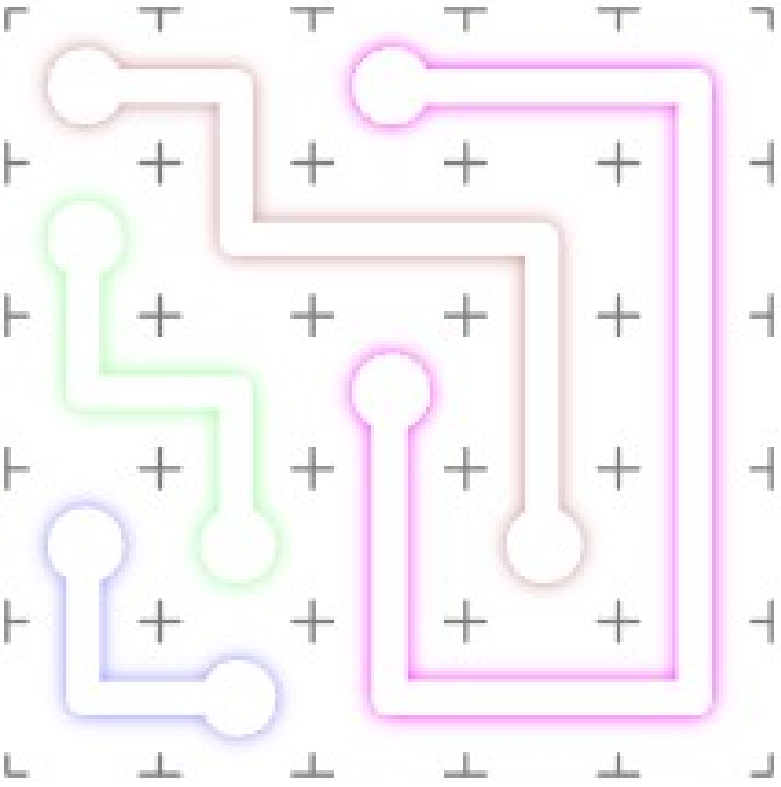
Rapport
SAE42_2025
“Relier les points”



Réalisé par Abed Bridja, Wilfried Brigitte, Christopher Dubreuil

Sommaire

Introduction

Dans le cadre de ce projet, nous avons développé une application Android en Java inspirée du jeu de puzzle "Relier les points" (Flow Free ou Dot Link). Ce jeu repose sur une grille carrée où des paires de points de même couleur doivent être reliées par des chemins continus, en respectant certaines contraintes de déplacement. L'objectif est de remplir la grille entièrement sans que les chemins ne se croisent ni ne se superposent.

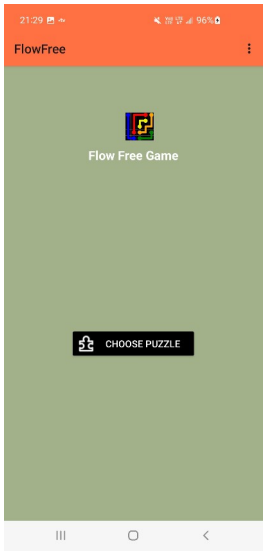
Notre projet vise à reproduire fidèlement cette mécanique en proposant une interface fluide et intuitive. L'utilisateur pourra sélectionner un puzzle parmi une liste de fichiers XML, interagir directement avec la grille tactilement et visualiser en temps réel l'état d'avancement du jeu. De plus, un mode achromate sera disponible pour les joueurs ayant des difficultés avec la perception des couleurs.

Ce rapport détaille les fonctionnalités de notre application, sa structure interne, l'algorithme de validation des solutions, ainsi qu'une analyse de notre progression et des difficultés rencontrées. Enfin, chaque membre de l'équipe proposera une conclusion personnelle sur son expérience au cours du développement de ce projet.

La description des fonctionnalités de votre programme

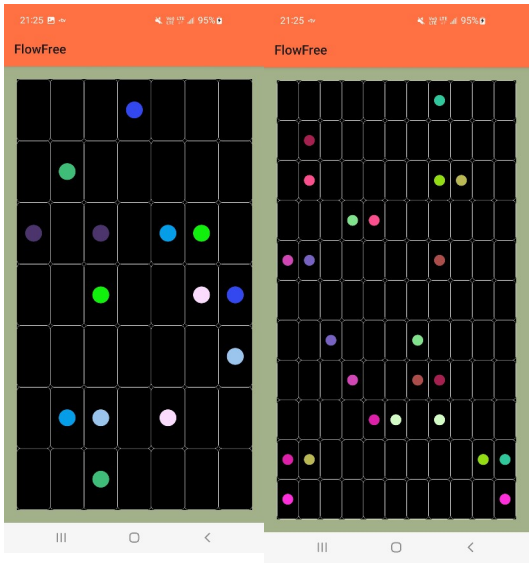
L’application Android est un jeu de puzzle où le joueur doit relier des paires de points sur une grille. Au démarrage, un menu principal permet de sélectionner un puzzle parmi ceux stockés dans app/src/main/assets/puzzles et d’accéder aux options. Les puzzles sont enregistrés sous forme de fichier XML et sont chargés dynamiquement via getAssets() et analysés avec XmlResourceParser. Un puzzle mal formé apparait dans le menu mais est désactivé pour éviter toute erreur. Une fois un puzzle sélectionné, l’utilisateur accède à une activité de jeu où il peut tracer un chemin en posant son doigt sur un point et en glissant jusqu’à l’autre point de la paire. Si le chemin passe par une case déjà occupée, sort de la grille ou si le joueur lève le doigt avant d’atteindre la destination, le tracé est annulé. Un chemin peut être remplacé en repartant de l’une de ses extrémités. Le jeu détecte automatiquement la résolution du puzzle et l’indique au joueur, après quoi il devient impossible de modifier les chemins tracés. Un retour arrière depuis l’écran de jeu ramène au menu principal, tandis qu’un retour arrière depuis le menu ferme l’application. Une option "Mode Achromate", accessible uniquement depuis le menu, permet de remplacer les couleurs des chemins par des tons de gris pour les joueurs atteints de daltonisme. L’état du jeu est conservé lorsqu’il est mis en pause afin d’éviter toute perte de progression.

Menu principal avec bouton pour choisir le puzzle

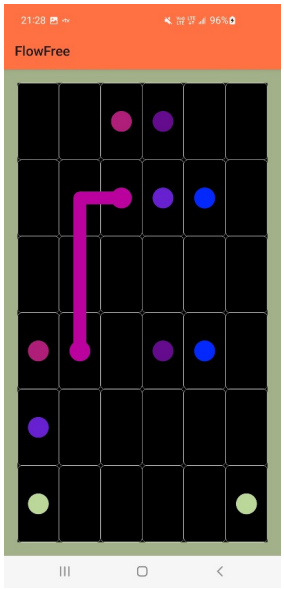


Menu principal avec listes des activités

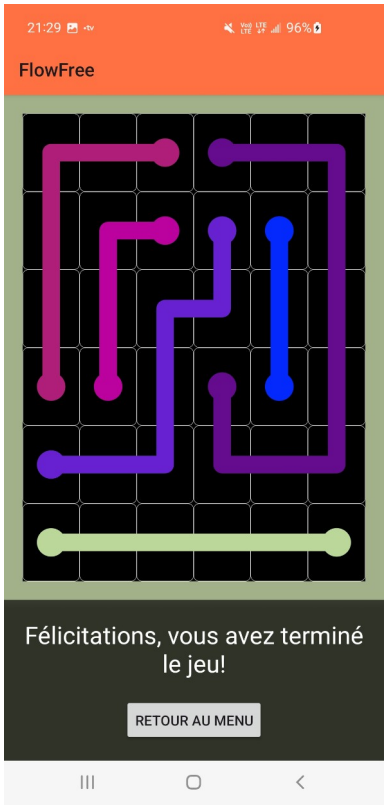
Activité du jeu mais avec des dimensions différentes en fonction de l'activités choisis dans le menu



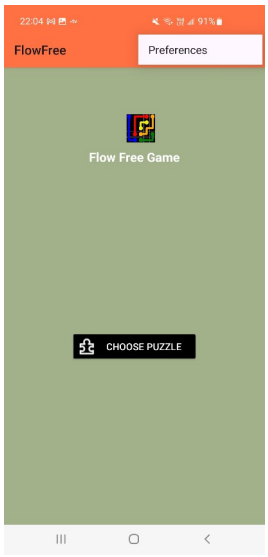
Activité du jeu Tracer un chemin



Activité du jeu: Fin du Jeu



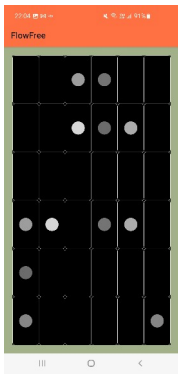
Mode Achromate ; Accés grace aux menus



Quand on clique sur **Preferences** on a :

et on peut cocher ou décocher l’option

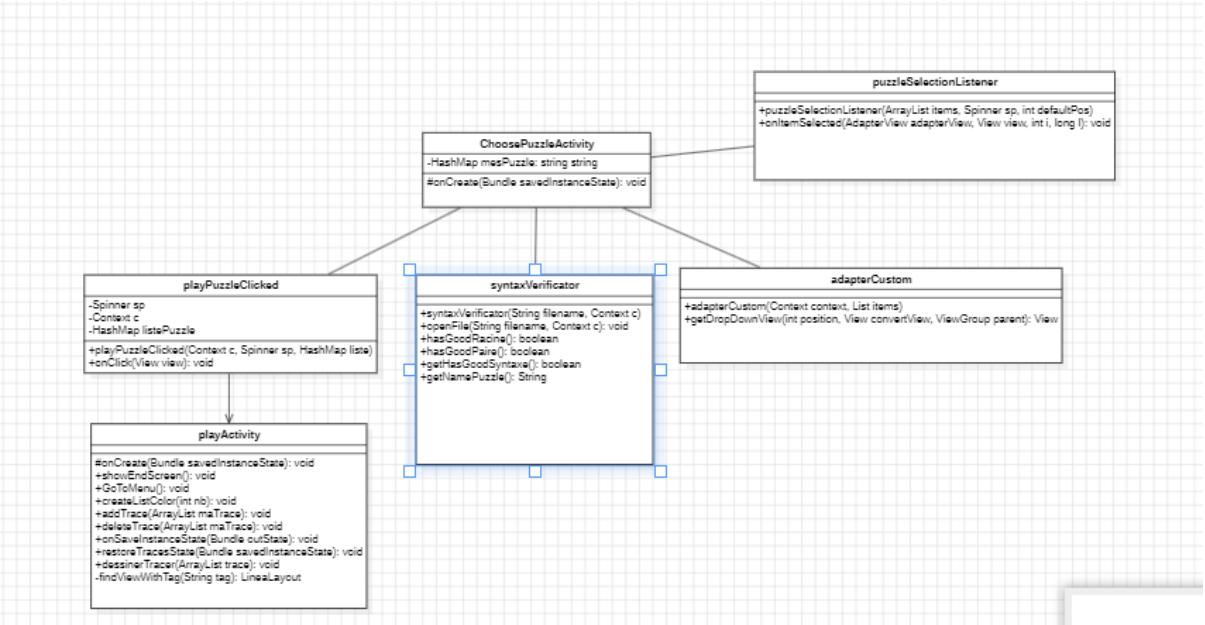
Visuel avec le Mode Achromat



Une présentation de la structure du programme pour chaque activité (diagramme de classes)

1ère activité

Le programme s'articule autour de plusieurs activités et classes qui interagissent pour gérer la sélection et le jeu des puzzles.L'activité principale **choosePuzzleActivity** affiche une liste de puzzles disponibles en vérifiant leur validité à l'aide de la classe **syntaxVerificator**. Si un puzzle est valide, un bouton interactif est généré pour permettre à l'utilisateur de le sélectionner.Lorsque l'utilisateur clique sur un bouton pour choisir un puzzle, l'événement est géré par la classe **playPuzzleClicked**, qui lance l'activité **playActivity**. Cette dernière est responsable de l'affichage du puzzle et de l'interaction avec l'utilisateur pour tracer les connexions entre les points.Pour améliorer l'interface utilisateur, la classe **adapterCustom** personnalise l'affichage des éléments du **Spinner**, tandis que **puzzleSelectionListener** empêche la sélection d'un puzzle invalide. Enfin, la classe **choosePuzzleClicked** est utilisée pour gérer le bouton qui redirige l'utilisateur vers **choosePuzzleActivity**.L'ensemble de ces classes travaille ensemble pour offrir une expérience fluide, allant du choix d'un puzzle jusqu'à sa résolution.

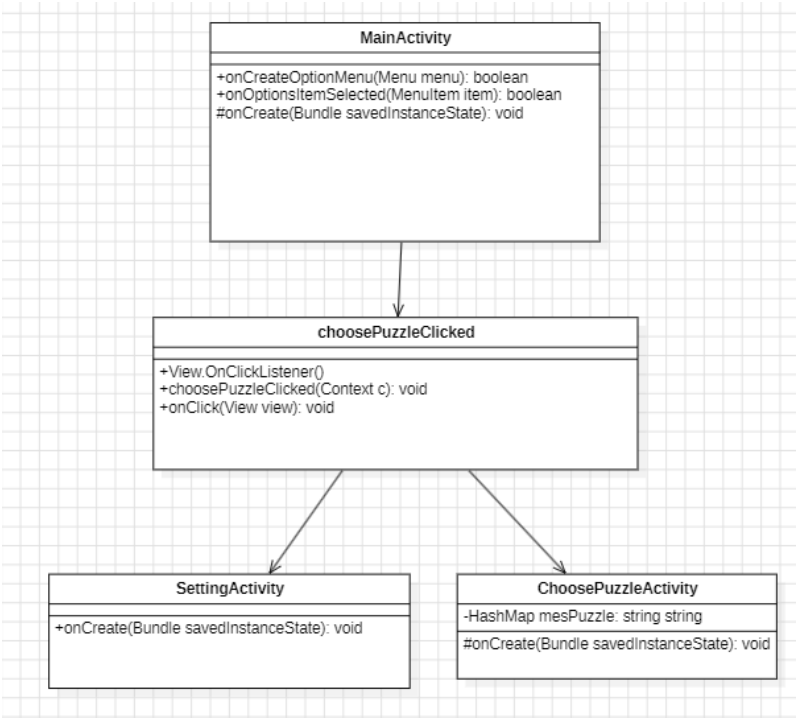


2ème activité

L’application est structurée autour de **MainActivity**, qui sert de point d’entrée et relie plusieurs autres activités. Lors de son lancement, elle affiche une interface utilisateur avec un bouton permettant d’accéder à **choosePuzzleActivity** grâce à l’écouteur **choosePuzzleClicked**. Elle intègre également un menu d’options permettant de naviguer vers **SettingsActivity** pour modifier les préférences de l’application.

L’activité **choosePuzzleActivity** affiche une liste de puzzles en vérifiant leur validité avec **syntaxVerificator**. Les puzzles valides sont activés et peuvent être sélectionnés via des boutons dynamiques. Lorsqu’un utilisateur clique sur un puzzle valide, l’événement est géré par **playPuzzleClicked**, qui lance l’activité **playActivity** pour afficher et jouer le puzzle.

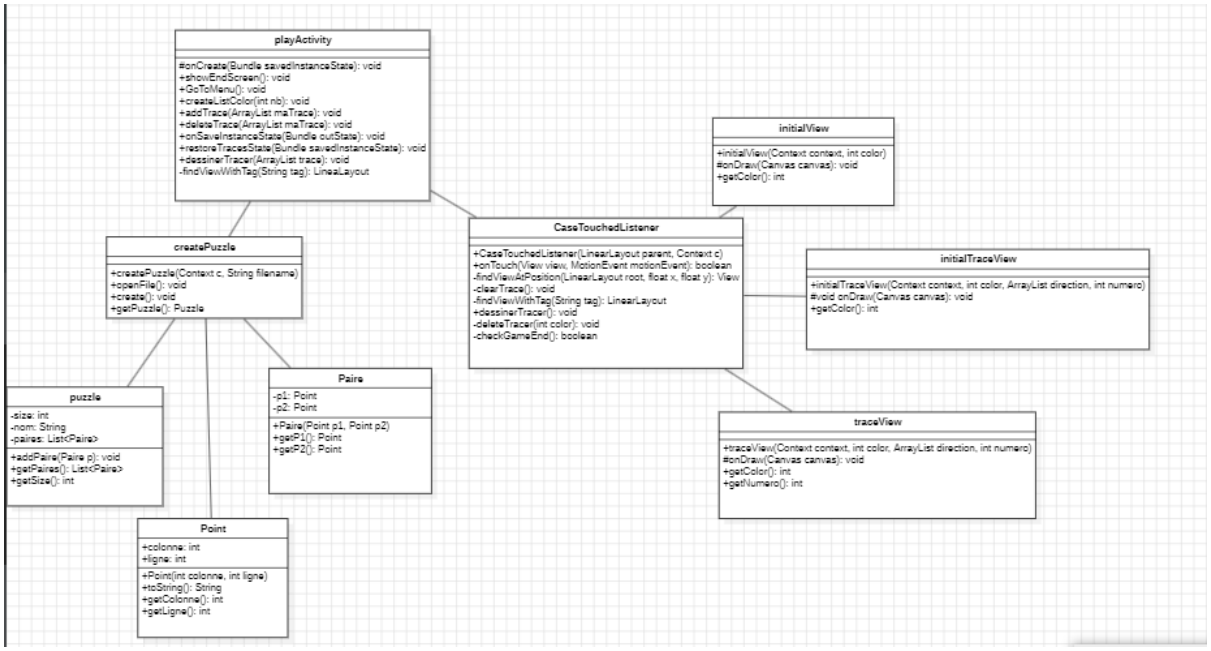
L’activité **SettingsActivity** permet aux utilisateurs de configurer les préférences de l’application en chargeant un fichier XML contenant les options modifiables.Ainsi, **MainActivity** joue un rôle central en connectant les différentes fonctionnalités, facilitant l’accès au choix des puzzles et aux paramètres de l’application.



3ème activité

La classe **PlayActivity** gère l'interface de jeu et les interactions de l'utilisateur. Elle est responsable de l'initialisation du puzzle, de l'affichage de la grille et de la gestion des événements tactiles. Elle est reliée à la classe **CreatePuzzle**, qui est chargée de la création d'un puzzle à partir d'un fichier XML. Ce fichier contient des

informations sur la taille du puzzle et les paires de points à relier. Une fois le fichier analysé, la classe **CreatePuzzle** renvoie un objet **Puzzle** qui contient les informations extraites. Le puzzle est constitué de paires de points, représentées par la classe **Paire**, où chaque paire associe deux objets **Point** définissant des positions dans la grille. La classe **PlayActivity** interagit également avec la classe **CaseTouchedListener**, qui est responsable de la gestion des événements tactiles lors du tracé des lignes entre les paires de points. Cette classe est liée à trois vues : **InitialView**, **InitialTraceView** et **TraceView**. **InitialTraceView** permet de dessiner des traces initiales sous forme de lignes et de cercles colorés, tandis que **TraceView** est utilisée pour afficher les tracés dynamiques du joueur en fonction de ses interactions avec la grille. Ainsi, la structure du programme repose sur l'interaction entre ces classes pour offrir une expérience de jeu fluide et interactiv



Une exposition de l'algorithme qui détecte si le puzzle est résolu

L’algorithme de vérification de la résolution du puzzle fonctionne en parcourant chaque case de la grille, qui est contenue dans un LinearLayout. Chaque case peut être vide, contenir un point de départ(initialView), un segment de chemin(traceView) ou un point de départ ou un point de départ connecté à un chemin initialTraceView. L’algorithme consiste donc à parcourir chaque case de la grille et à vérifier si elle est occupée par un initialView, un traceView ou un initialTraceView. Si une case vide est détectée, cela signifie que le puzzle n’est pas encore résolu. Si toutes les cases contiennent un élément valide, cela signifie que toutes les connexions entre les paires de points ont été effectuées et que le puzzle est terminé.

Conclusions personnelles

Wilfried : Personnellement, j'ai beaucoup apprécié travailler sur ce projet. L'une des principales raisons est qu'il m'a permis de créer quelque chose de concret et d'aller au-delà des exercices pratiques habituels. J'ai également aimé découvrir un nouveau support, en l'occurrence Android, et travailler avec un IDE. De plus, cette expérience m'a encouragé à adopter une approche plus rigoureuse dans ma manière de coder, afin de produire un code clair et compréhensible, non seulement pour moi, mais aussi pour mes deux camarades.

Abed : J'ai apprécié ce projet, parce que nous avons codé sur Android Studio, qui est une approche assez différente de la programmation classique et m'a permis d'explorer un environnement de développement spécifique à la création d'applications mobiles (interface graphique, gestion tactile, etc.), ce qui était très intéressant pour créer ce jeu interactif. Le concept du jeu m'a aussi beaucoup attiré, ce qui m'a motivé à approfondir mes compétences en développement mobile et à créer une expérience utilisateur immersive.

Christopher : Le fait est que le projet soit développé dans un tout nouvel environnement a pu me freiner dans la conception du projet mais le fait qu'on devait programmer un jeu que je connaissais m'a plu. Ainsi, j'ai pu me plonger dans la conception du jeu plus facilement et rapidement. De plus, le projet m'a permis de m'apercevoir de ce qu'il était possible de réaliser comme application mobile et de m'y intéresser davantage.