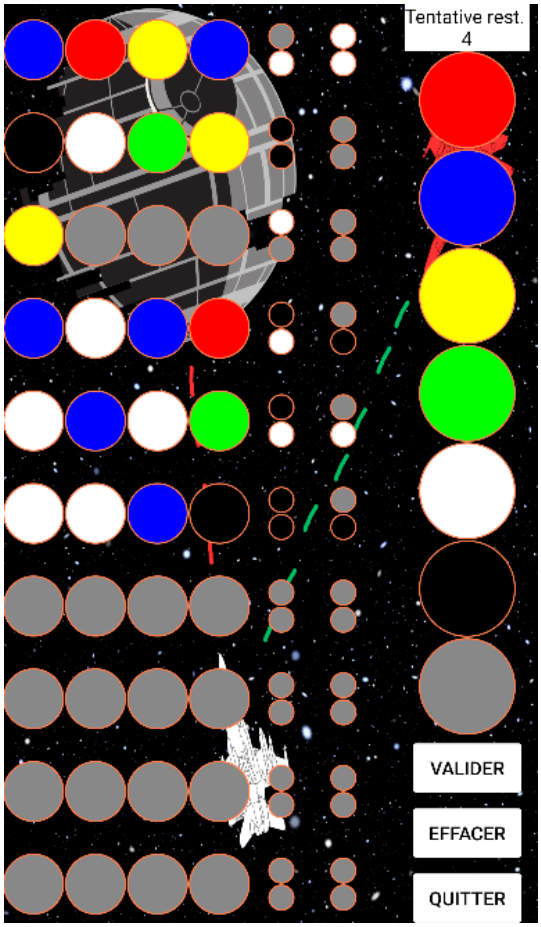
# Rapport - SAE41_2022

Bilal Boudjemline - Ethan Brezeky - Ethan Lefeuvre

9 Avril 2023

**Résumé**

Ce projet a pour but de créer un Mastermind en Java x Android. Pour ce faire, on devait se restreindre à l'API Officiel de Java (et celle d'Android) Ce jeu doit avoir trois fonctionnalités. La première, un menu permettant de sélectionner un des deux mode de jeu. La deuxième, un mode de jeu Joueur contre "Robot", la combinaison gagnante être généré aléatoirement. La troisieme, un mode de jeu Joueur contre Joueur en Hot Seat, la combinaison gagnante et les pions de vérification sont définis par un Joueur n. **Etant "chef" du GIT, Bilal Boudjemline a tout push sur la branche master.**



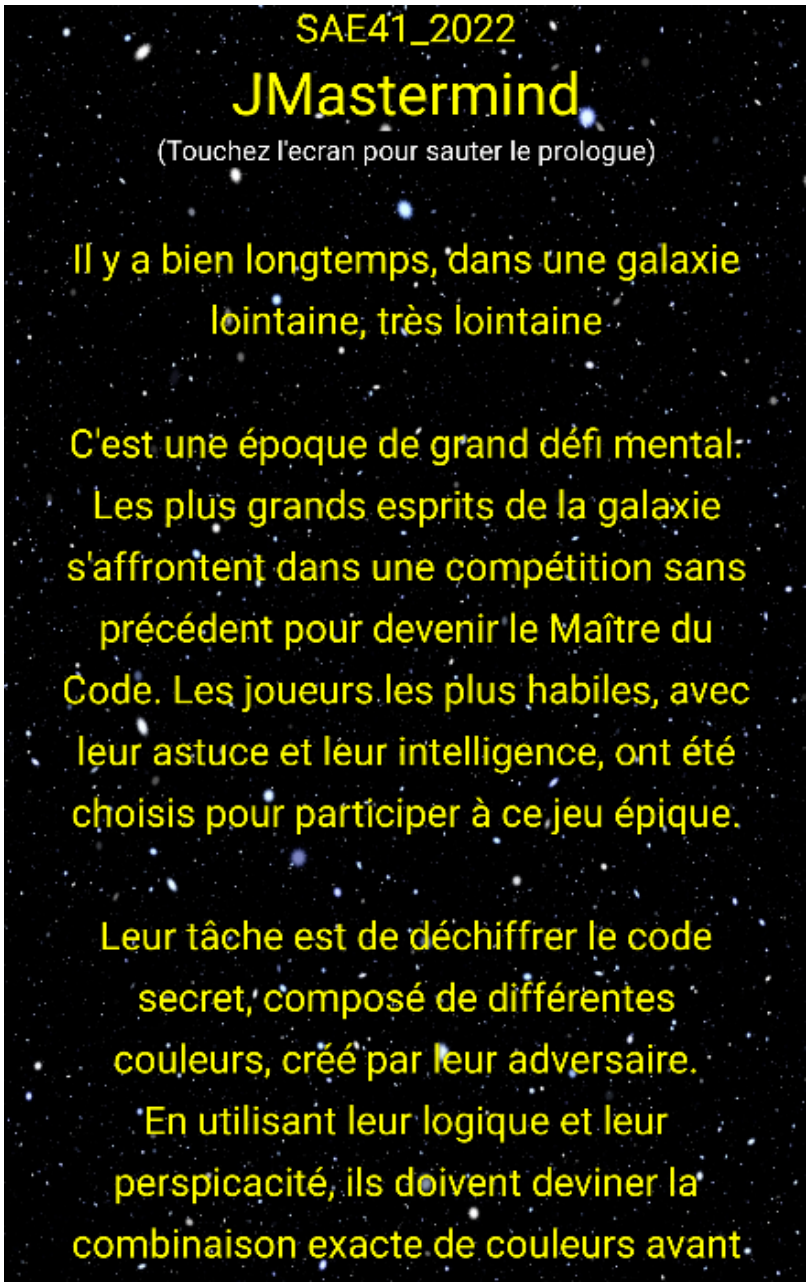

## 1 Inspiration

Ce Mastermind represente une sorte de tournoi se deroulant dans le monde de Star Wars. Les audios ont ete pris sur des videos youtubes de createurs ayant immite au piano les themes sonnores utiliser (credits en annexe). Les images sont fait main a l'aide de l'editeur de photo Canva.

# 2 Fonctionnalites

## 2.1 Prologue (Ethan L.)

Cette animation permet aux joueurs de se mettre dans le contexte "Star Wars". Ils peuvent decider si oui ou non ils veulent la voir. Il suffit juste de cliquer n'importe ou sur l'écran pour la sauter. Cette animation est régie par **Prologue/** contenu dans la source du package. L'événement du clique pour sauter l'animation s'est déclenché dans **MenuEvent/** de **Events/** qui va faire une redirection vers l'activite de **Menu/**. (Le texte défile du bas vers le haut à l'aide de certaines class de **android.view.animation/** et **PrologueAnimation/** de **Events/**).




Ce prologue a le générique de Star Wars version piano.

## 2.2 Menu (Ethan B.)

Le menu comporte uniquement 3 options explicites "Joueur vs Robot", "Joueur contre Joueur" et "Regles du Jeu". Ce menu est paramètre dans Menu/ du package **Menu/**. Il y a notamment la fonctionnalite "Autoriser un pion vide" qui permettera, une fois actif, de pouvoir jouer des combinaisons possedant les pions de couleur Gris (vide).

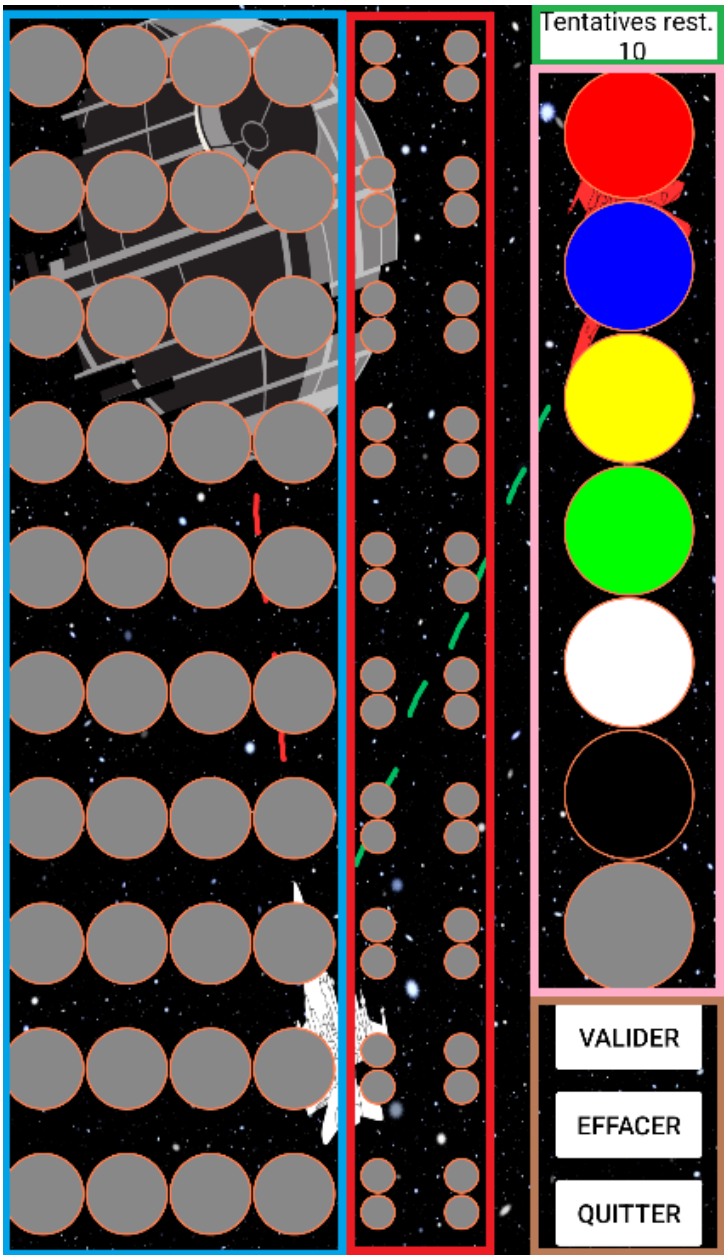

Ce menu a pour ambiance sonore "Duel of the Fate" version piano.

## 2.3 Joueur contre robot (Bilal B.)

Pour ce mode de jeu, nous avons génère aléatoirement la combinaison a l'aide du **Model/** du package **Game**/. Pour que la correction du projet soit plus facile, nous avons affiché un message dans le Logcat avec l'etiquette "COMBINAISON GAGNANTE".

```
COMBINAISON GAGNANTE    com.example.jmastermind          D  Jaune, Rouge, Blanc, Blanc,
```

Après la génération, le jeu donne sur un plateau vide comprenant 5 catégories. En bleu, la zone de dépôt de pions attaquant (**Circle/** de **Geometrics/**). En rouge, la zone de dépôt de pions défenseur (aussi **Circle/**). En vert, le nombre de tentatives restantes qui sera utilise pour déterminer le nombre de coups ($10 - tentativeRestantes$). En rose clair, la liste des couleurs disponibles, **le gris représentant "le pion vide"**. Pour finir, la zone des boutons sert à **valider** une combinaison, effacer la combinaison actuelle et **quitter** pour revenir au menu principal. (Voir image ci-dessous)



### 2.4 Joueur contre Joueur (Ethan B. Ethan L.)

Ce mode de jeu utilise la même la activité que "Joueur contre robot" a la différence qu'avant de lancer l'activité du jeu, il va lancer une activité **ColorDefSelect/** de **Game/** qui va demander au défenseur de choisir sa combinaison. La combinaison est alors envoyé au **Controller/** de **Game/** qui va orchestrer le plateau en fonction de chaque mode de jeu vien des "Extras" de la class **Intent/**.

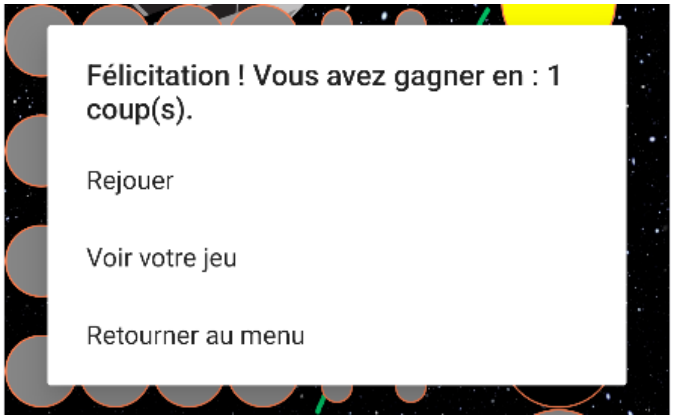

Le joueur peut **"Valider"** uniquement si tous les cercles sont rempli, **"Effacer"** pour effacer la combinaison et **"Retour"** pour retourner au menu principal.

## 2.5 Deroulement du jeu

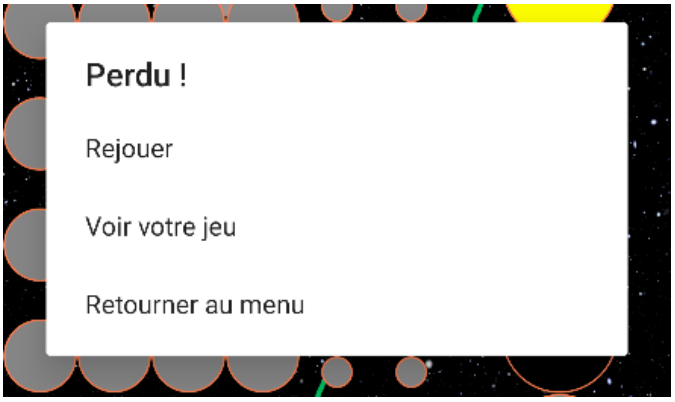
Quand l'utilisateur choisit une couleur, il clique sur un cercle ayant reçu pour événement **SelectionColorEvent/** de **Events/**. Une fois qu'il a fini sa combinaison compose de 4 pions maximum, il peut soit effacer sa combinaison pour x raisons, soit valider sa combinason et déclencher l'événement de **ControlEvent/** de **Events/** qui va declancher une fonction de **ColorSelectoEvent.goNext()**. Cette fonction s'occupera de placer tous les pions vérificateurs, stopper le jeu sur un pop-up si c'est gagne ou perdu, passer à la tentative suivante, décrémenter le nombre de tentatives et si la fonctionnalite "Autoriser un pion vide" n'est pas actif, stopper la fonction des le debut pour que l'attaquant finisse de remplir sa ligne.

## 2.6 Fin de jeu (Bilal B.)

Si l'attaquant gagne la partie, ce pop-up apparaîtra et lui proposera de faire ces trois actions ci-dessous.




Si l'attaquant perd la partie, ce pop-up apparaîtra et lui proposera de faire ces trois actions ci-dessous.




Ces actions sont directement reliées à **PopupEvent/** de **Events/**. "Rejouer" vas juste redémarrer l'activite **SoloGame/** de **Game/**, "Voir votre jeu" va enlever la notification de l'ecran pour pouvoir observer le jeu et empêcher le joueur de pouvoir re-cliquer sur "Valider". "Retourner au menu" va changer l'activité actuelle par **Menu/**.

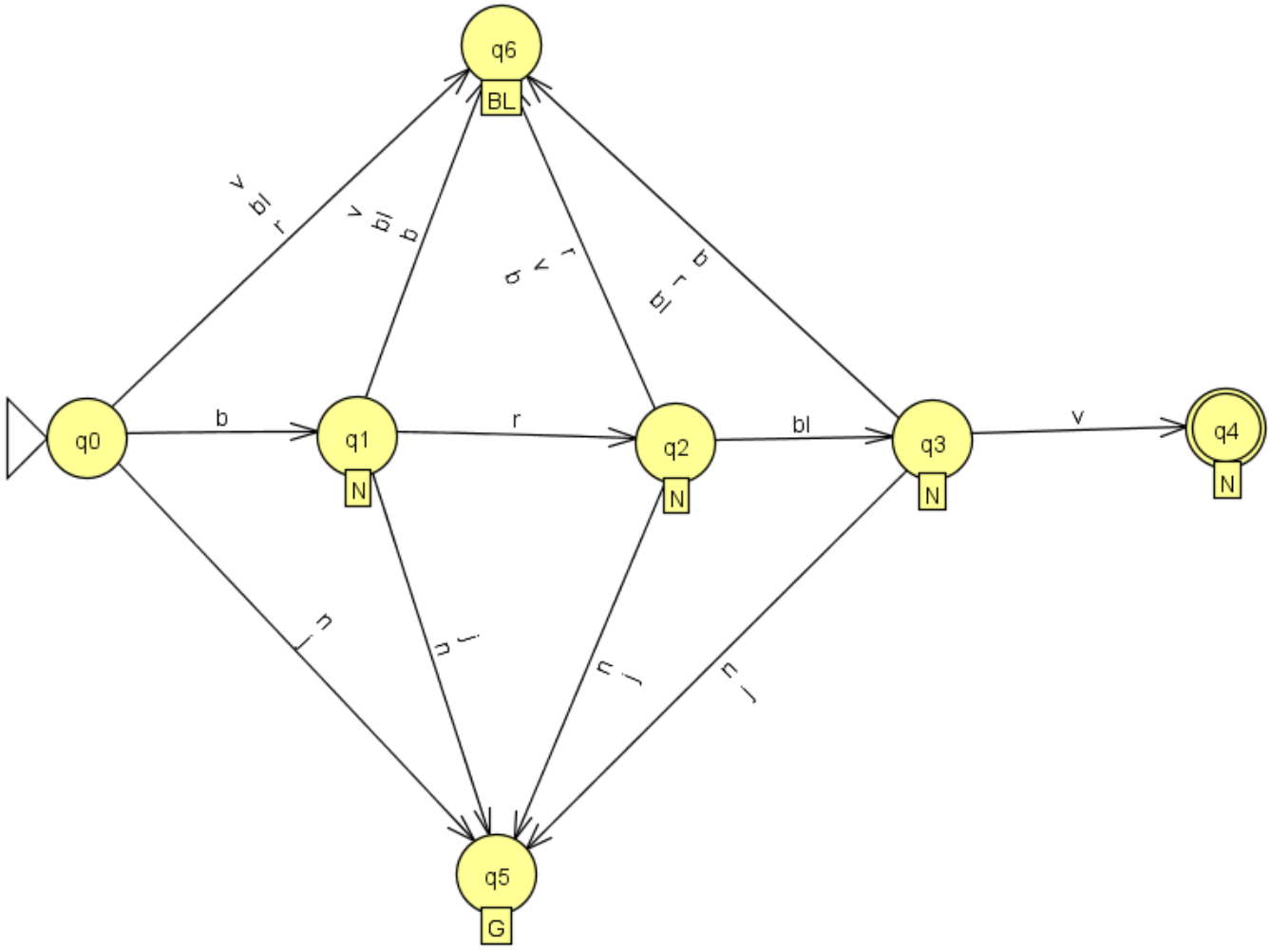
# 3 Fonctionnement du verificateur de pion (Bilal B.)

Le checkeur va passer au crible la combinaison de l'attaquant. Si le pion $\mathcal{P}i$ correspond au pion gagnant $\mathcal{W}i$, il va ajouter dans la liste des pions verifieur le pion noir. Si le pion $\mathcal{P}i$ est present dans la liste des pions gagnant $\mathcal{W}$, il ajoute dans la liste des pions verifieur le pion blanc. Si $\mathcal{P}i$ n'est ni egal a $\mathcal{W}i$ ou ni contenu dans $\mathcal{W}$, il ajoute dans la liste des pions verifieur le pion gris.

Voici un automate representant la combinaison gagnante **Bleu(b), Rouge(r), Blanc(bl), Vert(v)**.

Description :
$\sum = \{b, r, j, n, v, bl\}$
$\mathcal{Q} = \{0, 1, 2, 3, 4, 5, 6\}$
$\mathcal{F} = 4$

```
Les labels :
N   → Renvoie un pion noir (pion bien placé).
BL  → Renvoie un pion blanc (pion existant mais mal placé).
G   → Renvoie un pion grise (pion non existant).
```



# 4 Impression du projet

## 4.1 Bilal Boudjemline Groupe 1

Ce projet était vraiment plaisant à faire. Le monde du développement Android ne m'intéresser pas trop mais j'ai remarqué que la créativité qu'il me procurait était semblable a ma passion pour le web. J'espère que notre application vous plaira.

## 4.2 Ethan Brezeky Groupe 2

Malgres la la connaissance du jeu, j'ai apprécié travaillait sur ce projet. Les idées de Bilal pour le thème Star Wars me plaisent beaucoup.

## 4.3 Ethan Lefeuvre Groupe 2

Après avoir déjà developpé un mastermind en Java l'année dernière, je ne pensais pas que cette application poserait quelquoncque défi. Pourtant, je me suis bien pris la tête dessus, ce qui a rendu l'expérience assez pénible mais surtout très instructive. Avec Bilal qui nous a beaucoup aidé avec ses idée créatives et son vérificateur, je garde un bon souvenir de ce projet qui je l'espère saura vous plaire.

# 5 Annexe

## 5.1 Medias

Audio pour l'activite **R.layout.prologue**

Audio pour l'activite **R.layout.menu**

Audio pour le son de **victoire**

Audio pour le son de **defaite**

Montage photo

## 5.2 Directives

Consignes

Java Docs

Android Docs

## 5.3 Autre

StackOverFlow (Aides pour erreurs : Animation et Extras d'Intent.)

JFLAP (Pour l'automate)

StarUML (Pour les diagrammes UML)

# FIN