Python : Traitement du signal audio avec Pyo

Présentation

Lien :

- http://www.augmented-instruments.net/_media/pyo_as_python_dsp_toolbox.pdf
- Site Web : http://ajaxsoundstudio.com/software/pyo/
- Documentation : http://ajaxsoundstudio.com/pyodoc/

Pyo est un module Python dédié au **traitement de signal audio** en temps réel.

Sommaire

```
#!/usr/bin/python3
#-*- coding: utf-8 -*-

# Arnaud TECHER <arnaud.techer@laposte.net>
# SCEI 2019 N°43701
# Projet casque antibruit

import wx,pyo,math, time
from thread import *
# classe pour le serveur audio
class Serveur:
   def __init__(self):
      self.serveur=pyo.Server()
      #self.serveur.setOutputDevice(2)
      #self.serveur.setInputDevice(2)
      self.serveur.boot()
      #self.input = pyo.Input()
      self.serveur.amp = 0.4
      #parametre pour la generation du son
      self.freqG = pyo.Sine(freq=300, phase=0, mul=0.1, add=0).out(0)
      self.freqD = pyo.Sine(freq=300, phase=0, mul=0.1, add=0).out(1)
      # parametre pour l'enregistrement
      self.enr = pyo.Input(chnl=0, mul=4.0)
      self.file = "enregistrement_tipe.wav"
      self.serveur.recordOptions(filename=self.file, fileformat=0, sampletype=1)


# creation de la classe de l'application
class Fenetre(wx.Frame):
   # le constructeur de la classe fenetre herite de wx.Frame,
   # il faut appeler le constructeur de wx.Frame : wx.Frame.__init__().
   def __init__(self, parent, id, title, pos, size) :
      wx.Frame.__init__(self, parent, id, title,pos, size)
      self.parent = parent

      # var pour gerre slkider phase D
      self.iSliderD = 0

      self.serveur()
      self.initialise()

   def serveur(self):
      self.audio = Serveur()

   def initialise(self):
      # creation d'un menu
      self.menu()

      # creation de l'interface

      # creation du panel et des box
      self.panel = wx.Panel(self)
```

```
        mainSizer = wx.BoxSizer(wx.VERTICAL)
        boxFrequence = wx.BoxSizer(wx.VERTICAL)
        boxAmplitude = wx.BoxSizer(wx.HORIZONTAL)
        gBoxAmplitude = wx.BoxSizer(wx.VERTICAL)
        dBoxAmplitude = wx.BoxSizer(wx.VERTICAL)
        boxPhase = wx.BoxSizer(wx.HORIZONTAL)
        gBoxPhase = wx.BoxSizer(wx.VERTICAL)
        dBoxPhase = wx.BoxSizer(wx.VERTICAL)
        boxCommande = wx.BoxSizer(wx.HORIZONTAL)

        # self.panel est le parent du widget,
        # wx.ID_ANY pour laisser wxPython choisir un identifiant
        # wx.EXPAND pour agrandir la cellule si la fenetre est agrandie
        # ajout dans les boxSizer
        # Bind pour associer la méthode a exécuter a l'evenement du widget

        # Gestion de la frequence
        # label
        self.labelFreq = wx.StaticText(self.panel,wx.ID_ANY,
                                       label=u'Frequence',
                                       style=wx.ALIGN_CENTRE_HORIZONTAL)
        boxFrequence.Add(self.labelFreq,0, wx.ALL | wx.EXPAND, 5)
        # slider
        self.freq=wx.Slider(self.panel,wx.ID_ANY,value=300,minValue=50,maxValue=1000,
                            pos=(0,0),size=(200,-1),style=wx.SL_LABELS)
        boxFrequence.Add(self.freq,1, wx.ALL | wx.EXPAND, 5)
        mainSizer.Add(boxFrequence,1, wx.ALL | wx.EXPAND, 5)
        self.Bind(wx.EVT_SLIDER, self.changeFreq)

        # Gestion de l'amplitude
        # label gauche
        self.gLabelAmplitude = wx.StaticText(self.panel,wx.ID_ANY,
                                             label=u'Amplitude gauche',style=wx.ALIGN_RIGHT)
        gBoxAmplitude.Add(self.gLabelAmplitude,0, wx.ALL | wx.EXPAND, 5)
        # slider gauche
        self.gAmplitude=wx.Slider(self.panel,wx.ID_ANY,value=1,minValue=0,maxValue=10,
                                  pos=(0,0),size=(200,-1),style=wx.SL_LABELS)
        gBoxAmplitude.Add(self.gAmplitude,1, wx.ALL | wx.EXPAND, 5)
        boxAmplitude.Add(gBoxAmplitude,1, wx.ALL | wx.EXPAND, 5)
        self.Bind(wx.EVT_SLIDER, self.gChangeAmplitude, self.gAmplitude)
        # label droite
        self.dLabelAmplitude = wx.StaticText(self.panel,wx.ID_ANY,
                                             label=u'Amplitude droite',style=wx.ALIGN_RIGHT)
        dBoxAmplitude.Add(self.dLabelAmplitude,0, wx.ALL | wx.EXPAND, 5)
        # slider droite
        self.dAmplitude=wx.Slider(self.panel,wx.ID_ANY,value=1,minValue=0,maxValue=10,
                                  pos=(0,0),size=(200,-1),style=wx.SL_LABELS)
        dBoxAmplitude.Add(self.dAmplitude,1, wx.ALL | wx.EXPAND, 5)
        boxAmplitude.Add(dBoxAmplitude,1, wx.ALL | wx.EXPAND, 5)
        self.Bind(wx.EVT_SLIDER, self.dChangeAmplitude, self.dAmplitude)
        #ajout dans box main
        mainSizer.Add(boxAmplitude,1, wx.ALL | wx.EXPAND, 5)

        # Gestion de la phase gauche
        # label gauche
        self.gLabelPhase = wx.StaticText(self.panel,wx.ID_ANY,
                                         label=u'Phase gauche')
        gBoxPhase.Add(self.gLabelPhase,0, wx.ALL | wx.EXPAND, 5)
        # slider phase G
        self.gPhase=wx.Slider(self.panel,wx.ID_ANY,value=0,minValue=0,maxValue=100,
                              pos=(0,0),size=(200,-1),style=wx.SL_LABELS)
        gBoxPhase.Add(self.gPhase,1, wx.ALL | wx.EXPAND, 5)
        boxPhase.Add(gBoxPhase,1, wx.ALL | wx.EXPAND, 5)
        self.Bind(wx.EVT_SLIDER, self.gChangePhase, self.gPhase)
        # label droite
        self.dLabelPhase = wx.StaticText(self.panel,wx.ID_ANY,label=u'Phase droite')
        dBoxPhase.Add(self.dLabelPhase,0, wx.ALL | wx.EXPAND, 5)
        # Gestion de la phase droite
        self.dPhase=wx.Slider(self.panel,wx.ID_ANY,value=0,minValue=0,maxValue=100,
                              pos=(0,0),size=(200,-1),style=wx.SL_LABELS)
```

```
        dBoxPhase.Add(self.dPhase,0, wx.ALL | wx.EXPAND, 5)
        boxPhase.Add(dBoxPhase,1, wx.ALL | wx.EXPAND, 5)
        self.Bind(wx.EVT_SLIDER, self.dChangePhase, self.dPhase)
        #ajout dans box main
        mainSizer.Add(boxPhase,1, wx.ALL | wx.EXPAND, 5)

        # Gestion graphique du Scope dans une fenetre separee
        self.gscope = pyo.Scope([self.audio.freqG, self.audio.freqD])

        # Gestion des boutons de commande
        #bouton star/stop
        self.boutonStart = wx.Button(self.panel,wx.ID_ANY,label="Start")
        boxCommande.Add(self.boutonStart,1, wx.ALL | wx.EXPAND, 5)
        self.Bind(wx.EVT_BUTTON, self.start, self.boutonStart)

        #bouton enregistrement
        self.boutonEnr = wx.Button(self.panel,wx.ID_ANY,label="Enr")
        boxCommande.Add(self.boutonEnr,1, wx.ALL | wx.EXPAND, 5)
        self.Bind(wx.EVT_BUTTON, self.enregistrement, self.boutonEnr)

        #bouton auto
        self.boutonAuto = wx.Button(self.panel,wx.ID_ANY,label="Auto")
        boxCommande.Add(self.boutonAuto,1, wx.ALL | wx.EXPAND, 5)
        self.Bind(wx.EVT_BUTTON, self.auto, self.boutonAuto)

        # ajout box main
        mainSizer.Add(boxCommande,1, wx.ALL | wx.EXPAND, 5)

        self.panel.SetSizerAndFit(mainSizer)

    def changeFreq(self,event):
        # modifier les frequences gauche et droite
        self.audio.freqG.setFreq(event.GetInt())
        self.audio.freqD.freq=event.GetInt()

    def gChangeAmplitude(self,event):
        # modifier l'attribut mul
        self.audio.freqG.mul=event.GetInt()/10

    def dChangeAmplitude(self,event):
        # modifier l'attribut mul
        self.audio.freqD.mul=event.GetInt()/10

    def gChangePhase(self,event):
        # modifier l'attribut phase
        self.audio.freqG.setPhase(event.GetInt()/100)

    def dChangePhase(self,event):
        # modifier l'attribut phase
        self.audio.freqD.setPhase(event.GetInt()/100)

    def start(self,event):
        # demarer / arreter le serveur audio
        if self.boutonStart.GetLabel() == "Start":
            self.audio.serveur.start()
            self.boutonStart.SetLabel("Stop")
        else:
            self.audio.serveur.stop()
            self.boutonStart.SetLabel("Start")

    def enregistrement(self,event):
        # demarrer / arreter l'enregistrement
        if self.boutonEnr.GetLabel() == "Enr":
            self.audio.serveur.recstart()
            self.boutonEnr.SetLabel("Fin enr")
        else:
            self.audio.serveur.recstop()
            self.boutonEnr.SetLabel("Enr")

    def auto(self,event):
```

```
      # demarrer / arreter la variation automatique de la phase droite
       if self.boutonAuto.GetLabel() == "Auto":
           self.boutonAuto.SetLabel("Auto actif")
           duree = 0.01
           self.t =  Intervallometre(duree,self.augmenter)
           self.t.setDaemon(True)
           self.t.start()
       else:
           self.boutonAuto.SetLabel("Auto")
           self.iSliderD=0
           self.audio.freqD.setPhase(0)
           self.t.stop()

   def augmenter(self):
      # augmenter la phase
       if self.iSliderD < 100:
           self.iSliderD += 1
           self.audio.freqD.setPhase(self.iSliderD/100)
           self.dPhase.SetValue(self.iSliderD)

   def menu(self):
      # generation du menu
      filemenu= wx.Menu()

      # wx.ID_ABOUT et wx.ID_EXIT sont des IDs standards pour les wxWidgets.
      menuApropos = filemenu.Append(wx.ID_ABOUT, "&A propos","Information sur ce programme")
      filemenu.AppendSeparator()
      menuQuitter = filemenu.Append(wx.ID_EXIT,"&Quitter"," Quitter le programme")

      # Creation du menu.
      menuBar = wx.MenuBar()
      menuBar.Append(filemenu,"&Fichier") # Ajout de "filemenu" a la barre de Menu
      self.SetMenuBar(menuBar)  # Ajout de la bare de menu au contenu de la fenetre

      # lier les evenements aux methodes
      self.Bind(wx.EVT_MENU, self.OnApropos, menuApropos)
      self.Bind(wx.EVT_MENU, self.OnQuitter, menuQuitter)

   def OnApropos(self,event):
       # Afficher une boite de dialogue avec un bouton OK. wx.OK est un ID standard des wxWidgets.
       dlg = wx.MessageDialog( self, "Arnaud TECHER - TIPE 2019", "Gestion des haut-parleurs", wx.OK)
       dlg.ShowModal() # afficher la bopite de dialogue par dessus la fenetre
       dlg.Destroy() # detruire la bopite de dialogue quand on clique sur OK ou que l'on, la ferme.

   def OnQuitter(self,event):
       # arreter le moteur audio
       self.serveur.serveur.stop()
       self.Close(True)  # fermer la frenetre.


if __name__ == "__main__":
    app = wx.App()
    fenetre_1 = Fenetre(None,wx.ID_ANY,'Pannel son1', (50,50), (1000,600))
    # faire apparaitre la fenetre
    fenetre_1.Show()
    # boucle infinie qui attend les evenements utilisateur
    app.MainLoop()
```

```
#!/usr/bin/python3
#-*- coding: utf-8 -*-

# Arnaud TECHER <arnaud.techer@laposte.net>
# SCEI 2019 N°43701
# Gestion d'un thread

import threading

class Intervallometre(threading.Thread):

    def __init__(self, duree, fonction, args=[], kwargs={}):
```

```
        threading.Thread.__init__(self)
        self.duree = duree
        self.fonction = fonction
        self.args = args
        self.kwargs = kwargs
        self.encore = True  # pour permettre l'arret a la demande

    def run(self):
        while self.encore:
            self.timer = threading.Timer(self.duree, self.fonction, self.args, self.kwargs)
            self.timer.setDaemon(True)
            self.timer.start()
            self.timer.join()

    def stop(self):
        self.encore = False  # pour empecher un nouveau lancement de Timer et terminer le thread
        if self.timer.isAlive():
            self.timer.cancel()  # pour terminer une eventuelle attente en cours de Timer
```