Maxime ELIOT
Lucas POT
Informatique

Groupe TP3
BUT 1

# Rapport
## SAE 2.1

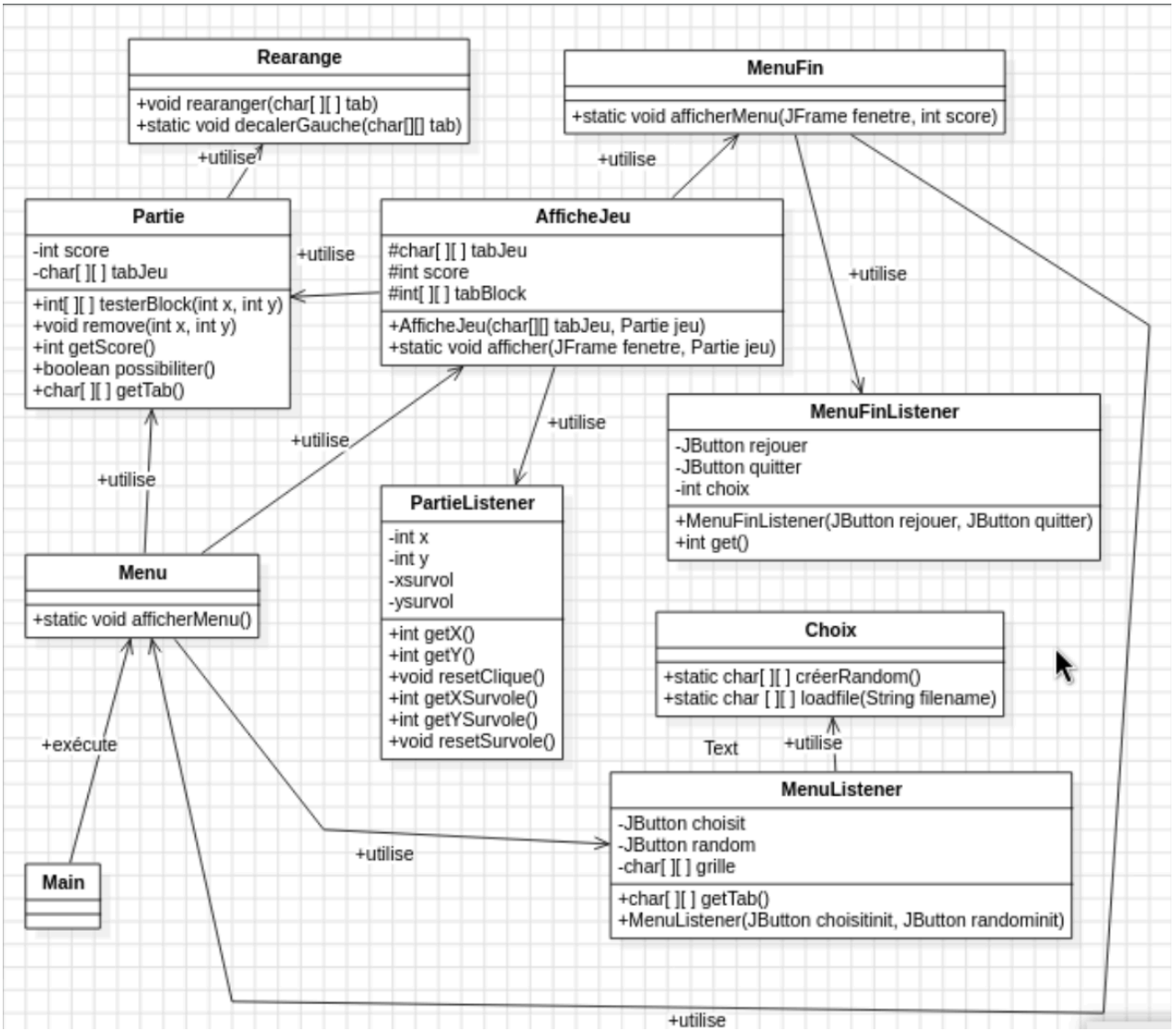
## Introduction :

Au cours de notre second semestre de BUT Informatique, nous avons réalisé un jeu programmé entièrement en Java. Ce jeu devait respecter les règles du SameGame.
Lorsque l'on exécute le programme, un écran d'accueille apparaît pour nous demander si l'on souhaite importer une grille depuis un fichier ou si on veut la générer aléatoirement. Suite à cela, une grille de jeu de 10 lignes et 15 colonnes est affichée. Chaque case est remplie soit de vert, soit de rouge ou de bleu. Lorsqu'au moins deux cases adjacentes sont de la même couleur, on peut cliquer sur le groupe de la couleur pour les supprimer. Tant qu'il existe des groupes, la partie continue. Le but du jeu est de faire le plus gros score possible sachant que plus le groupe supprimé est gros, plus il apporte de points. Il faut aussi noter que lorsqu'une ligne se vide, toutes les lignes du dessus descendent par effet de gravité. Lorsqu'une colonne est vide, toutes les colonnes à sa droite se déplacent vers la gauche.

# Tables des matières

# I - Organisation du code et fonctionnalités

## 1) Diagramme de classes :



## 2) Fonctionnalités :

Notre jeu comporte différentes fonctionnalités. Tout d'abord nous avons un menu d'accueil qui présente les "créateurs" du jeu (du moins ceux qui l'ont codé) et comporte deux boutons. Le premier bouton, appelé "Grille importée", utilise Jfilechooser pour permettre aux utilisateurs de générer une partie depuis un fichier. Il se place par défaut dans le répertoire "Pattern" qui est un sous-répertoire de celui où vous trouverez le makefile. Toutes les extensions de fichier sont acceptées mais le fichier doit respecter le format suivant : 10 lignes de 15 caractères, sachant que les caractères ne peuvent être que "R", "V" ou "B" pour rouge vert ou bleu.
Le deuxième bouton du menu d'accueil, appelé "Grille aléatoire", génère une grille pour vous de manière aléatoire.

Vient ensuite le déroulement de la partie. Lors de la partie, les items dans la grille prennent la forme de petits cercles qui sont d'une couleur parmi les trois disponibles. Lorsque l'on passe la souris sur un groupe de couleur, le fond de la case de chaque éléments du groupe devient noir pour faciliter la visualisation des coups possibles. Enfin, lorsque l'on clique sur un groupe d'au moins 3 éléments, on voit le score affiché en dessous de la grille de jeu augmenter. Vous pourrez toujours supprimer un groupe de deux cases mais cela n'aura pas d'influence sur le score car pour chaque groupe supprimé de n cases, le score augmente de $(n-2)^2$ points. Cela peut tout de même être stratégique de supprimer un groupe de deux pour augmenter la taille d'un groupe sur le coup suivant.

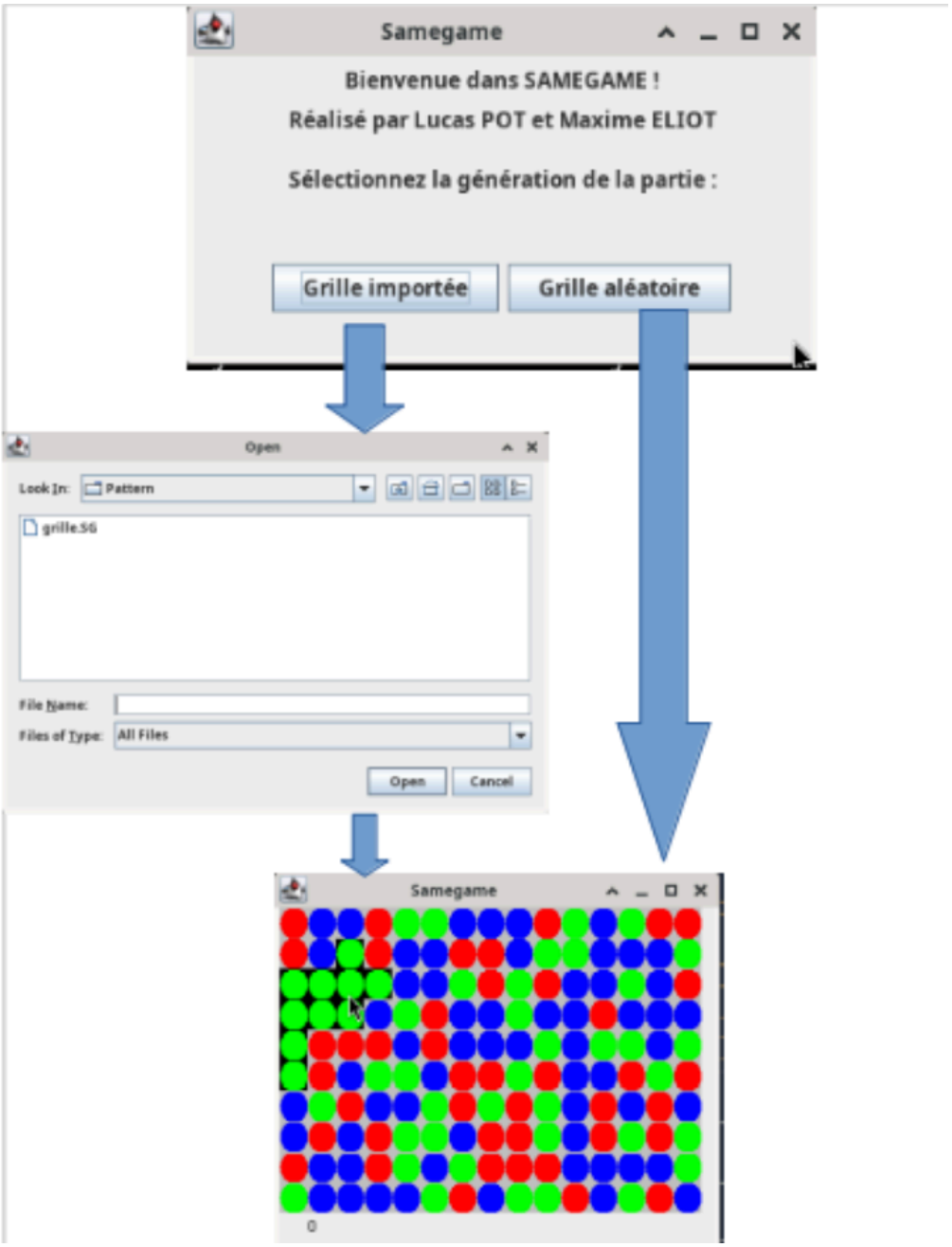
Pour finir le menu de fin qui apparaît automatiquement après avoir joué votre dernier coup possible. Il est très similaire au menu d'accueil. Il affiche le score de votre partie et met deux boutons à disposition : un pour rejouer, qui vous affichera le menu d'accueil, puis un bouton quitter qui fermera la fenêtre et arrêtera le programme.

Regardons maintenant comment se déroule une partie.

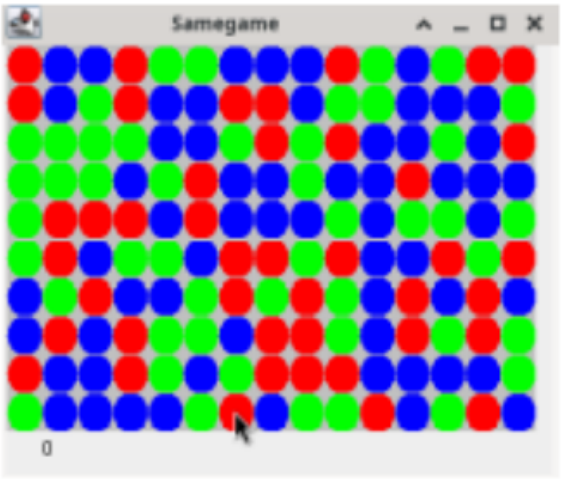
# II - Déroulement d'une partie

## 1) Initialisation de la partie

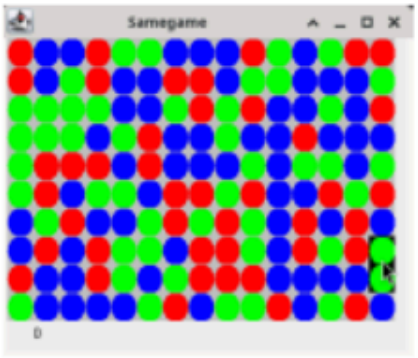
La génération de grille se fait de deux manières. Soit on choisit d'importer une grille depuis un fichier et dans ce cas un explorateur de fichier apparaît. Soit on choisit de générer une grille aléatoirement et la grille apparaît instantanément.



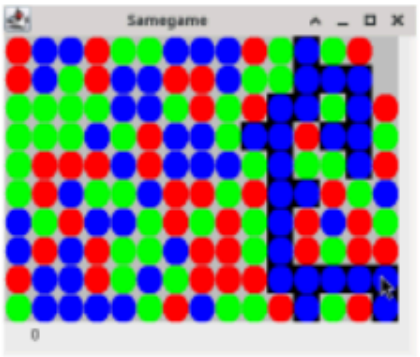
## 2) Au moment de jouer



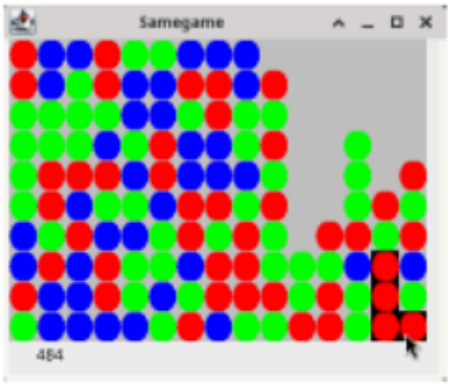

On peut observer que si on survole une case seule, son fond ne change pas car elle n'est pas cliquable.




Ici on clique sur un groupe de deux et on voit que les cases du dessus se sont décalées de deux cases vers le bas.

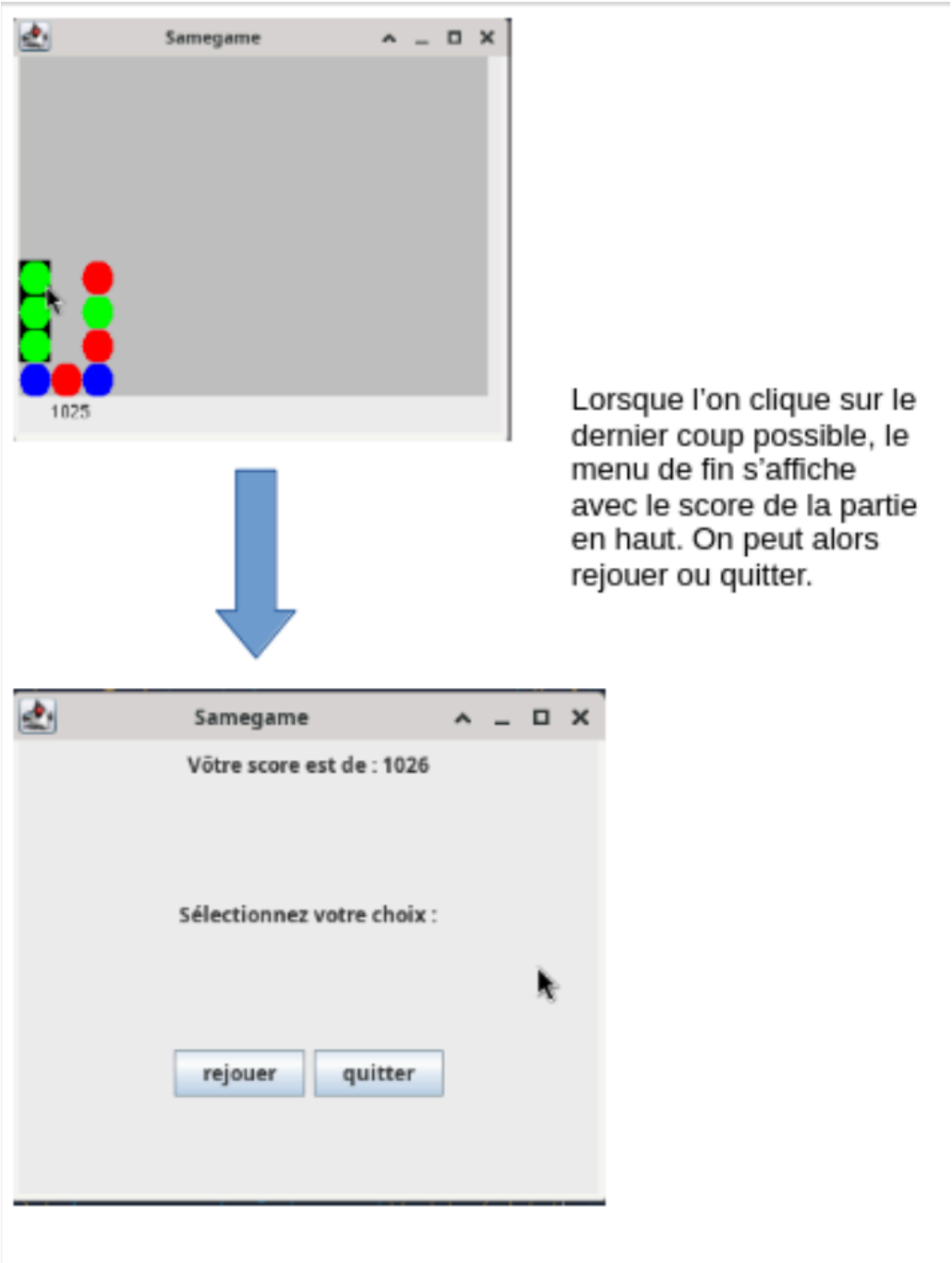



Ici on clique sur un groupe plus gros et on voit que le score augmente.



## 3) Fin de la partie

Après avoir jouer presque tous les coups possibles, nous arrivons au dernier coup :




Si on clique sur rejouer, tout le processus que nous venons de voir se répétera.

# III - Algorithme de détection des groupes

Pour la détection de groupe, nous utilisons un second tableau à double entrée, tabBlock, qui est constitué d'entiers initialisés à 0. A chaque fois que la souris change de position, un calcul est fait pour savoir sur quelle case elle se trouve. Dans tabBlock, on change alors la valeur de la case par un 1. Ensuite pour cette même case on va regarder ses voisins dans tabJeu (tableau de caractères où les couleurs sont indiquées). Pour chaque voisin qui est de la même couleur, on récupère ses coordonnées de case pour mettre la valeur de la case dans tabBlock à 1. Cela va durer jusqu'à ce que tous les voisins d'une même couleur aient été visités. Au final, on aura dans tabBlock que des 0 sauf pour le groupe survolé par la souris où il y aura des 1. Ainsi nous pouvons facilement changer la couleur de fond des éléments survolés puisque tabBlock nous donne le tableau des éléments à mettre en évidence. On a aussi un système qui compte les cases de tabBlock qui sont à 1 pour pouvoir savoir si on doit mettre en évidence les cases ou non ( ex : si il n'y a qu'une case à 1 ce n'est pas un groupe donc on ne change pas le fond) et pour pouvoir compter les points gagnés en cas de clique. Il va de soi aussi que l'algorithme ne prend pas en compte les groupes de "\0" dans tabJeu puisque ce sont les cases vides.

# IV - Conclusions personnelles

## 1) Conclusion Maxime

Pour conclure, je dirais que ce projet était intéressant à réaliser bien qu'il soit plutôt similaire à celui du premier semestre. Je trouve intéressant d'utiliser des classes graphiques qui soient universelles comparé au premier semestre. Ce projet m'a aussi permis d'approfondir mon organisation de travail en équipe avec Lucas. Je pense que nous sommes mieux organisés ce qui nous a permis d'avancer plus vite ensemble. En somme, je suis satisfait de notre travail et que le jeu fonctionne parfaitement en peu de temps de travail si on compare au projet du semestre 1.

## 2) Conclusion Lucas

Pour conclure, je dirais que ce projet nous a permis de mieux appréhender la programmation orienté objet même si par moment on ne réfléchissait pas systématiquement dans cette direction et qu'on pourrait mieux optimiser certains de nos programmes. Le résultat obtenu est quand même à mon goût et le java nous a permis d'enfin faire un programme facilement exécutable par tous et d'aussi rendre notre programme responsive. En somme, nôtre travail avec Maxime est, d'après moi, plutôt une réussite et un réel apprentissage qui nous a permis de mieux comprendre l'orienté objet.