Cahier des charges final

Présentation du Projet : Conception d'un Jeu de type 2048 en Python

Cours : Conception de logiciel

Responsable : Lucien Mousin

Équipe : Clément SZEWCZYK, Chahinez MEZOUAR, Emma STIEVENARD

Date de début : 13/09/2024

Date de soumission : 07/01/2025

1. Présentation du Projet

Contexte

Dans le cadre du cours **Conception de logiciel en Licence 3**, nous avons pour objectif de concevoir un projet informatique complet mettant en application les principes de modélisation, d'architecture logicielle et de gestion des versions. Le projet choisi est la réalisation d'une version du jeu **2048** en Python, ce qui permet de combiner plusieurs aspects de la conception logicielle, notamment la programmation orientée objet, l'algorithme de gestion des mouvements et des tuiles, ainsi que la création d'une interface utilisateur simple.

Client

Ce projet est réalisé dans un cadre pédagogique. Le client est donc l'enseignant du cours, qui évalue le projet en fonction de l'implémentation technique, la conception logicielle, la capacité d'organisation et création de livrables et enfin, la démo intermédiaire.

Historique

Le jeu **2048**, développé par Gabriele Cirulli en 2014, est un jeu vidéo de réflexion en solo, très populaire pour sa simplicité et son aspect addictif. Le joueur doit fusionner des tuiles numérotées pour atteindre la somme 2048. Nous nous inspirons de cette idée pour créer notre propre version en Python.

Règles et conditions de jeu 2048

1. **Objectif** :
    - Fusionner des tuiles portant des puissances de 2 afin d'atteindre une tuile numérotée **2048**. Vous pouvez continuer à jouer pour obtenir des tuiles plus grandes et des scores plus élevés.
2. **Plateau de jeu** :
    - Grille de **4x4**, composée de 16 cases.
    - Deux tuiles avec des valeurs "2" ou "4" apparaissent au début de la partie à des positions aléatoires.
3. **Déplacement des tuiles** :
    - Utilisez les flèches directionnelles pour déplacer toutes les tuiles dans une direction (haut, bas, gauche ou droite).
    - Toutes les tuiles bougent en même temps jusqu'à ce qu'elles rencontrent une autre tuile ou le bord de la grille.
4. **Fusion des tuiles** :
    - Lorsque deux tuiles de même valeur se rencontrent après un mouvement, elles fusionnent pour créer une tuile dont la valeur est la somme des deux tuiles.
    - Exemple : deux tuiles "8" forment une tuile "16".
5. **Apparition de nouvelles tuiles** :
    - Après chaque mouvement, une nouvelle tuile ("2" ou "4") apparaît aléatoirement sur une case vide du plateau.
6. **Conditions de victoire** :
    - Le jeu se termine si vous atteignez la tuile **2048**. Vous pouvez continuer à jouer après avoir atteint cet objectif.
7. **Conditions de fin de jeu (perte)** :
    - La partie est terminée lorsque plus aucune case n'est disponible et qu'aucun déplacement ne permet de fusionner des tuiles.

Points :

- Les points sont gagnés chaque fois que vous **fusionnez des tuiles**.
- **Calcul des points** : Les points gagnés sont égaux à la valeur de la tuile nouvellement formée après une fusion.
    - Exemple : Si vous fusionnez deux tuiles "16" pour créer une tuile "32", vous gagnez **32 points**.
    - Si vous fusionnez deux tuiles "512" pour obtenir une tuile "1024", vous gagnez **1024 points**.
- **Accumulation des points** : Le score est cumulé au fur et à mesure que vous réalisez des fusions. Plus vous réussissez de grandes fusions, plus votre score total augmente.
- Les plus grosses tuiles créées vous rapportent des points exponentiels, car la valeur de la tuile augmente de façon exponentielle (puissances de 2).

Niveaux :

- Le jeu ne comporte pas de niveaux au sens classique, mais chaque nouvelle tuile de valeur plus élevée peut être considérée comme un **niveau atteint**.
- Progression : **2 → 4 → 8 → 16 → 32 → 64 → 128 → 256 → 512 → 1024 → 2048**

---

2. Description de la Demande

Résultats attendus

Le projet vise à produire une version fonctionnelle du jeu 2048 avec une interface utilisateur graphique, en implémentant toutes les règles du jeu ainsi qu'une gestion des scores. Le code doit être bien structuré et accompagné de tests unitaires.

Fonctionnalités principales

1. **Authentification :** L'utilisateur peut se connecter / s'inscrire afin de sauvegarder son score.
2. **Grille de jeu 4x4** : Affichage d'une grille de jeu avec des tuiles de différentes valeurs.
3. **Déplacements des tuiles** : Permettre à l'utilisateur de déplacer les tuiles vers la gauche, la droite, le haut ou le bas.
4. **Fusion des tuiles** : Deux tuiles de même valeur fusionnent lorsqu'elles se rencontrent, et leur valeur est doublée.
5. **Génération de nouvelles tuiles** : Après chaque mouvement, une nouvelle tuile apparaît sur une case libre, avec une valeur de 2 ou 4.
6. **Calcul et affichage du score** : Le score augmente à chaque fusion de tuiles, et le score total est affiché.
7. **Conditions de victoire et de défaite** : Le jeu se termine si l'utilisateur atteint la tuile 2048 (victoire) ou si aucun mouvement n'est possible (défaite).
8. **Interface utilisateur** : Interface graphique (Pygame) ou version en mode terminal.
9. **Gestion des entrées utilisateur** : Les déplacements des tuiles doivent être contrôlables par le clavier.
10. **Bouton pour recommencer :** Bouton permettant de recommencer à zéro une partie
11. **Bandeau :** Contient le score actuel, le meilleur score de l'utilisateur connecté.

---

3. Contraintes

Coût

Ce projet n’a pas de coût financier direct. Tous les outils et environnements de développement utilisés (Python, bibliothèques Pygame) sont gratuits et open-source.

Durée de développement

La durée totale du projet est répartie sur tout le premier semestre. A savoir que la date de début est le 13 septembre et la date de rendu le 07 janvier.

---

4. Déroulement du Projet

Le projet est divisé en plusieurs grandes étapes, chacune associée à des tâches spécifiques. Chaque tâche a un livrable concret (sprint) pour valider son avancement.

Sprint 0 : Conception et Planification

Période : 13 au 23 septembre (Rédaction du cahier des charges)

- **Tâches :**
  - Définir l'architecture du logiciel (schémas UML, découpage en modules).
  - Planifier les différentes phases du projet.
- **Ressources :** Lucid Chart, Google Docs.
- **Livrables :** Schéma UML du projet, plan de développement détaillé.

Sprint 1 : Développement Initial

Période : 23 septembre au 8 octobre

- **Tâches :**
  - Réaliser le système de connexion.
  - Concevoir une première interface. (Grille et Tuile)
- **Ressources :** Python, Pygame, SQL.
- **Livrables :** Première interface, Interface système de connexion

---

Sprint 2 : Développement de la Logique du Jeu

Période : 8 octobre au 18 octobre

- **Tâches :**
  - Liaison entre la fonctionnalité de connexion et le jeu.
  - Continuation de l'interface utilisateur.
  - Début de la mise en place de la logique du jeu.

- **Ressources :** Python, éditeur de code (VSCode), documentation des règles du jeu.
- **Livrables :** Avancement de la logique du jeu, interface améliorée.

Sprint 3 : Complétion de la Logique et Mise en Place des Scores

Période : 18 octobre au 26 octobre

- **Tâches :**
    - Finition de la logique
    - Finition de l'interface
    - Ajout de la gestion du score
- **Ressources :** Pygame, Python.
- **Livrables :** Code complet pour la logique du jeu, gestion des scores.

Sprint 4 : Version finale du code et documentation

Période : 26 novembre au 04 décembre

- **Tâches :**
    - Vérification et ajout commentaire du code
    - Bouton redémarrer sur le bandeau
    - Ajout Modification mot de passe
- **Ressources :** Pygame, Python, Google Docs
- **Livrables :** Code sans bug, une documentation de projet bien entamée.

Sprint 5 : Documation et Tests

Période : 04 décembre au 17 décembre

- **Tâches :**
    - Fonction Perdant et Gagnant
    - Vérification Email
    - Relecture du code
- **Ressources :** Google Docs, Visual Studio Code, Python
- **Livrables :** Code final débogué, début de documentation et tests unitaires validés.

Sprint 6 : Finalisation de la documentation

Période : 17 décembre au 7 janvier

- **Tâches :**
    - Finir de rédiger la documentation du jeu
    - Préparation de la présentation
    - Correction des derniers bugs
- **Ressources :** Google Docs, Documentation
- **Livrables :** Documentation technique et utilisateur, présentation du projet.

---

5. Conventions de nommage

Chaque nom de classe devra commencer par une majuscule. Si un autre mot doit s'ajouter au premier, la première lettre du deuxième mot devra aussi être une majuscule. (ex : ClasseMère).
Pour les variables, on commence le nom par une minuscule et chaque nouveau mot par une majuscule (ex: variableExemple).
camelCase

---

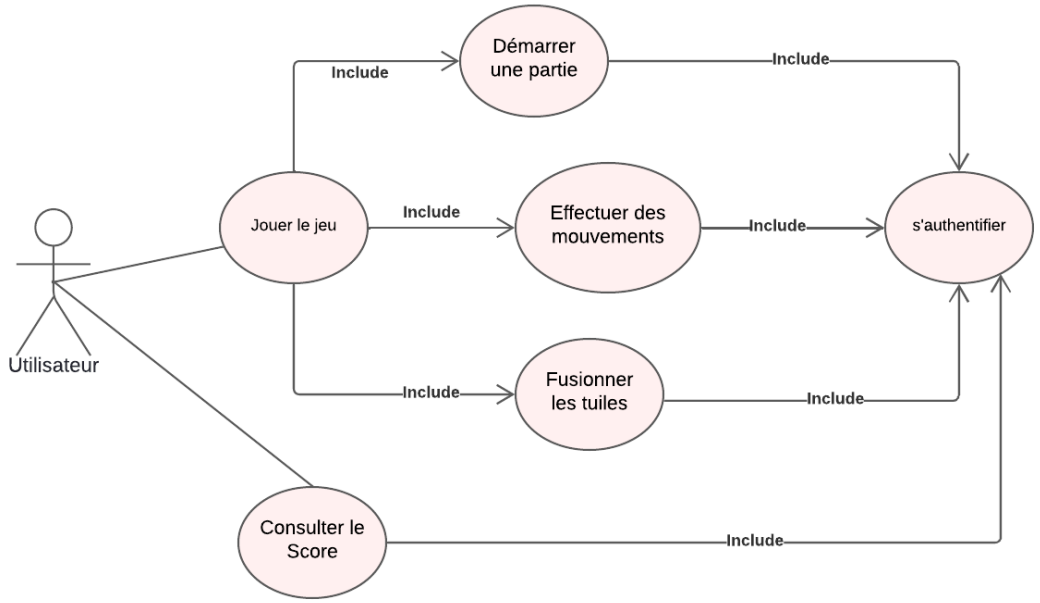
6. Annexes

- **Lien Github**

[Github du projet]

- **Diagramme de Gantt**

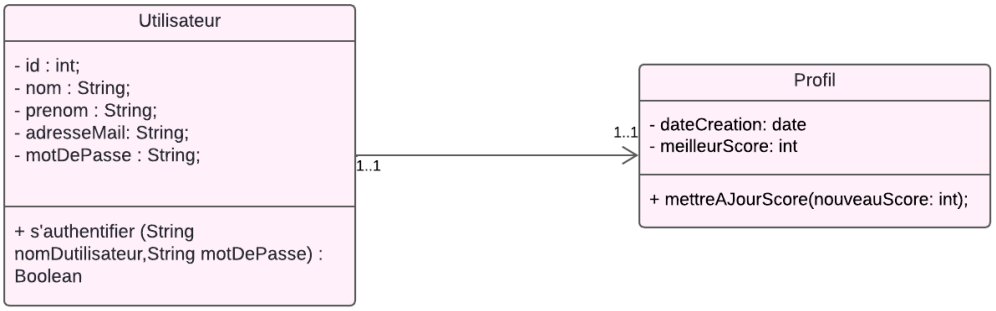
- **Diagramme de cas d'utilisation**



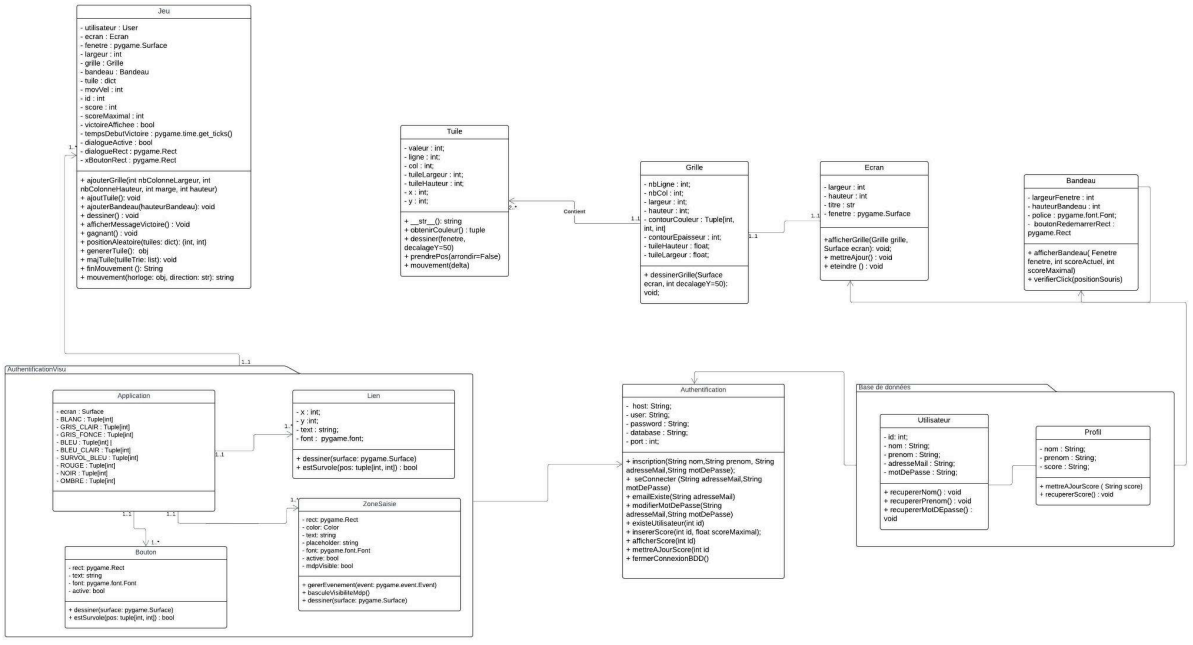
- **Diagramme de classes :**

  Cette figure illustre le diagramme de classes de notre projet, qui comprend deux classes : Utilisateur et Profil.

  La classe Utilisateur regroupe les attributs essentiels tels que le nom, le prénom, adresse mail et le mot de passe, ainsi qu'une méthode appelée s'authentifier. Quant à la classe Profil, elle contient les attributs date de création et meilleur score, avec une méthode permettant de mettre à jour le score.



- **Diagramme de classes (Objet)**



Lien :

https://lucid.app/lucidchart/5513ac22-c7bc-4837-b5d5-58c68368d0e0/edit?viewport_loc=-2105%2C-485%2C4869%2C1964%2C0_0&invitationId=inv_706cc932-aba4-44b3-b284-67f686a86519