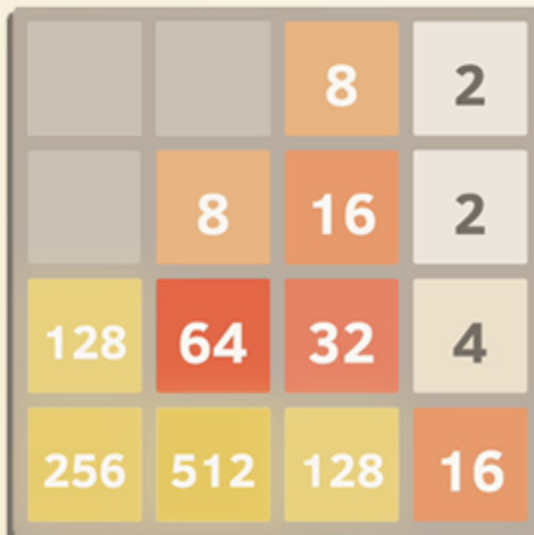

2048 en python

Rédigé par Stievenard Emma, Mezouar Chahinez et Szewczyk Clément
Réalisé sur la période : 13/09/2024 - 07/01/2025



		8	2
	8	16	2
128	64	32	4
256	512	128	16

2048

Table des matières

1	Résumé du projet	1
2	Installation / Prérequis	1
3	Structure du projet	2
4	Choix des technologies	2
5	Convention de Codage	3
6	Description des classes	3
7	Fonctionnement du jeu	4
7.1	Initialisation du plateau	4
7.2	Gestion des événements clavier	4
7.3	Logique des mouvements et des fusions	4
7.4	Conditions de fin de partie (victoire/défaite)	5
7.5	Mise à jour de l’affichage	5

1 Résumé du projet

Projet du cours Conception Logiciel réalisé par :

- Stievenard Emma,
- Mezouar Chahinez
- Szewczyk Clément

Ce projet implémente le jeu 2048 en Python en utilisant **Pygame**. Le but du jeu est de déplacer les tuiles sur une grille pour combiner des valeurs égales et atteindre la valeur cible de 2048. L'application inclut une gestion simple des tuiles, une connexion avec une base de données pour enregistrer les scores et permet de rejouer ou de quitter une partie.

Le projet est consultable sur le dépôt GitHub : [2048 en python](#)

2 Installation / Prérequis

- Avoir une version de **Python 3.11.4** ou supérieure installée sur votre machine.

Vous pouvez télécharger Python sur le site officiel : [Python](#)

- Avoir installé le module **Pygame**.

Pour installer Pygame, ouvrez un terminal et tapez la commande suivante :

```
pip install pygame
```

- Avoir installé le module **mysql-connector-python**.

Pour installer mysql-connector-python, ouvrez un terminal et tapez la commande suivante :

```
pip install mysql-connector-python
```

- Avoir installé le module **bcrypt**.

Pour installer bcrypt, ouvrez un terminal et tapez la commande suivante :

```
pip install bcrypt
```

- Récupérer le projet sur GitHub.

Pour récupérer le projet, vous pouvez cloner le dépôt GitHub en utilisant la commande suivante :

```
git clone <https://github.com/Clement-Szewczyk/2048_SDN>
```

```
cd 2048_SDN
```

- Avoir un gestionnaire de base de données (pouvant utiliser des Base de données MYSQL)

Vous devez importer la base de données du jeu. Pour cela, vous pouvez utiliser le fichier `2048.sql` situé dans le dossier BDD.

3 Structure du projet

- `2048_SDN` : Dossier principale
 - `doc` : Dossier contenant la documentation du projet
 - `doc.md` : Fichier contenant la documentation du projet au format Markdown
 - `BDD` : Dossier contenant le fichier SQL de la base de données
 - `2048.sql` : Fichier SQL de la base de données
 - `src` : Dossier contenant le code source du projet
 - `main.py` : Fichier principal du jeu 2048
 - `classTuile.py` : Fichier de la classe Tuile (Gestion des tuiles)
 - `classGrille.py` : Fichier de la classe Grille (Gestion de la grille)
 - `classJeu.py` : Fichier de la classe Jeu (Gestion de la logique du jeu)
 - `classeEcran.py` : Fichier de la classe Ecran (Gestion de l’affichage)
 - `classeBandeau.py` : Fichier de la classe Bandeau (Gestion de l’affichage du bandeau)
 - `authentificationVisu.py` : Fichier gérant le visuel de l’authentification
 - `authentification.py` : Fichier gérant l’authentification
 - `README.md` : Fichier README du projet
 - `.gitignore` : Fichier pour ignorer certains fichiers lors de l’ajout sur le dépôt GitHub

4 Choix des technologies

Pour développer le jeu 2048, nous avons choisi d’utiliser le langage Python. Ce choix s’est imposé naturellement, car il s’agit d’un langage dans lequel nous nous sentons à l’aise. De plus, grâce à la riche collection de bibliothèques disponibles, il est possible de créer une interface graphique tout en intégrant une connexion avec une base de données.

Pour l’interface graphique, notre choix s’est porté sur la bibliothèque Pygame. Cette dernière est spécialement conçue pour la création de jeux vidéo en 2D avec Python. Simple à utiliser, elle offre toutes les fonctionnalités nécessaires pour concevoir des jeux interactifs et visuellement attractifs.

En ce qui concerne la gestion des données, nous avons opté pour MySQL, un système de gestion de bases de données relationnelles bien connu pour sa fiabilité et sa facilité d’utilisation. Il nous permet de stocker les scores des joueurs ainsi que d’autres informations essentielles.

Notre base de données est structurée autour de deux tables principales :

- La table **Utilisateur**, destinée à stocker les informations personnelles des joueurs, telles que leur ID, nom, prénom, email et mot de passe.
- La table **Profil**, qui conserve les performances des joueurs, avec des colonnes comme l’ID, l’identifiant utilisateur (`utilisateurId`), la date de création (`dateCreation`) et leur meilleur score (`meilleurScore`).

Afin de garantir la sécurité des mots de passe des utilisateurs, nous utilisons la bibliothèque `bcrypt`. Cette dernière permet de hacher les mots de passe de manière sécurisée, rendant leur stockage et leur protection conformes aux bonnes pratiques en matière de cybersécurité.

5 Convention de Codage

Pour garantir une cohérence dans le code source du projet, nous avons défini une convention de codage à respecter. Cette convention inclut les éléments suivants :

- Utilisation de CamelCase pour les noms des variables, des fonctions
- Respect de la règle des 80 caractères par ligne
- Utilisation de commentaires pour expliquer le code
- Utilisation de noms de variables explicites
- Utilisation de noms de fichiers significatifs
- Utilisation du français pour les noms de variables, de fonctions, classe et de commentaires

6 Description des classes

Classe Tuile :

Gère les tuiles du jeu 2048. Cette classe inclut la création des tuiles, leur dessin à l’écran, leur mouvement. Elle définit également les attributs de chaque tuile, tels que la valeur numérique, la position sur la grille et la couleur en fonction de la valeur.

Classe Ecran :

Gère l’affichage de l’écran du jeu. Cette classe est responsable de la création de la fenêtre de jeu, de la mise à jour de l’affichage et de l’extinction de l’écran lorsque le jeu se termine.

Classe Grille :

Gère la grille du jeu 2048. Cette classe inclut la création de la grille, le dessin de la grille à l’écran. Elle définit les attributs de la grille, tels que le nombre de lignes et de colonnes, la largeur et la hauteur de chaque cellule.

Classe Jeu :

Gère la logique principale du jeu 2048. Cette classe permet de gérer la logique du jeu. Mais aussi l’ajout de nouvelles tuiles, le déplacement des tuiles existantes, la fusion des tuiles

de même valeur, le calcul du score du joueur, l’affichage des éléments et la vérification de la fin de la partie.

Classe Bandeau :

Gère le bandeau en haut de l’écran de jeu. Cette classe inclut l’affichage du score actuel du joueur, du score maximal atteint, et d’un bouton de redémarrage pour recommencer une nouvelle partie.

Authentification :

Gère l’authentification des utilisateurs. Cette classe inclut la connexion à la base de données des utilisateurs, la vérification des informations de connexion fournies par l’utilisateur, la gestion des comptes utilisateurs (création, modification, suppression) et la gestion des sessions utilisateur.

AuthentificationVisu : Gère l’interface graphique de l’authentification des utilisateurs. Cette classe inclut les champs de saisie pour la connexion et l’inscription, les boutons pour soumettre les informations de connexion ou d’inscription, et les liens pour naviguer entre les pages de connexion et d’inscription. Elle gère également les événements de clic de souris et les interactions de l’utilisateur avec l’interface.

7 Fonctionnement du jeu

7.1 Initialisation du plateau

Lors de l’initialisation du jeu, une grille vide est créée avec un nombre fixe de lignes et de colonnes. Deux tuiles sont placées aléatoirement sur la grille avec une valeur initiale de 2 ou 4. La classe `Grille` est responsable de cette initialisation de la grille et la classe `Jeu` pour l’initialisation des tuiles avec la fonction `genererTuile`.

7.2 Gestion des événements clavier

Dans le fichier `main.py`, qui le fichier de base pour lancer le jeu, nous avons une boucle principale qui gère les événements clavier. En fonction de ce que l’on fait, on appelle les fonctions de la classe `Jeu` pour déplacer les tuiles, redémarrer une partie ou quitter le jeu.

7.3 Logique des mouvements et des fusions

La logique des mouvements et des fusions est gérée par la classe `Jeu`. Lorsque le joueur appuie sur une touche directionnelle, on va d’abord trier les tuiles en fonction de la direction du mouvement (par exemple : si on appuie sur droite, les tuiles seront triées de droite à gauche). Ensuite, on va déplacer les tuiles en fonction de la direction du mouvement. Si deux tuiles de même valeur se rencontrent, elles fusionnent pour former une nouvelle tuile avec une valeur doublée. Le score du joueur est mis à jour en fonction des valeurs des tuiles fusionnées.

7.4 Conditions de fin de partie (victoire/défaite)

Les conditions de fin de partie sont vérifiées après chaque mouvement. La classe `Jeu` vérifie si une tuile avec la valeur 2048 est présente sur la grille, ce qui signifie que le joueur a gagné. Si la grille est pleine et qu'aucun mouvement ou fusion n'est possible, le jeu se termine par une défaite. Des messages de victoire ou de défaite sont affichés à l'écran.

7.5 Mise à jour de l'affichage

La mise à jour de l'affichage est gérée par la classe `Ecran`. Après chaque mouvement, l'écran est rafraîchi pour refléter les nouvelles positions des tuiles et les valeurs mises à jour.