RAPPORT Jeu 2048 avec Flutter

Table des matières

Le projet consiste à développer une version personnalisée du célèbre jeu "2048". Ce jeu de puzzle repose sur la combinaison de tuiles ayant les mêmes valeurs pour atteindre la valeur cible de 2048. L'objectif de ce projet est de créer une version améliorée avec des variantes sur le système de calcul, la grille, et des options permettant de personnaliser l'expérience du joueur.

1. Interface utilisateur

a) Flutter :

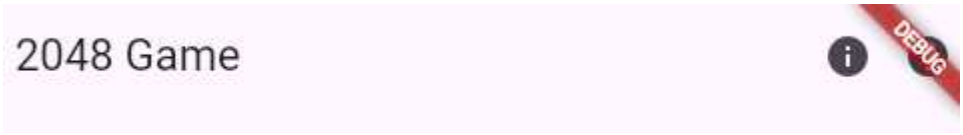
- Framework utilisé pour développer l'application. Il fournit une interface utilisateur fluide et réactive.

b) AppBar :

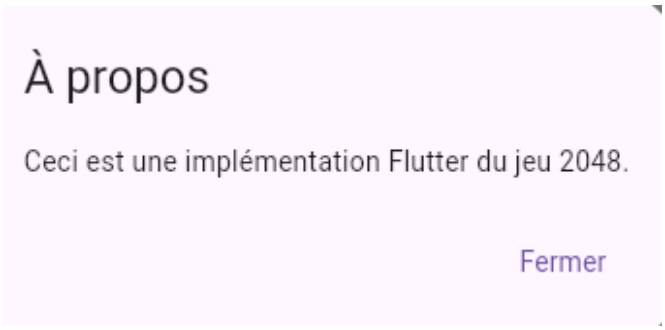
- Utilisée pour afficher le titre et les boutons "À propos" et "Aide".

```
appBar: AppBar(
  title: const Text('2048 Game'),
  actions: [
    IconButton(
      icon: const Icon(Icons.info),
      onPressed: _showAboutDialog,
    ),  // IconButton
    IconButton(
      icon: const Icon(Icons.help),
      onPressed: _showRulesDialog,
    ),  // IconButton
  ],
),  // AppBar
```

- Titre affichant "2048 Game".



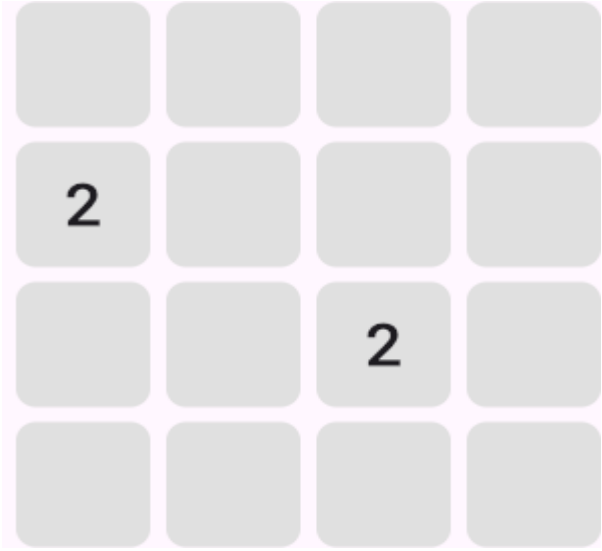
- Un bouton "À propos" qui affiche des informations sur le jeu.
- Un bouton "Aide" pour afficher les règles du jeu.



c) GridView.builder :

- Génération dynamique de la grille 4x4 pour afficher les tuiles.

```
child: GridView.builder(
  gridDelegate: const SliverGridDeleg
    crossAxisCount: 4,
    childAspectRatio: 1,
  ),  // SliverGridDelegateWithFixedC
```



- Un compteur de coups (**`moveCount`**) qui affiche le nombre de mouvements effectués.


```
Text(
  'Coups: $moveCount',
  style: const TextStyle(
),  // Text
```

2. Option et personnalisation

a) DropdownButton :

- Sélection de la valeur cible (128 à 2048)


```
DropdownButton<int>(
  value: targetValue,
  onChanged: _onDifficultyChange,
  items: const [
    DropdownMenuItem(value: 128, child: Text('128')),
    DropdownMenuItem(value: 256, child: Text('256')),
    DropdownMenuItem(value: 512, child: Text('512')),
    DropdownMenuItem(value: 1024, child: Text('1024')),
    DropdownMenuItem(value: 2048, child: Text('2048')),
```

b) Container :

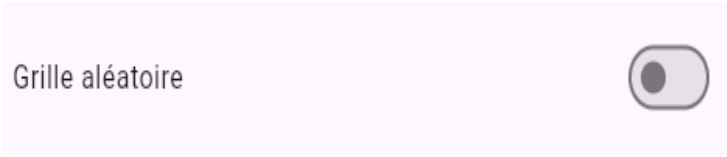
- Personnalisation visuelle des tuiles avec des couleurs basées sur leurs valeurs.

```
switch (value) {
  case 2:
    return Colors.grey[300]!;
  case 4:
    return Colors.yellow[400]!;
  case 8:
    return Colors.orange[400]!;
  case 16:
    return Colors.red[500]!;
  case 32:
    return Colors.pink[600]!;
```

c) Switch :

- Permet d'activer une grille initiale aléatoire (**`isRandomGrid`**).

```
SwitchListTile(
  title: const Text('Grille aléatoire'),
  value: isRandomGrid,
  onChanged: _onCheckBoxChanged,
),  // SwitchListTile
```



d) FloatingActionButton

- Bouton de réinitialisation pour redémarrer la partie.

```
floatingActionButton: FloatingActionButton(
  onPressed: _startNewGame,
  child: const Icon(Icons.refresh),
  tooltip: 'Nouvelle Partie',
),  // FloatingActionButton
```

3. Mécaniques de jeu :

a) GestureDetector :

- Capture des gestes de glissement (haut, bas, gauche, droite) pour déplacer les tuiles.

b) **setState** :

- Mise à jour de l'état de la grille et des scores après chaque mouvement.

c) **Random() :**

- Génération aléatoire des nouvelles tuiles après chaque action.

```
ody: GestureDetector(
 onVerticalDragEnd: (details) {
   if (details.velocity.pixelsPerSecond.dy > 0) {
     _onSwipe("down");
   } else {
     _onSwipe("up");
   }
 },
 onHorizontalDragEnd: (details) {
   if (details.velocity.pixelsPerSecond.dx > 0) {
     _onSwipe("right");
   } else {
     _onSwipe("left");
```

4. <u>Gestion des états :</u>

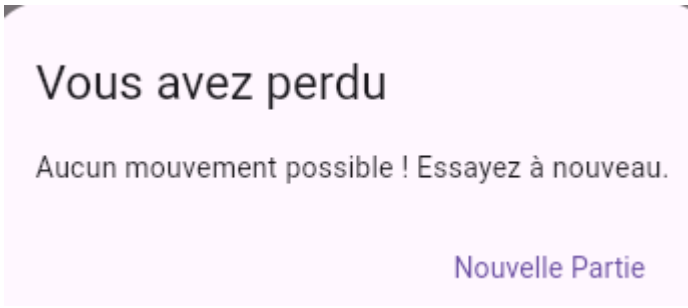
a) **StatefulWidget :**

- Gestion de l’état dynamique de la grille et des options du jeu.

b) **AlertDialog :**

- Affichage des fenêtres de victoire, défaite, aide et informations.

```
void _showCongratulationsDialog() {
  showDialog(
    context: context,
    barrierDismissible: false, // Empêche de fermer en cli
    builder: (context) {
      return AlertDialog(
        title: const Text('Félicitations !'),
        content: Text('Vous avez atteint $targetValue !'),
```



5. Animation

a) Animation :

Dans cette partie, une animation a été ajoutée pour créer un effet visuel dynamique avec des pétales animés, simulant un effet de dispersion de petites formes (pétales) qui bougent

- Utilisation du **`StatefulWidget`** pour l'animation
- **`AnimationController`** : Un contrôleur d'animation est utilisé pour gérer la durée et le type de l'animation.
- Animation avec **`Tween`** : La position verticale des pétales est animée avec un **`Tween`**
- Affichage des pétales avec **`AnimatedBuilder`** .