Jeu Anti-Virus: Une Présentation Technique

Ce projet de jeu anti-virus, développé par Noah Thiesset et Maxence Goutard, explore l'ingénierie logicielle à travers Pygame. Découvrez les défis de conception, l'architecture objet et fonctionnelle, et la synergie de notre équipe pour donner vie à cette expérience ludique et éducative.

Cahier des Charges Détaillé du Projet

Programmation Orientée Objet et Fonctionnelle

Le jeu doit impérativement utiliser des paradigmes de programmation orientée objet (POO) pour la structure des entités (virus, molécules) et fonctionnelle pour la gestion de certains comportements.

Niveaux de Jeu Spécifiés

Le premier défi reproduira un modèle existant, tandis qu'un second défi original intégrera 8 molécules, nécessitant une logique de jeu complexe et adaptable.

Utilisation de la Bibliothèque Pygame

La bibliothèque Pygame sera le moteur graphique et interactif, permettant une exploitation complète de ses fonctionnalités pour le rendu visuel et la gestion des événements.

Interface et Contrôles du Joueur

Le joueur pourra contrôler le virus via la souris, les touches du clavier, ou une combinaison des deux pour une flexibilité maximale. La sortie unique sera positionnée en haut à gauche.

Développement en Python

Le projet sera entièrement codé en Python, tirant parti de sa syntaxe claire et de son écosystème riche pour un développement efficace.

Immersion Graphique et Menu Complet

Le jeu intègrera un fond graphique soigné et un menu intuitif, offrant une expérience utilisateur complète et engageante.

Processus de Développement du Jeu

01

1. Initialisation de la Fenêtre et de l'Arrière-Plan

Nous avons commencé par la mise en place de la fenêtre principale du jeu et l'intégration d'un arrière-plan $pour créer l'ambiance visuelle du jeu.

02

2. Intégration des Mouvements du Virus

La phase suivante a consisté à développer les mécanismes de déplacement du virus, garantissant une interaction efficace et réactive avec les contrôles du joueur.

03

3. Création des Classes pour les Virus

Afin de gérer l'apparition et le comportement de multiples virus, nous avons conçu une architecture de classes, permettant une instantiation facile et une gestion cohérente de chaque entité.

04

4. Conception du Menu Interactif

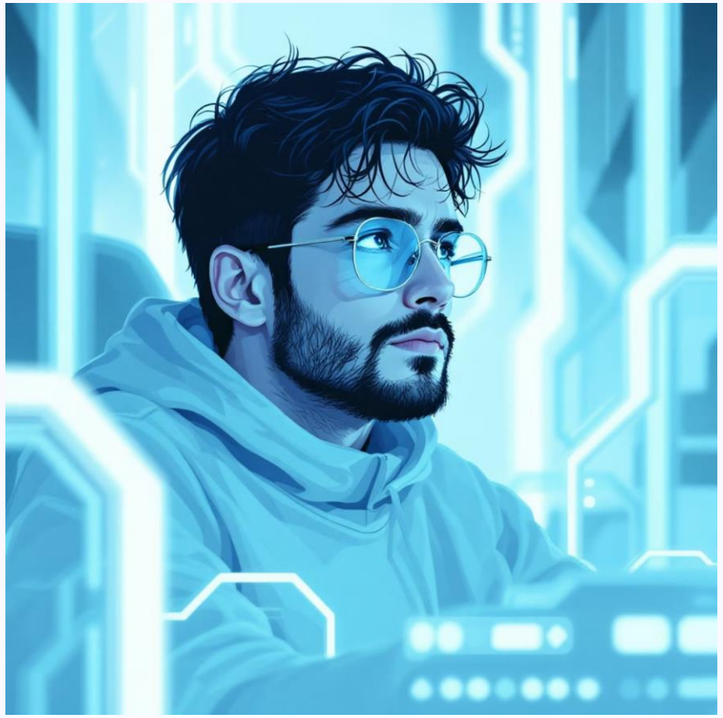
Le menu a été développé pour offrir une navigation intuitive, incluant les choix de niveau, améliorant ainsi l'expérience utilisateur.

05

5. Gestion des Collisions avec les Objets

Enfin, nous avons implémenté un moteur de collision, essentiel pour la détection des interactions entre le virus et les différents éléments du jeu, assurant une logique de jeu cohérente.

Répartition des Rôles et Responsabilités



Maxence Goutard: Le Front-End

Création et Développement du Menu

- Gestion du placement des différentes instances du menu d'affichage.
- Implémentation des systèmes de détection d'appui sur les différents boutons interactifs.
- Assurer une expérience utilisateur fluide et esthétique.

Noah Thiesset: Le Back-End

Création et Développement du Déplacement du Virus

- Gestion des mouvements complexes et de la logique sous-jacente du plateau de jeu.
- Développement du moteur de collisions précis pour toutes les interactions.
- Conception et implémentation des classes principales du jeu.

Diagramme de Classes et Fonctionnalités Clés

Notre architecture de jeu repose sur des classes robustes qui définissent les entités principales et leurs interactions, en parfaite adéquation avec les exigences fonctionnelles.

Contributions de Maxence Goutard

Conception et Développement du Menu

- Création d'un menu intuitif et esthétique, facilitant la navigation du joueur.
- Mise en place des boutons de retour pour une expérience utilisateur sans friction.
- Gestion efficace de la transition entre le menu principal et les phases de jeu actives.

Exploitation de la Bibliothèque Pygame

- Utilisation approfondie de Pygame pour l'initialisation des mouvements d'objets et le rendu graphique.
- Gestion des événements et des états du jeu via Pygame.

Outils de Développement

- Utilisation de Visual Studio Code pour un environnement de développement optimisé.
- Collaboration efficace grâce à des outils modernes de gestion de code.

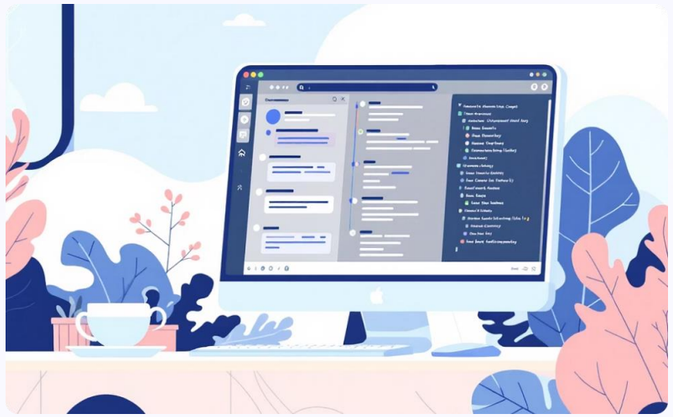
Contributions de Noah Thiesset

Systèmes de Mouvement et de Collision

- Mise en place d'un système de mouvement précis pour le virus et les molécules.
- Développement d'un algorithme de détection de collision pour les atomes, assurant l'interactivité.

Architecture des Classes et Niveaux

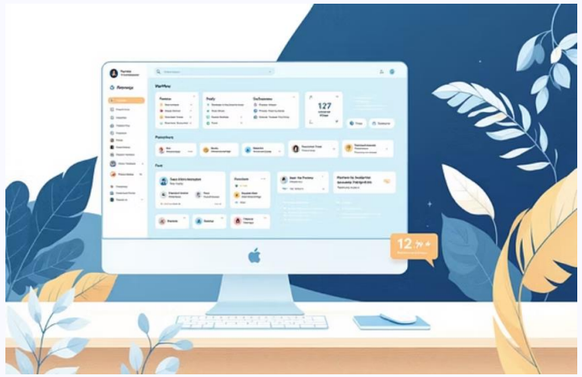
- Création des classes fondamentales et gestion de leurs relations pour une structure modulaire.
- Conception et implémentation des différents niveaux de jeu, chacun avec ses propres défis.

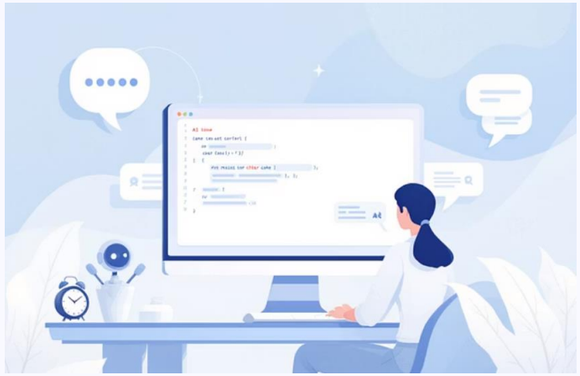


Intégration et Outils

- Intégration des assets graphiques avec la logique du jeu pour une expérience immersive.
- Exploitation de Pygame pour toutes les fonctionnalités back-end.
- Utilisation de GitHub pour le contrôle de version et de PyCharm comme IDE principal.

Références et Outils Utilisés





Ce projet a bénéficié de l'inspiration et des ressources de diverses sources. Nous avons notamment consulté le programme de jeu en Python de "Graven" pour des idées architecturales.

L'outil "Workflow" nous a aidés à organiser nos idées et à structurer le développement du jeu. Enfin, des assistants basés sur l'IA comme ChatGPT et Mistral ont été précieux pour le débogage de certaines erreurs et l'approfondissement de notre compréhension de lignes de code complexes.