

Rapport de projet

SAÉ 4.1 FI - CHUZZLE

2025 - 2026

Aylane SEHL

Séri-khane YOLOU

Jenson VAL



TABLE DE MATIÈRES

I. Introduction

Présentation du projet et contexte

Objectifs du projet

Technologies employées

III. Structure du programme

Structure et logique du code

Diagramme de classe

V. Difficultés rencontrées et solutions

Contraintes techniques

II. Présentation générale et fonctionnement

Menu principale et navigation

Grille de jeux et déplacement

Partie avec graine et Hard Mode

Calcul du score

IV. Algorithme de fin de partie

Principe de détection

Description de l'algorithme

VI. Conclusions

Conclusion personnelle de chaque membre

I. INTRODUCTION

1. Présentation du projet et contexte

L'objectif principal de cette SAE est de développer une **application Android jouable, fonctionnelle et bien structurée**, en reproduisant le fonctionnement général du jeu Chuzzle.

Ce projet a pour but de nous apprendre à concevoir une application mobile complète, en combinant à la fois la logique du jeu, l'affichage à l'écran et la gestion des interactions du joueur.

Un premier objectif important était de mettre en place une grille de jeu de 6 x 6, composée de plusieurs types d'éléments, et de permettre le déplacement circulaire des lignes et des colonnes.

Le projet devait également permettre de détecter automatiquement les séries d'au moins trois éléments identiques, de les supprimer, puis de faire descendre les éléments restants afin de remplir à nouveau la grille. Il fallait aussi gérer les éventuelles séries en cascade qui peuvent apparaître après une première suppression.

Un autre objectif était d'ajouter des éléments de jeu supplémentaires afin de rendre l'application plus complète, comme le score, le compteur de coups, les verrous, ainsi qu'un système de fin de partie lorsque plus aucun coup utile n'est possible.

Nous devions aussi proposer plusieurs modes de jeu, avec d'une part une partie normale, générée aléatoirement, et d'autre part une partie avec graine, permettant de rejouer exactement la même configuration.

Un objectif complémentaire consistait à intégrer un mode difficile, accessible depuis un écran d'options, avec des objectifs supplémentaires pour rendre la partie plus exigeante et plus variée. Cette SAE avait également pour objectif de nous faire progresser sur des aspects essentiels du développement Android, comme la gestion de plusieurs écrans, les interactions tactiles, la sauvegarde de l'état de l'application lors des interruptions, ainsi que l'organisation claire du projet.

Enfin, ce projet nous a permis de développer notre rigueur, notre autonomie et notre capacité à travailler en équipe sur une application concrète, plus complète et plus proche d'un vrai projet de développement mobile.

I. INTRODUCTION

2.Objectifs du projet

L'objectif principal de cette SAE est de développer une application Android jouable, fonctionnelle et bien structurée. Ce projet nous a amenés à implémenter l'ensemble des règles du jeu Chuzzle tout en respectant les bonnes pratiques de développement Android. Concrètement, les objectifs techniques étaient les suivants :

- Implémenter la logique complète du jeu : grille 6x6, 7 types d'éléments, déplacements circulaires, détection et suppression des séries, chute des éléments après suppression.
- Gérer deux modes de jeu : la partie normale avec une grille aléatoire, et la partie destinée initialisée à partir d'un nombre choisi par le joueur.
- Afficher en permanence le score courant et le nombre de coups joués, sans rendre le jeu inaccessible aux daltoniens grâce aux symboles.
- Implémenter un mécanisme de verrous : certaines cases deviennent bloquantes et empêchent tout glissement de la ligne ou colonne qui les contient, jusqu'à ce qu'elles soient éliminées par une série.
- Implémenter une activité de configuration accessible uniquement depuis le menu principal, permettant d'activer le mode difficile avec des objectifs.
- Gérer correctement le cycle de vie Android : retour en arrière entre les écrans, mise en pause et reprise de la partie sans perte de données.
- Détecter automatiquement la fin de partie lorsqu'aucun coup valide n'est possible, puis afficher le résultat et le numéro de graine de la partie.
- Accélérer l'apparition des verrous au fil du jeu pour assurer une difficulté croissante et un rythme qui s'intensifie.

I. INTRODUCTION

3. Technologies employées

Pour réaliser ce projet, nous avons utilisé le langage **Java** pour développer l'ensemble de l'application et mettre en place la logique du jeu.

Le développement a été effectué avec **Android Studio**, qui nous a permis de concevoir les écrans, de tester l'application et de corriger les erreurs rencontrées pendant le projet.

La gestion de version a été assurée à l'aide de la plateforme de l'IUT, ce qui nous a permis de mieux travailler en équipe, de suivre les modifications et de répartir les tâches plus efficacement.

Pour les diagrammes, nous avons utilisé **PlantUML**, afin de représenter plus clairement la structure générale du projet.

Enfin, nous avons utilisé **Canva** et Word pour rédiger et mettre en page le rapport de façon claire et soignée.



Pour rédiger, compiler et exécuter le code source de l'application, tout en bénéficiant d'un environnement de développement clair et efficace.



Pour versionner les codes et collaborer en équipe efficacement via Grond tout au long du développement



Pour rédiger, mettre en page et illustrer le rapport de projet, en assurant une présentation claire et professionnelle du travail réalisé.

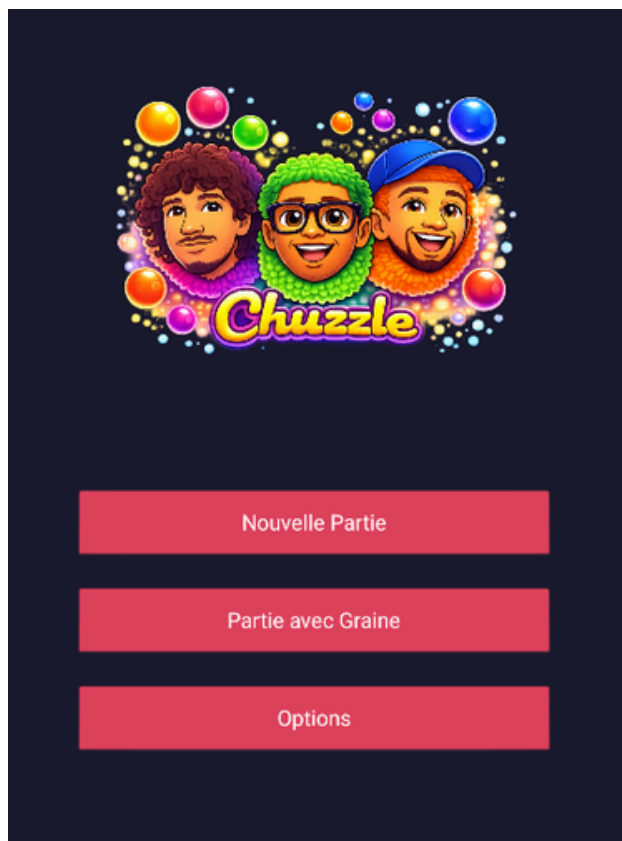
II. PRÉSENTATION GÉNÉRALE ET FONCTIONNEMENT

1. Menu principal et navigation

Lorsque l'application est lancée, le joueur arrive sur un **menu principal** composé de trois boutons :

- **Nouvelle Partie** : lance immédiatement une partie avec une grille générée de façon aléatoire.
- **Partie avec Graine** : affiche un écran intermédiaire permettant au joueur de saisir un nombre de son choix. Ce nombre sert à initialiser le générateur aléatoire, ce qui garantit une partie identique à chaque fois qu'on l'utilise.
- **Options** : ouvre l'écran de configuration du mode difficile, accessible uniquement depuis le menu.

Chaque écran de l'application est géré indépendamment. La navigation entre eux suit les règles Android habituelles : un retour depuis l'écran de jeu ramène au menu, et un retour depuis le menu ferme l'application. Une partie en cours n'est jamais perdue si l'application est mise en pause.



II. PRÉSENTATION GÉNÉRALE ET FONCTIONNEMENT

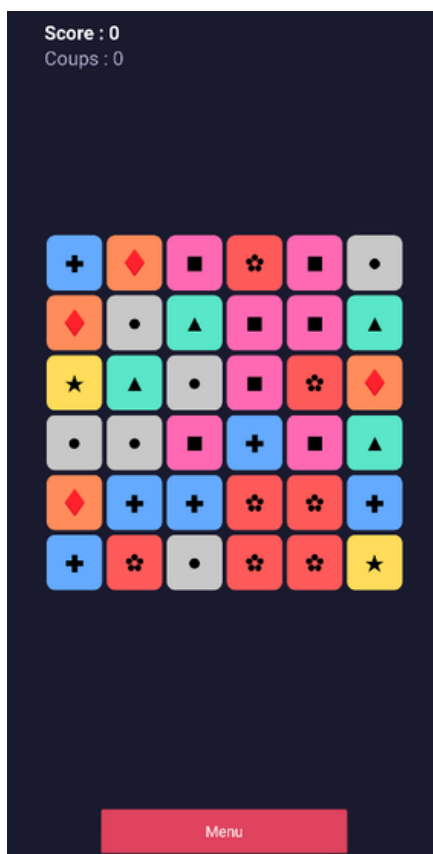
2. Grille de jeu et déplacements

L'écran de jeu affiche en permanence le score courant et le nombre de coups joués en haut de l'écran. La grille 6x6 occupe la partie centrale. Un bouton Menu permet de revenir au menu sans perdre la partie en cours.

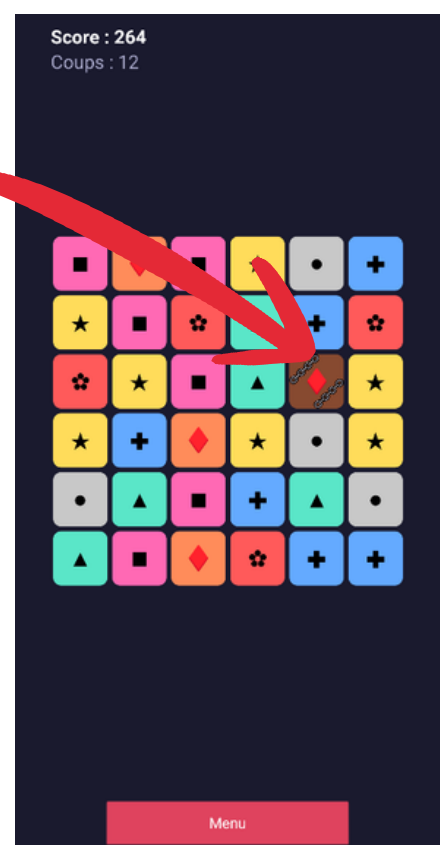
Un coup consiste à choisir une ligne ou une colonne et à la faire glisser horizontalement ou verticalement d'un ou plusieurs rangs. Le glissement est circulaire : l'élément qui sort d'un côté réapparaît de l'autre côté. Un coup n'est autorisé que s'il crée au moins une série de trois éléments identiques consécutifs. Dans le cas contraire, la grille revient à son état précédent et le coup n'est pas comptabilisé.

Toutes les séries créées lors d'un coup sont retirées en même temps. Les éléments situés au-dessus des cases vides tombent vers le bas, et les cases libérées en haut de chaque colonne sont remplies par de nouveaux éléments aléatoires. Ce processus se répète automatiquement en cascade jusqu'à ce qu'aucune nouvelle série n'apparaisse.

Parfois, à la fin de la résolution d'un coup, une case devient verrouillée. Il n'est alors plus possible de faire glisser la ligne ou la colonne qui la contient, jusqu'à ce que cet élément participe à une série et soit retiré de la grille. La fréquence d'apparition des verrous augmente au fil des coups.



Verrou



II. PRÉSENTATION GÉNÉRALE ET FONCTIONNEMENT

3. Partie avec graine et Hard Mode

Partie avec graine

En choisissant l'option correspondante depuis le menu, le joueur est dirigé vers un écran de saisie où il peut entrer un nombre entier. Ce nombre sert de graine au générateur aléatoire : deux parties lancées avec la même graine produiront exactement la même grille de départ et la même suite d'éléments générés au fil des coups. Ce mode permet de partager une partie précise ou de la rejouer à l'identique.

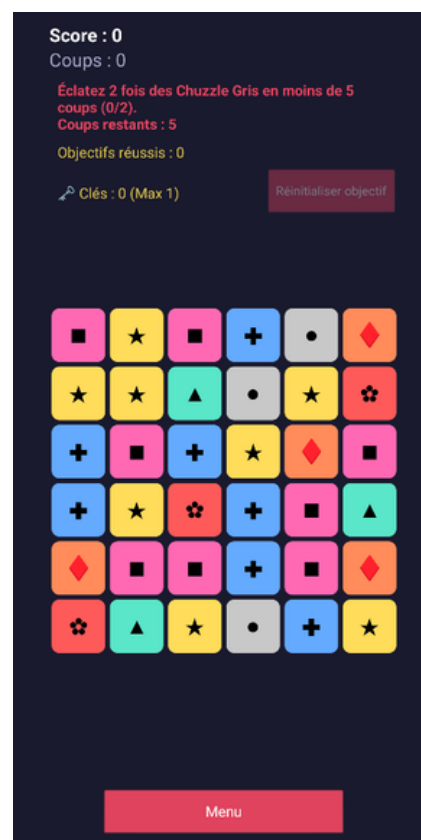
A la fin d'une partie normale, la graine utilisée est affichée afin que le joueur puisse la noter et reproduire cette partie ultérieurement.

Mode difficile (Hard Mode)

L'écran de configuration, accessible uniquement depuis le menu, permet d'activer le mode difficile via une case à cocher. Ce choix est mémorisé par l'application. En mode difficile, un objectif supplémentaire s'affiche en haut de l'écran de jeu, par exemple :

« *Faites exploser 1 fois des éléments roses en moins de 5 coups* ».

Le joueur dispose d'un stock de clé (au maximum 1) pour réinitialiser un objectif qu'il juge impossible à atteindre. Une clé est gagnée tous les deux objectifs réussis et tous les quatre coups réussis. Les objectifs sont générés selon un tirage pondéré qui favorise les défis équilibrés entre difficulté et faisabilité.



II. PRÉSENTATION GÉNÉRALE ET FONCTIONNEMENT

3. Calcul du score

Le score augmente à chaque coup valide. Chaque série retirée rapporte un certain nombre de **points de base** selon sa longueur :

Longueur de la série	Points de base
3 elements	8 points
4 elements	16 points
5 elements	32 points
6 elements ou plus	64 points

Lorsqu'un seul coup crée plusieurs séries en même temps, un **bonus de +50 %** est appliqué par série supplémentaire au-delà de la première. Autrement dit, plus un coup déclenche de séries simultanées, plus le multiplicateur est élevé.

Score du coup = total des points de base x (1 + nombre de séries supplémentaires x 0,5)

Exemple : si un coup retire deux séries de 3 éléments puis déclenche en cascade une série de 5 éléments, les points de base s'élèvent à $8 + 8 + 32 = 48$. Trois séries au total signifient deux séries supplémentaires, donc un multiplicateur de $1 + 2 \times 0,5 = 2$. Le score final du coup est donc **96 points**.

En mode difficile, les verrous apparaissent deux fois plus vite qu'en mode normal après chaque coup, ce qui rend la grille progressivement plus difficile à manoeuvrer.

III. STRUCTURE DU PROGRAMME

1. Structure et logique du code

L'application est organisée autour de **cinq écrans distincts**, chacun correspondant à une activité Android :

- **L'écran d'accueil** : point d'entrée de l'application. Il gère la navigation vers les autres écrans via les trois boutons du menu.
- **L'écran de saisie de graine** : permet au joueur d'entrer un nombre pour lancer une partie reproductible. Ce nombre est transmis à l'écran de jeu.
- **L'écran de jeu** : activité principale. Il affiche la grille, détecte les gestes de l'utilisateur, applique les règles du jeu et met à jour l'affichage en temps réel.
- **L'écran de configuration** : permet d'activer ou désactiver le mode difficile. Ce choix est enregistré localement et lu au démarrage de chaque partie.
- **L'écran de fin de partie** : affiche le score final et la graine utilisée. Il propose de rejouer ou de revenir au menu.

La **logique du jeu** est entièrement séparée de l'affichage. Cette séparation garantit que les règles du jeu ne dépendent d'aucun élément visuel, ce qui facilite les tests et la maintenance du code :

- La **grille** gère les déplacements circulaires, la détection des séries, la chute des éléments et l'ajout de verrous.
- Le **gestionnaire d'état** conserve le score, le nombre de coups, l'état de fin de partie et assure la sauvegarde lors des interruptions.
- Le **gestionnaire d'objectifs** génère les défis du mode difficile et gère le stock de clés du joueur.
- La **vue de la grille** se charge uniquement de dessiner la grille à l'écran, avec les couleurs et symboles de chaque case.

III. STRUCTURE DU PROGRAMME

1.2 Architecture du code

1) Le projet contient plusieurs écrans principaux :

- MenuActivity : affiche le menu principal et permet d'accéder à une nouvelle partie, à la partie avec graine ou aux options.
- SeedActivity : permet au joueur de saisir une graine avant de lancer une partie reproductible.
- OptionsActivity : permet d'activer ou de désactiver le mode difficile.
- MainActivity : correspond à l'écran principal de jeu. Il initialise la partie, la grille, le contrôleur et la gestion tactile.
- FinPartieActivity : affiche le score final, le nombre de coups, la graine utilisée et la grille finale.

2) Organisation en blocs (architecture) + MVC

Pour garder le projet lisible, nous avons structuré le code en plusieurs parties, en nous appuyant sur une organisation proche du modèle MVC (Modèle / Vue / Contrôleur).

- Modèle (Model) : gère les données et les règles du jeu.
- Vue (View) : affiche les écrans et dessine la grille.
- Contrôleur (Controller) : relie les actions du joueur à la logique du jeu.

Nous avons choisi cette organisation car elle permet de séparer plus clairement les responsabilités :

la logique du jeu reste indépendante de l'affichage, et l'interface ne contient pas directement les règles du jeu. Cette séparation rend le projet plus clair et plus simple à maintenir.

3) Rôle des classes (résumé)

Écrans de l'application

- MenuActivity : écran d'accueil de l'application.
- SeedActivity : écran de saisie d'une graine.
- OptionsActivity : écran des options.
- MainActivity : écran de jeu principal.
- FinPartieActivity : écran de fin de partie.

Logique du jeu

- EtatJeu : gère l'état global de la partie, le score, le nombre de coups, les séries, les cascades et la fin de partie.
- Plateau : gère la grille, les déplacements des lignes et des colonnes, les verrous, la détection des séries et la chute des éléments.
- Objectif : représente un objectif du mode difficile, avec sa couleur cible, son nombre de séries à réaliser et le nombre de coups disponibles.
- GestionnaireObjectifs : génère les objectifs, compte les réussites et gère le stock de clés du joueur.

Affichage

- VueGrille : dessine la grille, les couleurs, les symboles, les verrous et les effets de glissement.

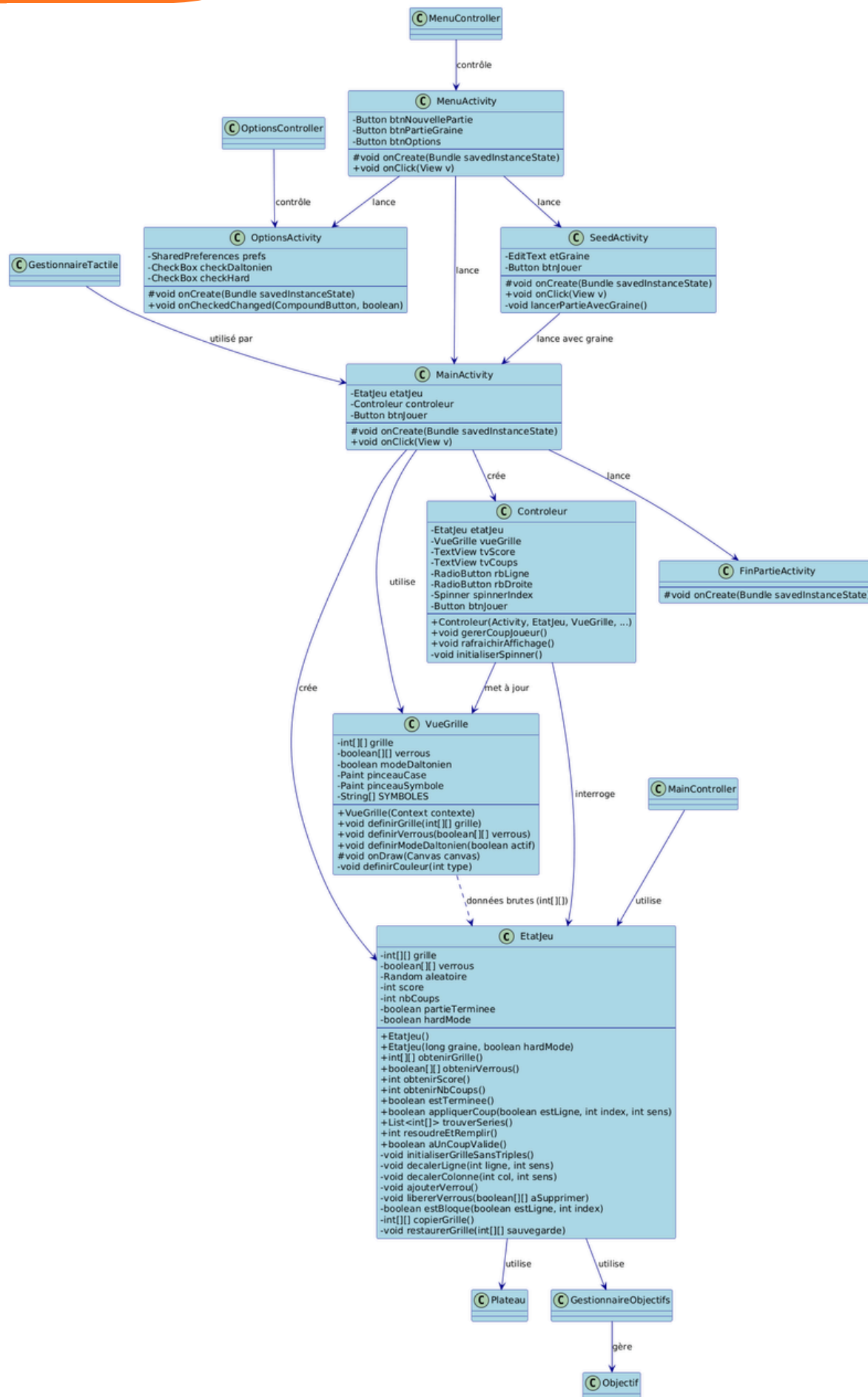
Contrôleurs / interactions

- Controleur : met à jour l'affichage, gère les objectifs, contrôle la progression de la partie et déclenche la fin de partie.
- GestionnaireTactile : interprète les gestes du joueur sur la grille et transforme les glissements en déplacements de lignes ou de colonnes.
- MenuController : gère les actions des boutons du menu principal.
- MainController : gère le retour au menu depuis l'écran de jeu.
- OptionsController : gère l'activation du mode difficile et le bouton retour dans l'écran des options.

Dans l'ensemble, cette organisation nous a permis d'obtenir un projet plus clair, mieux structuré et plus facile à faire évoluer. Chaque partie du programme possède un rôle bien défini, ce qui nous a aidés à mieux répartir les tâches et à mieux comprendre le fonctionnement global de l'application.

III. STRUCTURE DU PROGRAMME

2. Diagramme de classe simplifié



IV. ALGORITHME DE FIN DE PARTIE

1. Principe de détection

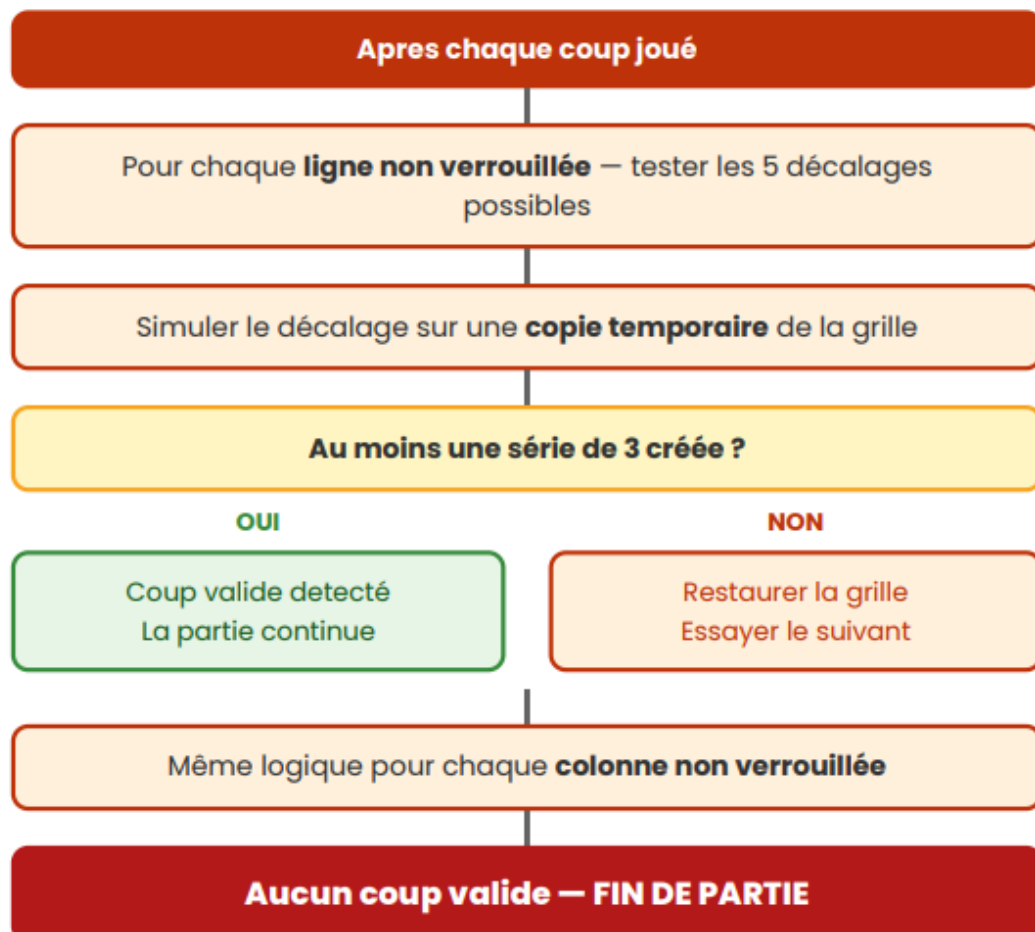
La partie est déclarée terminée lorsqu'**aucun coup valide n'est encore possible** sur la grille courante. Un coup est considéré comme valide s'il permet de créer au moins une série de trois éléments identiques, horizontalement ou verticalement.

Le jeu vérifie automatiquement cette situation après chaque coup. Lorsqu'aucun mouvement ne permet plus de former une série, la partie s'arrête et le résultat final est affiché au joueur.

2. Description de l'algorithme

Pour savoir s'il reste au moins un coup possible, le programme teste les différents déplacements encore envisageables sur la grille. Il examine les lignes et les colonnes qui ne sont pas bloquées, puis simule leurs décalages possibles sur une copie temporaire de la grille.

Après chaque simulation, le jeu vérifie si une série de trois éléments ou plus est créée. Si au moins un déplacement produit une série, alors la partie continue. En revanche, si aucun test ne donne de résultat, cela signifie qu'il ne reste plus de coup utile et que la partie est terminée.



1. Contraintes techniques

Compréhension du jeu et des attentes du sujet

L'une des premières difficultés rencontrées a été de bien comprendre le fonctionnement exact du jeu ainsi que le résultat attendu dans le cadre de la SAE. Au début, certains points n'étaient pas totalement clairs, notamment sur la manière dont les déplacements devaient fonctionner, sur la gestion des séries, ou encore sur les contraintes liées à la fin de partie. Nous avons donc dû relire le sujet, tester plusieurs idées et clarifier progressivement ce qui était réellement attendu.

Mise en place de la logique du jeu

Une fois le fonctionnement mieux compris, la mise en place de l'algorithme a représenté une difficulté importante. Il ne suffisait pas de faire apparaître une grille à l'écran : il fallait aussi gérer correctement les déplacements, détecter les séries, déclencher les cascades, mettre à jour le score et conserver un comportement cohérent dans tous les cas. La détection de la fin de partie a été l'un des points les plus délicats, car il fallait vérifier précisément s'il restait encore au moins un coup valide sur la grille.

Gestion du tactile

Distinguer un simple appui d'un glissement horizontal ou vertical s'est avéré complexe. Il a fallu définir un seuil minimal de déplacement pour ignorer les micro-mouvements involontaires, et déterminer la direction principale du geste selon le rapport entre le déplacement horizontal et vertical.

Choix et équilibrage du mode difficile

Enfin, nous avons beaucoup réfléchi au mode difficile à implémenter. Il ne s'agissait pas seulement d'ajouter une difficulté artificielle, mais de proposer quelque chose d'intéressant, compréhensible et réellement jouable. Nous avons donc pris du temps pour choisir une forme de Hard Mode qui apporte un vrai défi au joueur, sans rendre la partie injuste ou trop frustrante.

1.1 Conclusion personnelle de chaque membre

Jenson VAL

Cette SAE a été pour moi une bonne opportunité d'approfondir mes connaissances en Java tout en découvrant Android. Ayant déjà de l'expérience avec le langage, j'ai pu me concentrer davantage sur la logique du jeu et les spécificités du développement mobile.

La partie qui m'a le plus intéressé est la conception de l'algorithme de détection de fin de partie. Le principe consiste à simuler tous les mouvements possibles sur une copie de la grille, vérifier si au moins l'un d'eux crée une série, puis effacer cette copie. C'est une approche simple mais efficace, et son application dans un jeu réel m'a permis de mieux comprendre l'intérêt des algorithmes de recherche par simulation.

En revanche, l'un des aspects les plus délicats a été de bien organiser les classes qui participent à la logique du jeu et de réussir à les faire fonctionner ensemble de manière cohérente. Il ne suffisait pas que chaque partie marche séparément : il fallait aussi que la gestion de la grille, les déplacements, la détection des séries et la fin de partie restent bien liées entre elles, sans rendre le code confus. Ce travail m'a permis de mieux comprendre l'importance d'une bonne structure de programme et du lien entre les différentes classes dans un projet plus complexe.

Ce projet m'a montré qu'un jeu en apparence simple cache une vraie complexité technique dès lors qu'on cherche à le rendre robuste et agréable. C'est une leçon que je retiendrai pour mes futurs projets.

1.2 Conclusion personnelle de chaque membre

Séri-Khane YOLOU

Cette SAE m'a permis de découvrir le développement Android de manière concrète et approfondie. Au début du projet, la gestion des différents écrans et le cycle de vie d'une application mobile me semblaient complexes, mais en avançant étape par étape, j'ai pu mieux comprendre comment les différentes parties d'une application communiquent entre elles et comment l'état d'une partie survit aux interruptions.

La partie qui m'a demandé le plus d'efforts est la gestion des séries en cascade : quand la chute des éléments après une suppression déclenche de nouvelles suppressions, il faut gérer chaque étape dans le bon ordre. Il a fallu plusieurs essais avant d'aboutir à un résultat fiable, notamment pour synchroniser la suppression des séries, la chute des éléments et la levée des verrous.

Ce projet m'a appris l'importance de bien séparer la logique du jeu de l'affichage. Lors de nos premières versions, tout était mélangé dans le même écran, ce qui rendait le code difficile à faire évoluer. Reorganiser le code pour que chaque partie ait un rôle précis a été une leçon précieuse pour l'avenir.

Travailler en trinôme sur Gitea nous a permis de répartir efficacement les tâches. Cette SAE m'a permis d'appliquer des concepts algorithmiques dans un contexte concret et motivant.

1.3 Conclusion personnelle de chaque membre

Aylane SEHL

Pour ma part, cette SAE a été une expérience enrichissante, bien que parfois difficile. J'ai particulièrement apprécié le côté concret du projet : développer un jeu qui tourne réellement sur téléphone est très motivant par rapport aux exercices habituels. Voir l'application fonctionner sur un appareil réel a été une vraie satisfaction.

La gestion des interactions tactiles a été l'une des parties les plus complexes à mettre en place. Il a fallu distinguer un simple appui d'un glissement, et ignorer les mouvements trop courts pour être intentionnels. Plusieurs versions ont été nécessaires avant d'obtenir une détection fiable et agréable pour le joueur.

J'ai beaucoup appris sur la structure d'un projet Android : comment les écrans communiquent entre eux, comment on enregistre des préférences localement, comment on transmet des informations d'un écran à l'autre. Ces mécanismes me semblaient abstraits au début, mais leur utilisation répétée m'a permis de les intégrer naturellement.

La gestion du mode difficile, avec ses objectifs qui changent dynamiquement et son système de clés, a été la fonctionnalité qui m'a demandé le plus de réflexion en termes de conception. Globalement, cette SAE m'a permis de progresser significativement en développement mobile.

Projet réalisé par



- **Aylane SEHL**



- **Séri-Khane YOLOU**



- **Jenson VAL**

Enseignant : LUC HERNANDEZ

