Projet Meutre au manoir

Documentation technique

CONFIDENTIAL

Sommaire

Affaire n°1

Démarrage de l'Aventure : Vue d'ensemble Technique

Contexte et Objectifs

Le projet Meurtre au Manoir s'inspire du célèbre jeu de société Cluedo, en reprenant son univers mystérieux et ses intrigues criminelles pour proposer une expérience interactive et immersive. Dans Meurtre au Manoir, le joueur endosse le rôle d'un enquêteur qui doit interroger divers personnages pour résoudre un meurtre. Cette démarche vise à créer une ambiance captivante où chaque indice et chaque interaction contribuent à l'enquête.

Les objectifs techniques de ce projet sont multiples :
Performance en temps réel : Garantir une latence minimale dans la transmission des requêtes et des réponses pour assurer une fluidité d'interaction indispensable à l'immersion.

Robustesse de l’architecture : Mettrez en place des mécanismes de gestion des erreurs et des exceptions pour maintenir la continuité du service, même en cas de surcharge ou de défaillance d'un composant.

Scalabilité : Concevoir une architecture capable de supporter une augmentation du nombre d'utilisateurs et d'interactions sans compromettre la qualité de l'expérience.

Présentation de l’Infrastructure IA et des Technologies Employées

Pour donner vie à cette aventure, Meurtre au Manoir repose sur une infrastructure intégrant plusieurs technologies complémentaires :

- **Modèle IA (Gemma2/GPT) :** Le cœur du système repose sur un modèle d'intelligence artificielle de type GPT, ici représenté par Gemma2, qui génère des réponses contextuelles et cohérentes lors des interactions. Ce modèle est intégré via une API dédiée, permettant une communication efficace entre le module IA et le reste de l'architecture.

- **Framework Flask pour le back-end :** L'application serveur est développée avec Flask, un framework Python léger et performant, qui gère les requêtes HTTP/WebSocket et orchestre les différents services du jeu (gestion des endpoints, communication avec l'IA, etc.). La gestion des endpoints, tels que ceux contenus dans le fichier api.py, illustre cette approche modulaire et orientée service.

- **Interface utilisateur avec Pygame :** L'expérience de jeu est rendue possible grâce à Pygame, qui offre une interface graphique interactive et dynamique. Ce moteur de jeu, dont l'implémentation principale se trouve dans le fichier main.py, gère le rendu visuel, les animations, et l'interaction en temps réel avec le joueur.

- **Intégration de Whisper pour la transcription audio :** Afin de permettre des interactions vocales, le projet intègre également le module Whisper, dédié à la transcription audio. Cette solution permet de convertir les messages enregistrés en texte, facilitant ainsi leur traitement par le modèle IA.

Affaire n°2

L'Envers du Décor du Jeu : Architecture et Conception Technique

Infrastructure Technique du Jeu

(voir annexe 1 & 2 - page 21-22)

- **Composants Matériels et Logiciels :** Le cœur technique du projet "Meutre au manoir" repose sur une architecture répartie entre un serveur back-end et un client interactif. L'application serveur est développée avec Flask, comme illustré dans le fichier api.py , et expose plusieurs endpoints essentiels (par exemple, /whisperiser, /repondre et /quitueur). Ces points d'accès gèrent la réception des requêtes, la transcription des fichiers audio grâce à Whisper, ainsi que l'envoi des prompts enrichis vers le modèle d'intelligence artificielle (Gemma2 via l'API Ollama) pour générer des réponses adaptées au contexte de l'enquête. Du côté client, l'interface est réalisée avec Pygame (cf. main.py), qui permet de gérer le rendu graphique, les interactions utilisateur en temps réel et l'animation des personnages. Cette répartition permet de répondre aux exigences de performance en temps réel et de garantir une expérience immersive tout en assurant la modularité du système.

Organisation Logique du Projet "Meutre au manoir"

- **Modularité du Système :** Le système est soigneusement découpé en modules fonctionnels afin de séparer les différentes responsabilités et de faciliter la maintenance ainsi que l'évolution du projet. Un module dédié à l'enregistrement et au traitement de l'audio, géré par Pygame et des bibliothèques telles que SoundDevice et Pydub, se charge de capturer et de convertir la voix du joueur en fichiers exploitables. Un autre module, implanté dans Flask, se concentre sur la gestion des dialogues et la communication avec l'IA. Celui-ci utilise une structure de données de type deque pour conserver un historique des échanges, ce qui permet d'enrichir le contexte lors de la génération de réponses. Enfin, le moteur de rendu graphique assure l'affichage dynamique des personnages, des dialogues et des animations, créant ainsi une interface de jeu cohérente et immersive.

- **Intégration des Services Externes :** L'architecture intègre également des services externes qui renforcent les fonctionnalités du jeu. Le module Whisper est utilisé pour transcrire les messages audio en texte, garantissant ainsi une interaction fluide entre le joueur et le système. Par ailleurs, l'intégration de Gemma2 via Ollama permet de générer des réponses contextuelles en temps réel, basées sur les prompts enrichis par l'historique de la conversation et les indices récoltés durant l'enquête. Cette intégration assure une cohérence narrative et permet d'offrir une expérience de jeu riche et personnalisée.

Scénarios d'Interaction et Diagrammes de Séquence

(voir annexe 4 - page 24)

- **Cycle de Vie d'une Requête :** Le processus d'interaction dans "Meutre au manoir" est soigneusement orchestré pour garantir une réactivité et une immersion totales. Tout commence par la saisie d'une requête par le joueur, que ce soit par le biais d'un enregistrement audio ou d'un message texte. Cette saisie est ensuite capturée et convertie en données exploitables, puis envoyée au serveur via Pygame. Le serveur Flask prend le relais en utilisant l'endpoint /whisperiser pour transcrire l'audio et enrichir le prompt en intégrant le contexte de l'enquête ainsi que l'historique des échanges. Le prompt ainsi constitué est transmis à l'endpoint /repondre, qui interroge le modèle IA afin de générer une réponse cohérente. Enfin, la réponse est renvoyée au client, où elle est affichée à l'écran, bouclant ainsi le cycle d'interaction.

- **Gestion des Exceptions et des Erreurs :** Afin de garantir une expérience utilisateur sans interruption, des mécanismes robustes de gestion des exceptions ont été mis en place. Chaque endpoint de l'API est conçu pour vérifier la validité des données reçues et renvoyer des messages d'erreur explicites en cas d'anomalies, tels que des problèmes de timeout ou des erreurs de format. Ces stratégies de récupération assurent que, même en cas de défaillance d'un composant, le système peut rapidement isoler le problème et continuer à fonctionner de manière stable.

Chemins de l'Aventure (Flux de Données et Traitement)

(voir annexe 3 - page 23)

- **Traçabilité des Données :** Le parcours des données dans "Meutre au manoir" est minutieusement tracé pour garantir la transparence et la qualité des échanges. Dès la saisie de la requête par le joueur, les informations transitent par une série de points de contrôle : elles sont capturées via l'interface Pygame, converties en fichiers audio, puis envoyées au serveur Flask où elles sont transcrites et enrichies par le contexte de l'enquête. Ce flux complet, qui s'étend du joueur au modèle IA et inversement, est documenté et surveillé afin de permettre un suivi détaillé, facilitant ainsi le diagnostic en cas d'incident et garantissant une réponse en temps réel.

- **Optimisation du Passage de Données :** Pour offrir une expérience de jeu fluide, l'optimisation des flux de données est une priorité. L'utilisation de formats légers comme le JSON permet de réduire la surcharge lors de l'échange d'informations entre les modules. De plus, l'adoption de protocoles asynchrones et l'intégration d'appels non bloquants (via asyncio dans le traitement des requêtes du chatbot) assurent une gestion efficace des interactions concurrentes. Ces optimisations permettent non seulement de minimiser la latence, mais également d'améliorer la réactivité globale du système, garantissant ainsi une immersion continue pour le joueur.

Affaire n°3

Affinement des Interactions Narratives : Optimisation Technique du Prompt Engineering

Conception de Prompts Immersifs et Techniques

- **Approche Méthodologique :** Le développement de prompts immersifs repose sur la structuration et la personnalisation de contenus issus du fichier prompts_v5.json. Ce fichier sert de référentiel pour définir la tonalité, le style et la complexité des interactions narratives. L'approche méthodologique consiste à analyser en profondeur la syntaxe et la sémantique des requêtes afin d'adapter les prompts aux différents contextes de jeu et aux caractéristiques des personnages. Chaque prompt est conçu pour non seulement fournir une réponse cohérente et contextuelle, mais aussi pour renforcer l'immersion du joueur dans l'univers de Meurtre au Manoir. Des techniques avancées de parsing et de traitement linguistique sont employées pour identifier les éléments clés des messages utilisateurs, permettant ainsi une formulation précise et adaptée des requêtes transmises au modèle IA.

Analyse Comparative des Formulations de Prompts

- **Tableaux de Bord et Indicateurs de Performance :** Une analyse comparative approfondie est menée pour évaluer l'efficacité des différentes formulations de prompts. Des tableaux de bord dédiés compilent des indicateurs de performance tels que les scores de similarité (BLEU, ROUGE) et d'autres métriques internes, permettant ainsi une comparaison objective des formats de prompts utilisés. Les retours d'expérience recueillis auprès des utilisateurs et des tests en conditions réelles fournissent des données précieuses qui orientent les itérations futures.

Analyse Comparative des Formulations de Prompts

- **Tableaux de Bord et Indicateurs de Performance :** Une analyse comparative approfondie est menée pour évaluer l'efficacité des différentes formulations de prompts. Des tableaux de bord dédiés compilent des indicateurs de performance tels que les scores de similarité (BLEU, ROUGE) et d'autres métriques internes, permettant ainsi une comparaison objective des formats de prompts utilisés. Les retours d'expérience recueillis auprès des utilisateurs et des tests en conditions réelles fournissent des données précieuses qui orientent les itérations futures.

- **L'analyse du cas d'usage dans Meurtre au Manoir :** met en lumière deux scénarios distincts de meurtre au sein du manoir Duval. Le premier, "La Nuit de la Disparition", suit un schéma classique avec un empoisonnement discret, tandis que le second, "La Nuit des Secrets", introduit des éléments plus immersifs, comme un anniversaire caché et une arme ancienne brisée, rendant l'intrigue plus riche et captivante. L'évaluation par Gemma2 conclut que le second scénario est plus original et détaillé, avec des personnages et des alibis mieux construits. Les scores de performance confirment ces observations : BERTScore (F1 : 0.7496) indique une bonne similarité sémantique, tandis que ROUGE-L (0.2268) et BLEU (0.1226) révèlent un manque de diversité lexicale. METEOR (0.3594) montre une cohérence linguistique correcte mais améliorable. En conclusion, bien que les scénarios soient immersifs, un travail supplémentaire sur la variation des structures et l'optimisation des prompts permettrait d'améliorer la fluidité et l'originalité du récit

Affaire n°4

Quêtes et Obstacles : Défis Techniques et Solutions

Optimisation de la Pertinence des Réponses

- **Identification des Défis Techniques :** Le projet "Meutre au manoir" a nécessité de relever d'importants défis pour garantir la pertinence des réponses générées par le système d'IA. Parmi ces défis, on compte la gestion de la latence, l'assurance d'une cohérence dans les dialogues et la précision des réponses dans un contexte narratif complexe inspiré de l'univers de Cluedo. En effet, les retours de l'IA doivent être suffisamment détaillés pour alimenter l'intrigue tout en restant en phase avec le contexte historique de la conversation. L'analyse des flux de données, telle qu'illustrée par le parcours depuis la saisie audio par Pygame jusqu'à la réponse générée par Gemma2 via l'API Ollama, a permis d'identifier les goulets d'étranglement et les incohérences potentielles qui pouvaient impacter la fluidité de l'expérience.

- **Ajustements Techniques :** Pour améliorer la pertinence des réponses, plusieurs ajustements ont été réalisés, notamment dans la calibration des paramètres du modèle IA. L'optimisation des prompts, enrichis du contexte de l'enquête et de l'historique de la conversation, a permis de réduire les erreurs de compréhension. La mise en place de mécanismes de réessai en cas d'erreurs (timeouts, réponses JSON invalides) garantit que le système reste réactif, même en cas de surcharge ou d'incident isolé. Cette approche itérative a permis de raffiner le processus de génération de réponses et d'assurer une meilleure adéquation entre la demande de l'utilisateur et la réponse délivrée.

Équilibre du Jeu : Gestion des Biais et Limitations du Modèle

- **Détection et Correction des Biais :** Le projet a également confronté la problématique des biais inhérents aux modèles de langage. L'analyse régulière des réponses générées a permis de détecter des tendances qui pourraient nuire à l'expérience narrative, notamment dans la représentation des personnages ou dans l'attribution des rôles. Des techniques d'analyse linguistique ont été mises en place pour identifier ces biais et ajuster les formulations dans les prompts, en modulant par exemple la pondération des données d'entrée afin d'obtenir des réponses équilibrées et fidèles à l'univers du jeu.

- **Stratégies d'Atténuation :** Afin de compenser les limitations intrinsèques du modèle, des stratégies telles que le rééchantillonnage des données et le filtrage post-réponse ont été implémentées. Ces techniques permettent d'atténuer l'impact des biais et d'améliorer la cohérence globale du récit. La documentation technique intègre également des retours d'expérience issus des tests en conditions réelles, qui servent à affiner en continu les paramètres et la structuration des prompts. Cette approche proactive garantit que les réponses fournies restent non seulement pertinentes, mais également impartiales, renforçant ainsi la crédibilité du jeu.

Affaire n°5

Scores et Résultats : Analyse Technique et Performance

Comparaison Avant/Après Optimisation

- **Études de Cas Techniques :** Des comparaisons systématiques ont été réalisées pour mesurer l'impact des optimisations sur le système. En comparant les performances avant et après l'optimisation des prompts et l'amélioration des flux de données, il a été possible de constater une réduction significative de la latence et une amélioration notable de la cohérence des réponses. Des graphiques et des statistiques détaillent ces évolutions, mettant en lumière les gains en rapidité et en stabilité du système. Ces études de cas permettent de valider l'efficacité des ajustements techniques et de fournir des bases solides pour les futurs développements.

Défis Futurs et Axes d'Amélioration Technique

- **Identification des Limites Actuelles :** Malgré les améliorations obtenues, certaines limites persistent dans l'architecture actuelle. Des goulots d'étranglement liés à la gestion asynchrone des requêtes et à la complexité de l'intégration des services externes (comme Whisper et Ollama) ont été identifiés. Ces points faibles, qui impactent la scalabilité et la réactivité du système, nécessitent des recherches et des tests supplémentaires.

- **Propositions d'Améliorations Futures :**
  - Dockerisation
  - Optmiser Whisper
  - Limite du hardware, optimisation des ressources hardware utilisées par Ollama
  - Distiller gemma II 9B pour le spécialisé dans la création de contenu dédié à notre jeu
  - Développer la storyline
  - Développer environnement du jeu (monde, pnj, etc.)

Affaire n°6

Fin de Partie et Nouveaux Niveaux : Conclusion Technique et Perspectives d’Évolution

Fin de Partie et Nouveaux Niveaux : Conclusion Technique et Perspectives d'Évolution

- **Synthèse Technique des Résultats :** La phase finale du projet "Meutre au manoir" offre une rétrospective détaillée des solutions techniques mises en place. La documentation présente un bilan complet des améliorations apportées, des défis surmontés et des performances obtenues. Le système, qui s'appuie sur une architecture modulaire robuste et une intégration fine entre Pygame, Flask, Whisper et Gemma2, a démontré sa capacité à gérer des interactions en temps réel avec une qualité de réponse remarquable. Les mécanismes de monitoring, de gestion des erreurs et d'optimisation continue ont permis de créer une expérience immersive fidèle à l'esprit du jeu Cluedo.

- **Perspectives d'Évolution Technique :** Les perspectives d'évolution du projet "Meutre au manoir" se concentrent sur plusieurs axes majeurs. Tout d'abord, l'optimisation du modèle IA reste une priorité, avec une exploration des dernières avancées en matière de traitement du langage naturel et d'apprentissage automatique. Par ailleurs, l'évolution des frameworks utilisés (tant côté serveur que côté client) permettra d'améliorer la scalabilité et la performance du système. L'enrichissement de l'interface graphique et l'automatisation du pipeline de déploiement contribueront également à pérenniser le projet. Enfin, l'intégration de nouvelles technologies émergentes ouvrira la voie à l'extension du gameplay, offrant ainsi de nouveaux niveaux d'interaction et d'immersion aux joueurs.

Affaire n°8

Annexes

Architecture technique

```
racine du projet
├── audios → Contient les fichiers audio enregistrés
│   ├── new_recording.mp3
│   ├── new_recording.wav
│   ├── temp_audio.mp3
│
├── prompts → Stocke les fichiers JSON pour les prompts des personnages
│   ├── prompts_v1.json
│   ├── prompts_v2.json
│   ├── prompts_v3.json
│   ├── prompts_v4.json
│   ├── prompts_v5.json
│
├── res → Contient les ressources visuelles et audio du jeu
│   ├── audio → Effets sonores et musiques
│   ├── background → Images des décors
│   ├── dead → Sprites liés au corps de la victime
│   ├── fonts → Polices utilisées dans le jeu
│   ├── player → Sprites et animations du personnage principal
│   ├── png1 → Animations pour le premier PNJ
│   ├── png2 → Animations pour le second PNJ
│   ├── png3 → Animations pour le troisième PNJ
│   ├── title → Contient des images pour l'écran titre
│
├── venv → Environnement virtuel Python
│
├── .gitignore → Fichiers à exclure du versionnement
├── api.py → Serveur Flask qui gère les requêtes du jeu
├── compare_prompts.py → Script d'analyse des performances des prompts
├── lancer.py → Fichier potentiellement utilisé pour exécuter le jeu
├── main.py → Contient le moteur du jeu sous Pygame
├── README.md → Documentation générale du projet
├── requirements.txt → Dépendances Python nécessaires
├── transcript.json → Fichier généré pour stocker les transcriptions audio
```

Diagramme de logique

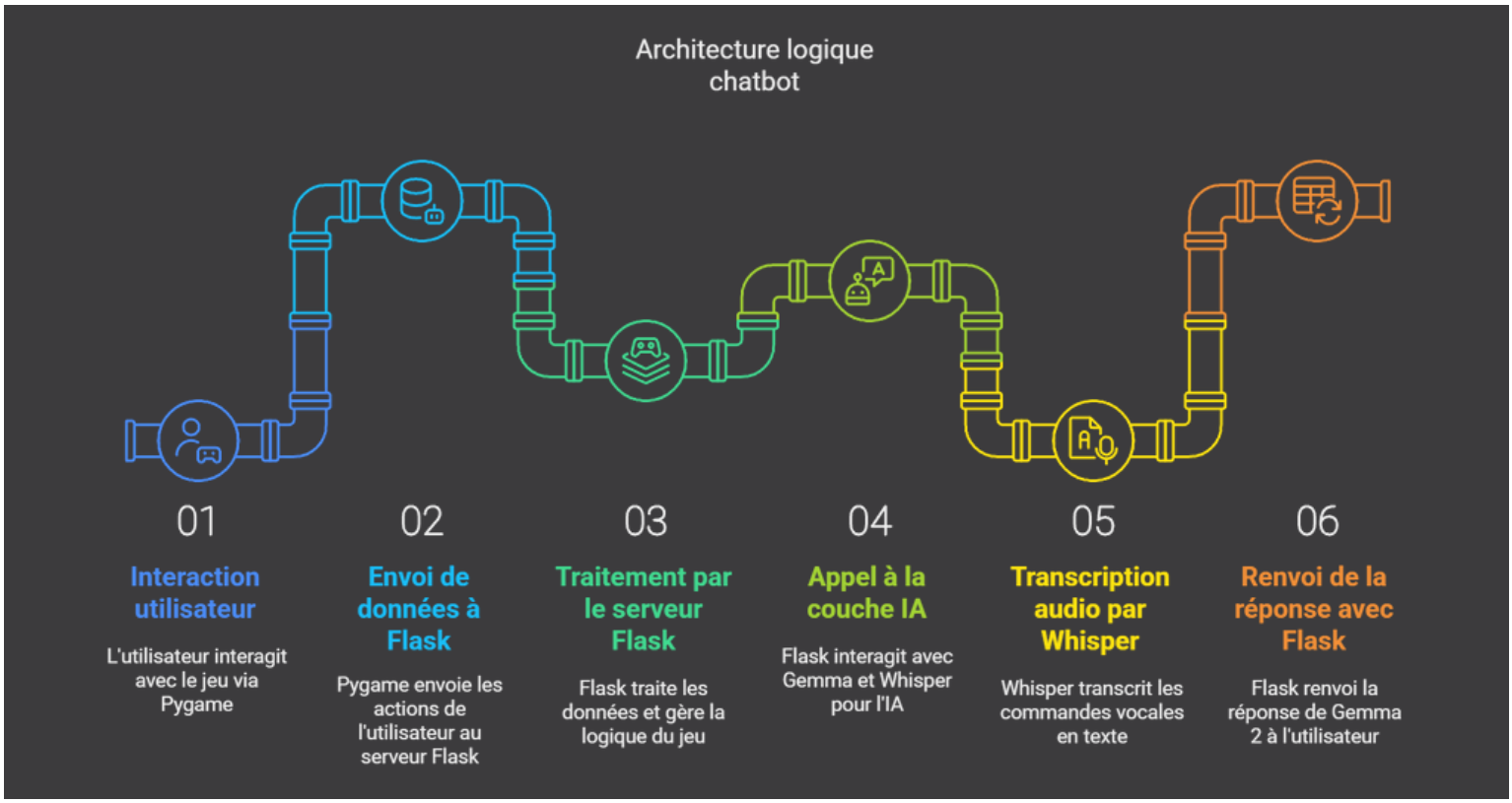

Diagramme de flux

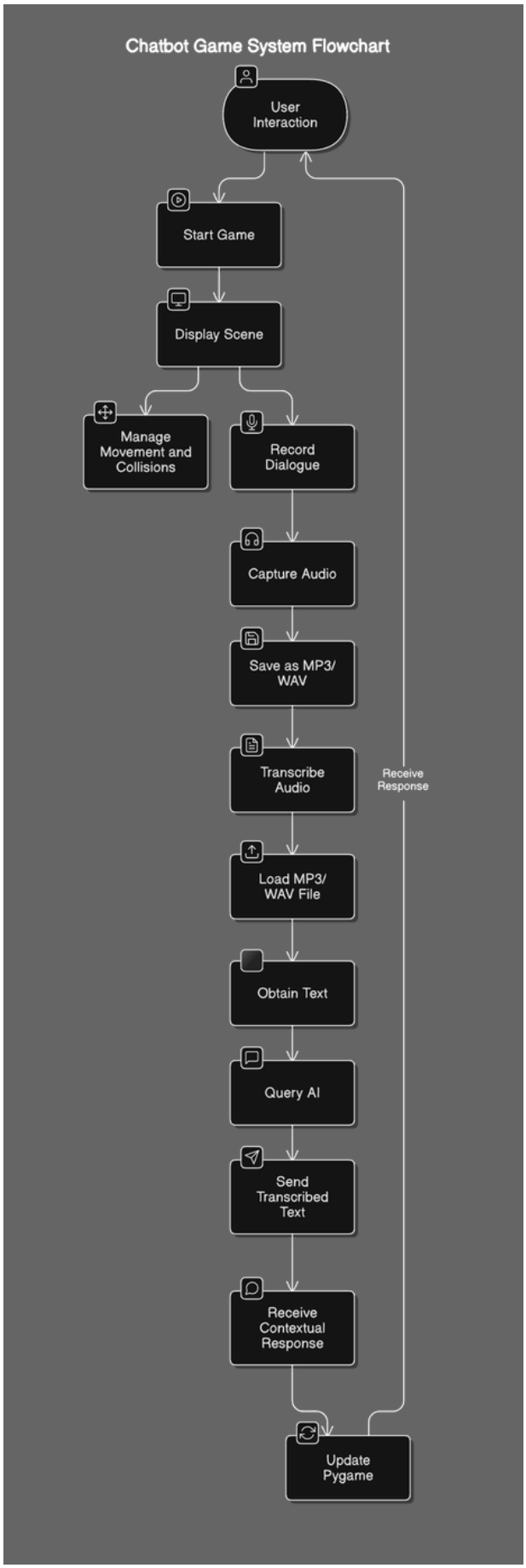

Diagramme de séquence