Projet Pokémon
Spécifications Techniques

Mohamed CHINE
Nicolas CLEMENT
Louis DUBOIS
Aurélia HUISSE-DALBOUSSIERE
Sirine NADIFI

Table des Matières

Spécification Techniques

Matériel Ciblé

PC : Le projet sera développé et exécuté sur des ordinateurs personnels. Le choix des PC comme matériel principal se justifie par la disponibilité des ressources nécessaires au développement logiciel, telles que la puissance de calcul, la compatibilité avec les environnements de développement et la capacité de gérer les outils de collaboration (GitHub, IDE, etc.). Les ordinateurs permettent également une flexibilité en termes de systèmes d'exploitation (Windows, Linux, macOS) et offrent un accès facile à des logiciels de développement, de tests et de suivi du projet.

Système d’exploitation

Windows 10 ou 11, versions les plus récentes de Windows. De plus, Windows 10 et 11 facilitent l'intégration avec des outils de gestion de version comme Git, ainsi qu'avec des IDE tels que Visual Studio Code ou PyCharm, essentiels pour le bon déroulement du projet.

Interface externe

L'interface graphique : Elle nous permet d'interagir avec l'application de manière intuitive et visuelle. Contrairement à une interface en ligne de commande, l'interface graphique simplifie l'expérience utilisateur en offrant des éléments interactifs tels que des boutons, des menus, des fenêtres, etc. Pour ce projet, l'interface graphique sera implémentée à l'aide de **Pygame**, une bibliothèque Python spécialisée dans la création d'interfaces pour les jeux vidéo. Pygame permet de créer des fenêtres graphiques, de gérer les événements (clavier, souris) et d’afficher des éléments visuels, ce qui est essentiel pour offrir une interface fluide et engageante

Stockage des données

Fichier : Le choix de **fichiers** pour le stockage des données est une solution simple et adaptée aux besoins de ce projet. Les données seront stockées sous forme de fichiers de configuration ou de sauvegarde, dans un format tel que JSON ou TOML (via la bibliothèque `tomlkit`). Cette méthode est suffisante pour gérer les paramètres de l'application et les sauvegardes d'état, sans nécessiter la complexité d'une base de données complète. Le format fichier est facile à lire et à manipuler, tout en étant léger et compatible avec l'ensemble des plateformes choisies.

Langages de programmation

Python 3.12.6 : Le projet sera développé en **Python 3.12.6**, une version récente de Python, largement utilisée pour le développement d'applications et de jeux grâce à sa simplicité d'utilisation et sa vaste communauté. Python offre un large éventail de bibliothèques spécialisées pour le développement d'interfaces graphiques, la gestion des fichiers, et la création de jeux vidéo. Cette version de Python inclut des améliorations de performance et de nouvelles fonctionnalités qui rendent le développement plus fluide et permettent de tirer profit des dernières avancées du langage.

Bibliothèques utilisées

Pygame : C'est la bibliothèque principale utilisée pour la gestion de l'interface graphique et des événements du projet. Elle permet de gérer la création de fenêtres, le rendu des éléments graphiques, la gestion des entrées utilisateur (clavier, souris), et le rendu des sprites. C'est une bibliothèque idéale pour des projets de jeux ou des applications interactives nécessitant une interface riche.

Pytmx : C'est utilisé pour charger et manipuler des fichiers de tuiles au format **Tiled Map Editor** (TMX). Cette bibliothèque permet de charger facilement des cartes en 2D pour des jeux, tout en offrant des fonctionnalités avancées pour gérer les couches de tuiles et les objets. Elle facilite ainsi la création de niveaux ou de scènes basées sur des cartes.

Pyscroll: Une bibliothèque complémentaire à Pygame qui permet de gérer le défilement (scrolling) d'une carte en 2D. C'est particulièrement utile pour les jeux où les niveaux ou les environnements sont plus grands que l'écran, permettant à la caméra de suivre le joueur ou de se déplacer dans le monde du jeu.

Tomlkit : Utilisée pour gérer les fichiers de configuration au format **TOML**. TOML est un format de fichier de configuration facile à lire et à écrire pour les humains, similaire à JSON mais plus lisible. Cette bibliothèque est essentielle pour manipuler les fichiers de configuration, notamment pour gérer les préférences du jeu ou les états de sauvegarde dans un format léger.

Autres Technologies

En plus de toutes les technologies citées, nous utilisons une librairie réalisée par nos soins qui permet d'intégrer le pattern model view controller à la librairie pygame et quelques widgets supplémentaires comme des boutons ou des menus.

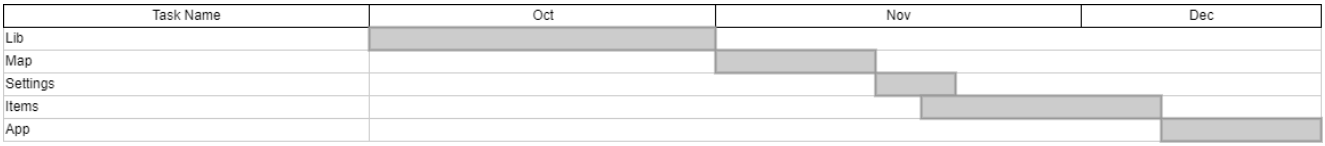
Organisation du projet

Découpage en lot

Nous avons décidé de diviser notre projet en module qui peuvent être réalisé à l'aide de la librairie. Nous avons établi l'organisation suivante. Louis a codé tout ce qui était technique dans la librairie et les autres personnes se sont occupés chacun d'un module dans lequel il y a réalisé la tâche portée par le nom du module.

Planning

Le planning a été réalisé en se basant sur le diagramme de Gantt ci-dessous :



Environnement de Développement

Pycharm : L'**IDE PyCharm** est choisie comme environnement de développement pour ce projet. PyCharm est un outil puissant pour le développement Python, offrant des fonctionnalités telles que l'auto-complétion de code, la gestion des erreurs en temps réel, et une intégration facile avec les environnements virtuels et les outils de gestion de projet. Il permet également une intégration directe avec GitHub, facilitant ainsi le suivi des modifications du code.

Pytest : **Pytest** est utilisé pour les tests unitaires, ce qui permet de s'assurer que chaque partie du code fonctionne correctement individuellement. Pytest simplifie l'écriture et l'exécution des tests, et son intégration avec PyCharm permet d'exécuter facilement des tests automatisés après chaque mise à jour du code. Les tests réguliers garantissent que les nouvelles fonctionnalités n'introduisent pas de bugs dans le projet.

Pyproject.toml : Le fichier **pyproject.toml** est utilisé pour définir les métadonnées du projet, y compris les dépendances du projet (comme Pygame, Pytmx, Pyscroll, etc.). Ce fichier standardise la configuration du projet et facilite l'installation des dépendances dans des environnements différents. Il centralise toutes les informations liées à la gestion des packages et à la configuration de l'application.

Venv : L'utilisation d'un **environnement virtuel (venv)** est essentielle pour isoler les dépendances du projet des autres projets ou installations Python du système. Cela permet de s'assurer que chaque développeur travaille avec les mêmes versions de bibliothèques, évitant ainsi les conflits de version ou d'incompatibilité. Le venv est facile

à configurer avec PyCharm, ce qui assure une configuration homogène pour tous les membres de l'équipe.

Méthodes de suivi du projet

Discord (call et messagerie) : Nous utilisons **Discord** pour la communication en temps réel, à la fois via des appels vocaux et des messages écrits. Discord permet de créer des canaux dédiés aux discussions spécifiques sur les différentes parties du projet (développement, tests, bugs, etc.), facilitant ainsi l'organisation des échanges. Les appels vocaux réguliers permettent des réunions d'équipe rapides et efficaces pour synchroniser l'avancement du projet, clarifier les objectifs, et résoudre rapidement les éventuels problèmes.

Référentiel du projet

GitHub : **GitHub** sera le **référentiel principal** du projet, hébergeant tout le code source ainsi que la documentation du projet. Chaque modification du code sera suivie par des commits qui permettent de conserver un historique clair de l'évolution du projet. L'utilisation de branches facilitera le travail en parallèle sur différentes fonctionnalités, et les **pull requests** permettront de revoir et de valider le code avant son intégration dans la branche principale. GitHub assure également la collaboration en ligne, rendant les contributions plus fluides et organisées. En plus de cela nous utilisons OneDrive pour tous les documents de spécifications et les tests d'intégrations.