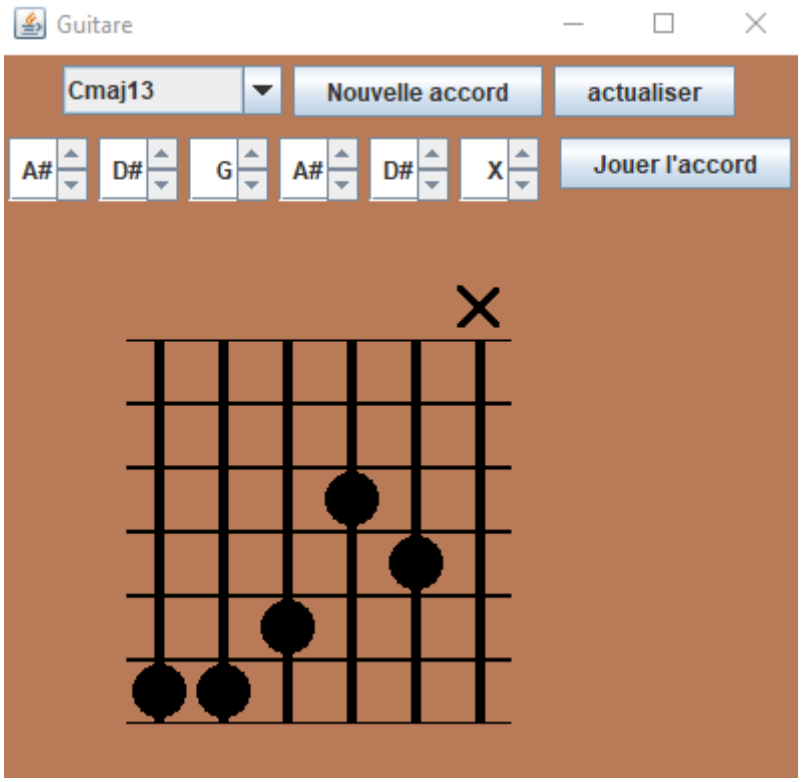
RAPPORT PROJET JAVA

FAMECHON Hugo

TRINQUART Matthieu

Visualisation graphique d'accords de guitare



SOMMAIRE

Introduction

Présentation du projet :

Dans ce projet nous avions pour but de réaliser un logiciel permettant de visualiser des accords de Guitare en utilisant Java.

Pour pouvoir facilement éditer ou rajouter de nouveaux accords à la liste des accords connus du logiciel, nous devons les stocker dans un fichier texte.

Ceci permettra de rajouter des accords sans modifier le programme.

De plus notre Logiciel possède d'autres fonctionnalités qui n'était pas forcement demandé comme :

-Enregistrer des accords via l'application

-Pouvoir modifier les accords via un JSpinner.

-Reconnaitre un accord s'il existe dans la base de données des accords.

-Jouer l'accord actuellement sélectionné.

Analyse

Classes principales utilisées :

Dans la conception de notre application nous avons utilisé plusieurs classes :

-Principal.java -> Le main qui s'occupe de simplement instancier une Fenêtre.

-Fenetre.java -> Contient un BorderLayout qui vas afficher les JPanels de manière correcte.

-Guitare.java -> classe qui contient 6 entiers statiques représentant les notes des cordes affichées.

-ZoneDessin.java -> Affiche les Sprites des Cordes et des Points du manche de la guitare et permet la modification de ceux-ci par un clic de la souris.

-MenuGuitare.java -> S’occupe d’afficher les JSpinner des notes de chaque corde.

-Menu.java -> S’occupe d’afficher et de gérer la JComboBox pour afficher l’accord sélectionné ainsi que le bouton pour sauvegarder un accord.

-LectureFichier.java -> S’occupe de lire le fichier data.txt où les accords sont stockés.

-EcouterAccord.java -> Un JButton qui, lorsque cliqué, joue l’accord de guitare sélectionné.

Classes secondaires :

Ces classes sont des petites classes qui ne rajoutent qu’une seule fonctionnalité :

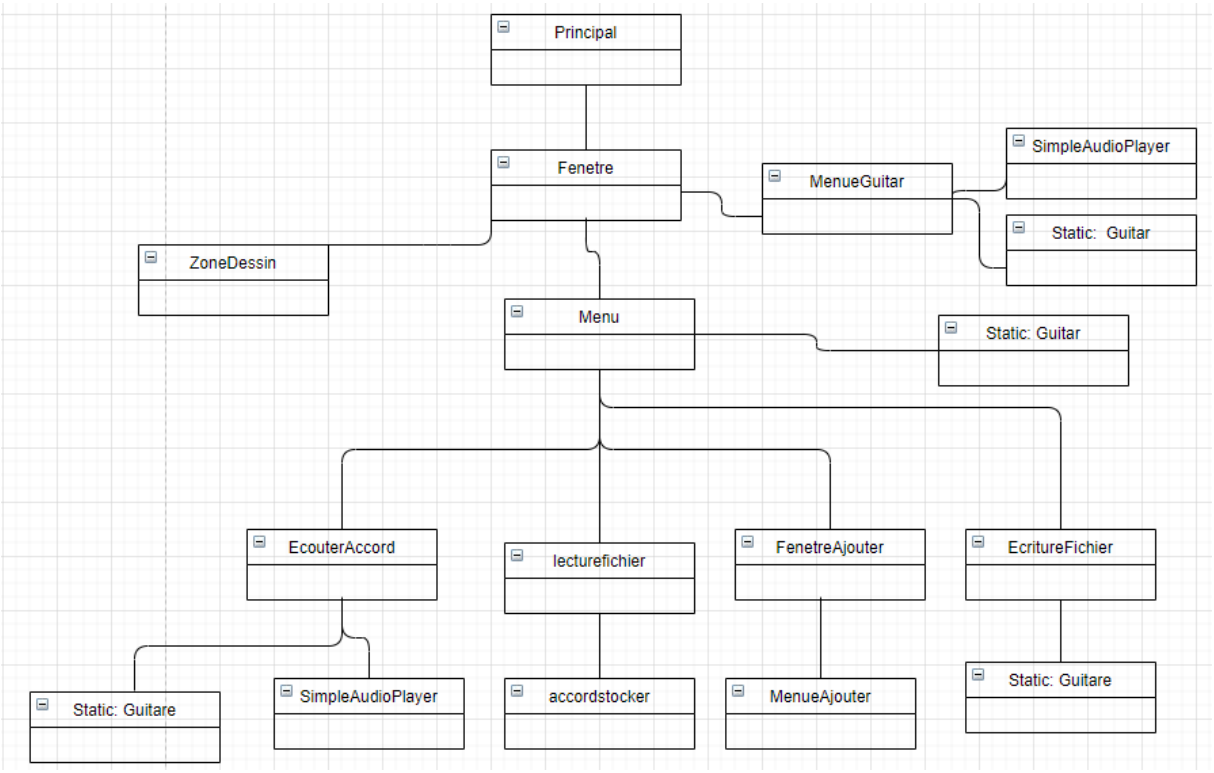
-MenueAjouter.java -> Le JPanel qui contient le JTextField et un JButton pour enregistrer un accord

-FenetreAjouter.java -> Affiche la fenêtre contenant MenueAjouter.java

-SpinnerCircularListModel.java -> Permet de faire boucler les JSpinners lorsqu’ils arrivent à la fin du tableau.

-SimpleAudioPlayer.java -> s’occupe de la lecture de fichiers .WAV

Relations entre les classes



Fonctionnement global

Dans notre projet, on a Fenetre.java qui va afficher tous les JPanels dans un BorderLayout et Guitare.java qui vas être utilisé par toute les classes manipulant les notes des Cordes.

Réalisation

Choix techniques :

De manière générale les classes affichant des informations comme le nom de l'accord ou bien les notes de la guitare s'actualisent avec un Timer. Chaque fois que l'évènement du Timer est appelé alors les classes vont relire Guitare.java (qui stocke les notes des cordes actuelles avec des variables statiques) et les notes affichées vont être mis à jour.

Bien que très loin d'être optimale c'est malheureusement la seule solution que nous avons trouvée pour actualiser les notes.

Présentation des algorithmes utilisés

Modification de l'affichage de la guitare. (ZoneDessin)

- Quand on clique on vérifie si on est dans la zone des cordes de la guitare.
- Ont converti les coordonnées en pixel vers des coordonnées en "block" qui correspondent aux positions du tableau.
- On nettoie le tableau précèdent en stockant la valeur de la corde à l'endroit cliqué
- Vérification avec une série de if/else if de tous les cas possibles.
- Lorsque l'on modifie le tableau on sauvegarde l'endroit que l'on a modifié.
- On actualise la classe Guitare avec la note nouvellement modifiée grâce à des switchs.
- On repeint la classe ZoneDessin

Voir code ZoneDessin ::modifieTableau()

Détection de l'accord actuellement affiché. (Menu.changeSelect())

- A chaque fois que le Timer est appelé, on vérifie si l'accord affiché correspond à un accord stocké et, si notre String nous dit que l'accord de la dernière vérification est diffèrent de l'accord actuel.
- Si c'est le cas, on stocke une chaine qui correspond aux notes actuelles ainsi qu'un String qui permet de savoir si la dernière fois que l'on a vérifié, les notes étaient différentes de la chaine ou non.
- On change donc la valeur du JComboBox par le nom de l'accord si l'accord correspond et par "N'existe pas" si l'accord n'est pas reconnu.

La raison pour laquelle nous vérifions si l'accord précèdent est différent de l'accord actuel est pour éviter que le programme ne rechange la JComboBox si l'utilisateur ne touche à rien. En effet les inputs de l'utilisateur pouvaient être "mangé" par le programme qui actualise la JComboBox résultant à une modification qui "s'annule".

```
public void changeSelect()
{
    int i = 0;
    while( i< test.liste.size() )//&& avant != test.liste.elementAt(i).Nom)
    {
        if( Guitare.Corde1 == test.liste.elementAt(i).corde1 &&                 //la il rentre que si ça correspond et que P est different de "avant etait le meme"
            Guitare.Corde2 == test.liste.elementAt(i).corde2 &&
            Guitare.Corde3 == test.liste.elementAt(i).corde3 &&
            Guitare.Corde4 == test.liste.elementAt(i).corde4 &&
            Guitare.Corde5 == test.liste.elementAt(i).corde5 &&
            Guitare.Corde6 == test.liste.elementAt(i).corde6 &&
            !P.equals("avant etait le meme"))
            {
                System.out.println("Ceci correspond a un truc du JComboBox");
                P = "avant etait le meme";

                chaineCle =
                Integer.toString(test.liste.elementAt(i).corde1) +
                Integer.toString(test.liste.elementAt(i).corde2) +
                Integer.toString(test.liste.elementAt(i).corde3) +
                Integer.toString(test.liste.elementAt(i).corde4) +
                Integer.toString(test.liste.elementAt(i).corde5) +
                Integer.toString(test.liste.elementAt(i).corde6) ;

                LeNomDeTaComboBox.setSelectedItem(test.liste.elementAt(i).Nom);
            }
            ++i;
    }
    if(P.equals("avant etait le meme"))//la il rentre que si "avant etait le meme"
    {
        String EtatActuel =
        Integer.toString(Guitare.Corde1) +
        Integer.toString(Guitare.Corde2) +
        Integer.toString(Guitare.Corde3) +
        Integer.toString(Guitare.Corde4) +
        Integer.toString(Guitare.Corde5) +
        Integer.toString(Guitare.Corde6) ;

        if(!chaineCle.equals(EtatActuel))
        {
            P = "Differents";
            LeNomDeTaComboBox.setSelectedItem("N'existe pas");
            System.out.println("Il y a eu un changement comparer a la derniere selection");
        }
    }
```

Format de fichier

Le fichier data.txt est composé du nom de l'accord, puis, sur la ligne d'après de la composition de celui-ci.

0 -> note en haut de la corde

1 -> note 1 de la corde

...

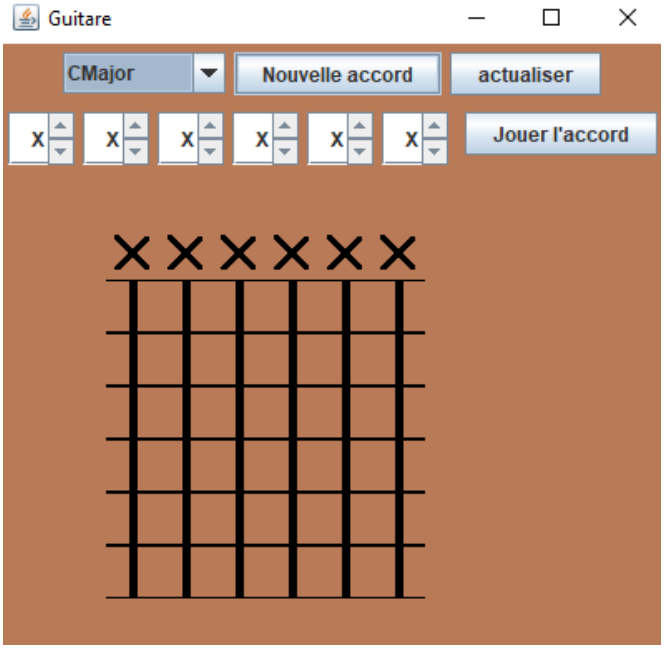
6 -> note 6 de la corde

7 -> On ne joue pas la corde

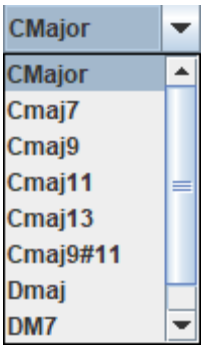
Utilisation.

Mode d'emploi

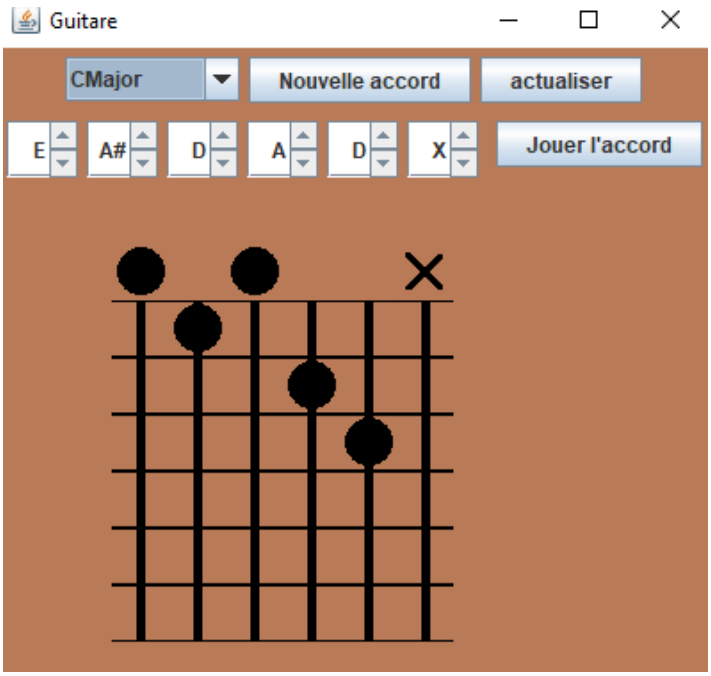
Lancer le logiciel.
Une fenêtre s'ouvre avec l'interface si dessous



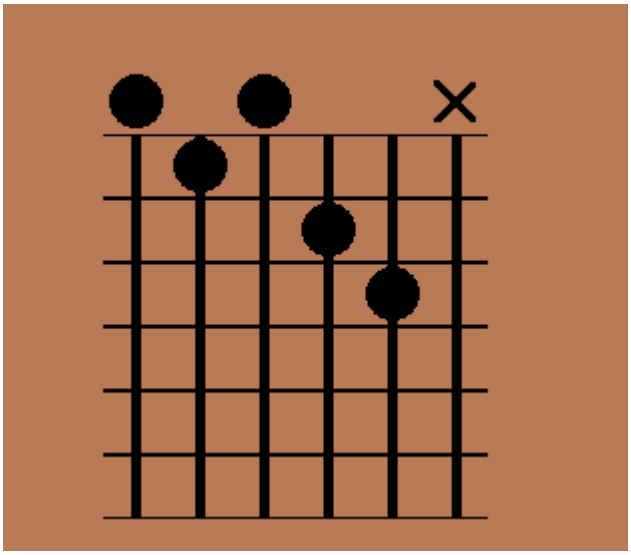
Si vous voulez afficher un accord sélectionner la JComboBox (La case a coter du bouton "Nouvelle accord") puis sélectionner l'accord que vous voulez afficher



ET LA MAGIE L'ACCORD S'AFFICHE

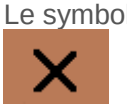


Si vous cliquer sur la partie graphique du manche de guitare vous pouvez sélectionner une corde précise ou même les placer avec les JSpinners.
Manche de Guitard :

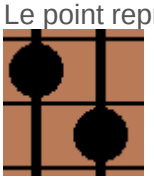


JSpinners :

Le symbole de la croix représente qu'on ne gratte pas la corde.

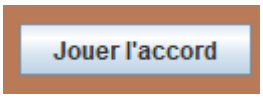


Le point représente que l'on gratte la corde en appuyant sur un endroit précis de la corde.



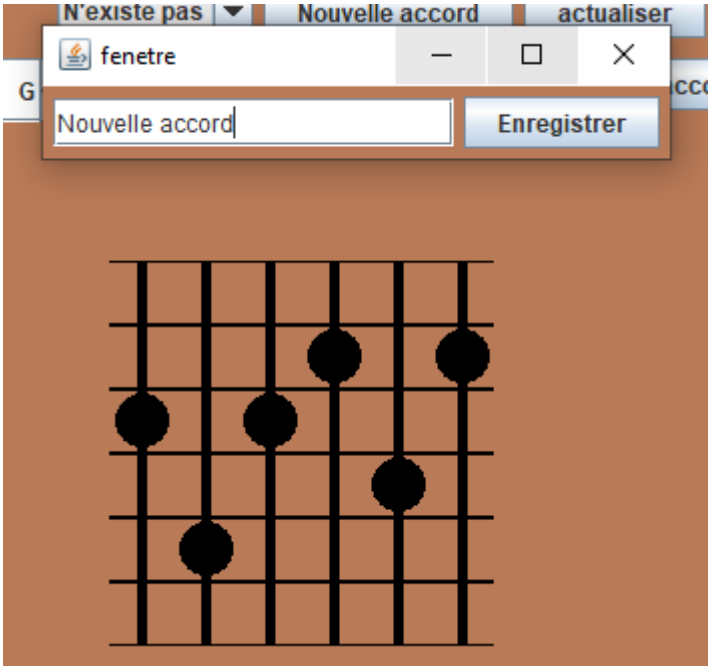
Si le point est tout en haut du manche cela symbolise que l'on gratte la corde mais qu'on n'appuie pas dessus

Quand vous avez sélectionner un accord soit avec les JSpinners soit avec le manche de guitare le nom de l'accord sera afficher sur la JComboBox.
Vous pouvez écouter l'accord dessiner avec le bouton "Jouer l'accord"



Il est possible aussi de créer sont propre accord en appuyant sur le bouton "Nouvelle accord"

Ce bouton va afficher une nouvelle fenêtre ou rentrer le nom de votre nouvel accord et créer votre accord à partir du manche de la première fenêtre. Appuyer sur le bouton "Enregistrer" pour enregistrer votre accord.



Appuyer ensuite sur le bouton actualisé et vous pouvez afficher l'accord que vous avez créé.
BRAVO.

Conclusion

Bilan

En bilan on a fini tout ce que le projet nous demandé plus quelque bonus comme le fait que l'utilisateur puisse créer son propre accord ou même qu'il puisse déplacer c'est corde avec des Jspiner

Optimisation possible

Le fait que l'actualisation entre la position des points et les Jspiner soit fait avec un timer n'est pas très optimal. Il aurait mieux fallu que l'actualisation se fasse à chaque fois qu'on met un point sur le manche de guitare ou quand on modifie le Jspiner.

Extensions possibles

Comme extensions possibles on pourrait imaginer que le son d'une corde ce déclenche à chaque fois que l'utilisateur modifie la position d'une corde. On pourrait aussi imaginer que l'utilisateur puisse assembler plusieurs accords à la suite et que le son de c'est accord se fasse pour qu'il puisse créer sa propre musique.