EPHEC

Louvain-la-neuve

Traitement de signal

Projet - Accordeur de guitare

Rapport de projet

3TL1

Herrier Lucie Juckler Christian Musuvaho Grace Nyssens Sylvain

13 décembre 2014

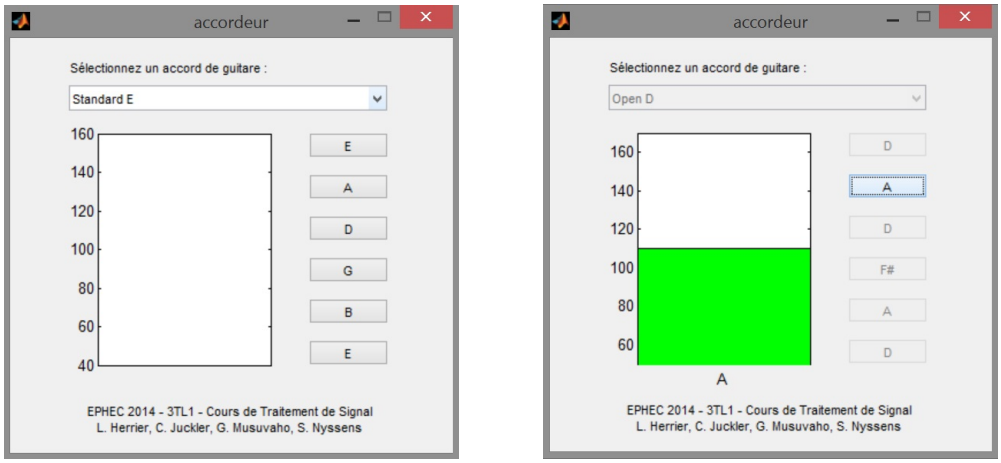
Table des matières

1 Introduction

Dans le cadre du cours de travaux pratiques de traitement de signal, nous avons dû réaliser un projet mettant en application nos acquis. Notre groupe a choisi de créer un accordeur de guitare. Celui-ci a été codé à l'aide de Matlab, l'outil utilisé lors des exercices à réaliser durant le cours. Ce rapport présente tout d'abord en quelques lignes en quoi consistait le projet. Nous expliquons par la suite la méthodologie que nous avons utilisée, et nous apportons des idées d'améliorations possibles pour notre accordeur. Enfin, chaque membre du groupe apporte sa conclusion personnelles concernant ce projet.

2 Présentation du projet

Le projet que nous avons réalisé est un accordeur de guitare. Celui-ci a été codé à l'aide du logiciel utilisé lors des TPs, Matlab. Nous avons choisi d'implémenter une interface graphique, plus intuitive à utiliser. Celle-ci propose d'accorder sa guitare selon plusieurs accords fréquemment utilisés. La figure 1 montre l'interface utilisateur de l'accordeur.



(a) Accordeur au démarrage (b) Accordeur en train d'accorder

FIGURE 1 – Interface utilisateur

On peut voir sur les images ci-dessus la manière dont nous avons composé notre accordeur. Dans le haut de la fenêtre se trouve le menu de sélection des accords. Nous proposons divers accords fréquents pour la guitare :

- L'accordage classique en *E*
- Le *Drop D*
- L'*Open D*
- L'*Open G*
- L'accordage en *quartes*

Les boutons de droite représentent les notes composant l'accord. Celui du dessus correspond à la corde du dessus lorsqu'on tient une guitare, autrement dit la note la plus grave. Enfin, le grand rectangle à gauche sert à visualiser si la corde de la guitare sélectionnée est bien accordée ou non. Le code couleur utilisé est classique : vert si la corde est correctement accordée, à 2Hz près, jaune-orange si la corde est trop haute ou trop basse à 15Hz près, et rouge si elle est hors des limites précédentes.

Lorsqu'une note est sélectionnée pour être accordée, un timer de 5 secondes s'enclenche. Il est possible pendant ce temps de jouer la corde pour l'accorder. De plus, les autres notes sont bloquées, ainsi que la sélection des accords, afin d'éviter les conflits.

3 Méthodologie utilisée

Lors de la réalisation de notre accordeur, nous sommes passés par plusieurs étapes avant d'arriver au produit actuel. Cette partie du rapport explique ces différentes phases de développement, ainsi que les difficultés rencontrées au cours de celles-ci.

En premier lieu, nous avons tout d'abord cherché à enregistrer un son dans Matlab. En effet, nous voulions pouvoir visualiser le spectre fréquentiel de l'enregistrement du son produit par une corde de guitare sur une durée déterminée.

4 Améliorations possibles

Notre projet d'accordeur pourrait être amélioré de plusieurs manières. Tout d'abord, nous pourrions rajouter plusieurs autres accords à la liste de ceux déjà proposés. Cela permettrait à l'accordeur d'être encore plus efficace pour les guitaristes aimant jouer plusieurs styles musicaux différents.

Nous avions également pensé, à la base, à laisser l'accordage de la corde actif jusqu'à ce qu'une autre note soit sélectionnée. N'ayant pas réussi à mettre cette technique en place, nous avons opté pour l'implémentation d'un timer. Un amélioration possible serait donc de revenir à notre idée initiale d'accordage en continu pour une corde. Un clic sur la note lancerait le processus d'accordage, et un clic sur une autre note stopperait le premier et lancerait le nouveau.

Enfin, nous aurions pu réaliser une interface permettant, une fois l'accord sélectionné, d'accorder n'importe quelle corde. C'est-à-dire qu'au lieu des notes sur la droite et d'un graphique à gauche, nous aurions eu 6 graphiques (un par note). Nous n'avons pas pu mettre en place un tel système à cause du problème des harmoniques mentionné plus haut. Cependant, une amélioration de l'accordeur pour parvenir à une telle version serait intéressante. En effet, plus besoin de choisir la note. Le guitariste jouerait la corde qu'il veut, et il lui suffirait de regarder le graphique correspondant pour l'accorder.

5 Conclusions personnelles

5.1 Herrier Lucie

5.2 Juckler Christian

5.3 Musuvaho Grace

5.4 Nyssens Sylvain

Bibilographie

1. La documentation Matlab, `http://nl.mathworks.com/help/matlab/`, consultée durant tout le projet.

A Code

```
1  function varargout = accordeur(varargin)
2  % ACCORDEUR MATLAB code for accordeur.fig
3  % ACCORDEUR, by itself, creates a new ACCORDEUR or raises the existing
4  % singleton*.
5  %
6  % H = ACCORDEUR returns the handle to a new ACCORDEUR or the handle to
7  % the existing singleton*.
8  %
9  % ACCORDEUR('CALLBACK',hObject,eventData,handles,...) calls the local
10 % function named CALLBACK in ACCORDEUR.M with the given input arguments.
11 %
12 % ACCORDEUR('Property','Value',...) creates a new ACCORDEUR or raises the
13 % existing singleton*. Starting from the left, property value pairs are
14 % applied to the GUI before accordeur_OpeningFcn gets called. An
15 % unrecognized property name or invalid value makes property application
16 % stop. All inputs are passed to accordeur_OpeningFcn via varargin.
17 %
18 % *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
19 % instance to run (singleton)".
20 %
21 % See also: GUIDE, GUIDATA, GUIHANDLES
22
23 % Edit the above text to modify the response to help accordeur
24
25 % Last Modified by GUIDE v2.5 11-Dec-2014 22:40:42
26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name', mfilename, ...
30                    'gui_Singleton', gui_Singleton, ...
31                    'gui_OpeningFcn', @accordeur_OpeningFcn, ...
32                    'gui_OutputFcn', @accordeur_OutputFcn, ...
33                    'gui_LayoutFcn', [] , ...
34                    'gui_Callback', []);
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if nargout
40     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});
43 end
44 % End initialization code - DO NOT EDIT
45
46 % --- Executes just before accordeur is made visible.
47 function accordeur_OpeningFcn(hObject, eventdata, handles, varargin)
48 % This function has no output args, see OutputFcn.
49 % hObject handle to figure
```

```
50  % eventdata reserved - to be defined in a future version of MATLAB
51  % handles structure with handles and user data (see GUIDATA)
52  % varargin command line arguments to accordeur (see VARARGIN)
53
54  % Choose default command line output for accordeur
55  handles.output = hObject;
56
57  % Update handles structure
58  guidata(hObject, handles);
59
60  % This sets up the initial plot - only do when we are invisible
61  % so window can get raised using accordeur.
62  if strcmp(get(hObject,'Visible'),'off')
63      %% Dclaration des variables
64      La = 440;
65      ajoutDemiTon = 2^(1/12);
66      ajoutTon = 2^(1/6);
67      Octave3 = cell(2,12);
68      Octave3(1,:)={'C','C#','D','D#','E','F','F#','G','G#','A','A#','B'};
69
70      %% Gnration du tableau de la 3me octave
71      Octave3{2,10}=La;
72
73      for i = 10:11
74          Octave3{2,i+1}=Octave3{2,i}*ajoutDemiTon;
75      end;
76      for i = 10:(-1):2
77          Octave3{2,i-1}=Octave3{2,i}/ajoutDemiTon;
78      end;
79
80      %% Gnration des tableaux des diffrents accords
81      % Classique E
82      AccordE = cell(2,6);
83      AccordE(1,:)={'E','A','D','G','B','E'};
84      AccordE = initAccordEOpen(AccordE, Octave3);
85
86      % Drop D
87      AccordDropD = AccordE;
88      AccordDropD{1,1}='D';
89      for j= 1:12
90          if (Octave3{1,j}==AccordDropD{1,1})
91              AccordDropD{2,1} = Octave3{2,j}/4;
92          end;
93      end;
94
95      % Quartes
96      AccordQuartes = AccordE;
97      AccordQuartes{1,5}='C';
98      AccordQuartes{1,6}='F';
99      for i = 5:6
100         for j= 1:12
101             if (Octave3{1,j}==AccordQuartes{1,i})
```

```
102                     AccordQuartes{2,i} = Octave3{2,j};
103                 end;
104             end;
105         end;
106
107         % Open D
108         AccordOpenD = cell(2,6);
109         AccordOpenD(1,:)={'D','A','D','F#','A','D'};
110         AccordOpenD = initAccordEOpen(AccordOpenD, Octave3);
111
112         % Open G
113         AccordOpenG = cell(2,6);
114         AccordOpenG(1,:)={'D','G','D','G','B','D'};
115         AccordOpenG = initAccordEOpen(AccordOpenG, Octave3);
116
117         %% Memoriser les variables pour l'interface graphique
118         handles.AccordE = AccordE;
119         handles.AccordDropD = AccordDropD;
120         handles.AccordQuartes = AccordQuartes;
121         handles.AccordOpenD = AccordOpenD;
122         handles.AccordOpenG = AccordOpenG;
123         handles.Accord = AccordE;
124         handles.Fs = 4000;
125         handles.nBits = 8;
126         handles.nChannel = 1;
127         handles.length = 0.5;
128         handles.L=10000;
129         guidata(hObject, handles);
130
131         %% Barre de dpart et bouton de dpart avec l'accord
132         accordBar(40,100, '');
133         setBoutonsNotes(AccordE, handles);
134 end
135 % Effacer l'cran
136 clc;
137
138 % UIWAIT makes accordeur wait for user response (see UIRESUME)
139 % uiwait(handles.figure1);
140
141
142 % --- Outputs from this function are returned to the command line.
143 function varargout = accordeur_OutputFcn(hObject, eventdata, handles)
144 % varargout cell array for returning output args (see VARARGOUT);
145 % hObject handle to figure
146 % eventdata reserved - to be defined in a future version of MATLAB
147 % handles structure with handles and user data (see GUIDATA)
148
149 % Get default command line output from handles structure
150 varargout{1} = handles.output;
151
152
153
```

```
154 % --------------------------------------------------------------------
155 function FileMenu_Callback(hObject, eventdata, handles)
156 % hObject handle to FileMenu (see GCBO)
157 % eventdata reserved - to be defined in a future version of MATLAB
158 % handles structure with handles and user data (see GUIDATA)
159
160
161 % --------------------------------------------------------------------
162 function OpenMenuItem_Callback(hObject, eventdata, handles)
163 % hObject handle to OpenMenuItem (see GCBO)
164 % eventdata reserved - to be defined in a future version of MATLAB
165 % handles structure with handles and user data (see GUIDATA)
166 file = uigetfile('*.fig');
167 if ~isequal(file, 0)
168     open(file);
169 end
170
171 % --------------------------------------------------------------------
172 function PrintMenuItem_Callback(hObject, eventdata, handles)
173 % hObject handle to PrintMenuItem (see GCBO)
174 % eventdata reserved - to be defined in a future version of MATLAB
175 % handles structure with handles and user data (see GUIDATA)
176 printdlg(handles.figure1)
177
178 % --------------------------------------------------------------------
179 function CloseMenuItem_Callback(hObject, eventdata, handles)
180 % hObject handle to CloseMenuItem (see GCBO)
181 % eventdata reserved - to be defined in a future version of MATLAB
182 % handles structure with handles and user data (see GUIDATA)
183 selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
184                      ['Close ' get(handles.figure1,'Name') '...'],...
185                      'Yes','No','Yes');
186 if strcmp(selection,'No')
187     return;
188 end
189
190 delete(handles.figure1)
191
192
193 % --- Executes on selection change in popupmenu1.
194 function popupmenu1_Callback(hObject, eventdata, handles)
195 % hObject handle to popupmenu1 (see GCBO)
196 % eventdata reserved - to be defined in a future version of MATLAB
197 % handles structure with handles and user data (see GUIDATA)
198
199 % Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
200 % contents{get(hObject,'Value')} returns selected item from popupmenu1
201 cla;
202
203 popup_sel_index = get(handles.popupmenu1, 'Value');
204 % Slection de l'accordage dfini par l'utilisateur
205 switch popup_sel_index
```

```
206         case 1
207             Accord = handles.AccordE;
208         case 2
209             Accord = handles.AccordDropD;
210         case 3
211             Accord = handles.AccordOpenD;
212         case 4
213             Accord = handles.AccordOpenG;
214         case 5
215             Accord = handles.AccordQuartes;
216     end
217     jouerAccord(Accord);
218     %Modification des boutons pour les notes
219     setBoutonsNotes(Accord, handles);
220     handles.Accord = Accord;
221     guidata(hObject, handles);
222
223
224
225     % --- Executes during object creation, after setting all properties.
226     function popupmenu1_CreateFcn(hObject, eventdata, handles)
227     % hObject handle to popupmenu1 (see GCBO)
228     % eventdata reserved - to be defined in a future version of MATLAB
229     % handles empty - handles not created until after all CreateFcns called
230
231     % Hint: popupmenu controls usually have a white background on Windows.
232     % See ISPC and COMPUTER.
233     if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
234         set(hObject,'BackgroundColor','white');
235     end
236
237     set(hObject, 'String', {'Standard E', 'Drop D', 'Open D', 'Open G', 'Quartes'});
238
239     % --- Executes on button press in note1.
240     function note1_Callback(hObject, eventdata, handles)
241     activerNote(1, handles);
242
243     % --- Executes on button press in note2.
244     function note2_Callback(hObject, eventdata, handles)
245     activerNote(2, handles);
246
247     % --- Executes on button press in note3.
248     function note3_Callback(hObject, eventdata, handles)
249     activerNote(3, handles);
250
251     % --- Executes on button press in note4.
252     function note4_Callback(hObject, eventdata, handles)
253     activerNote(4, handles);
254
255     % --- Executes on button press in note5.
256     function note5_Callback(hObject, eventdata, handles)
257     activerNote(5, handles);
```

```
258
259 % --- Executes on button press in note6.
260 function note6_Callback(hObject, eventdata, handles)
261 activerNote(6, handles);
262
263 %% Fonctions personnelles
264 % Fonction d'initialisation des accords de type E ou Open sur base de
265 % l'octave 3.
266 function [Accord] = initAccordEOpen(Accord, Octave3)
267 for i=1:6
268     for j=1:12
269         x=0;
270         if (Octave3{1,j}==Accord{1,i})
271             if (i==6)
272                 x=1;
273             elseif (i<=2)
274                 x=4;
275             else
276                 x=2;
277             end;
278             Accord{2,i} = Octave3{2,j}/x;
279         end;
280
281     end;
282 end;
283
284 % Fonction permettant de jouer l'accord en paramtre
285 function jouerAccord(Accord)
286     tsec=0.5;
287     a=2;
288     F=44100;
289     t = linspace(0,tsec,tsec*F);
290     y=a*sin(2*pi*Accord{2,1}*t);
291     for i=2 :6
292         y = [y (a*sin(2*pi*Accord{2,i}*t))];
293     end;
294     sound(y,F);
295
296 % Calcul de la couleur de la barre d'accordage qui affiche o on en est par
297 % rapport l'accord
298 function accordBar(frequence,x, note)
299 if(frequence>=(x+30) || frequence<=(x-30))
300     color = [1 0 0]; % Rouge
301 elseif(frequence>=(x+15) || frequence<=(x-15))
302     color = [1 0.5 0]'; % Orange
303 elseif(frequence>(x+2) || frequence<(x-2))
304     color = [1 1 0]'; % Jaune
305 else
306     color=[0 1 0]; % Vert
307 end
308 bar(frequence,'facecolor', color);
309 xlim([0.6,1.4]);
```

```
310 set(gca,'xtick',[]);
311 xlabel(note);
312 ylim([(x-60),(x+60)]);
313
314 % Dfinition des noms des boutons de notes
315 function setBoutonsNotes(Accord, handles)
316 set(handles.note1,'string',Accord{1,1});
317 set(handles.note2,'string',Accord{1,2});
318 set(handles.note3,'string',Accord{1,3});
319 set(handles.note4,'string',Accord{1,4});
320 set(handles.note5,'string',Accord{1,5});
321 set(handles.note6,'string',Accord{1,6});
322
323 % Accordage d'une note en particulier pendant 5 secondes
324 function accorderNote(position,handles)
325 clc;
326 Accord = handles.Accord;
327 x = Accord{2,position};
328 % Rcupration de variables
329 Fs = handles.Fs;
330 nBits = handles.nBits;
331 nChannel= handles.nChannel;
332 % Enregistrer le son
333 soundRecord = audiorecorder(Fs, nBits, nChannel);
334 record(soundRecord);
335 pause(0.3);
336 compte = 1;
337 % Pendant 5 sec, traiter le son enregistr pour accorder la guitare
338 while (compte < 50)
339     myRecording = getaudiodata(soundRecord);
340     myRecording = xcorr(myRecording);
341     NFFT = Fs;
342     Y = fft(myRecording,NFFT);
343     f = (Fs/2*linspace(0,1,NFFT/2+1));
344     [valeur, frequence]=max(Y(f>0 & f<(x+50)));
345     accordBar(frequence,x, Accord{1,position});
346     pause(0.1);
347     compte = compte+1;
348 end
349
350 % Fonction pour enable et disable les boutons pendant l'accordage d'une
351 % note
352 function boutonOffOn(note, etat, handles)
353 if (note ~= 1) set(handles.note1,'Enable',etat); end
354 if (note ~= 2) set(handles.note2,'Enable',etat); end
355 if (note ~= 3) set(handles.note3,'Enable',etat); end
356 if (note ~= 4) set(handles.note4,'Enable',etat); end
357 if (note ~= 5) set(handles.note5,'Enable',etat); end
358 if (note ~= 6) set(handles.note6,'Enable',etat); end
359 set(handles.popupmenu1,'Enable',etat);
360
361 % Fonctions excuter lors de la slection d'une note
```

```
362 function activerNote(pos, handles)
363 boutonOffOn(pos, 'off', handles);
364 accorderNote(pos,handles);
365 boutonOffOn(pos, 'on', handles);
366 xlabel('');
```