Rapport BE C++ : Piano Hero

Alexandre Collobert - Corentin Tatger

I. Introduction

Piano Hero est un projet de jeu vidéo se basant sur le concept de Guitar Hero. Le joueur doit parvenir à jouer des chansons avec le moins de fausses notes possibles parmi une Playlist prédéfinie.

Plusieurs modes sont disponibles dans Piano Hero à partir de l'interface principale du jeu. Le premier mode est de jouer simplement au jeu. Le second est le mode Jukebox qui consiste à écouter la musique sélectionnée. Le jeu se joue à l'aide de 5 boutons auxquels des notes sont automatiquement attribuées pour chaque titre.

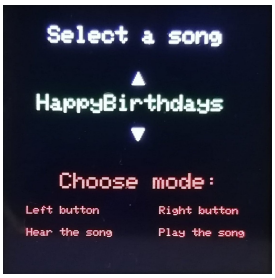
Côté écran, l'interface assez minimaliste du jeu montre un bandeau représentant les 5 boutons. Sur la partie haute de l'écran apparaissent les notes de la chanson sélectionnée qui doivent être jouée au moment au moment où elles sont à la place la plus proche du bandeau.

II. Matériel

Lors du projet, nous avons eu à notre disposition une carte ESP8266, un écran LCD Adafruit ST7789 (240x240), un buzzer passif, un joystick ainsi que 5 boutons poussoir.

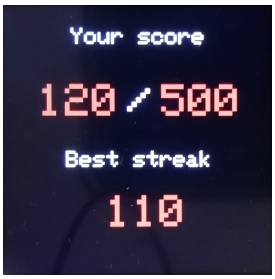
III. Fonctionnement

Le système comprend une page d'accueil dans lequel le nom du jeu s'affiche. Après un délai, on passe au menu suivant, celui du choix de la chanson, sur lequel on peut se déplacer à l'aide du joystick dans la playlist en affichant le nom des chansons.



On valide de nouveau avec les boutons. Un pour choisir de lancer le mode "JUKEBOX" qui joue la musique à l'aide du buzzer, un autre pour lancer le jeu. Au lancement du jeu, les boutons restants deviennent des touches de piano et le joueur doit appuyer sur la bonne pour augmenter son score.

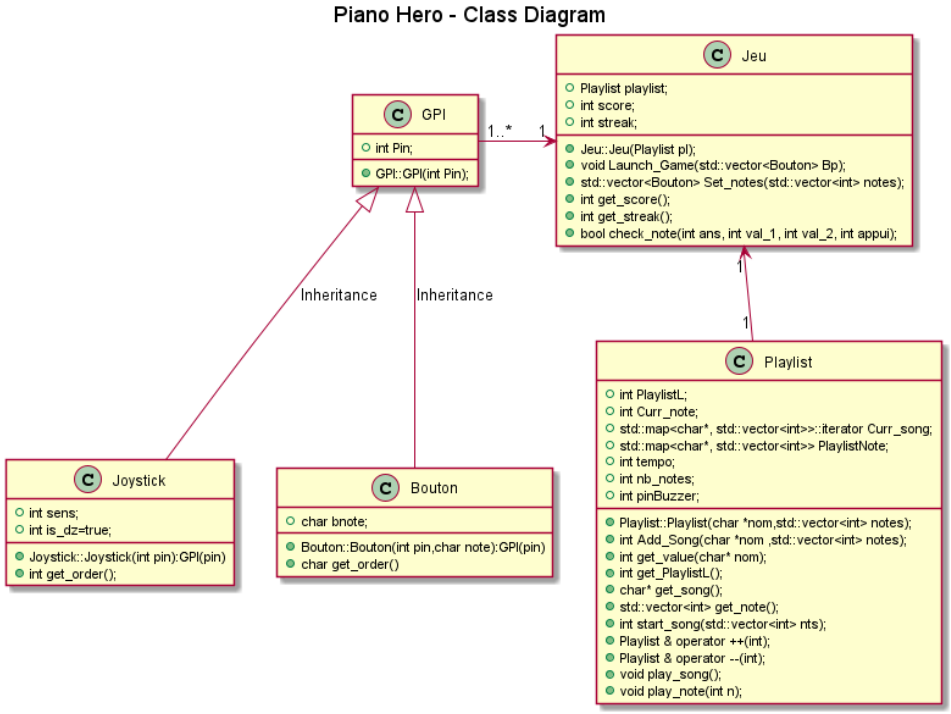
Le joueur peut voir à l'écran les prochaines notes de la chanson, ainsi qu'une interface indiquant sur quel bouton appuyer pour jouer chaque note. Au cours de la partie, l'écran affiche également en bas de l'écran le score en cours, qui augmente de à chaque bonne note, et le streak qui indique la série de notes justes en cours. A la fin de la chanson, l'écran affiche un menu de fin de partie avec le score final et le dernier streak, après une temporisation le jeu retourne au menu du choix du titre.



IV. UML

Le projet est composé de 5 classes :

- Playlist
  - Gère la liste de titres ainsi que des notes qui compose ce titres
- Jeu
  - Gère le mécanisme du jeu, comment il fonctionne
- GPI
  - Gère les ports qui recevront des informations
- Bouton
  - Hérite de GPI, elle gère l'utilisation des boutons durant les différentes phases du jeu
- Joystick
  - Hérite de GPI, elle gère l'utilisation du joystick dans l'interface d'accueil



V. Conclusion

Le projet est encore améliorable et de nombreuses pistes d'amélioration sont envisageables. Le problème majeur rencontré est que même si codée, la partie jeu du programme n'a pas pu être testée et implémentée. Faute de temps de debuggage important sur les parties précédentes. Le système fonctionne donc uniquement en mode "JUKEBOX" en l'état actuel. Afin de respecter le cahier des charges il est aussi possible d'inclure un mécanisme d'héritage, présenté sur le diagramme de classe. Tous les capteurs peuvent hériter d'une classe mère "GPI", ce qui permettrai d'ajouter n'importe quel type périphérique dans une version future du projet.

Une gestion des exceptions est également envisageable. Par exemple le stockage des notes étant sur un format bien particulier (Une note associé à sa durée et le tempo au début), la taille de cet élément doit toujours être impaire. Un simple try/catch sur la fonction d'ajout d'une musique et le constructeur de playlist vérifie cette contrainte.

Les pistes d'améliorations sont multiples, changer les types d'actionneur pour le jeu en plaçant des capteurs tactiles plutôt que des boutons poussoirs pour la simplicité d'utilisation, créer un pcb et une vraie maquette du produit. Pour ce qui est du code, un système de highscore, un affichage plus attrayant et un système de partie en ligne en multijoueur semblent être les plus adaptés à ce projet.