Killian GRISLIN
Léa COURMONTAGNE
Sara TERRIER

Cahier des charges

I. Introduction

A. Présentation du projet

Le projet consiste à développer le jeu vidéo classique, Pac-Man, en respectant les normes de codage modernes et les bonnes pratiques de développement. Le jeu Pac-Man est un jeu où le joueur contrôle un personnage, Pac-Man, qui doit collecter tous les points dans un labyrinthe tout en évitant des fantômes. Ce projet inclut la création du jeu, l'écriture d'une documentation technique complète, des tests unitaires, et la mise en place d'une procédure de gestion de code via GitHub.

B. Objectifs du cahier des charges

L'objectif principal de ce cahier des charges est de définir les spécifications techniques et fonctionnelles nécessaires au développement du jeu Pac-Man. Il doit également décrire les étapes nécessaires pour garantir la qualité du code (tests unitaires, revue de code), ainsi que les outils de gestion du projet (GitHub pour le versionnement et les pull-requests).
Les objectifs spécifiques sont :

1. Fournir une description claire du contexte et des besoins du projet.
2. Garantir que le code source est propre, lisible et conforme aux normes de codage.
3. Définir les méthodes de test et les outils utilisés pour garantir la stabilité du jeu.
4. Mettre en place un plan de développement collaboratif avec gestion des versions et revue de code.

II. Contexte du projet

Ce projet a pour objectif de recréer un jeu classique en utilisant des technologies modernes tout en appliquant les standards actuels de conception et de développement. Le processus de développement suivra une approche itérative, intégrant des tests unitaires dès le début. La gestion du code sera assurée par une plateforme de versionnement comme GitHub, facilitant le travail collaboratif. Par ailleurs, des revues de code via les pull-requests seront mises en place pour garantir la qualité du code produit.

III. Objectifs du projet

A. Objectifs généraux

- ➔ Développer un jeu Pac-Man fonctionnel et divertissant tout en respectant les bonnes pratiques de codage et de gestion de projet.
- ➔ Garantir la qualité du code grâce à l'application de tests unitaires et à la revue de code systématique sur GitHub.

➔ Produire une documentation technique claire et détaillée pour faciliter l'évolution du projet.

B. Objectifs spécifiques

Eléments du jeu :

- **Labyrinthe :** Représentation visuelle fidèle d'un labyrinthe avec différents chemins où le personnage principal (Pac-Man) peut se déplacer.
- **Personnage principal (Pac-Man) :** Le joueur doit pouvoir contrôler Pac-Man à l'aide des touches de direction du clavier.
- **Fantômes :** Différents ennemis (fantômes) avec des comportements uniques qui poursuivent Pac-Man dans le labyrinthe.
- **Points à collecter :** Dispersés dans le labyrinthe, Pac-Man doit ramasser des points pour gagner des points et des vies supplémentaires.
- **Système de vies et du score :** Une structure permettant de gérer les points de vie du joueur et la progression du score.

Normes de codage :

- Respect des normes de développement (naming convention, structures de classes, modularité, commentaires, etc.).
- Mise en place de tests unitaires avec le framework Unitest pour le langage de programmation Python.

Outils et technologies :

- Utilisation du langage de programmation Python avec le framework Unitest
- Utilisation de GitHub pour le contrôle de version, avec une stratégie de branches et de pull-requests pour les contributions.
- Revue de code pour s'assurer que les normes sont respectées avant de fusionner les branches.
- Documentation complète du code (commentaires, summary…).

Tests :

- Définir un plan de test couvrant les tests unitaires, d'intégration, et éventuellement des tests d'interface utilisateur.
- Utilisation d'un framework de tests adapté, Unitest, au langage Python.

IV. Fonctionnalités requises

A. Liste exhaustive des fonctionnalités

Déplacements de Pac-Man :

- Le joueur doit pouvoir contrôler Pac-Man à l'aide des touches directionnelles du clavier pour se déplacer dans les quatre directions (haut, bas, gauche, droite).
- Gestion des collisions avec les murs du labyrinthe.

Fantômes ennemis (IA des fantômes) :

- Quatre fantômes au comportement différent (comme dans le jeu original : Chasseur, Aléatoire, Suiveur, Fuyant).
- Les fantômes se déplacent dans le labyrinthe en poursuivant Pac-Man ou errant aléatoirement selon des phases distinctes (chasse, fuite).

Labyrinthe :

- Conception du labyrinthe avec des chemins clairs et des obstacles (murs).
- Pac-Man et les fantômes ne peuvent traverser les murs.

Points à collecter (Pac-Gums) :

- Pac-Man doit ramasser des points répartis dans le labyrinthe pour accumuler des points de score.
- Différentes tailles de points, certains offrant des pouvoirs temporaires (exemple : "Power Pellets" pour rendre Pac-Man invincible et lui permettre de manger les fantômes).

Système de vie et de score :

- Pac-Man commence chaque partie avec un nombre déterminé de vies (par exemple, 3).
- Le joueur gagne des points en ramassant des Pac-Gums et en mangeant des fantômes lorsqu'il est invincible.
- Le score et le nombre de vies doivent être affichés à l'écran en temps réel.

Interface utilisateur (UI) :

- Écran de démarrage, pause et redémarrage du jeu.
- Affichage des scores et des vies en temps réel pendant la partie.
- Écran de fin de partie (Game Over) avec le score final et possibilité de redémarrer une nouvelle partie.

B. Priorisation des fonctionnalités

Priorité 1 (Fonctionnalités essentielles) :

- **Déplacements de Pac-Man** : Contrôle du personnage principal.
- **Labyrinthe** : Gestion des collisions et conception du labyrinthe.
- **Fantômes ennemis (IA de base)** : Comportement simple de poursuite.
- **Points à collecter (Pac-Gums)** : Collecte de points et progression dans la partie.
- **Système de vie et de score** : Gestion des vies et affichage du score.

Priorité 2 (Fonctionnalités importantes) :

- **IA avancée des fantômes** : Comportement plus complexe (poursuite, fuite, stratégie de groupe).

- **Power Pellets** : Points spéciaux permettant d'inverser les rôles (Pac-Man chasse les fantômes).

Priorité 3 (Fonctionnalités secondaires) :

- **Écrans d'interface (pause, Game Over)** : Ajout d'une interface utilisateur intuitive et attractive.
- **Amélioration des graphismes** : Optimisation des sprites et du design visuel du labyrinthe, des personnages et de l'environnement.

C. Interactions entre les fonctionnalités

Pac-Man et le labyrinthe :

- Les mouvements de Pac-Man doivent être restreints par les murs du labyrinthe, créant ainsi des chemins délimités que Pac-Man peut emprunter.
- Pac-Man interagit également avec les Pac-Gums dans le labyrinthe, qu'il doit collecter pour gagner des points.

Fantômes et Pac-Man :

- Les fantômes poursuivent Pac-Man en suivant différents algorithmes d'IA. S'ils entrent en collision avec Pac-Man, une vie est retirée.
- En cas d'ingestion de "Power Pellets", les fantômes changent de comportement et fuient Pac-Man, permettant à ce dernier de les manger.

Système de vie et score :

- Chaque interaction entre Pac-Man et les fantômes (soit capture de Pac-Man par un fantôme, soit consommation d'un fantôme par Pac-Man) impacte le nombre de vies de Pac-Man et le score du joueur.
- Le score s'accumule également grâce à la collecte des Pac-Gums et des Power Pellets dans le labyrinthe.

V. Contraintes et limitations

A. Contraintes de temps

Le projet doit être terminé et livré au plus tard le 20 décembre 2024. Cela inclut le développement complet du jeu, la rédaction de la documentation technique, la mise en place des tests unitaires, ainsi que le dépôt du code final sur GitHub.

B. Contraintes techniques

- **Budget** : Le projet ne dispose d'aucun budget. Toutes les technologies, outils, frameworks et ressources nécessaires doivent donc être gratuits ou open-source. Aucun achat de logiciels, licences ou ressources externes ne pourra être envisagé.

- **Matériel** : Le développement se fera sur les les ordinateurs personnels des membres du groupe.

- **Complexité du jeu** : La complexité du jeu devra être limitée pour tenir compte du temps restreint disponible pour le développement et des contraintes techniques. Il est important de se concentrer sur les fonctionnalités essentielles (déplacement de Pac-Man, fantômes, labyrinthe, points à collecter) avant d'ajouter des améliorations visuelles ou sonores plus avancées.

- **Qualité du code et normes de codage** : Le respect des normes de codage est un critère important pour ce projet. Le code doit être lisible, structuré et documenté, afin de faciliter la maintenance et les revues de code sur GitHub. Les tests unitaires devront être mis en place dès le début du développement pour valider la robustesse et la fiabilité du code au fur et à mesure de son avancement.

VI. Tests et validation

A. Stratégie de test

Tests unitaires :

- **Objectif :** Vérifier le bon fonctionnement de chaque composant individuel du jeu (ex. : mouvements de Pac-Man, gestion des collisions, IA des fantômes).
- **Outils :** Utilisation du framework Unitest de tests adaptés au langage Python
- **Méthodologie :** Chaque fonctionnalité critique (ex. : gestion des déplacements, ramassage de points, comportement des fantômes) aura un test unitaire dédié, exécuté automatiquement après chaque modification du code.

Tests d'intégration :

- **Objectif :** S'assurer que les différents modules du jeu (ex. : Pac-Man, fantômes, labyrinthe, score) fonctionnent ensemble de manière cohérente.
- **Méthodologie :** Après validation des tests unitaires, des tests seront réalisés pour vérifier que les interactions entre ces modules se déroulent comme prévu (ex. : interaction entre Pac-Man et les fantômes, gestion du score).

Tests fonctionnels :

- **Objectif :** Vérifier que le jeu respecte les fonctionnalités définies dans le cahier des charges, du point de vue de l'utilisateur (ex. : interface utilisateur, démarrage du jeu, fin de partie).
- **Méthodologie :** Tests manuels pour valider que le joueur peut correctement interagir avec le jeu (se déplacer, collecter des points, échapper aux fantômes), et qu'il peut accéder aux fonctionnalités comme la pause ou le redémarrage de la partie.

Tests de performance :

- **Objectif :** Garantir que le jeu fonctionne de manière fluide et réactive sur des configurations matérielles basiques.
- **Méthodologie :** Mesure des temps de réponse, utilisation des ressources processeur et mémoire. Vérification que le jeu ne subit pas de ralentissements ou de bugs lorsque plusieurs fantômes et objets sont présents à l'écran.

B. Critères de réussite des tests

Tests unitaires :

- 100 % des tests unitaires doivent réussir. Chaque module doit répondre correctement aux attentes, sans erreur ou bug. Par exemple, Pac-Man doit se déplacer correctement dans le labyrinthe, et les fantômes doivent respecter les comportements définis (poursuite, fuite).

Tests d'intégration :

- Toutes les interactions entre les modules doivent fonctionner de manière cohérente. Par exemple, lorsque Pac-Man entre en collision avec un fantôme, une vie doit être retirée, et le score doit correctement augmenter après la collecte de tous les Pac-Gums.

Tests fonctionnels :

- Le jeu doit offrir une expérience utilisateur fluide : Pac-Man doit répondre immédiatement aux commandes du joueur, et toutes les fonctionnalités (collecte de points, IA des fantômes) doivent être accessibles et fonctionnelles.
- L'interface utilisateur (affichage du score, système de pause) doit être claire et réactive.

Tests de performance :

- Le jeu doit fonctionner sans ralentissement perceptible. Le temps de réponse aux commandes utilisateur doit être optimal.

C. Procédure de validation du projet

1. **Exécution des tests automatiques :**
    - Avant chaque nouvelle version ou pull-request sur GitHub, tous les tests automatisés (unitaires, d'intégration) doivent être exécutés et validés.
2. **Revue de code et tests manuels :**
    - Une revue de code systématique sera effectuée sur chaque pull-request pour valider la qualité du code et vérifier que les modifications respectent les normes de développement.
    - Des tests manuels seront réalisés pour s'assurer que l'expérience utilisateur reste fluide et que le jeu est fonctionnel dans son ensemble.

3. **Documentation des résultats de test :**
    - Un rapport de tests détaillant les résultats de chaque phase de test sera produit, incluant les succès, les échecs et les corrections apportées.
    - Ce rapport sera utilisé lors de la validation finale pour démontrer que le jeu respecte les exigences définies et qu'il est prêt pour la livraison.
4. **Validation finale :**
    - Le projet sera validé si tous les tests sont réussis, si les objectifs du cahier des charges sont atteints.