L2 INFORMATIQUE
PROJET POO : 2020
Bouanem Yani Akli /
Le Franc Matthieu
Groupe : 25

Table des matières

Description du projet

Développé par :

- Le Franc Matthieu, N°Etudiant : 71800858, e-mail : matthieu.le-franc@etu.u-paris.fr

- Bouanem Yani Akli, N°Etudiant : 21962029, e-mail : yani-akli.bouanem@etu.u-paris.fr

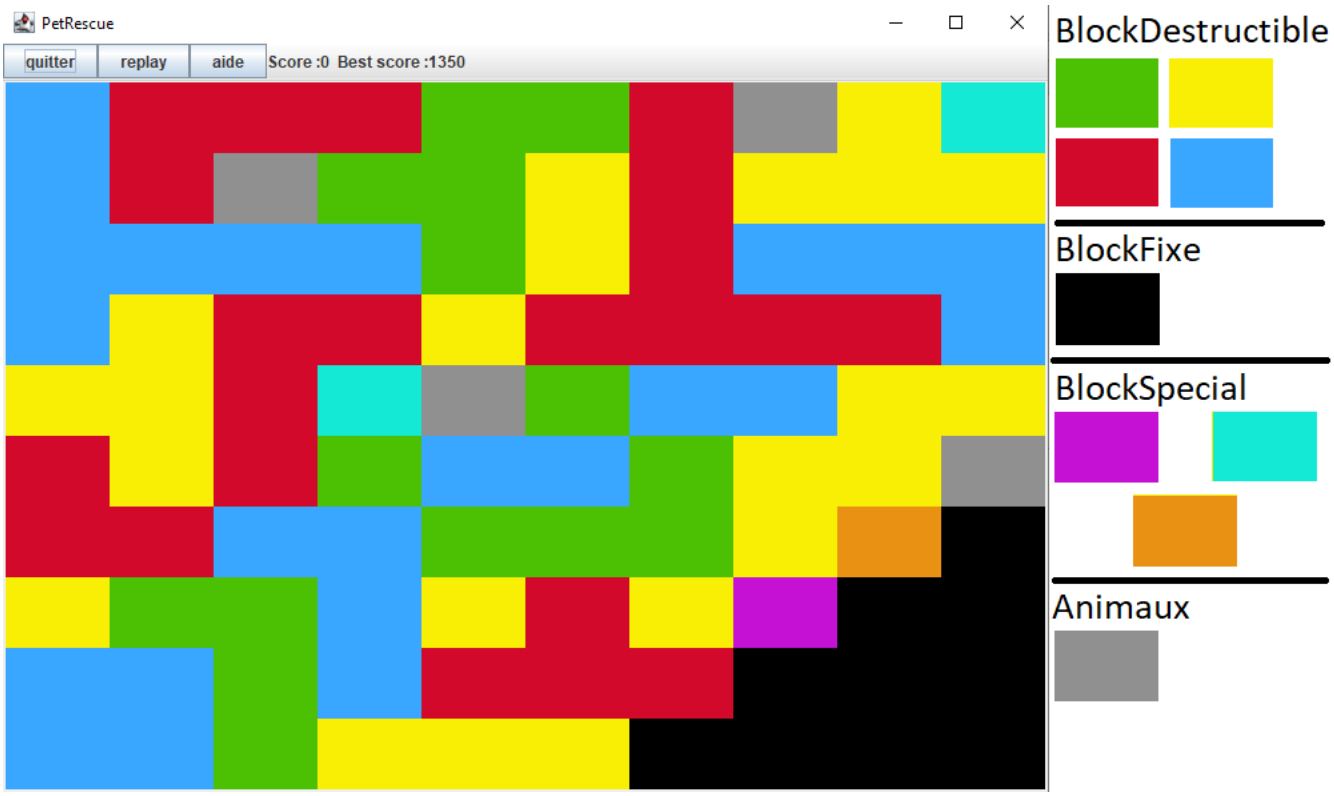
Mini jeu s'inspirant du principe de **Pet Rescue Saga**. A travers différents niveaux, le joueur devra ramener tous les bloques de types *Animaux* (représentés par des bloques gris) au bas du tableau. Il n'y a pas de réelles possibilités d'échec, la bonne réussite d'un niveau se réalise en fonction du score une fois le niveau complété.

Dépot github du projet : https://github.com/SirHenryAllen/PROJET_POOIG_2020.git

Université de Paris

L2 INFORMATIQUE
PROJET POO : 2020
Bouanem Yani Akli /
Le Franc Matthieu
Groupe : 25

Les différents types de bloques :



Les BlockDestructible : en vert, jaune, rouge et bleu, ces bloques se détruisent lorsqu'on clique dessus en entraînant la destruction de leurs voisins s'ils sont du même type (**BlockDestructible**) et de la même couleur.

Les BlockFixe : sont des bloques indestructibles et indéplaçables.

Les BlockSpecial : en violet, bleu ciel et orange, permettent d'interagir et de provoquer la destruction de respectivement : leurs bloques adjacents et bloques en diagonales, leur colonne, leur ligne ; à condition que les les bloques à détruire soient du type **BlockDestructible**.

Les Animaux : en gris, sont les bloques dont la position est évaluée pour confirmer la fin du niveau. Lorsqu'un bloque de type **Animaux** est évaluée à la position « sol » du tableau ([20][x]), le bloque est supprimé. Une fois tous les bloques **Animaux** supprimés, le jeu s'arrête.

Les null : éléments vide du tableau représentés en blanc illustre lorsqu'une case est inoccupée par un des types de bloques décrits plus tôt.

Les règles et mécaniques du jeu :

Le déplacement des bloques : Lorsqu'un bloque (à l'exeption du type **BlockFixe**) se retrouve au-dessus d'un **null** ou emplacement vide (en blanc), il chute jusqu'à se placer au-dessus d'un autre bloque.

Dans le même principe, lorsqu'une colonne est inoccupée, la clone de bloques à sa droite vient remplir cette espace manquant en entraînant toutes ses voisines de droite. NB : une colonne de **BlockFixe** est considérée comme vide et entraîne donc un décalage à gauche.

Le score : A chaque bloque détruit, 20 points sont attribués au joueur (plus on détruit de gros groupes de bloques, plus les points augmentent) et lorsqu'un animal atteint le sol et disparaît, 50 points sont attribués.

L2 INFORMATIQUE
PROJET POO : 2020
Bouanem Yani Akli /
Le Franc Matthieu
Groupe : 25

Le plateau :

```
######################
Plateau format développeur
######################

    |  A  B  C  D  E  F  G  H  I  J  K  L
 0  |  #  #  #  #  #  #  #  #  #  #  #  #
 1  |  #  .  .  .  .  .  .  .  .  .  .  #
 2  |  #  .  .  .  .  .  .  .  .  .  .  #
 3  |  #  .  .  .  .  .  .  .  .  .  .  #
 4  |  #  .  .  .  .  .  .  .  .  .  .  #
 5  |  #  .  .  .  .  .  .  .  .  .  .  #
 6  |  #  .  .  .  .  .  .  .  .  .  .  #
 7  |  #  .  .  .  .  .  .  .  .  .  .  #
 8  |  #  .  .  .  .  .  .  .  a  .  .  #
 9  |  #  .  .  .  .  .  b  .  d  .  .  #
10  |  #  .  .  .  d  .  c  a  b  .  .  #
11  |  #  .  .  .  b  .  a  c  a  .  .  #
12  |  #  .  b  .  b  .  +  b  c  .  .  #
13  |  #  d  a  a  d  .  d  d  d  .  .  #
14  |  #  b  a  a  d  d  c  a  d  .  .  #
15  |  #  c  c  a  b  a  b  b  b  .  .  #
16  |  #  a  a  +  a  b  b  d  d  .  .  #
17  |  #  a  b  a  a  c  c  a  d  .  .  #
18  |  #  b  d  b  b  b  b  +  b  .  .  #
19  |  #  a  c  b  a  +  +  a  d  .  .  #
20  |  #  c  b  b  b  d  b  c  a  .  .  #
21  |  #  #  #  #  #  #  #  #  #  #  #  #
```

On peut voir ici à quoi ressemble le plateau de jeu dans son intégralité. Seules les cases comprise [10 ; 20] & [B ; K] sont visibles par l'utilisateur sur la version graphique du jeu.

Le contour du tableau (représenté par les #) est remplit de BlockFixe. Ainsi, lorsque l'on cherche à parcourir les éléments, on évite les mauvaises surprises et risques de sortir du tableau

Pour chaque niveau, nos différents bloques sont rangés dans un tableau de taille [22][12]. Cela permet d'ajouter des bloques au-dessus de la zone visible par le joueur et donner l'illusion que des bloques apparaissent lorsque ceux d'en-dessous sont détruits.

Les niveaux :

Quatre niveaux ont été créés pour tester notre jeu. Vous pouvez retrouver leur design dans le dossier *Annexe/Niveau*

L2 INFORMATIQUE
PROJET POO : 2020
Bouanem Yani Akli /
Le Franc Matthieu
Groupe : 25

Les considérations techniques

Les packages :

Nous avons décidé de créer et utiliser 5 principaux packages :

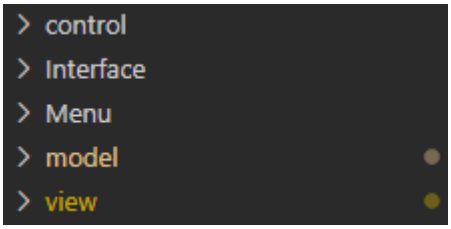
- control

- Interface

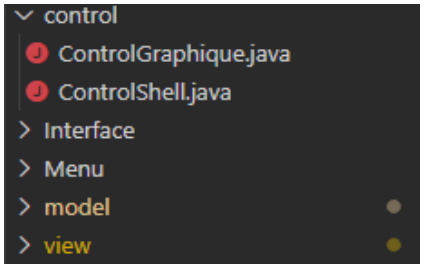
- Menu

- model

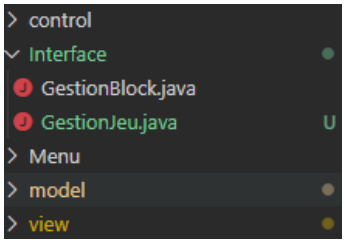
- view



control : regroupe toutes les classes permettant l'interaction avec le jeu en mode graphique ou shell.

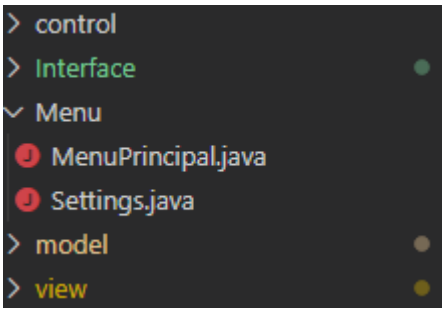


Interface : regroupe deux interfaces comportant les fonctions vitales au fonctionnement du projet

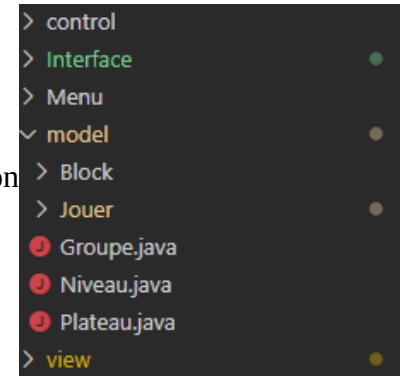


L2 INFORMATIQUE
PROJET POO : 2020
Bouanem Yani Akli /
Le Franc Matthieu
Groupe : 25

Menu : Menu est un peu à part du reste du projet car cette partie contient la fonction *main* du programme ainsi que l'interface graphique permettant de choisir son mode de jeu ainsi que son niveau. Une exception dans l'architecture du projet est accordée à Menu car ses classes contiennent des méthodes permettant à la fois de l'affichage graphique du control.

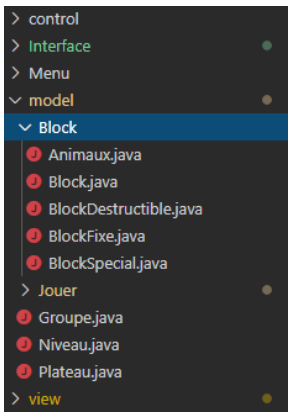


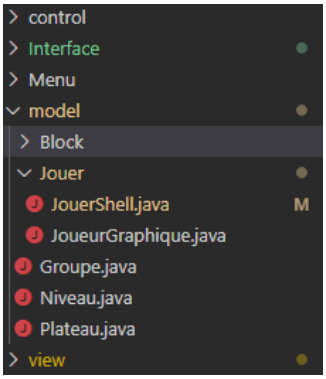
model : ce package contient toutes les classes permettant de faire fonctionner le système du jeu (la construction de la grille de jeu, des niveaux, la suppression et le déplacement des différents bloques…) ainsi que deux sous packages : **Block** & **Jouer**.



Block : contient toutes les classes des différents bloques

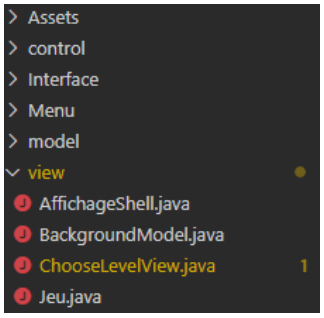
Jouer : contient les classes pour lancer le déroulement d'un niveau sur shell ou interface graphique





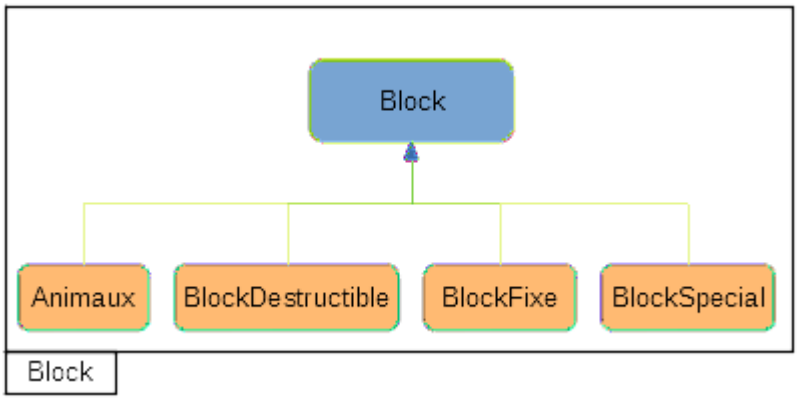
L2 INFORMATIQUE
PROJET POO : 2020
Bouanem Yani Akli /
Le Franc Matthieu
Groupe : 25

view : englobe toutes les classes s'occupant de la vue du jeu



Les différentes classes :

Descriptif de l'héritage de Block :



Les classes principales :

- model.Plateau : la classe **Plateau** contient toutes les fonctions nécessaires à la gestion de la grille de jeu, avec :

  1. la destruction des **Block**

  2. le déplacement des **Block** (chute, glissement à gauche)
  3. l'ajout des **Block**
  4. le traitement des informations affichées sur l'écran de jeu (comme le score)

- model.Niveau : permet la création de chaque **Niveau** avec soit une attribution manuelle de chaque **Block** à une case du tableau, soit une génération aléatoire du Niveau.
- view.Jeu : s'occupe de l'affichage graphique de la grille de jeu et de son actualisation. Les principales fonctions de cette classe vont venir évaluer l'état des éléments du tableau **Block[][]** (dans la classe model.Plateau) et générer l'affichage de carrés de couleurs en fonction du type des **Block** lu à chaque case. NB : à chaque actualisation du tableau **Carre[][]**, on vient repeindre les carrés et non les déplacer.
- view.AffichageShell : permet l'affichage de la grille de jeu complète (**Block[][]**) sur console de commande.
- Menu.Settings : classe contenant l'entrée du programme, la fonction main et générant les instances élémentaires pour le fonctionnement du jeu.
- Control.ControlShell : permet d'interagir avec la grille de jeu (va de paire avec l'affichage shell) et de compléter un niveau sans affichage graphique. Cette classe est l'une des première à avoir été développée, elle permettait de faire les premiers tests des fonction de **Plateau**, avant l'implémentation de l'interface graphique.
- Control.ControlGraphique : cette classe a pour but de gérer les inputs de la souris afin de commander des actions sur la grille de jeu (**Block[][]**). Elle englobe une sous-classe implémentant l'interface **MouseInputListener**.

Pistes d'améliorations :

Fonctionnalités de notre feuille de route, non implémentées :

Il y a un certain nombre d'autres options que nous aurions aimés implémenté, mais, prit par le temps, nous avons dû nous résoudre à les abandonner.

Replay : Un bouton recommencer sur chaque niveau afin de relancer la partie si le joueur le souhaite.

Des images : A la place des carrés de couleurs, il aurait en effet été intéressant de disposer des images afin d'avoir un affichage plus digeste.

Un robot : Nous voulions aussi donner la possibilité de compléter un niveau entièrement grâce à l'ordinateur. Il aurait simplement fallu créer une classe **Robot** dans le package *control* qui aurait fait appel aux différentes fonctions vitales pour jouer le niveau.

Idées de fonctionnalités supplémentaires :

Quoi que pouvant paraître superflu, on aurait également pu réfléchir à des moyens pour ajouter des bruitages à la destructions des différents bloques ou encore des animations.