Piano Hero

2013

Jean Dessane

Dossier ISN

Cahier des charges :

Guitar hero est un jeu musical qui se joue grâce a une manette en forme de guitare, celle-ci possède 5 boutons au niveau du manche de la guitare ainsi qu'une autre bouton au niveau de la main droite simulant les cordes.

A l'écran on voit des notes défiler verticalement sur un manche de guitare.
Le but est d'appuyer sur le bouton correspondant a la bonne note au moment ou elle arrive a un certain point sur l'écran.

Notre projet est de réaliser un jeu suivant le même principe que guitar hero à la seule différence que la manette sera un piano au lieu d'une guitare.

Nous comptons utiliser une carte arduino pour créer le piano (7 boutons poussoirs y seront reliés).
Pour l'interface graphique nous utiliserons un programme en C avec la librairie SDL couplée à la librairie FMOD pour la gestion du son.

Le piano communiquera avec le programme sur ordinateur qui se chargera de vérifier que les boutons pressés par le joueur correspondent bien a celles affichées a l'écran, gérera le défilement des notes a l'écran et la lecture des fichiers sonores.

Decoupage du travail :

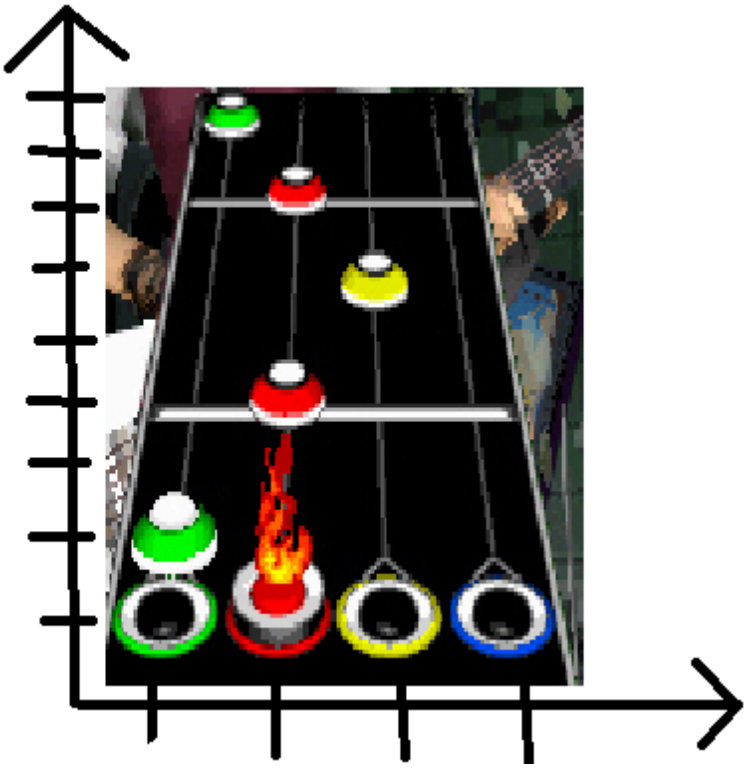
Ce projet a l'avantage d'être facilement découpable en plusieurs parties presque complètement distinctes, permettant a chaque membre du groupe de travailler séparément pour n'assembler toutes nos parties qu'a la fin.
Après avoir réalisé notre cahier des charges nous avons tout de suite pu déterminer 4 grandes parties :

Le menu:

Ce menu permet de naviguer dans notre jeu au clavier et a la souris afin d'accéder a une liste de choix de morceaux, mais également à un tutoriel et aux crédits.

Defilement des notes a l'ecran :

Pour chaque morceau, toutes les notes sont caractérisées par des coordonnées.
On affiche ensuite les notes sur le repère suivant :



L'abscisse correspond au bouton poussoir qui doit etre enfoncé par le joueur, chaque bouton poussoir correspond a une note de piano (do, ré, mi, fa, sol, la, si).
Le defilement étant vertical (les notes descendent vers le bas), l'ordonnée correspond au temps que la note mettra a arriver au niveau de l'ordonnée 1 (ou 0), c'est à ce moment la que le joueur doit appuyer sur le bon bouton poussoir et sur l'interrupteur.

A chaque fin de la boucle principale du programme on décrémente l'ordonnée de toutes les notes du morceau.

Les notes doivent parvenir à une certaine ordonnée au moment précis où le joueur l'entend, cette partie suppose donc une maîtrise parfaite du temps.

Le score:

Lorsque les notes sont « jouées » par le joueur on augmente le score, on comptabilisera à la fin du morceau le nombre de notes réussies pour afficher un pourcentage.

La gestion des evenements clavier :

Gestion des évenements clavier pour savoir quand le joueur appuie sur la touche qui correspond à la note et si elle est jouée au bon moment.

Détermination des coordonnées des notes:

Pour que le jeu ait un intérêt musical, il faut évidemment que les notes affichées à l'écran correspondent a celles que l'on entend dans la musique jouée. Il faut donc déterminer précisément comment placer ces notes.
Pour cela nous avons choisi de récuperer les dates et la fréquence de chaque note à afficher à partir de fichiers midi. Contrairement aux autres formats de fichiers audio, les fichiers midi ne contiennent pas un signal mais ne stockent que des commandes de volume, de déclenchement de tel instrument musical, d'arrêt. On peut donc utiliser ces fichiers comme « partitions » pour notre jeu.
Nous aurions pu extraire toutes ces informations des fichiers midi directement sous forme binaire mais cela nous a paru beaucoup trop difficile a traiter.
Nous avons donc utilisé un progamme externe qui permet de traduire ces instructions en fichier texte afin de n'avoir que des chaines de caractères plus faciles à analyser.

La plus grosse difficulté de cette partie consiste donc a repérer puis extraire les informations qui nous intéressent dans un fichier texte.

Réalisation et programmation de la manette:

La manette est constituée de 7 boutons poussoirs (un pour chaque note d'un octave) et d'une carte arduino.
Celle-ci a pour seul rôle de communiquer à l'ordinateur quels boutons sont enfoncés, ensuite le programme du jeu se charge de traiter ces informations afin de déterminer si le joueur a bien appuyé sur le bon bouton au bon moment.

Il a donc fallu programmer la carte afin qu'elle envoie toutes ces informations par le port série et écrire un morceau de programme qui permet de recevoir et analyser ces informations sur l'ordinateur.

Parties traitées :

Intro

Pour l'introduction du jeu on affiche deux images à la suite, la 1ere comme dans les vrais jeux commercialisés (comme dans guitar hero) et la deuxieme propose d'appuyer sur une touche pour lancer le menu d'accueil. Si on appuie sur un touche le menu d'accueil se lance. On a fait clignoter une image « appuyez sur une touche » en fondu.

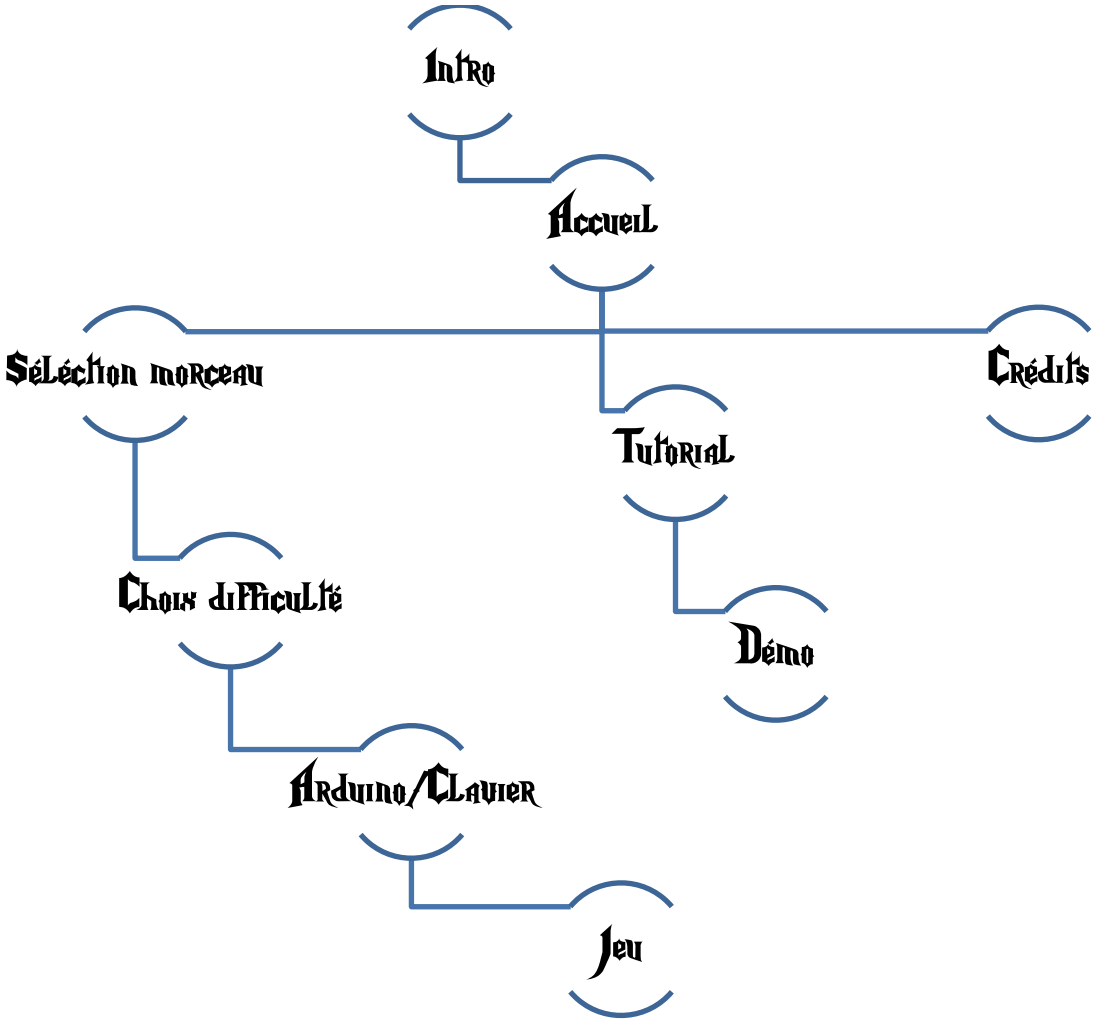
Menus

Pour notre jeu codé en C, nous avons utilisé la librairie SDL qui nous a permis de créer une fenêtre et d'afficher les images. Pour le menu d'accueil nous avons donc juste eu besoin d'afficher une image de fond sur laquelle le texte était déjà inscrit puis on déplace un curseur qui est une autre image grâce à la souris ou au clavier. Pour la souris on créé des intervalles de coordonées du curseur dans le repère qui correspondent à chaque fonction, pour le clavier on augmente ou on diminue les coordonnées du curseur. 
On lance ensuite les autres fonctions en appuyant sur ENTREE ou avec la souris en fonction des coordonnées du curseur.

Les autres menus sont fait de la même manière pour le curseur

Pour la séléction des morceaux, de la difficulté et de Arduino/Clavier c'est un peu plus complexe car on doit lancer le jeu en prenant en compte les choix, ainsi chaque choix correspond à une valeur que l'on renverra à la fonction jeu .

Les menus se succèdent de cette manière :



Musiques

On a utilisé la librairie pour la SDL *fmod* qui lit les fichier musicaux. Notre jeu lit exclusivement des fichiers midi, ceux à partir desquels on a fait la partition. Pour être synchronisé avec les notes qui défilent on lance la musique au bout de 2.7sec, temps que met une note pour descendre jusqu'à la barre ou l'on est sencé jouer la note.

Jeu – Lecture Partition

En reprenant une partie de la fonction qui lit les fichier textes midi on lit la partition donnée par le programme qui fait les partitions. Ainsi on donne à chaque note les positions dans l'espace et dans le temps au moyen de deux tableaux : note [ ] et début[ ] . On les réutilisera ensuite pour les afficher. On prend aussi la fin du morceau (*tempsFin*) qui correspond aux 8 dans la partitions. Il y a plusieurs 8 à la fin car pour les niveaux de difficulté on analyse la partition par exemple uniquement toutes les 3 notes pour facile.

Jeu – Gestion du temps

Pour gerer le temps, on utilise la fonction de la SDL : *SDL_GetTicks().* Cette fonction donne le temps qui s'est écoulé depuis le lancement du programme. Ici ce n'est pas ce temps là qui nous interesse mais le temps auquel la fonction jeu a commencé. On donne donc à une variable le temps auquel la fonction jeu a débuté pour la soustraire à *SDL_GetTicks().*
On a donc : *int tempsDebut=SDL_GetTicks();*
Puis dans la boucle on utilise une méthode simple :
On a deux variables de temps : tempsActuel , tempsPrecedent .
En faisant *tempsActuel = SDL_GetTicks() - tempsDebut;* On a le temps actuel réel du morceau
En faisant *tempsPrecedent = tempsActuel;* On pourra obtenir le temps qu'il s'est passé entre les deux tours de boucles (cela servira pour faire descendre les notes)

Jeu – Descente des notes

Afin d'avoir une descente précise des notes et que les notes descendent toutes à la même vitesse quelque soit la puissance de l'ordinateur, on les fait descendre en fonction du temps grâce à cette comande : *positionNote[l].y= positionNote[l].y+tempsActuel/10*2-tempsPrecedent/10*2;*
Pour l'incrémentation de la position en ordonée on ajoute la différence de temps entre les deux tours de boucle qu'on divise par 10 pour obtenir une valeur de bonne grandeur et on le mutiplie par 2 pour que la note ne descende pas trop lentement (on pourrait diviser par 5 mais c'est plus compréhensible comme cela)
On utilise un tableau car il y a une boucle do/while qui permet d'afficher toutes les notes qui doivent être affichées.

Jeu – Evenements clavier

On récupère les touches appuyées et on donne à une variable la valeur 1 pour afficher l'image de la touche appuyée et pour compter le score.
Le gros problème de la SDL est qu'elle ne peut gérer qu'un seul evenement en même temps. Pour notre jeu cela posait un problème car les accords (plusieurs notes à la fois) au piano sont très courants.
Ainsi pour contourner le problème on a laissé la valeur 1 aux variables pendant 250ms ce qui donne l'impression que les touches sont toutes prises en compte alors qu'elles sont prisent une par une.

Jeu – Score

On utilise la variable qui correspond à la note jouée qui est égale a 1 si on appuie et 0 si on appuie pas et les positions des notes. Si la position de la note est dans l'intervalle de tolérance et que la variable de la note correspondante est égale à 1 alors on augmente le score et la note à été jouée. On avait ajouté un son de fausse note si la note n'était pas jouée mais on ne l'utilise pas car sur un morceau difficile avec beaucoup de notes c'est insuportable.
Ensuite comme on a récuperé le nombre de notes total et le nombre de notes jouées on fait le pourcentage qu'on affichera après.

Textes

Pour le texte on a utilisé une autre librairie de la SDL : *SDL_ttf*. Cette fonction crée des surface avec les caractères que l'on souhaite. On a juste besoin d'une police.
La fonction est : *TTF_RenderText_Solid(police, caracteres, couleurNoire);*
On utilise Solid car même si la qualité est inférieur aux autres modes elle est plus rapide à exécuter et comme on change le texte du score souvent Solid est plus adapté.
Les paramètres sont (la police utilisée qui a été définie avant, les caractères à afficher qui ont aussi étés définis avant, la couleur du texte à afficher aussi définie avant).

Icone

Pour associer une icône au fichier.exe on a ajouté un .rc au projet dans lequel on a mis le chemin vers l'icone, il l'associe ainsi au .exe.

Choix

Pour les differents choix on a mis en arguments de la fonction jeu les valeurs renvoyées par les differents menus. Ensuite au moyen d'un switch on fait correspondre les fichiers .txt et .mid au choix et le nom du morceau qu'on affiche en haut à gauche dans la fonction jeu.

Partitions

Les partitions données par le programmes sont présentées de manière spécifiques et simples.

Temps en ms

Note

Le temps est donné en ms et à chaque note est associé une valeur : 0 = do
1 = ré
2 = mi
3 = fa
4 = sol
5 = la
6 = si

Le 8 correspon à la fin du morceau. On a déjà expliqué qu'on met au moins quatres 8 pour les niveaux de difficulté.

Exemple - Frère Jacques :

```
0 - 0 do
500 - 1 ré
1000 - 2 mi
1500 - 0 do
2000 - 0 do
2500 - 1 ré
3000 - 2 mi
3500 - 0 do
4000 - 2 mi
4500 - 3 fa
5000 - 4 sol
6000 - 2 mi
6500 - 3 fa
7000 - 4 sol
8000 - 4 sol
8250 - 5 la
8500 - 4 sol
8750 - 3 fa
9000 - 2 mi
9500 - 0 do
10000 - 4 sol
10250 - 5 la
10500 - 4 sol
10750 - 3 fa
11000 - 2 mi
11500 - 0 do
12000 - 0 do
12500 - 4 sol
13000 - 0 do
14000 - 0 do
14500 - 4 sol
15000 - 0 do
19000 - 8 fin
19000 - 8 fin
19000 - 8 fin
19000 - 8 fin
```

Autres parties :

Démo

On a réutilisé la fonction jeu et fait un perfect pour montrer comment jouer.

Crédits

On fait monter les noms qui sont dans une image grâce à la même commande que celle qui fait descendre les notes dans le jeu sauf qu'ici on fait monter l'image.

Arduino

On a fait un montage avec la plaque arduino et on utilise des boutons poussoirs. On a programmé l'arduino pour qu'elle renvoit des chaines de caractères, elles sont ensuite annalysées et retransmises à la fonction jeu.

Ecriture de partitions

On utilise un programme qui transcrit les fichier .mid en fichier texte. Ces fichiers textes sont ensuite analysés puis on crée un autre fichier texte qui contient la partition.

Captures d ecran :