

# **Rapport de projet**


## **Morpion solitaire**

### **2024**

**Assia MZYENE**  
**Nathan Bouzon**  
**Emmannuella Lawson-Lartego**



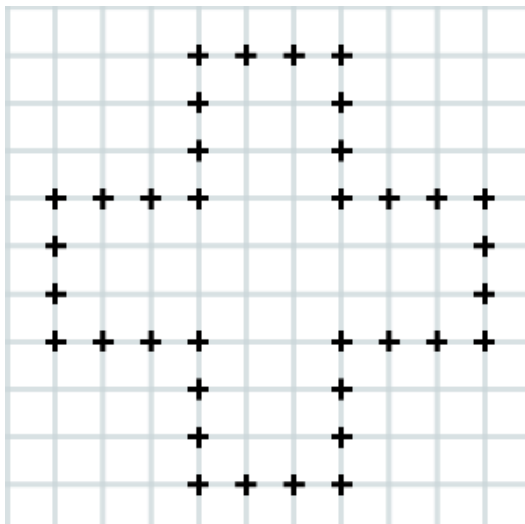
# Sommaire



Introduction .....	2
Description des fonctionnalités .....	3
Structure du programme .....	8
Explication de l'algorithme .....	13
Conclusion personnelle .....	16

# Introduction

Le morpion solitaire est, comme son nom l'indique, un jeu solitaire. La grille de jeu est composée d'un certain nombre de croix qui forment elles même une croix (voir fig. 1).

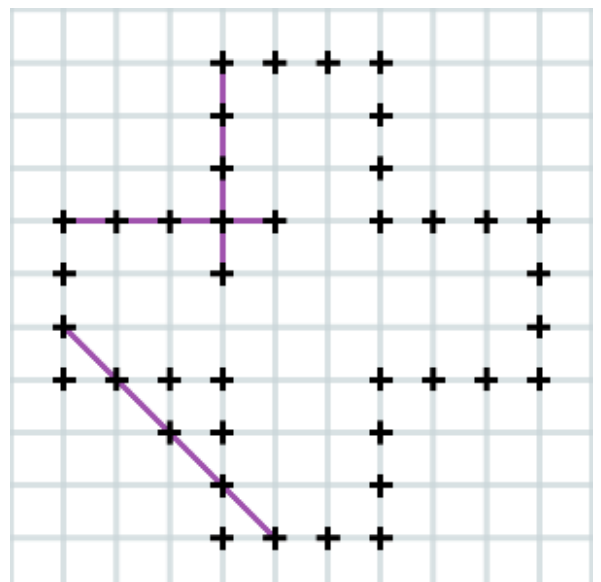


*figure 1*

Le jeu s'arrête lorsque plus aucun mouvement n'est disponible.

Le but est simple : le joueur doit relier quatre croix grâce à une ligne et l'étendre pour obtenir une nouvelle croix (voir fig. 2).

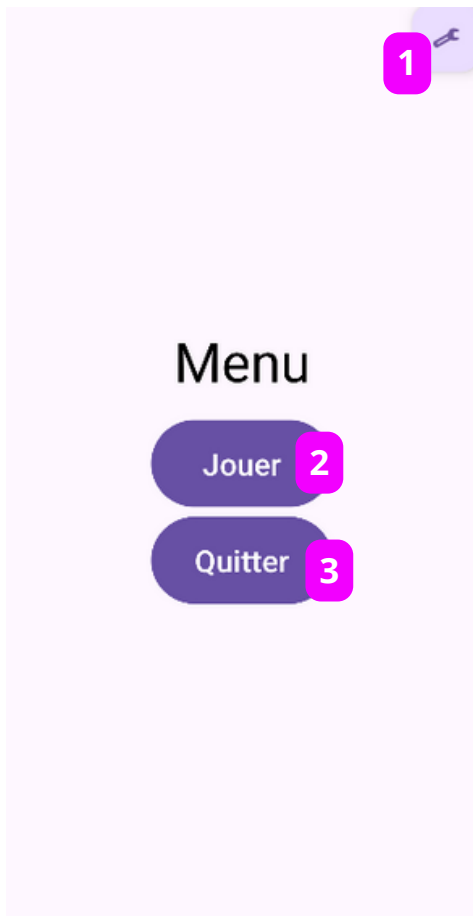
*figure 2*



# **Description des fonctionnalités**

L'écran d'accueil .....	4
Les paramètres .....	5
La feuille de jeu .....	6
La fin du jeu .....	7

# L'écran d'accueil



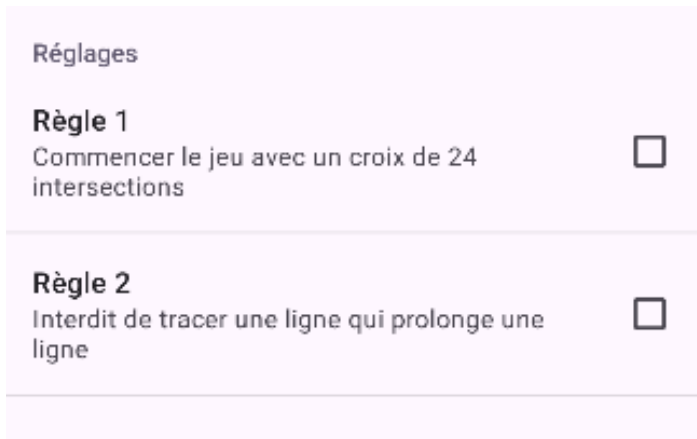
L'écran d'accueil est composé de trois boutons et d'une zone de texte. Lorsqu'on lance l'application, MainActivity créer une instance de HomeController puis associe les différents listeners aux boutons.

HomeController permet de contrôler les événements de click sur un bouton et de passer les paramètres à l'activité qui lance le jeu.

*figure 3*

1. Mène vers les paramètres (fig. 4)
2. Mène vers un terrain de jeu (fig. 5)
3. Ferme l'application

# Les paramètres



Réglages

**Règle 1**  
Commencer le jeu avec un croix de 24 intersections ☐

**Règle 2**  
Interdit de tracer une ligne qui prolonge une ligne ☐

*figure 4*

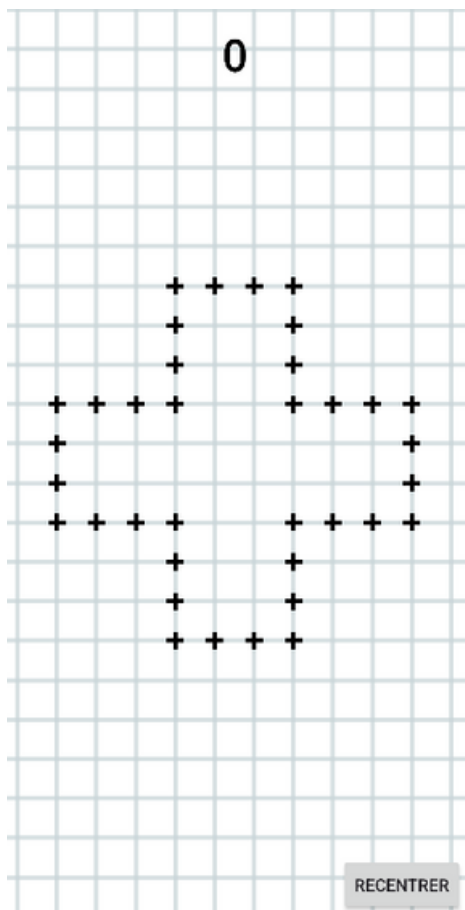
Comme son nom l'indique, ce sont les paramètres. Elles permettent d'activer deux règles.

La première, permet de commencer la partie avec une croix plus petite.

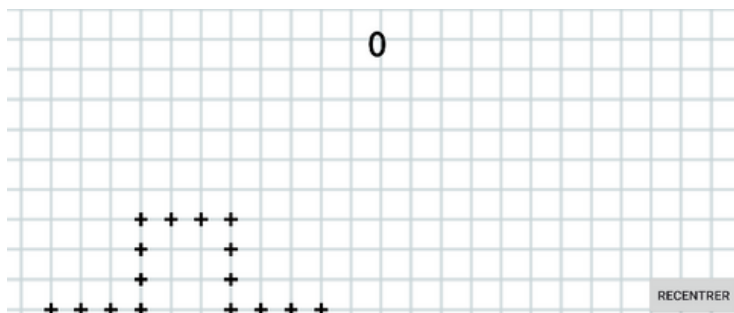
La deuxième permet d'interdire de tracer une ligne qui prolonge une ligne existante.

Lorsqu'on active une des option, un toast s'affiche afin de notifier que le changement à été pris en compte.

# La feuille de jeu



*figure 5*



*figure 6*

La croix de base compte 36 points. Le score est affiché en haut de la feuille.

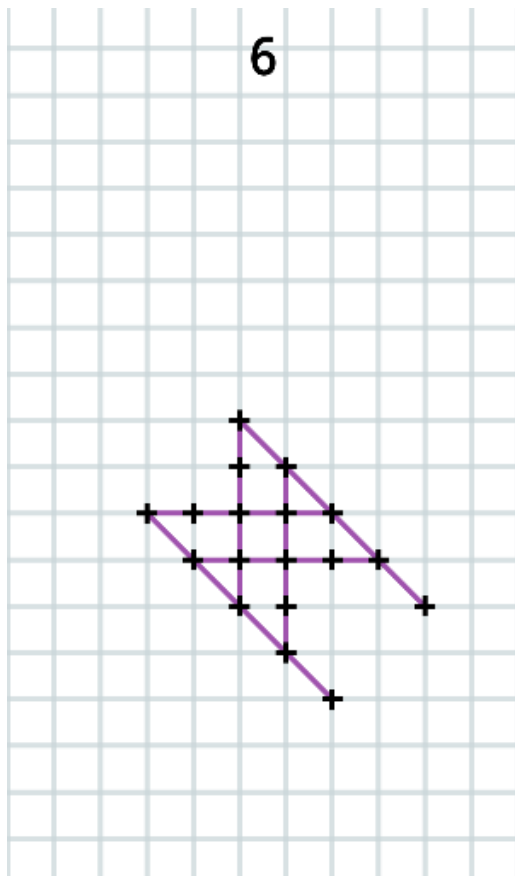
On peut naviguer dans la feuille en posant deux doigts.

On peut appuyer sur le bouton recentrer pour revenir sur la croix de base.

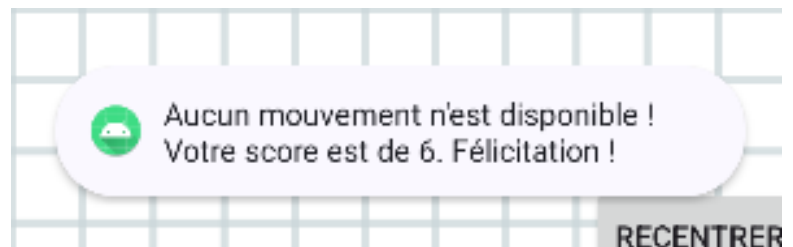
On peut jouer un coup en reliant quatre points et une intersection avec un doigt.

Lorsqu'on tourne l'écran, on garde la position dans laquelle on était (fig. 6).

# La fin du jeu



*figure 7*



*figure 8*

Lorsqu'aucun mouvement est disponible (fig. 7), la partie s'arrête et un message est affiché (fig. 8).

On peut toujours se déplacer sur la carte.



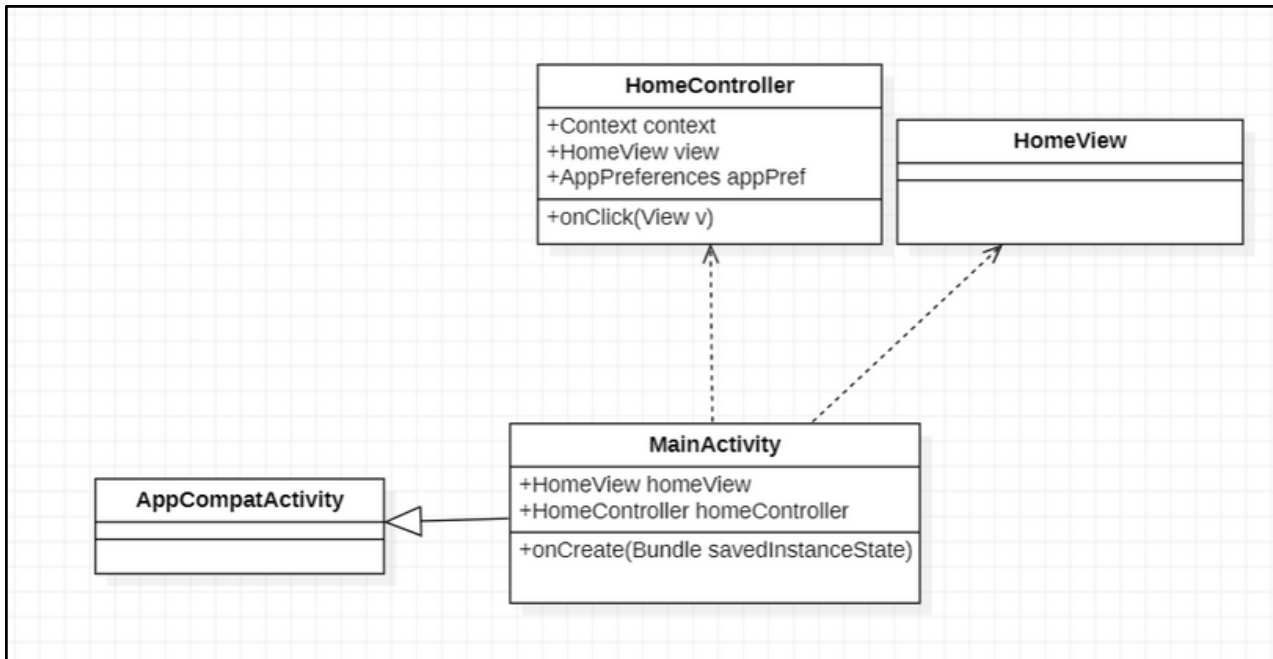
*figure 8*

Si on essaie de faire un coup (ou un mouvement avec un doigt), un autre message s'affichera (fig. 9).



# **Structure du programme**

# Accueil

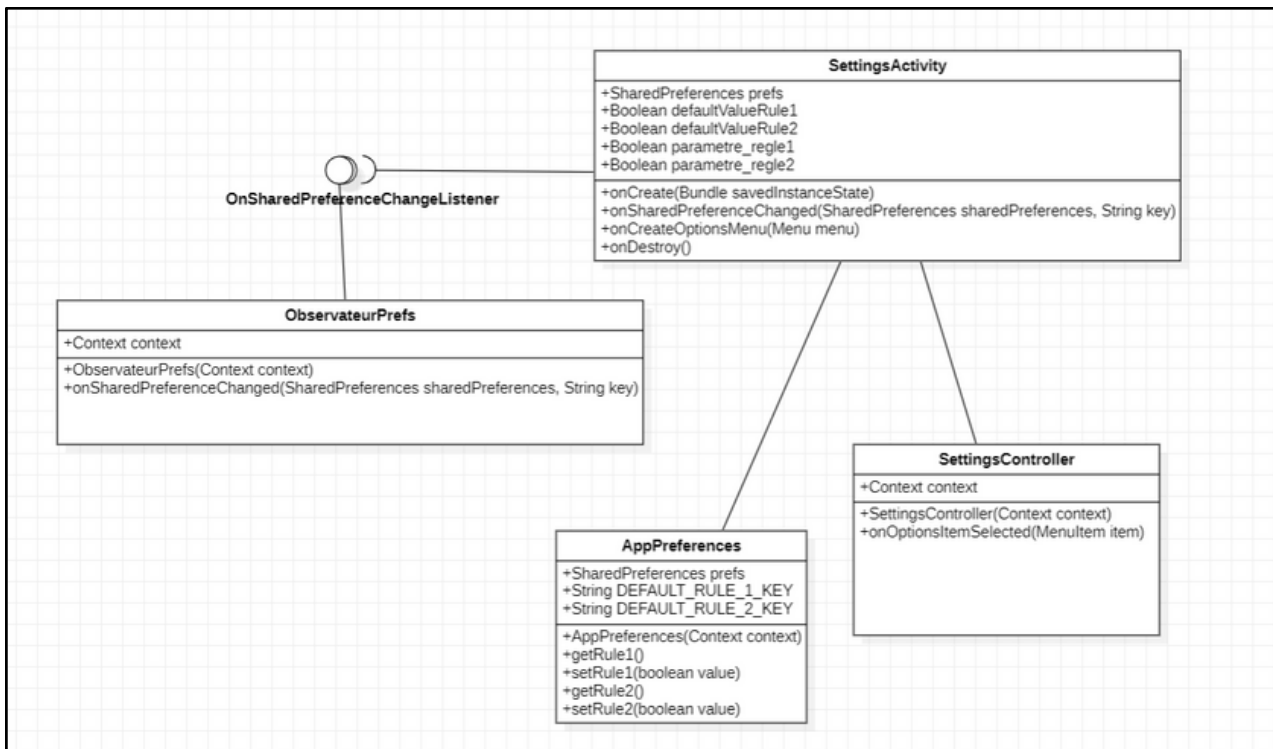


Le menu de base de l'application constitue le point d'entrée pour les utilisateurs, offrant des options pour démarrer le jeu, accéder aux paramètres et quitter l'application.

La classe **MainActivity** est associée à cette activité, agissant en tant qu'hôte pour la vue **HomeView** et le contrôleur **HomeController**.

Lorsque l'activité est créée, la vue **HomeView** est instanciée pour afficher les boutons de jeu, de paramètres et de sortie. Le contrôleur **HomeController** est responsable de gérer les interactions utilisateur avec ces boutons, en déclenchant des actions appropriées en réponse aux clics.

# Paramètres



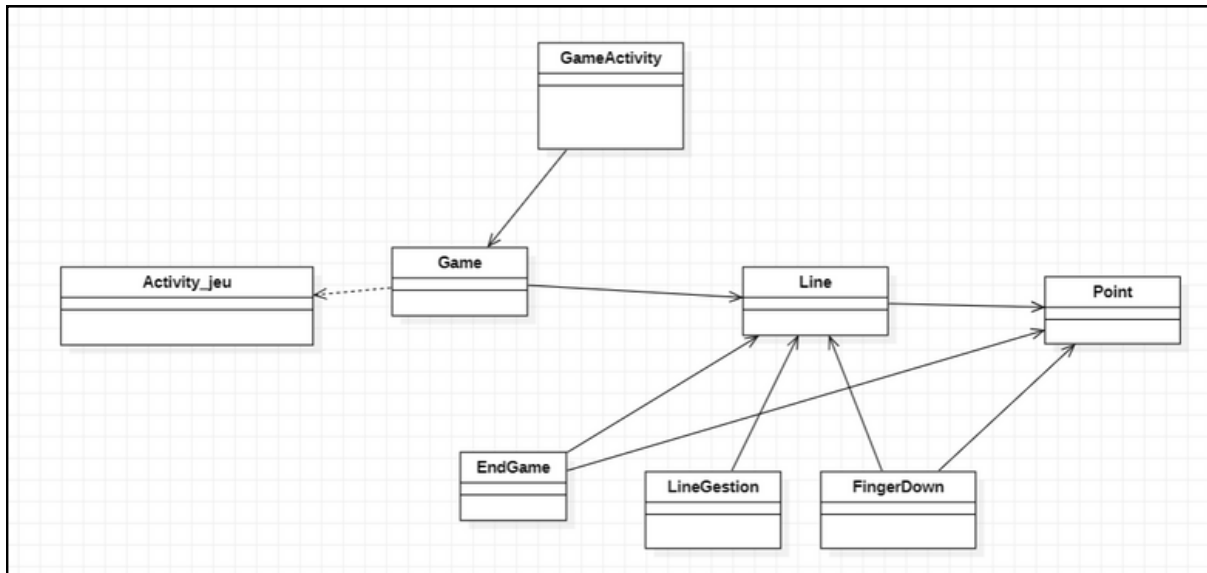
L'activité des paramètres permet aux utilisateurs de personnaliser certaines fonctionnalités du jeu selon leurs préférences.

La classe `SettingsActivity` gère cette fonctionnalité, affichant une liste d'options de paramètres à partir d'un fichier XML. Lorsque cette activité est lancée, elle charge les préférences par défaut et enregistre un observateur (`ObservateurPrefs`) pour détecter les changements dans les préférences partagées.

Les préférences sont stockées et récupérées à l'aide de la classe `AppPreferences`, qui utilise `SharedPreferences` pour gérer les données de configuration de l'application.

Le contrôleur `SettingsController` gère les interactions utilisateur avec les options de paramètres, permettant aux utilisateurs de modifier les réglages du jeu selon leurs préférences.

# Accueil



L'activité de jeu (GameActivity) constitue le noyau de l'expérience de jeu dans l'application. Elle orchestre l'ensemble du processus de jeu en reliant les différentes parties du code. Au cœur de cette activité se trouve la vue de jeu (GameView), responsable de l'affichage de la carte de jeu et de la gestion des interactions utilisateur. La classe GameView communique étroitement avec la logique du jeu, encapsulée dans la classe Game. Cette dernière est chargée de maintenir l'état actuel du jeu, y compris la disposition des points, des lignes et du score, ainsi que l'application des règles du morpion solitaire. Pour détecter et traiter les actions de l'utilisateur sur la carte de jeu, un écouteur de toucher (FingerDown) est utilisé, fournissant une rétroaction en temps réel à l'utilisateur.

# **Jeu**

Dans le cadre de la logique du jeu, deux autres classes principales sont utilisées : Point et Line. La classe Point représente les coordonnées d'un point sur la carte de jeu, tandis que la classe Line représente une ligne reliant deux points.

Ces classes sont utilisées pour modéliser les éléments fondamentaux du jeu et pour effectuer des opérations telles que la vérification de la validité des lignes.

En outre, une classe utilitaire appelée LineGestion est utilisée pour gérer les lignes sur la carte de jeu. Elle fournit des méthodes pour vérifier si une ligne est valide et l'ajouter au dictionnaire des points et des lignes si elle est valide, en conformité avec les règles spécifiques du morpion solitaire.

Enfin, la classe AppPreferences est responsable de la gestion des préférences de l'application, telles que les règles spécifiques au jeu, tandis que la classe SettingsController facilite la gestion des actions liées aux paramètres de l'application. Ces deux composants permettent à l'utilisateur de personnaliser son expérience de jeu en modifiant les paramètres tels que les règles du jeu.

# **Explication de l'algorithme**

L'algorithme commence par vérifier si la longueur de la ligne correspond à une valeur spécifiée, qui est définie comme 4 dans ce jeu. Cette vérification est réalisée en comparant la longueur de la ligne avec la valeur définie, en utilisant la méthode `haveLongueur()` de la classe `Line`. Si la longueur est correcte, l'algorithme continue.

Ensuite, l'algorithme calcule la direction de la ligne en déterminant la différence entre les coordonnées `x` et `y` du point de départ et du point d'arrivée de la ligne. Cela permet de savoir dans quelle direction la ligne se déplace sur le plateau de jeu.

Si la règle 2 est activée, l'algorithme vérifie s'il y a un prolongement de ligne. Cela signifie qu'il vérifie s'il existe déjà une ligne connectée au point de départ de la nouvelle ligne, sauf si cette ligne est elle-même. Si un prolongement est détecté, la vérification échoue et l'algorithme arrête son exécution.

# **Explication de l'algorithme**

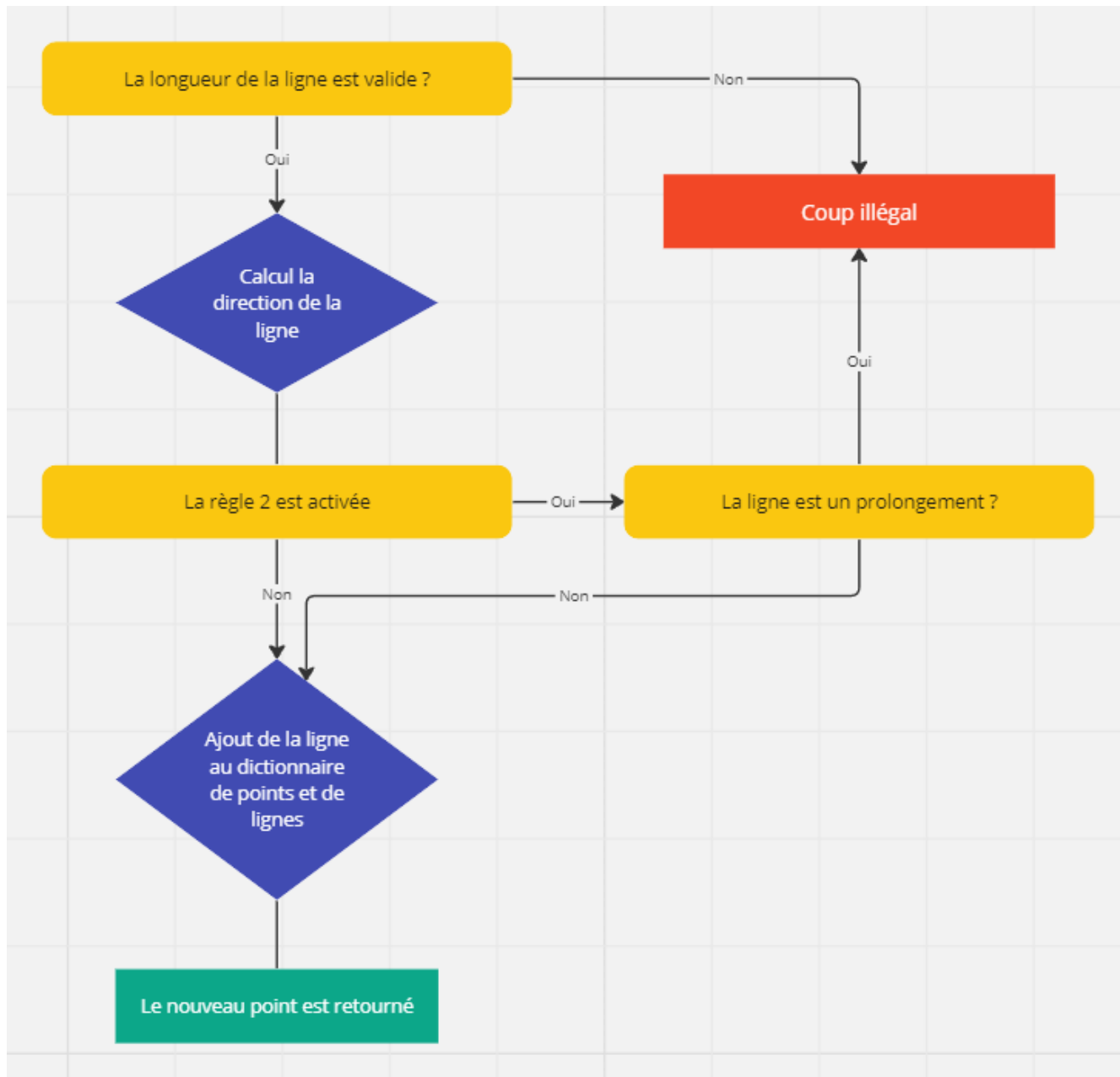
Ensuite, l'algorithme parcourt les points le long de la ligne et vérifie s'il existe déjà des lignes connectées à ces points. Si des croisements sont détectés, la vérification échoue et l'algorithme arrête son exécution.

Si la ligne satisfait à toutes les conditions précédentes, elle est considérée comme légale et est ajoutée au dictionnaire associant les points aux lignes. Ce dictionnaire est utilisé pour représenter les lignes sur le plateau de jeu.

Si la ligne est ajoutée avec succès au dictionnaire, un nouveau point est retourné. Ce point représente la fin de la nouvelle ligne, et il est utilisé pour dessiner de nouvelles lignes à partir de ce point dans le jeu.

Vous pouvez retrouver l'explication sous forme de schéma sur la page suivante.

# Explication de l'algorithme



*Schéma de l'algorithme qui détermine si un coup est légal*



# **Conclusion**

## **Nathan Bouzon**

Ce fut un projet intéressant. C'était notre première véritable application sur Android, donc nous avons rencontré quelques difficultés à nous adapter, notamment à comprendre les erreurs. Nous avons également plusieurs projets en même temps et nous avons décidé de répartir équitablement le travail entre nous pour ces projets. Ainsi, j'ai moins travaillé sur ce projet, mais j'ai quand même appris beaucoup de choses.

## **Emmanuella Lawson-Lartego**

Ce n'était pas la première fois que j'utilisais Android Studio, mais c'était la première fois que je l'utilisais avec Java. Grâce au cours et à ce projet, j'ai appris à mieux maîtriser le Java. Comme l'a dit Nathan, nous avons réparti notre charge de travail en fonction des projets, étant donné que nous avons le même groupe pour les trois. Je me suis moins concentrée sur Android, mais j'ai quand même pu acquérir de nouvelles connaissances.

# **Conclusion**

## **Assia Mzyene**

Ce sujet a été un véritable défi. J'ai rencontré des difficultés au début et mes camarades des autres groupes m'ont beaucoup aidé, notamment Lucile et Alexis à qui je dois beaucoup. Une fois que j'avais pris connaissance du terrain et des bases, il a été plus facile d'avancer et j'ai beaucoup plus apprécié le processus. J'avoue avoir été déçue en voyant le sujet, j'espérais un jeu un peu plus stimulant. Cependant, cela m'a quand même permis d'acquérir beaucoup plus de connaissances, et je me sens beaucoup plus à l'aise avec le logiciel.