de créer un système d'héritage afin que chaque sous-classe ait ses propres données . C'est-à-dire que chaque vector<vector<bool>> serra dans son propre fichier .

58434 - Lukasz Matusewicz (D212)
58183 - Ayoub Nedjar (D112)

Bag :

La classe Bag aura comme attribut un vecteur de Brique qui serra donc l'ordre dans lequel les Briques sont joués .
Cette classe aura un constructeur :
Un constructeur par défaut qui créera le jeu avec les briques standards .

La classe a aussi plusieurs méthode tel que :
-Shuffle() qui mélange le Bag afin que les briques ne tombent pas sur le board tout le temps dans le même ordre .
-getBag() qui est le getter du Bag .
-nextShape() qui permet de passer à la prochaine brique dans le Bag . Elle enlèvera la première brique de ce vecteur afin que la prochaine soit la première .

Board :

La classe board est le tableau où sont posées les briques . Elle est faite d'une taille , d'une largeur et d'un vector<vector<CaseType>> .
On a décidé d'utiliser un vector de vector d'énumération afin de savoir quel brique l'on pose sur le board ainsi que de savoir si la case est vide ou pas .
La taille et la largeur sont les dimensions qu'aura notre board .
Dans les constructeurs l'utilisateur aura le choix soit de jouer avec le board par défaut soit avec les dimensions qu'il aura choisi .
Elle aura aussi :

- Des getter de ses attributs .
- Une méthode pour savoir si la brique est dans le board .
- Une méthode pour vérifier la ligne si elle a été complétée .
- Une méthode qui supprimera les lignes complétées .

CheckRules :

C'est une classe qu'on a créé afin de ne pas surcharger la classe Game . Elle contiendra les conditions de victoire du joueur ainsi que des méthodes qui vont vérifier si ces conditions ont été atteintes .

Level :

Cette classe gérera les niveaux du jeu et serra celle qui pourra augmenter la vitesse à laquelle les briques défileront vers le bas .

58434 - Lukasz Matusewicz (D212)
58183 - Ayoub Nedjar (D112)

Différentes Énumération du model :

Direction :

Cette énumération a été créée pour que le joueur puisse choisir dans quelle direction la brique pourra bouger sur le board . On l'utilisera aussi pour savoir à quel moment la brique est bloquée sur le board .

SensRotation :

Cette énumération sert à choisir dans quel sens la brique pourra être tournée (horloger / anti-horloger)

State :

Cette énumération nous permet de savoir dans quelle situation se trouve le joueur car l'on retrouve différent état du jeu (entrain de jouer , fini , en pause) et si l'on nous demande de rajouter des états de jeu ou pourra le faire facilement dans cette énumération .

CaseType :

Cette énumération nous permet de savoir sur le tableau quel type de brique a été utilisé pour pouvoir implémenter par la suite par exemple des couleurs ou n'importe quel attribut d'une brique .

Position :

Cette énumération simplifie le fait de savoir à quel position se trouve le centre d'une brique et avec l'aide de l'énumération Direction on aura des calculs simplifiés pour la vérification des collisions .

Diagramme de séquences

Lorsque le joueur entre une direction dans laquelle il veut que la brique bouge , le controller va envoyer la direction souhaitée par le joueur . On ferra appel à la méthode contains(Position p) pour savoir si la position souhaitée est toujours contenue dans le board et envoie une réponse pour savoir si oui ou non elle est toujours dans le board .
Ensuite il y a un appel de méthode isOccupied(Direction) qui nous enverra une réponse pour savoir si une brique se trouve dans la direction demandée . Si les 2 méthodes d'avant n'ont pas retourné d'erreur , on ferra appel à la méthode verifCompleteLines() qui va nous dire si toute une ligne a été créer , si oui la ligne sera supprimer .
Une fois tout cela fini , on ferra appel aux méthodes qui vérifient si les conditions de fin de jeux sont atteintes . Dans le cas où elles ne sont pas atteintes , l'appel à la méthode nextShape() est exécuté et cette boucle recommence .