

Projet Android: Coloris

1) Descriptif et règles du projet

Le but de notre projet est de créer une application Android qui permet de jouer au Coloris. Le Coloris est un jeu de type casse-tête dont les règles sont assez simples. L'interface du jeu est composée d'une grille de 8*8 cellules que l'on nomme la carte centrale et de trois triplets qui nous serviront à remplir cette carte et marquer des points. Un triplet est un élément qui comporte 3 cases de couleurs et pouvant subir des rotations, les couleurs sont initialisées aléatoirement lors de la création du triplet. Pour marquer des points, on fait glisser un triplet sur la carte, si 3 cases ou plus de la même couleurs se touchent, alors elles disparaissent de la carte et le score augmente de 1 point pour chaque cellule détruite. La partie se termine lorsque la carte est remplie ou si le temps est écoulé, une partie dure au maximum 60 secondes.

L'application est créée avec l'IDE Android Studio sur lequel on vient 'linker' un projet GITHUB. Ce lien nous permettra de gérer facilement le versionning et le travail collaboratif. Avant de travailler sur le projet, on pull pour obtenir la dernière version, après avoir travaillé on effectue un commit puis un push pour mettre à jour le dossier sur le serveur GIT.

2) Modélisation de l'application

Notre application est composée de 4 classes:

Menu: cette classe est utilisée pour gérer le menu de l'application

p8.demo.p8coloris::Menu
-state: int -userData: UserData
+onCreate(Bundle savedInstanceState): void +New_game(View view): void +Continue(View view): void +Best_score(View view): void +System_settings(View view): void +Song_on(View view): void +Song_off(View view): void +Exit_game(View view): void +onBackPressed(): void +onResume(): void +onPause(): void +onStop(): void +onDestroy(): void #onActivityResult(int requestCode, int resultCode, Intent data): void

UserData: cette classe gère la persistance des données utilisateurs, elle enregistre dans la mémoire interne les préférences et la dernière partie en cours si elle existe

p8.demo.p8coloris::UserData
<div><div>-tabHighScore = new int[10]: int[] -tabNameHighScore = new String[10]: String[] -activeSound = true: boolean -prefCharged = false: boolean -gameSaved = false: boolean -timer: int -score: int -gameGrid = new int[8][8]: int[][] -tripletTab = new int[3][3]: int[][] -orientationTab = new int[3]: int[] -context: Context</div><div>+UserData(Context context): ctor +getTimer(): int +setTimer(int timer): void +setTripletTab(int tripletTab[][]): void +getTripletTab(): int[][] +setOrientationTab(int orientationTab[]): void +getOrientationTab(): int[] +setGameGrid(int gameGrid[][]): void +getGameGrid(): int[][] +getTabHighScore(): int[] +getHighScoreAtIndex(int index): int +newHighScore(int highScore, String name): void +getTabNameHighScore(): String[] +getNameHighScoreAtIndex(int index): String +setActiveSound(boolean value): void +getActiveSound(): boolean +setPrefCharged(boolean value): void +getPrefCharged(): boolean +setGameSaved(boolean value): void +getGameSaved(): boolean +readUserData(): void +writeUserData(): void +writeUserConfigData(): void +writeUserGameData(): void +debugLog(): void -readConfigFile(): boolean -writeConfigFile(): void -readGameFile(): boolean -writeGameFile(): void +getScore(): int +setScore(int score): void</div></div>

Coloris: cette classe est l'activité dans laquelle le jeu se déroulera

p8.demo.p8coloris::Coloris
-mColorisView: ColorisView -userData: UserData
+ onCreate(Bundle savedInstanceState): void + saveUserOk(View view): void + onBackPressed(): void + onResume(): void + onPause(): void + onStop(): void + onDestroy(): void

ColorisView: cette classe est la surface du jeu, c'est elle qui contient les éléments pour afficher le jeu et gérer l'interaction avec l'utilisateur. (voir tableau page suivante)

Le déroulement de la classe ColorisView s'effectue en 3 grandes étapes:

- Initialisation des données:
C'est durant cette étape que toutes les données (images, sons, grille 8*8, triplets, timer) sont chargés/rechargés/initialisées à 0 ou à leur valeurs d'initialisation correspondantes et la création puis le lancement du thread d'affichage.
Cela s'effectue dans le constructeur de la classe et la sous fonction « initparameters »
- Exécution du thread d'affichage:
Le thread d'affichage est le thread lancé à l'étape d'initialisation des données, il permet de décrémenter le timer ainsi que d'afficher tous les éléments du jeu à l'écran (grille de 8*8, triplet, score, temps, nextHighScore, Game Over).
Cela s'effectue dans la fonction run() du thread d'affichage et cette fonction s'arrête quand on passe le booléen globale « in » à false.
- Interaction et gestion des événements utilisateurs:
C'est ici que les inputs sur l'écran tactiles (le scrolling/rotation/re-génération d'un triplet, l'ajout d'un triplet sur la grille 8*8, la détection d'un alignement de couleur, les bruitages du jeu et le clique permettant de passer à la saisie du pseudo pour le score) sont réalisés.
Cela se s'effectue dans un onTouchListener.

```

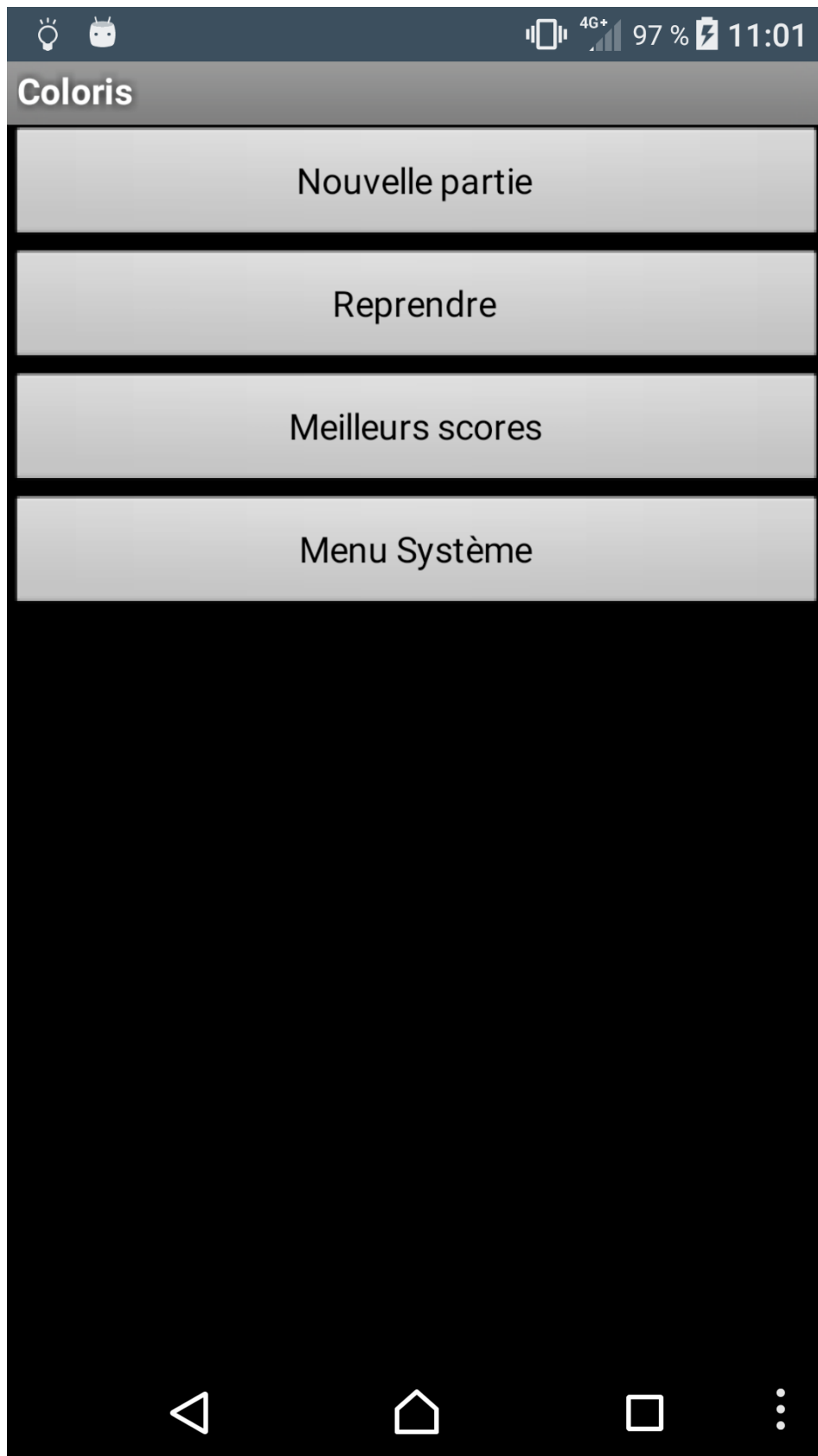
-userData: UserData
-songPose: MediaPlayer
-songAlignement: MediaPlayer
-songFlip: MediaPlayer
-block_vide: Bitmap
-block_bleu: Bitmap
-block_blanc: Bitmap
-block_jaune: Bitmap
-block_rose: Bitmap
-block_rouge: Bitmap
-block_vert: Bitmap
-fullGrid, endTime: Bitmap
-mRes: Resources
-mContext: Context
-reload = true: Boolean
-score = 0: int
-time: Integer
~t1 = 0: long
~t2 = 0: long
~tDiff = 0: long
~pause = false: Boolean
~carte = new int[8][8]: int[][]
-tripletTab = new int[3][3]: int[][]
-orientation[] = { 0, 0, 0 }: int
-ontouchtab = { { 0, 0 }, { 0, 0 }, { 0, 0 } }: float[][]
~carteTopAnchor: int
~carteLeftAnchor: int
~carteWidth = 8: int
~carteHeight = 8: int
~carteTileSize = 33: int
~CST_block_vide = 0: int
~CST_block_blanc = 1: int
~CST_block_bleu = 2: int
~CST_block_jaune = 3: int
~CST_block_rose = 4: int
~CST_block_rouge = 5: int
~CST_block_vert = 6: int
~ref = { { /* reference des blocks */ } }
+in = true: boolean
-cv_thread: Thread
~holder: SurfaceHolder
~paint: Paint
~lastindex = -1: int
~index = -1: int
~_otc = new OnTouchListener() {
}; OnTouchListener

+ColorisView(Context context, AttributeSet attrs): ctor
+loadlevel(): void
+initparameters(): void
-paintEndTime(Canvas canvas): void
-paintFullGrid(Canvas canvas): void
-paintcarte(Canvas canvas): void
-paintTripletTab(Canvas canvas): void
-paintInfoBar(Canvas canvas): void
~nDraw(Canvas canvas): void
+surfaceChanged(SurfaceHolder holder, int format, int width, int height): void
+surfaceCreated(SurfaceHolder arg0): void
+surfaceDestroyed(SurfaceHolder arg0): void
+run(): void
+setOrientation(int[] tabOrientation): void
+setCarte(int[][] carte): void
+setTriplet(int[][] triplet): void
~hitTripletTab(MotionEvent event): int
+colorAlignement(int resx, int resy, boolean orientation): void
~hitcarte(MotionEvent event): boolean
+onTouchEvent(MotionEvent event): boolean
~reverseTab(int indice): void
+setUserData(UserData userData): void
+setScore(int score): void
+setTime(int time): void
+setReload(Boolean reload): void
+getScore(): int
+playSong(MediaPlayer song): void
+isFullGrid(): boolean
+exit(): void
+setPause(Boolean pause): void

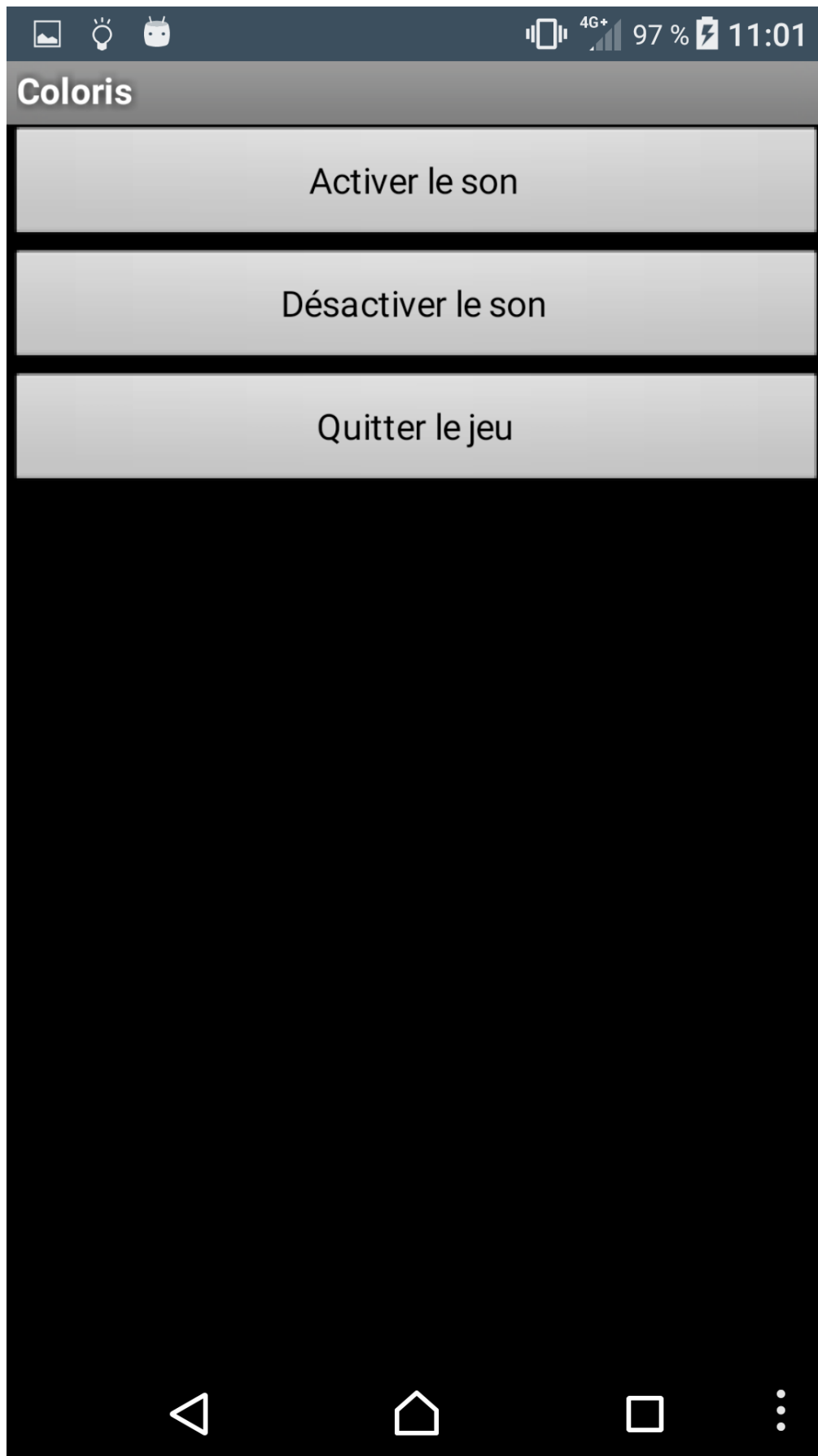
```

3) Vues de l'application

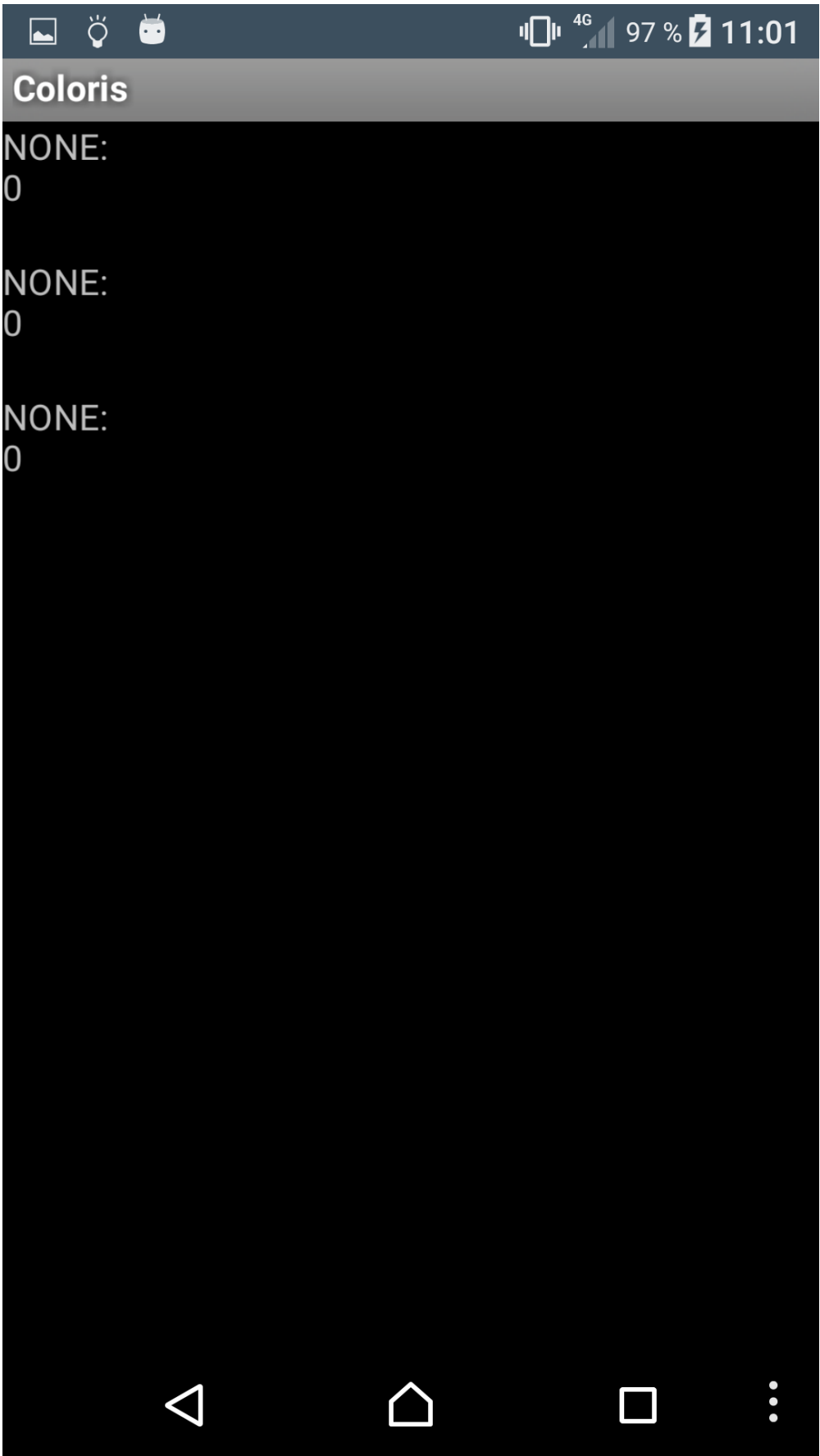
L'écran du menu d'accueil



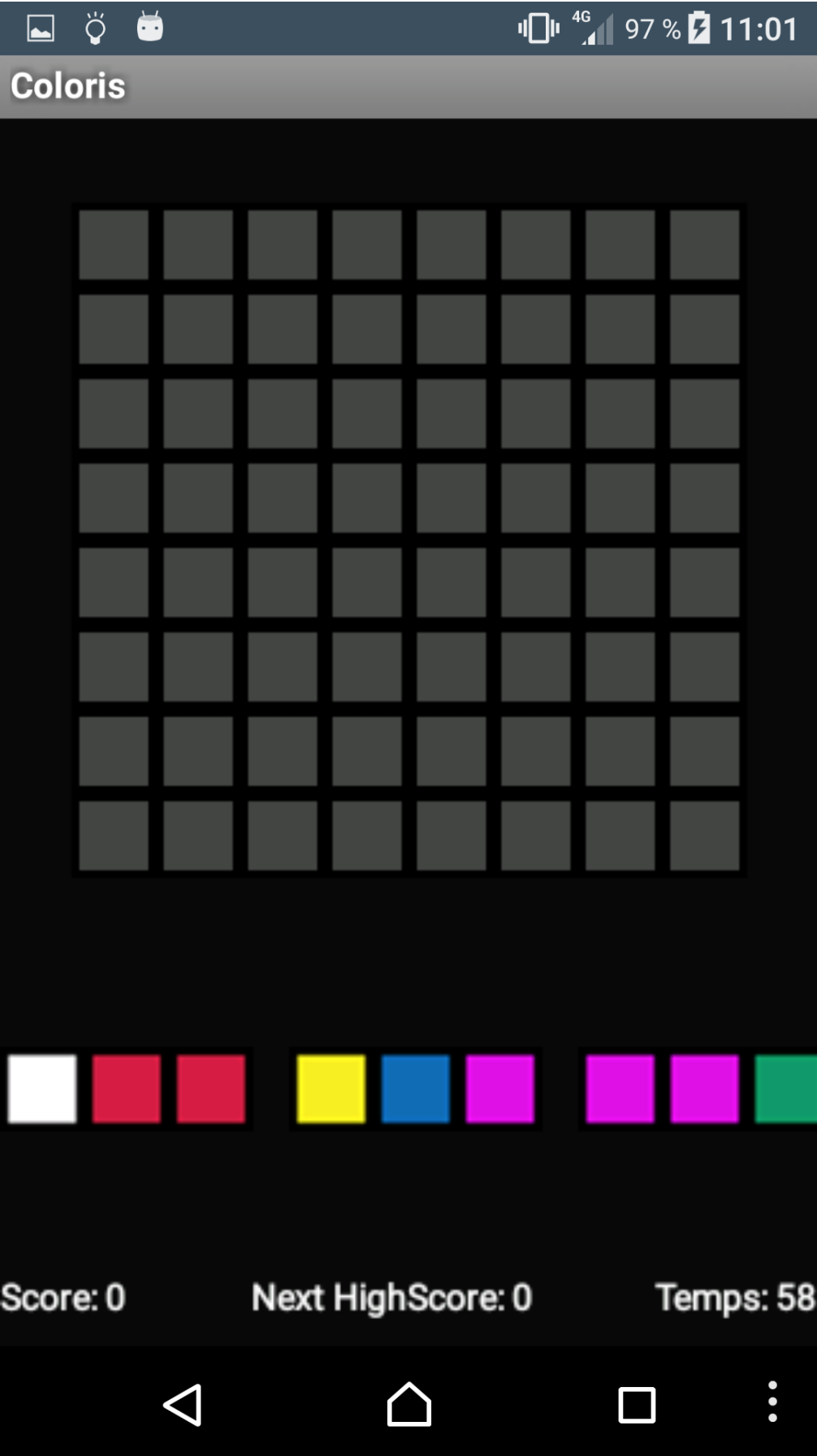
Sous menu pour le son



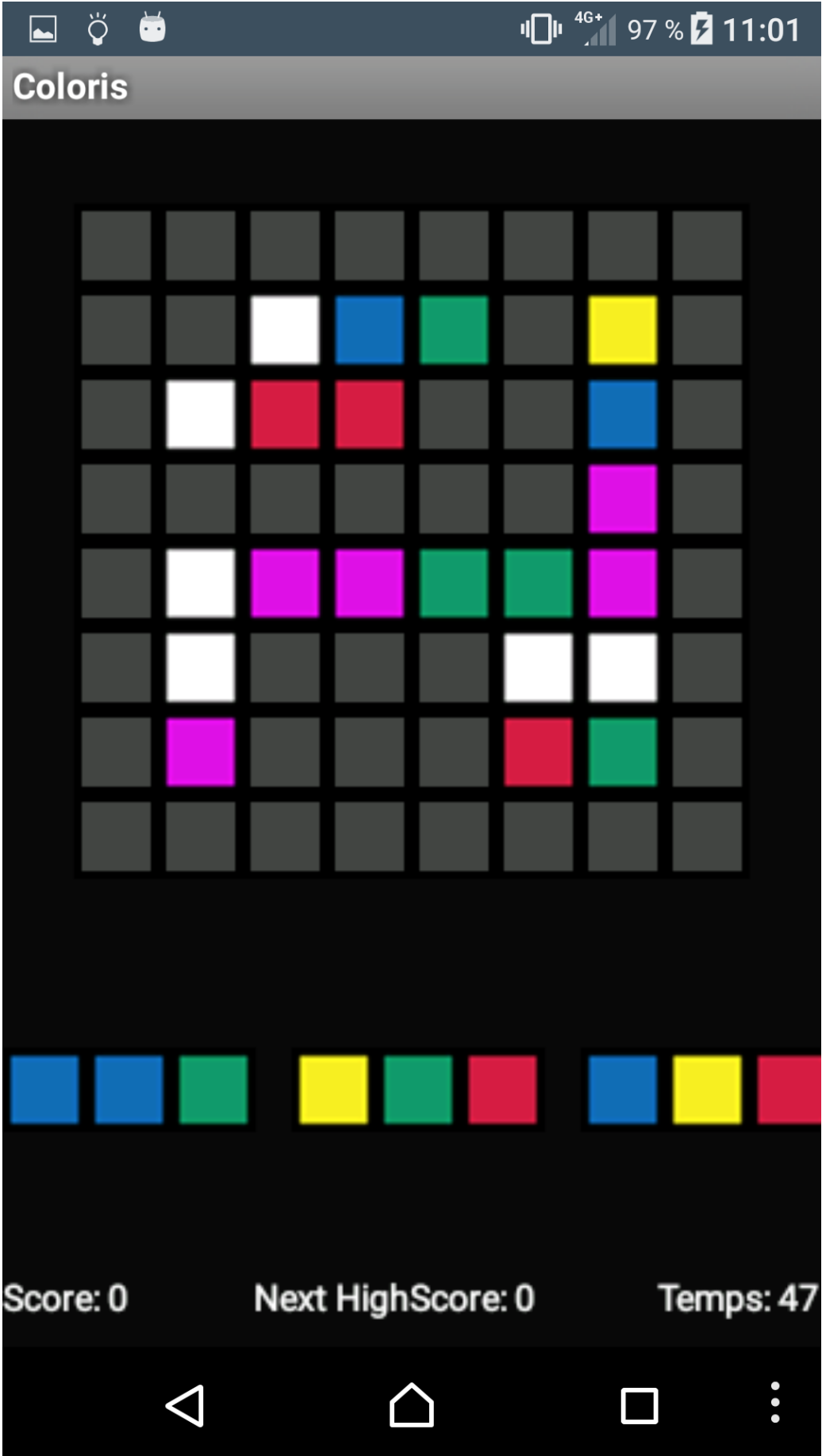
Sous menu Highscore:



Jeu à l'initialisation:



Jeu en progrès :



Fin de jeu :



Enregistrement du score :


Coloris

Nouveau Record !

Score: 34

Veuillez entrer votre nom:

OK



4) Problèmes rencontrés

Durant l'ensemble du projet, nous n'avons pas rencontré de problème majeur à part quelques petits conflits de versionning pendant l'utilisation de github. Cependant, un point nous à semblé intéressant à développer dans ce rapport, il s'agit du passage d'information dans un intent. De manière générale, un intent permet à une activity de faire une demande d'intention à une autre activity, cette demande peut s'accompagner de variables pour transmettre des informations. Un exemple d'intent qu'on utilise très souvent sur nos téléphone est l'intent de l'activity qui gère les contacts à l'activity qui gère les appels téléphoniques avec en extra la variable indiquant le numéro de téléphone. Dans notre cas, nous avons un intent de l'activity menu pour l'activity coloris pour passer du menu au jeu. Le jeu doit récupérer différentes informations selon si on crée une nouvelle partie ou si on charge une ancienne partie. Notre première idée ici était de passer dans l'intent non pas un ensemble de variable standard (string, int, float, ect...) mais directement une instance de notre classe UserData qui contient les informations nécessaire. Cependant, pour passer une classe en intent il faut que cette classe soit Serializable, en Android une classe est dite Serializable si elle implémente de façon fiable la classe Parcelable. Il nous faudrait donc implémenter la classe Parcelable à notre classe UserData et effectuer des Overrides pour 'couper' notre classe en un ensemble de string lors de la création du Parcelable et 'découper' ces string pour la recreation de la classe. Ce procédé nous a semblé un peu lourd par rapport à la faible masse d'information que nous devons passer. On a donc décidé de passer en Intent seulement les préférences de l'utilisateur et son choix de créer une nouvelle partie ou reprendre l'ancienne, l'activity coloris vas ensuite créer une nouvelle instance de UserData et lire les informations en mémoire interne.