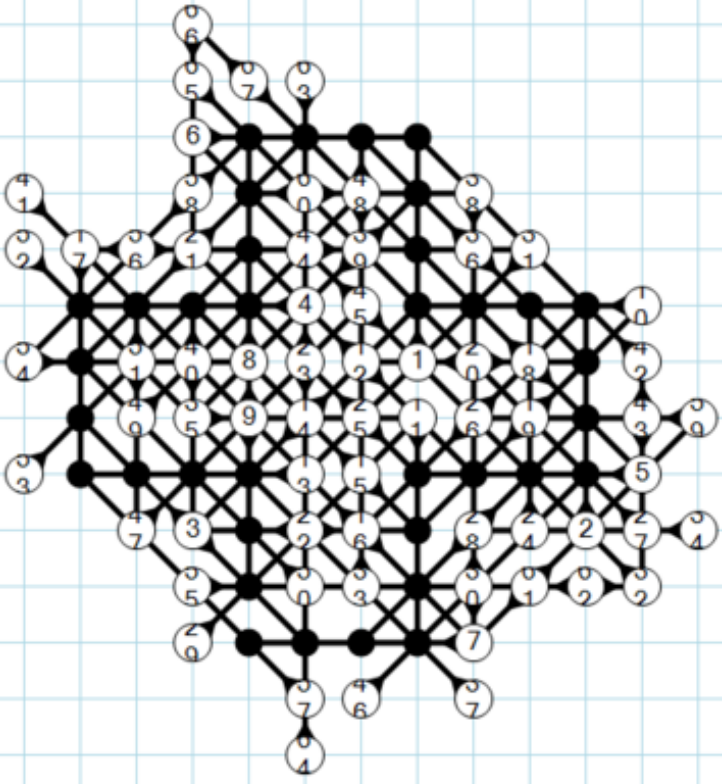
SAE41_2023

MORPION SOLITAIRE EN JAVA



Baptiste Blanchon, Emilien LEDUN & Samet DURAN

SOMMAIRE

INTRODUCTION

Nous développons une application Android du jeu du Morpion Solitaire en utilisant Java et XML.

Le Morpion Solitaire est un jeu de réflexion pour un seul joueur, où l'objectif est de relier des intersections marquées sur une grille à l'aide de lignes, jusqu'à ce qu'aucun coup valide ne soit possible. À chaque coup, le joueur relie quatre intersections marquées avec une intersection non marquée en traçant une ligne horizontale, verticale ou diagonale.

Le jeu se déroule sur une feuille à carreaux, et le score du joueur est déterminé par le nombre de coups joués. L'application offre une expérience utilisateur fluide, avec un menu principal permettant de démarrer une partie, d'accéder aux options et de consulter les règles du jeu.

Nous mettons également en place des fonctionnalités telles que le défilement de la grille, la possibilité de recentrer la vue sur la croix de départ, et la détection automatique de la fin de partie.
En outre, une activité de configuration permet d'activer des règles optionnelles pour personnaliser l'expérience de jeu. Ce projet nous permet d'explorer la conception et le développement d'une application Android interactive, tout en mettant en pratique les concepts de programmation orientée objet et d'interface utilisateur.

FONCTIONNALITÉS

Le Morpion Solitaire possède plusieurs fonctionnalités, en premier lieu lorsque vous lancer l'application depuis votre appareil Android, vous arrivez sur une page d'accueil :

Page Menu

Ici, vous pouvez observer le menu d'accueil qui vous mène à deux possibilités :

- Option : Lorsque vous cliquez sur ces trois petits points, vous arrivez sur la page des options
- Jeu : Lorsque vous cliquez sur le bouton "JOUER", cela vous fera commencer une partie

Page Option

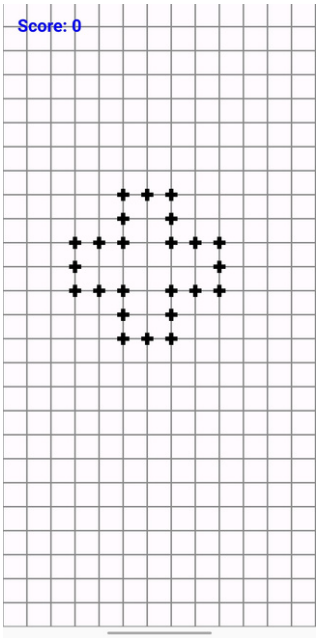
Sur cette page, vous aurez trois choix possibles :

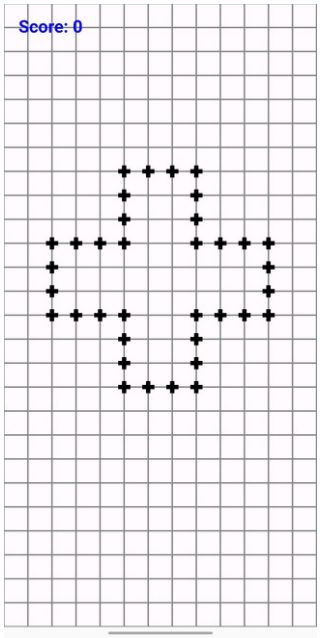
- Menu : en appuyant sur ce bouton, cela vous fera revenir au menu d'accueil
- Taille : il s'agit d'une liste d'objets ("spinner") qui permet de choisir la taille de la croix de jeu (choix entre "Petite" et "Grande")
- Continuité : il s'agit d'une case à cocher ("checkbox") qui est une autre option (pas finie par manque de temps)

Il est bon de savoir que les options se sauvegardent, c'est-à-dire que lorsque l'utilisateur quitte la page des options ou ferme l'application ou autre, les options sélectionnées restent les mêmes qu'avant la fermeture de la page

Page Jeu

“Petite” **“Grande”**





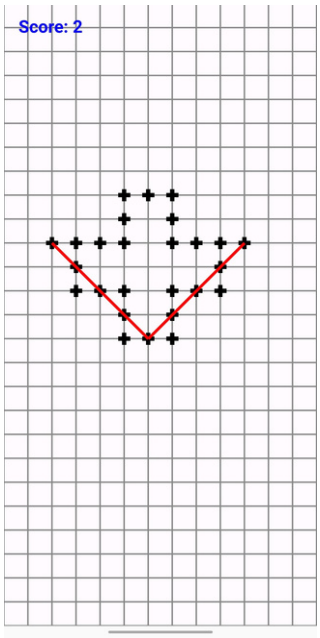
Taille de la croix

La **taille** “Petite” contient 24 petites croix alors que la **taille** “Grande” en contient 36.

Vous pouvez aussi voir en haut à gauche de ces images un **score** écrit en bleu qui est actuellement de 0, lorsqu’un trait est tracé le score augmente de 1

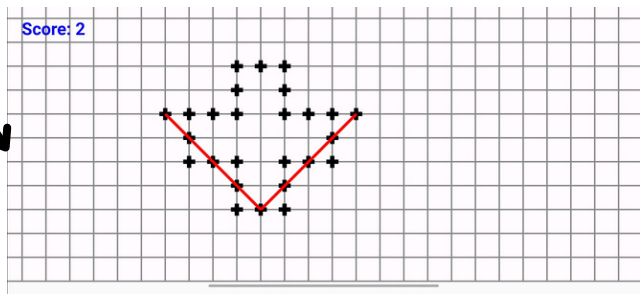
Scoring

Pour qu’un trait tracé (tracé en rouge) par l’utilisateur soit validé, il faut qu’il contienne 5 cases dont 4 petites croix ainsi qu’une intersection vide.
Lorsqu’un trait est validé, le **score** augmente de +1 et une croix est ajoutée sur l’intersection vide du trait tracé par l’utilisateur.



Aussi, lorsque l’utilisateur tourne son écran, le problème était que les informations n’étaient pas sauvegardées, c’est-à-dire que les traits tracés, les croix disparaissaient et le score revenait à 0. Il a donc fallu faire en sorte que ces informations soient sauvegardées pour qu’une erreur de l’utilisateur ne puisse pas lui causer de soucis.

(Ici, il s’agit bel et bien de la même partie, l’écran de l’utilisateur a simplement été tourné.)



Déplacements

Dans le cas où l'utilisateur souhaite se balader dans la grille car la croix qu'il a dessiné devient trop grande ou pour une quelconque autre raison, celui-ci peut se balader sur la grille en glissant deux doigts sur l'interface.

Aussi, pour éviter que celui-ci se perde à force de se déplacer, il peut se recentrer sur le centre de la grille en glissant deux doigts tout en les laissant immobiles (sinon cela s'annule) pendant 3 secondes piles sur l'interface du jeu.

Fin de jeu

Une fonctionnalité de "Fin de jeu" est aussi présente dans le jeu, un algorithme vérifie s'il existe encore des coups légaux traçables par l'utilisateur. S'il n'y en a pas, alors la partie s'arrête, un message s'affiche sur l'écran et l'utilisateur peut se balader sur la grille avec la croix et l'oeuvre d'art qu'il a créé de ses propres mains.

(par question de flemme d'essayer de finir une partie car ce jeu est trop long à finir, il n'y a pas d'image de ce qu'il se passe, mais cela est calculé avec les méthodes "afterMove" qui vérifie s'il existe encore un mouvement légal pour l'utilisateur, "checkForEndGame" qui vérifie donc si la partie est finie via "afterMove" et pour finir "showEndGameMessage" qui affiche le message à l'utilisateur et l'empêche de tracer un quelconque autre trait, il ne peut que se balader sur la grille)

STRUCTURE DU PROGRAMME

Avant de démarrer notre projet Android, une réflexion approfondie sur la structure du code a été primordiale. Notre objectif était de concevoir une application claire et maintenable, ce qui nous a conduit à diviser notre code en plusieurs classes, chacune assumant des responsabilités spécifiques au sein de l'application. Voici un aperçu des classes principales de notre projet et de leur rôle :

La classe MainActivity sert de point d'entrée principal pour notre application. Elle est responsable de l'interface utilisateur initiale où l'utilisateur peut naviguer vers différentes parties de l'app. Dans notre cas, MainActivity inclut un bouton pour démarrer le jeu et un menu pour accéder aux options. Elle utilise également une Toolbar pour une expérience utilisateur cohérente et moderne.

GameActivity est l'activité où le cœur du jeu se déroule. Elle héberge le GameView, qui est responsable de dessiner le jeu à l'écran et de gérer l'interaction utilisateur pendant une partie. Cette classe est relativement légère, car elle délègue la logique de rendu et de gameplay à la vue qu'elle contient.

Dans OptionsActivity, l'utilisateur peut modifier les paramètres du jeu, tels que la taille des croix dans le jeu ou activer/désactiver certaines extensions de jeu. Cette classe utilise des SharedPreferences pour enregistrer les préférences de l'utilisateur, garantissant que les choix de configuration sont persistants entre les sessions de jeu.

GameView est la classe centrale du jeu, responsable de toute la logique de rendu et d'interaction. Elle gère le dessin de la grille de jeu, l'affichage des croix, et prend en charge les entrées tactiles des utilisateurs pour jouer. Cette classe travaille avec une variété de données membres pour suivre l'état du jeu, comme la position des croix, le score du joueur, et si le jeu est terminé. Elle implémente également la logique pour sauvegarder et restaurer l'état du jeu à travers les rotations de l'écran ou d'autres interruptions.

Chaque classe a été conçue avec le principe de responsabilité unique en tête, permettant une séparation claire des préoccupations dans notre application. Cela rend notre code plus facile à comprendre, à tester, et à maintenir.

Diagramme de GameActivity

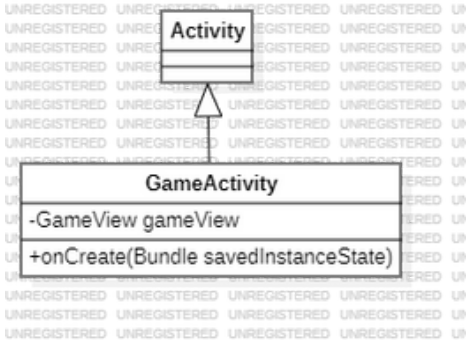


Diagramme de OptionActivity

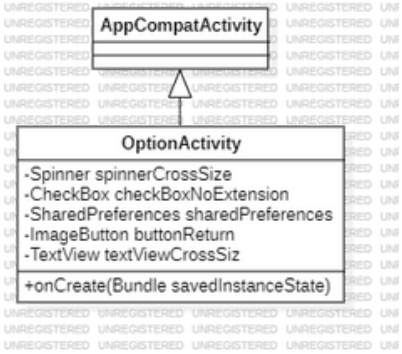


Diagramme de GameView

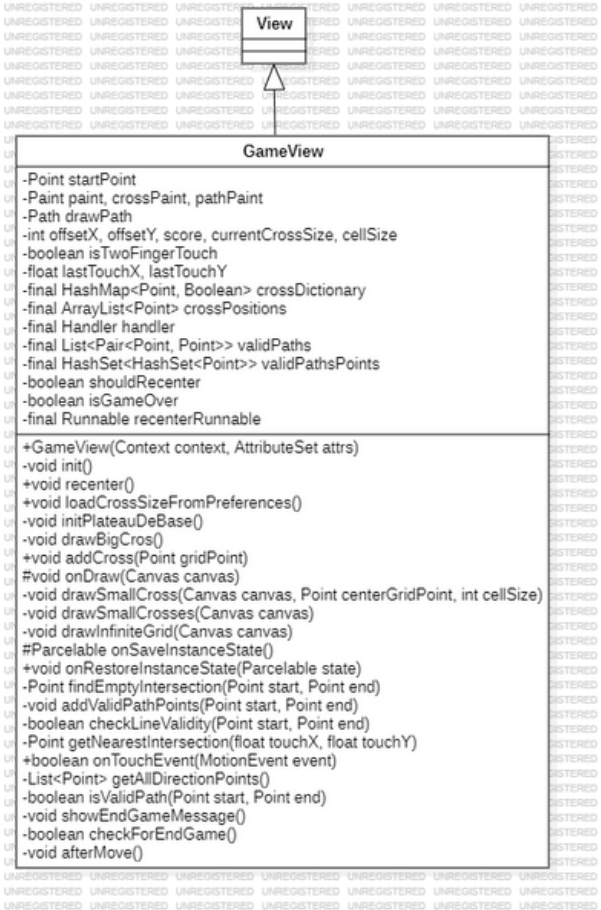
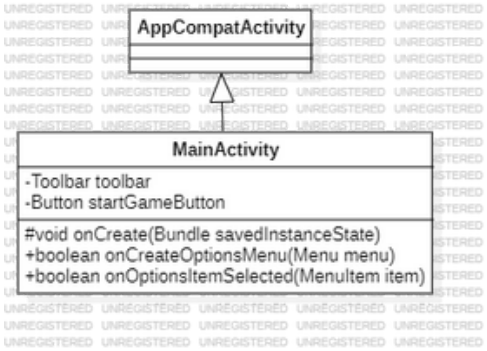


Diagramme de MainActivity



ALGORITHME DE VALIDATION DES COUPS

Dans le cadre du développement de notre jeu de morpion, nous avons dû implémenter une logique permettant de déterminer la légalité d'un coup joué par un participant. Un coup est considéré comme légal s'il remplit les conditions suivantes :

1. Détection de l'interaction utilisateur (onTouchEvent) :
Cette méthode est le point de départ pour toute action dans le jeu. Elle détecte les interactions tactiles et détermine si l'utilisateur essaie de tracer une ligne (un "coup").
Si l'utilisateur touche l'écran et déplace son doigt, la position de départ (**startPoint**) et la position actuelle sont utilisées pour tracer une ligne temporaire qui représente le coup potentiel.

2. Vérification de la légalité (checkLineValidity) :
Une fois que l'utilisateur lève son doigt, la méthode **checkLineValidity** est appelée avec le point de départ et le point de fin de la ligne dessinée.

Cette méthode vérifie plusieurs conditions pour s'assurer que le coup est légal :

- Alignement : La ligne doit être horizontale, verticale, ou diagonale. Ceci est vérifié en s'assurant que le déplacement en x est égal au déplacement en y (pour une ligne diagonale), ou que l'un des deux déplacements est zéro (pour une ligne horizontale ou verticale).
- Longueur : La longueur de la ligne doit être exactement de 5 cases.
- Intersection : La ligne doit passer par exactement 4 croix existantes et doit avoir une intersection vide où une nouvelle croix peut être ajoutée. Cela est vérifié en comptant le nombre de croix traversées et en s'assurant qu'il y a une case vide dans la ligne.
- Chevauchement : La ligne ne doit pas partager plus de deux points avec une ligne déjà tracée. Cette vérification empêche les joueurs de simplement réutiliser des segments de lignes existantes pour marquer facilement des points supplémentaires.

3. Ajout de la nouvelle croix (addCross) et mise à jour du score :

- Si le coup est légal, une nouvelle croix est ajoutée à l'intersection vide trouvée par **findEmptyIntersection**, qui parcourt la ligne pour trouver un emplacement sans croix.
- Le score est incrémenté pour chaque ligne valide ajoutée.
- La nouvelle croix est ensuite dessinée sur la grille par **drawSmallCrosses**.

4. Fin de jeu (checkForEndGame) :

Après chaque coup, le jeu vérifie s'il reste des coups légaux possibles en parcourant la grille et en vérifiant pour chaque croix si une ligne valide peut encore être tracée. Si aucun coup légal n'est possible, le jeu affiche un message de fin de partie.

En résumé, la légalité d'un coup dans votre version du morpion solitaire est déterminée par sa direction, sa longueur, sa capacité à intersecter correctement avec des croix existantes sans trop chevaucher avec des lignes déjà tracées, et la présence d'une intersection vide pour une nouvelle croix. Ce processus assure que le jeu reste un défi stratégique, obligeant les joueurs à planifier soigneusement leurs coups pour maximiser leur score tout en gardant le jeu en cours.

CONCLUSIONS PERSONNELLES

Baptiste: Ce projet de développement d'un jeu de morpion solitaire avec Android Studio a été un défi intéressant, surtout parce que je suis plus habitué à iOS qu'à Android. Utiliser un vieux téléphone Android a transformé un obstacle initial en une belle opportunité d'apprentissage. Travailler avec Java et XML dans un nouvel environnement logiciel m'a permis d'explorer de nouveaux horizons. Ce que j'ai le plus apprécié, c'est la satisfaction de voir les modifications apportées au code se refléter immédiatement, que ce soit via l'émulateur ou sur un vrai téléphone. Ce projet a été une expérience enrichissante qui a stimulé ma passion pour le développement.

Samet: Le développement d'un jeu de morpion solitaire sur Android représente un projet passionnant, particulièrement pour quelqu'un ayant une expérience préalable surtout dans le domaine d'iOS. L'usage de Java et XML au sein d'Android Studio, un IDE dédié au développement d'applications Android, a posé un défi intéressant. Ce projet a non seulement permis d'acquérir de nouvelles compétences dans le développement d'applications mobiles mais a aussi offert l'opportunité d'explorer diverses méthodes de programmation. L'expérience globale a apporté une contribution notable à l'évolution professionnelle dans le secteur du développement d'applications sur Android.

Emilien: Ce projet a pour moi une belle expérience car il m'a permis d'en apprendre plus sur la langage Java et de devenir meilleur dans ce langage. La difficulté a été de mélanger XML qui est un nouveau langage bien qu'il soit facile avec du Java afin de concevoir une application Android (bien que je sois un utilisateur iOS). Cela m'a permis d'en apprendre un peu plus sur le développement d'applications mobiles. En revanche, je n'ai pas particulièrement apprécié ce projet du au fait que pour tester les changements apportés il faut à chaque fois le relancer sur l'appareil Android. En bref, je n'ai pas particulièrement apprécié le projet qui n'a pas relevé la même réflexion que le Petit Tableur, mais cela m'a tout de même permis d'acquérir plus de connaissances en Java sur la partie algorithme et dessins.