Projet du Same Game

Table des matières

1. Présentation du projet

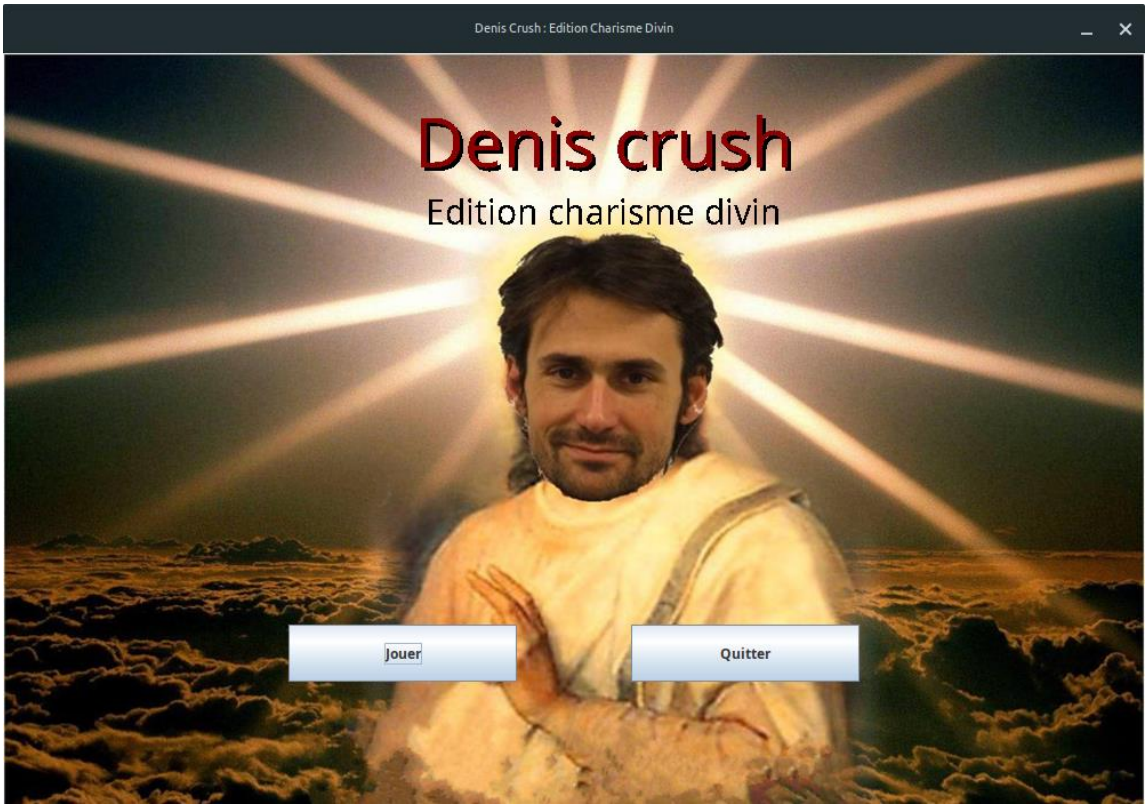
Le projet du Same Game est un jeu réalisé en Java, où l'on doit faire le plus haut score en détruisant le plus grand nombre de blocs d'un seul coup, le jeu se finit lorsqu'il n'y a plus de groupe à détruire, c'est-à-dire qu'il ne reste plus de blocs ou alors que les blocs restants sont isolés des autres au niveau de leur ressemblance. Lorsqu'un groupe de blocs a été détruit, une gravité de haut vers le bas est appliquée, de plus s'il y a une colonne vide à gauche de blocs alors la ligne de blocs sera déplacée vers
la gauche pour remplir le vide en place. Le jeu ne se fait qu'à la souris sur une grille 10 x 15 (10 lignes, 15 colonnes). Ce projet doit fonctionner avec un Makefile, dont avec la commande « make » on compile les fichiers et la commande « make run » pour lancer le programme. Le joueur, avant le début
de la partie, a le choix de choisir une grille aléatoire ou une grille qui existe dans un fichier dans le format suivant :

RRRRRRRRRRRRRR
RRRRRRRRRRRRRR
RRRRRRRRRRRRRR
RRRRRRRRRRRRRR
RRRRRRRRRRRRRR
RRRRRRRRRRRRRR
RRRRRRRRRRRRRR
RRRRRRRRRRRRRR
RRRRRRRRRRRRRR
RRRRRRRRRRRRRR

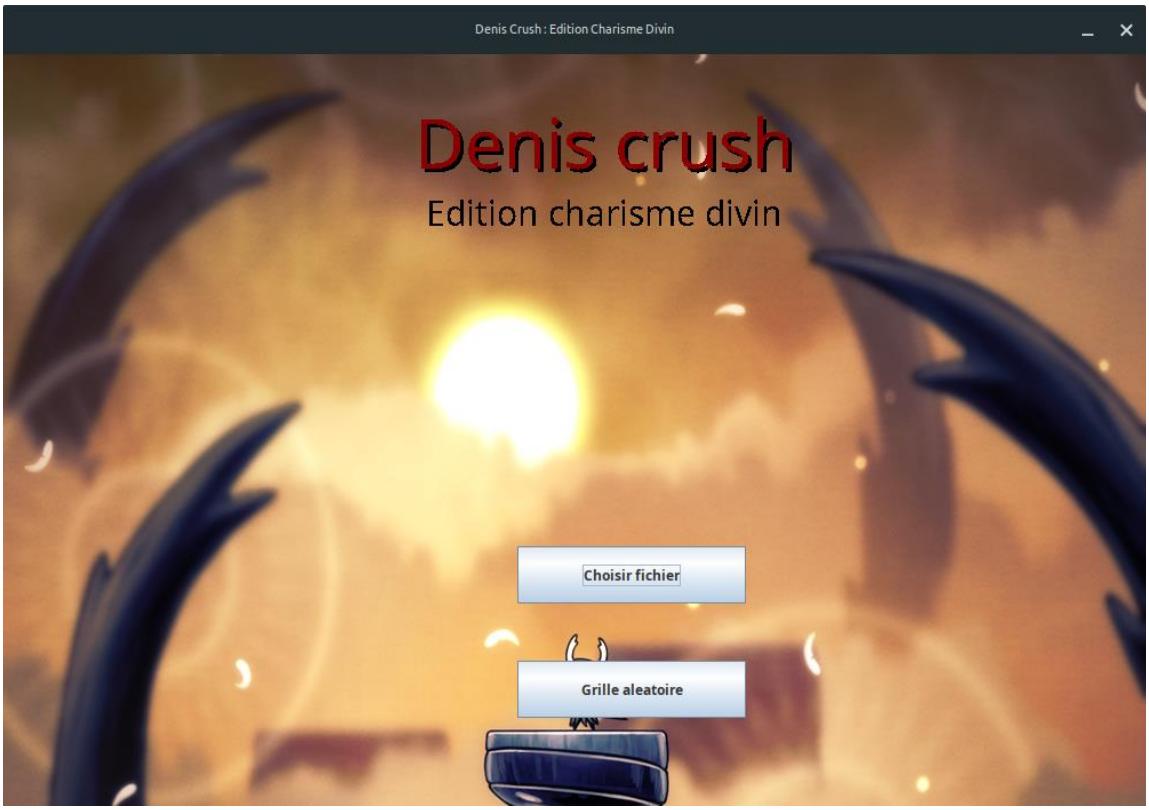
R ou V ou B pour choisir entre les différents blocs.

2. Présentation des fonctionnalités du programme

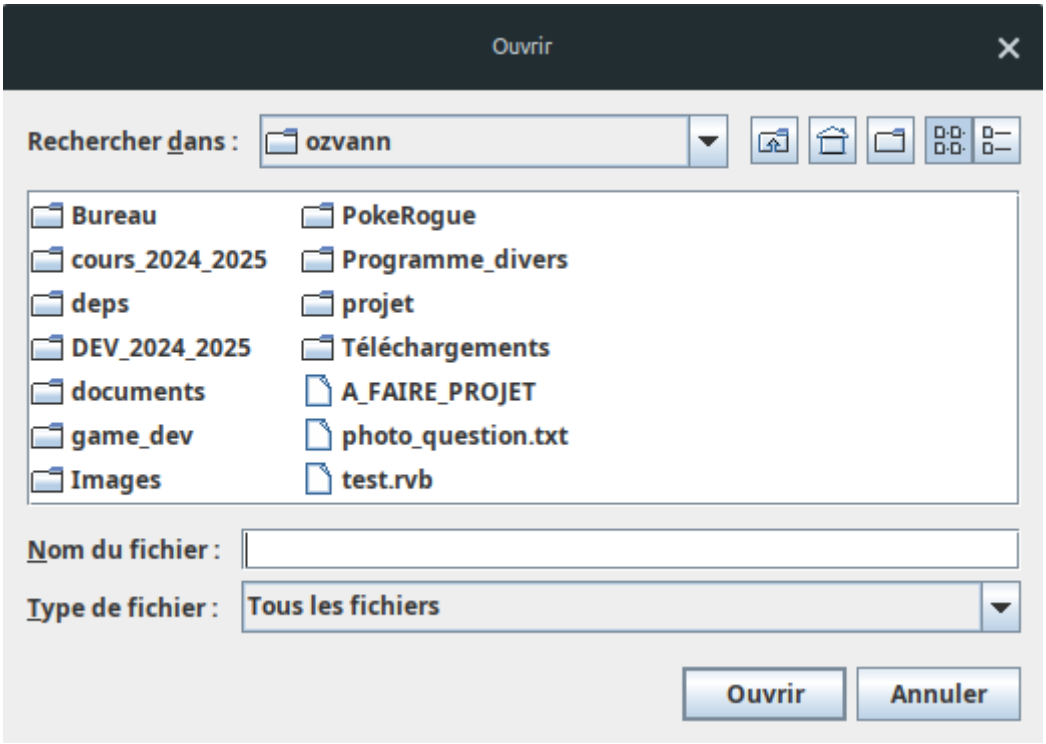
Au lancement du programme, on peut choisir entre jouer ou quitter :
Denis Monnerat est sur cette image et sur plusieurs autres images, donc nous devons préciser que nous lui avons demandé son accord et qu'il a dit que ça ne lui posait pas de problème d'être présent sur notre jeu.



Une fois que nous avons choisi de jouer, on a le choix de soit utiliser une grille aléatoire soit utiliser un fichier existant :

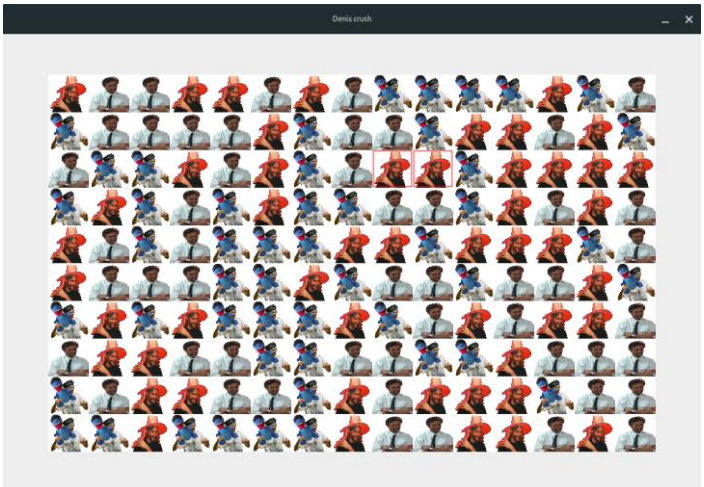


Si le joueur choisit de jouer avec une grille aléatoire alors le jeu se lancera immédiatement, mais si le joueur décide de choisir une grille alors une fenêtre s'ouvre pour qu'il choisisse le fichier :

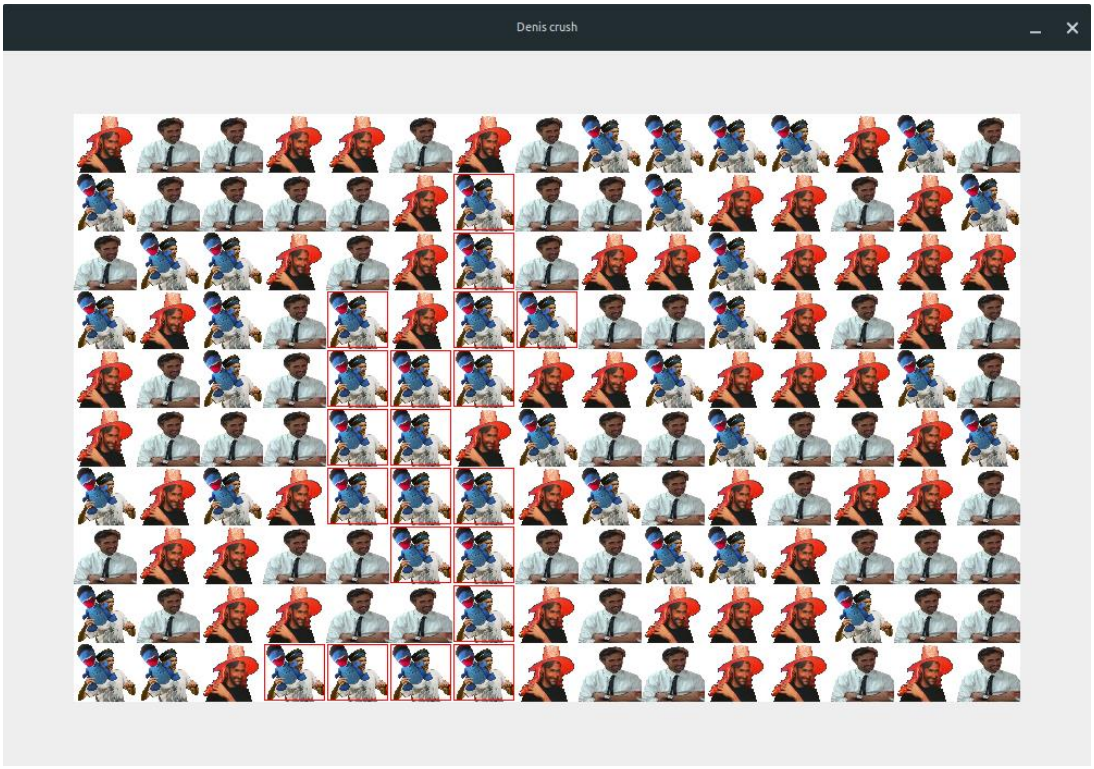


Si le fichier n'a pas le bon format, un message d'erreur apparaît :

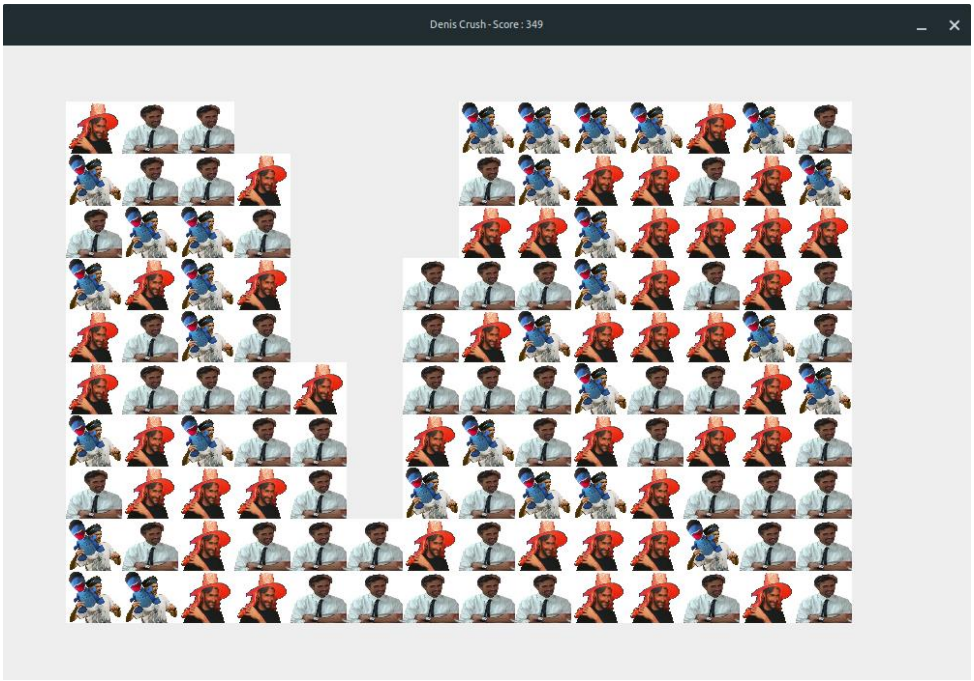
Une fois que le joueur a choisi un fichier correct avec le bon format alors le jeu se lance :



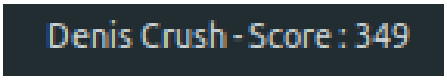
On peut sélectionner un groupe d'images :



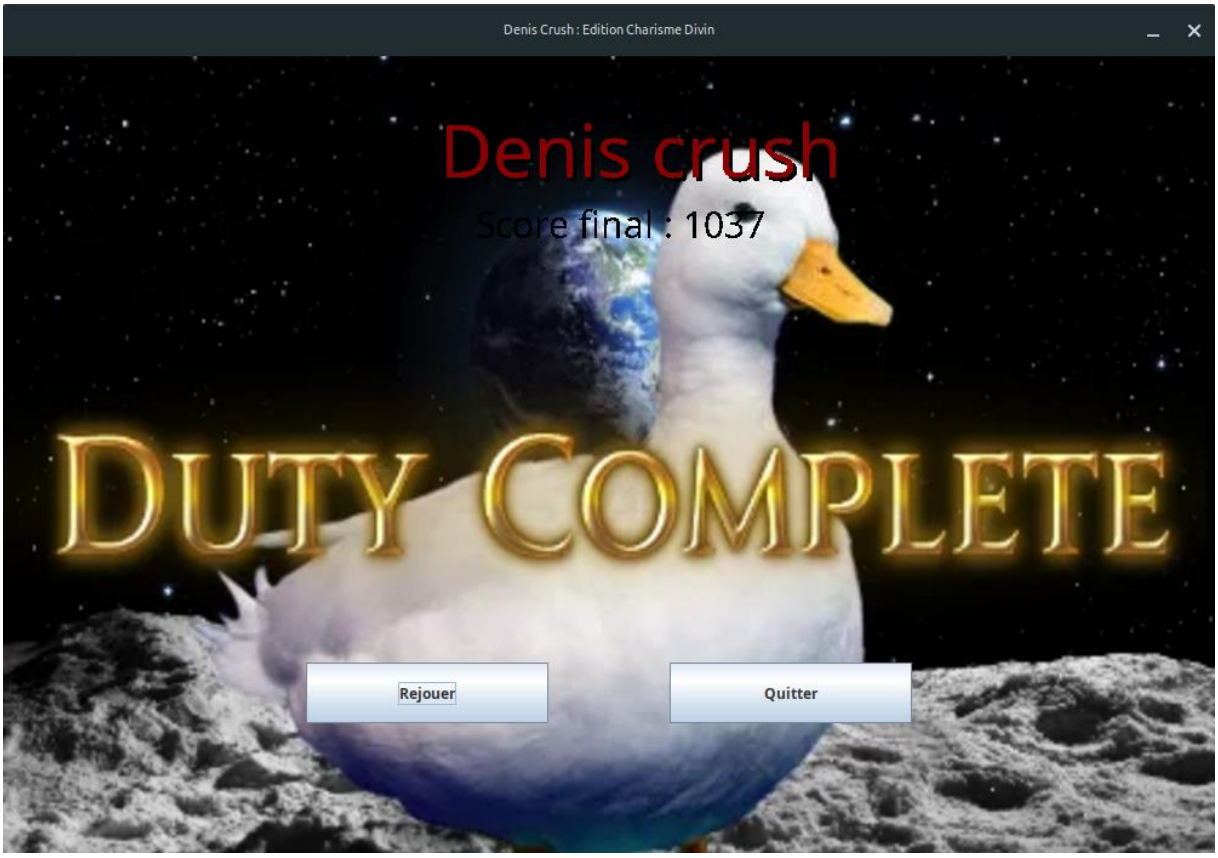
Et si le groupe est cliqué et que le groupe contient plus de 2 cases, alors ils sont détruits et une gravité est appliquée, si une colonne est détruite alors la colonne vide disparaît et les images se déplacent vers la gauche :



Quand les blocs sont détruits, le score est mis à jour dans le titre de la fenêtre :



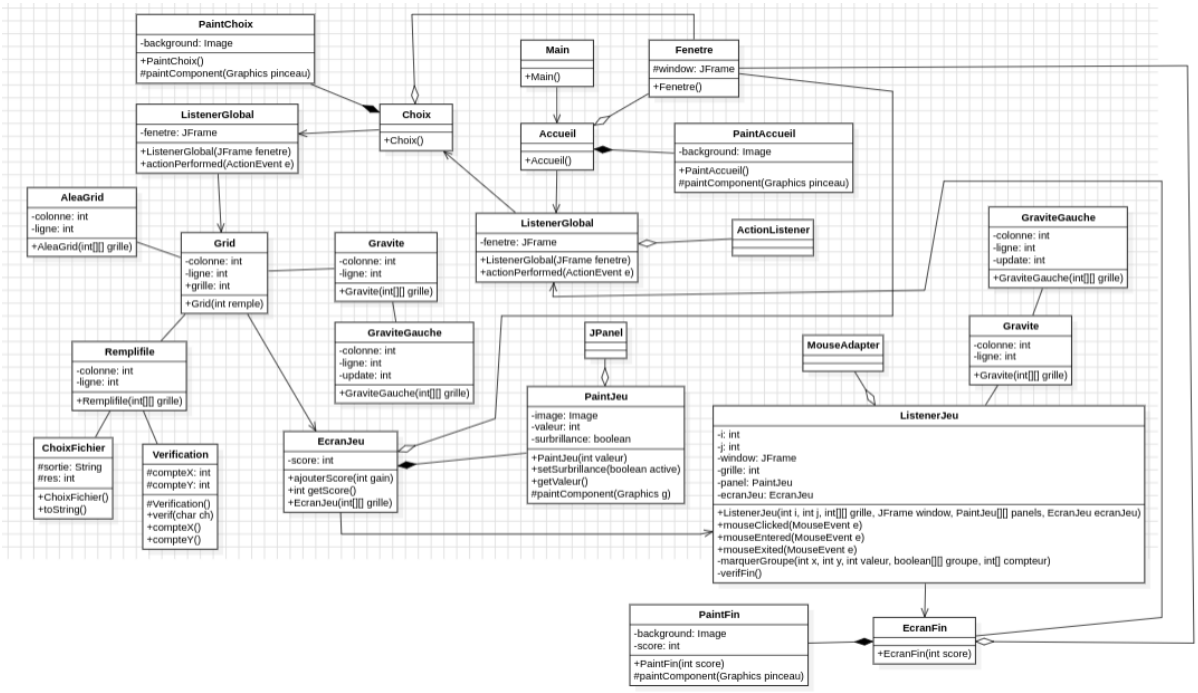
La partie se termine une fois qu'il n'y a plus de groupe à détruire ou qu'il ne reste que des images qui sont seules, il y aura alors la fenêtre de fin :



Selon le score, l'image change et, depuis cet écran, vous pouvez soit quitter le jeu, soit rejouer. Vous avez évidemment le score final que vous avez eu. Bonne partie.

3. Le Diagramme de classe

Pour ce projet, nous avons fait un diagramme de classe représentant l'interaction entre les différents fichiers. Le fichier *Main* appelle *Accueil*. *Accueil* dépend de *Fenetre* et *PaintAccueil*. *Accueil* et *PaintAccueil* ne peuvent exister sans l'autre. *Fenetre*, quant à lui, existe pour créer et gérer une fenêtre, donc est indépendant, mais *Accueil* existe car *Fenetre* existe. *Accueil* appelle *ListenerGlobal*. *ListenerGlobal* étend *ActionListener*. *ListenerGlobal* appelle *Choix*. *Choix* dépend, comme *Accueil*, de *Fenetre*. *Choix* existe grâce à *PaintChoix* et *PaintChoix* existe grâce à *Choix*. *Choix* appelle *ListenerGlobal*, qui lui appelle *Grid*. *Grid* communique avec *AleaGrid*, *RempliFile*, *Gravite*. *RempliFile* communique, lui, avec *ChoixFichier* et *Verification*. *Gravite* communique avec *GraviteGauche*. *Grid* appelle *EcranJeu*. *EcranJeu* existe grâce à *PaintJeu*, comme pour *Accueil* avec *PaintAccueil*. *EcranJeu* fait la même chose avec *Fenetre*. *EcranJeu* appelle *ListenerJeu*. *ListenerJeu* existe grâce à *MouseAdapter*. *ListenerJeu* communique avec *Gravite*, qui lui communique avec *GraviteGauche*. *ListenerJeu* appelle ensuite *EcranFin*. *EcranFin* existe grâce à *PaintFin* et *Fenetre*, comme *Accueil*. *EcranFin* communique ensuite avec *ListenerGlobal* pour refaire une partie ou arrêter en fonction du choix utilisateur :



4. **Exposition de l'algorithme qui identifie les groupes**

Quand on clique sur un élément dans la grille, l'algorithme parcourt récursivement ses voisins (haut, bas, gauche, droite) pour trouver tous ceux qui ont le même identifiant, formant ainsi un groupe connecté. Une fois le groupe détecté, sa taille détermine le score (score = (nombre d'éléments - 2)²), puis ces cases sont supprimées (mises à 0) tout en gardant les emplacements déjà visités en mémoire. Ensuite, la gravité fait tomber les éléments restants pour combler les vides. S'il y a une colonne vide, la colonne vide se retrouve à droite et les colonnes entre elle et la droite se retrouvent poussées vers la gauche.

5. Conclusions des auteurs

Rayan BISSON :

Pour ma part, ce projet fut notablement meilleur que le premier, notamment par sa qualité et par ma facilité à le faire, ce qui doit sûrement être dû au fait que mon camarade a été d'une gigantesque aide. Je dirais même qu'il a été le pilier de ce projet, et que je n'ai été que d'assistance (il s'occupait plutôt du back-end et moi du front-end principalement). Malgré la fierté que je ressens en voyant ce projet, je ne suis toujours pas satisfait de mon travail. Je me dis toujours que j'aurais pu en faire plus et que je pourrais mieux comprendre. Je me suis dit que c'était bien de faire le front-end, mais la partie la plus ardue pour moi reste le back-end. C'est pourquoi j'aimerais énormément refaire des projets de ce genre pour que je puisse vraiment être un développeur polyvalent, et dans l'espoir qu'un jour je n'aie aucun regret sur le travail que je puisse faire. Ce projet est encore pour moi un pas de plus dans ma connaissance de l'informatique, mais avant tout dans la connaissance de ma personne et de ma manière de travailler. J'ai l'espoir de repartir sur de bonnes bases, mais évidemment, cela ne tient qu'à moi.

Merci encore pour ce projet.

Ozvann ABRAHAM :

Ce projet du Same Game m’a vendu du rêve et je n’ai pas été déçu. Ce qui me rendait triste dans le projet du Blocus était la limite de personnalisation via la bibliothèque graphique de l’IUT, mais pour la Java, la personnalisation était beaucoup plus fluide. La répartition des tâches lors de ce projet a été faite conjointement avec mon camarade, la communication et l’entraide durant ce projet ont été une bouchée d’air frais comparé au projet du Blocus. Le plus gros défi de ce projet, en somme, était de ne pas voir trop haut. Pour ma part, au début, je voulais en faire toujours plus : un tableau des scores, différents modes de jeu, etc. Cependant, par la limite de temps, on a dû couper court à ces idées-ci. Je ne serais pas contre d’autres projets si c’est fait avec des camarades comme Rayan, qui est bosseur et investi dans le projet. Je le remercie donc chaleureusement pour sa participation et son aide dans la partie où j’ai le plus de mal : le back-end.