RETOUR D'EXPERIENCE SUR LE PROJET « MORPION SOLITAIRE »

CHAUVEAU Arthur

BRANCO-GOMES Hugo

BEN-THAIER Majdi

Table des matières

Présentation du projet

Introduction générale du sujet :

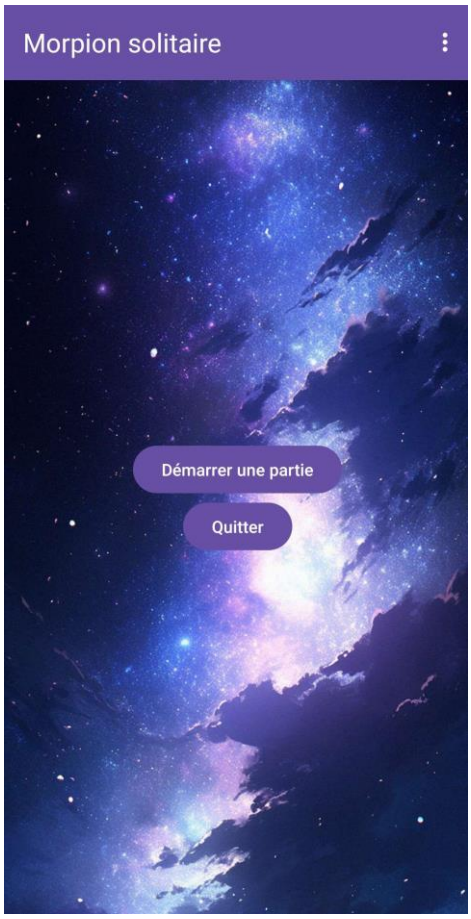
L'objectif du projet consistait à développer une application Android en Java permettant de jouer au Morpion Solitaire. Le Morpion Solitaire est un jeu solitaire se déroulant sur une grille composée de 24 ou 36 intersections, formant une croix grecque. Au début du jeu, le joueur peut choisir la taille de la croix (24 ou 36 intersections) ainsi que d'autres options via les paramètres de l'application. La partie se déroule tant qu'il reste des coups possibles à jouer.

Pour vérifier la validité d'un coup, il est nécessaire de s'assurer qu'il passe par quatre intersections déjà occupées et que sa longueur est correcte.

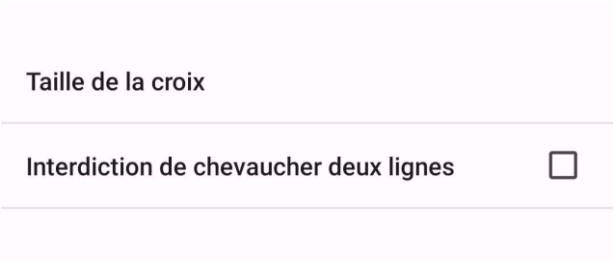
Fonctionnalités du programme

Notre programme de Morpion Solitaire propose un menu au démarrage du jeu.

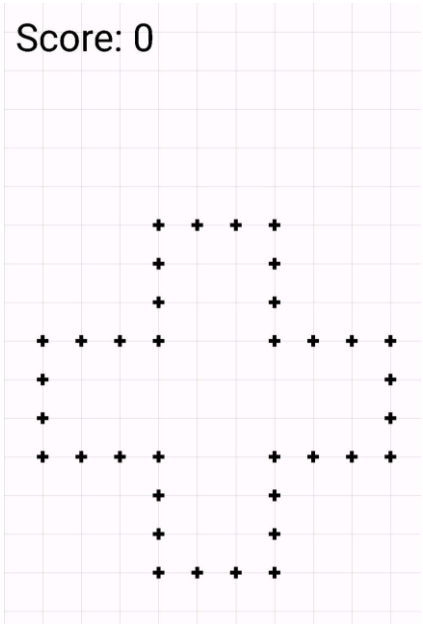
L'utilisateur peut choisir d'accéder au jeu ou aux options en appuyant sur les trois points en haut à droite du menu. Il peut également quitter le menu pour revenir sur son téléphone en appuyant sur le bouton "Quitter" ou en retournant en arrière.



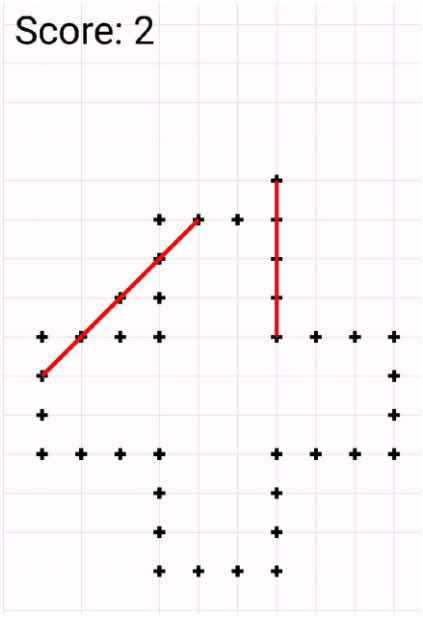
Dans les options, il est possible de modifier la taille de la croix et d'activer l'interdiction de chevaucher deux lignes entre elles. De plus, il est possible de revenir au menu en retournant en arrière. Les changements d'options prendront effet jusqu'au prochain changement.



En accédant au jeu, l'utilisateur peut voir son score en haut à gauche ainsi que la croix de 36 ou 24 intersections (36 dans l'exemple ci-dessous).



Pour jouer, l'utilisateur peut tracer une ligne en restant appuyé entre la position de départ et la position d'arrivée de la ligne. Le coup sera ajouté à la grille s'il est légal, sinon il sera ignoré. Le déplacement doit être effectué en un seul geste.



Pour se déplacer sur la grille, l'utilisateur doit poser deux doigts et il peut naviguer sur la grille. Pour revenir à la position de départ, il doit effectuer un appui long avec un doigt.

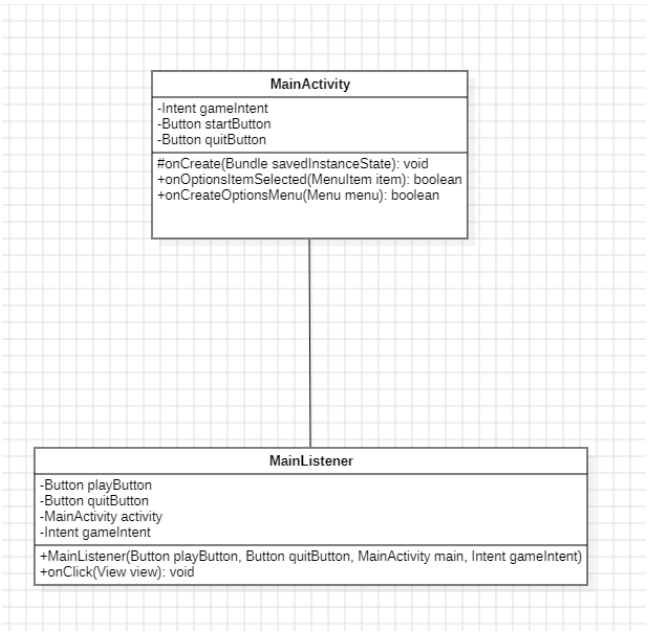
Si l'utilisateur met en pause l'application, il pourra y revenir sans perdre l'avancement. Nous avons tenté de gérer la rotation de l'écran, mais n'avons réussi qu'à sauvegarder une seule option à la fois. La sauvegarde de rotation est donc en commentaire dans notre fichier "GameActivity".

Structure du programme

Le programme principal est composé de 2 classes :

-Le MainActivity qui va correspondre au menu d'accueil du jeu.

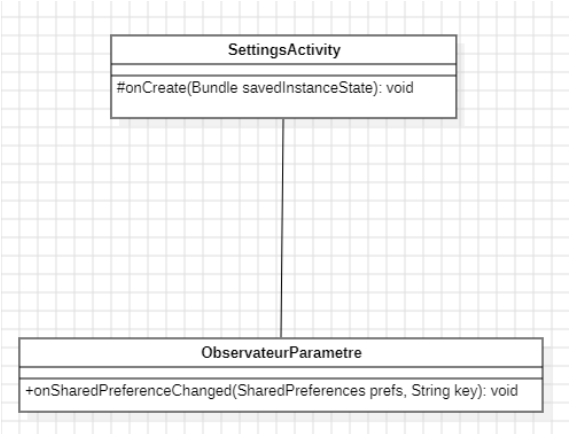
-MainListener qui va voir si des clics sont effectuées sur le menu pour aller vers les autres activités.



Le programme qui gère les paramètres est composé de 2 classes :

-Le SettingsActivity qui va gérer l'affichage des paramètres

-ObservateurParametre



Le programme qui va gérer le jeu est lui composé de 6 programmes :

-Le GameActivity qui va permettre de lancer le programme du jeu

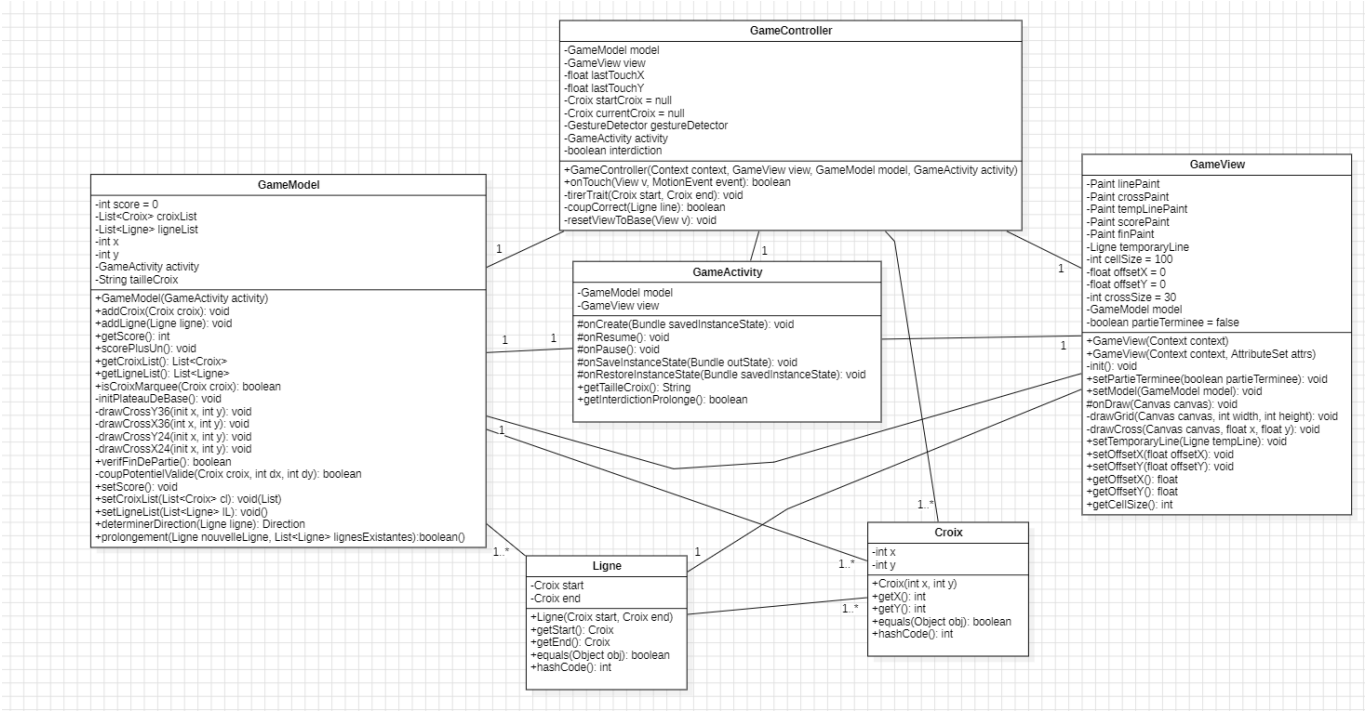
-GameView, c'est la classe qui permet de créer l'écran de jeu

-GameController, c'est donc la classe qui représente le controller de l'écran de jeu et qui va controller tous les clics qui sont fait ce même écran.

-GameModel, permet la création du modèle ainsi des méthodes pour interagir avec des objets des classes Croix et Ligne.

-Croix, comme son nom l'indique cette classe permet la représentation d'une croix et stocke des données tel que sa position

-Ligne, comme son nom l'indique cette classe permet la représentation d'une ligne et stocke des données tel que sa position de départ et sa position d'arrivée.



Algorithme pour savoir si un coup est légal

Pour valider un coup de ligne, nous avons mis en place un processus méthodique.

Tout d'abord, nous déterminons les points de départ et de fin de la ligne proposée. Ensuite, nous calculons la différence en termes de coordonnées X (diffX) et Y (diffY) entre ces points afin de comprendre l'orientation et la longueur de la ligne. Une liste nommée pointsLigne est initialisée pour stocker tous les points (représentés par des objets Croix) que la ligne traversera.

À l'aide des valeurs de deltaX et de deltaY, qui indiquent la direction de la ligne sur les axes X et Y, nous ajoutons à pointsLigne les points par lesquels la ligne passera.

Pour chaque point dans pointsLigne, nous vérifions si une croix est déjà marquée à cette position en utilisant la fonction model.isCroixMarquee(point). Si c'est le cas, nous augmentons le compteur croixComptees.

Nous déterminons ensuite si la ligne est horizontale, verticale ou diagonale en examinant les valeurs de diffX et diffY.

Si la ligne est horizontale, verticale ou diagonale, nous procédons à la vérification de sa validité. Si l'interdiction de prolonger une ligne existante est activée, nous utilisons model.Prolongement(line, model.getLigneList()) pour vérifier si la ligne commence ou finit à l'extrémité d'une ligne existante tout en ayant la même direction. Si la ligne respecte cette condition et qu'il y a exactement 4 croix comptées, le coup est considéré comme correct. Nous parcourons alors pointsLigne pour trouver le point non marqué et y ajoutons une croix à l'aide de model.addCroix(point), signifiant ainsi que le coup est valide et effectué.

Si l'interdiction de prolonger n'est pas activée, nous vérifions simplement si 4 croix sont comptées le long de la ligne proposée. Si c'est le cas et qu'il y a un point non marqué, le point est marqué comme une nouvelle croix, validant ainsi le coup.

La méthode retourne true si le coup est validé et effectué, sinon false.

Conclusions personnelles

CHAUVEAU Arthur :

Pour ma part, ce projet m'a permis de réaliser comme tous les autres projets qu'il est essentiel de débuter un projet dès qu'on nous le confie, car celui-ci sera souvent très chronophage, comme ce fut le cas pour le morpion solitaire que nous avons dû réaliser.

Même en ayant entamé le projet dès le début, j'ai pu constater qu'il était plus exigeant en termes de temps que ce que j'avais initialement envisagé.

Ce projet m'a également permis d'améliorer mes compétences en java en xml. En particulier, en élaborant la javadoc pour tous les fichiers, j'ai compris toute son utilité en visualisant l'ensemble des méthodes présentes dans un seul fichier. Elle facilite la compréhension des fonctionnalités des méthodes pour toute personne consultant le code ultérieurement. Mais aussi en séparant bien les fichiers en MVC afin de se retrouver plus facilement.

Enfin, cela m'a fait prendre conscience qu'un projet est souvent plus gérable en équipe ou en groupe, avec une bonne organisation.

BRANCO-GOMES Hugo :

Ce projet fut une nouvelle expérience enrichissante sur un nouveau langage, Android Studio. De plus, réaliser ce projet avec mes camarades m'a permis d'améliorer mon travail en équipe, notamment la répartition du travail. Ce projet m'ai permis d'apprendre au mieux à comment faire la partie xml notamment ainsi que la java même moindre.

BEN-THAIER Majdi :

Ce projet était pour moi très enrichissant car il y avait une sorte de défi, certaines classes ont été plus difficiles à modéliser que d'autres mais quand tout fonctionne l'on a cette sensation de travail accompli et de satisfaction.

Il m'a également permis de renforcer mes acquis du langage et d'aborder d'une autre manière les travaux de programmation en Java/Android.