wxPython : Création des widgets

Utiliser un gestionnaire de layout

Le **positionnement** des widgets dans une fenêtre ou un conteneur) peut se faire de **plusieurs manières**, par ajout horizontal, verticalement, etc.

Il existe donc différentes classes appelées **gestionnaires de layout** (layout managers) capables de placer les widgets dans les conteneurs de différentes manières. Il est possible d'utiliser des gestionnaire de layout différents selon les conteneurs.

Pour cette première application, le gestionnaire grid (grille) sera utilisé. C'est une simple grille où vous positionnez vos widgets dans des cases à la manière d'un tableur (Excel, OpenOffice Calc...). Par exemple :

- Mettre un bouton à la colonne 2, ligne 1,
- Mettre une checkbox colonne 5, ligne 3. etc.

Pour définir le gestionnaire avec wxPython :

- **création** du gestionnaire avec **sizer=wx.GridBagSizer()**
- **utilisation** du gestionnaire par la fenêtre avec **self.SetSizerAndFit(sizer)**.

```
import wx
# création de la classe de l'application
class Fenetre(wx.Frame):
   def __init__(self, parent, id, title, pos, size) :
      super(wx.Frame, self).__init__(parent, id, title,pos, size)
      self.parent = parent
      self.initialise()

   def initialise(self):
      # forcer l'apparition de la fenetre
      grille = wx.GridBagSizer()
      self.SetSizerAndFit(grille)
      self.Show(True)

if __name__ == "__main__":
    app = wx.App()
    fenetre_1 = Fenetre(None,-1,'Première application', (25,25), (400,100))
    app.MainLoop()
```

Ajout d'un champ de saisie texte

Pour ajouter des widgets :

- on crée d'abord le widget : ici création du widget TextCtrl → self.entree=wx.TextCtrl().
- puis on l'ajoute à un conteneur : .

```
   def initialise(self):
      # forcer l'apparition de la fenetre
      grille = wx.GridBagSizer()
      # self est le parent du widget, -1 pour laisser wxPython choisir un identifiant
      self.entree = wx.TextCtrl(self, -1, value="Entrez une valeur.")
      # ajout à la grille en précisant les coordonnées,
      # l'étendue (1,1) pour ne pas déborder suer les cellules voisines
      # wx.EXPAND pour agrandir la cellule si la fenêtre est agrandie
      grille.Add(self.entree,(0,0),(1,1),wx.EXPAND)
      self.SetSizerAndFit(grille)
      self.Show(True)
```

Ajout d'un bouton

Création et ajout du bouton

```
    def initialize(self):
        grille = wx.GridBagSizer()
```

```
        self.entree = wx.TextCtrl(self,-1,value=u"Entrez un texte.")
        grille.Add(self.entry,(0,0),(1,1),wx.EXPAND)

        self.bouton = wx.Button(self,-1,label="Cliquez sur moi !")
        grille.Add(bouton, (0,1))

        self.SetSizerAndFit(sizer)
        self.Show(True)
```

```
#!/usr/bin/python3
# -*- coding: utf8 -*-

import wx,pyo,math
# 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.serveur.amp = 0.4
      #parametre pour la generation du son
      self.freqG = pyo.Sine(freq=100, phase=0, mul=1, add=0).out(0)
      self.freqD = pyo.Sine(freq=100, phase=0, mul=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)


# création de la classe de l'application
class Fenetre(wx.Frame):
   # le constructeur de la classe fenetre hérite 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
      self.serveur()
      self.initialise()

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

   def initialise(self):
      # creation d'un menu
      self.menu()
      grille = wx.GridBagSizer(6,2)
      # self est le parent du widget, wx.ID_ANY pour laisser wxPython choisir un identifiant
      # ajout à la grille en précisant les coordonnées,
      # l'étendue (1,1) pour ne pas déborder sur les cellules voisines
      # wx.EXPAND pour agrandir la cellule si la fenêtre est agrandie
      # grille.Add(self.entree,(0,0),(1,1),wx.EXPAND)

      # Gestion de la frequence
      # label
      self.labelFreq = wx.StaticText(self,wx.ID_ANY,label=u'Fréquence',style=wx.ALIGN_CENTRE_HORIZONTAL)
      grille.Add(self.labelFreq, pos=(0,0), span=(1,2),flag=wx.EXPAND)
      # slider
      self.freq=wx.Slider(self,wx.ID_ANY,value=100,minValue=50,maxValue=1000,
                          pos=(0,0),size=(200,-1),style=wx.SL_LABELS)
      grille.Add(self.freq,pos=(1,0),span=(1,2),flag=wx.EXPAND)
      self.Bind(wx.EVT_SLIDER, self.changeFreq)
      #afficher info
      #self.labelFreq=wx.StaticText(self,wx.ID_ANY,label=u"100")
      #grille.Add(self.labelFreq,(1,1))

      # Gestion de l'amplitude
      # label
      self.labelAmplitude = wx.StaticText(self,wx.ID_ANY,label=u'Amplitude',style=wx.ALIGN_RIGHT)
```

```
        grille.Add(self.labelAmplitude, pos=(2,0), span=(1,2), flag=wx.EXPAND)
        # slider
        self.amplitude=wx.Slider(self,wx.ID_ANY,value=5,minValue=0,maxValue=10,
                                 pos=(0,0),size=(200,-1),style=wx.SL_LABELS)
        grille.Add(self.amplitude, pos=(3,0), span=(1,2),flag=wx.EXPAND)
        self.Bind(wx.EVT_SLIDER, self.changeAmplitude, self.amplitude)
        #afficher apmplitude
        #self.labelAmplitude=wx.StaticText(self,wx.ID_ANY,label=u"Amplitude")
        #grille.Add(self.labelAmplitude,(3,1))

        # Gestion de la phase gauche
        # label
        self.labelPhase = wx.StaticText(self,wx.ID_ANY,label=u'Phase')
        grille.Add(self.labelPhase, (4,0),(1,2), wx.EXPAND )
        # slider phase G
self.phaseG=wx.Slider(self,wx.ID_ANY,value=0,minValue=0,maxValue=100,pos=(0,0),size=(200,-1),style=wx.SL
_LABELS)
        grille.Add(self.phaseG,(5,0),(1,1))
        self.Bind(wx.EVT_SLIDER, self.changePhaseG, self.phaseG)
        # Gestion de la phase droite
self.phaseD=wx.Slider(self,wx.ID_ANY,value=0,minValue=0,maxValue=100,pos=(0,0),size=(200,-1),style=wx.SL
_LABELS)
        grille.Add(self.phaseD,(5,1),(1,1))
        self.Bind(wx.EVT_SLIDER, self.changePhaseD, self.phaseD)

        #bouton star/stop
        self.boutonStart = wx.Button(self,wx.ID_ANY,label="Start")
        grille.Add(self.boutonStart, (6,0))
        self.Bind(wx.EVT_BUTTON, self.start, self.boutonStart)

        #bouton enregistrement
        self.boutonEnr = wx.Button(self,wx.ID_ANY,label="Enr")
        grille.Add(self.boutonEnr, (6,1))
        self.Bind(wx.EVT_BUTTON, self.enregistrement, self.boutonEnr)


        self.SetSizerAndFit(grille)

        # forcer l'apparition de la fenetre
        self.Show(True)

    def changeFreq(self,event):
         self.audio.freqG.setFreq(event.GetInt())
         self.audio.freqD.freq=event.GetInt()
         #self.labelFreq.SetLabel(str(event.GetInt()))

    def changeAmplitude(self,event):
        # modifeir l'attribut mul
        self.audio.freqG.mul=event.GetInt()/10
        self.audio.freqD.mul=event.GetInt()/10

    def changePhaseG(self,event):
         self.audio.freqG.setPhase(event.GetInt()/100)
         #self.labelPhaseG.SetLabel(str(event.GetInt()/10))

    def changePhaseD(self,event):
         self.audio.freqD.setPhase(event.GetInt()/100)
         #self.labelPhaseD.SetLabel(str(event.GetInt()/10))

    def start(self,event):
        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):
        if self.boutonEnr.GetLabel() == "Enr":
```

```
            self.audio.serveur.recstart()
            self.boutonEnr.SetLabel("Fin enr")
        else:
            self.audio.serveur.recstop()
            self.boutonEnr.SetLabel("Enr")


    def menu(self):
        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" à la barre de Menu
        self.SetMenuBar(menuBar)  # Ajout de la bare de menu au contenu de la fenêtre

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

    def OnApropos(self,event):
        # Afficher une boîte 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 fenêtre
        dlg.Destroy() # detruire la bopite de dialogue quand on clique sur OK ou que l'on, la ferme.

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


if __name__ == "__main__":
    app = wx.App()
    fenetre_1 = Fenetre(None,wx.ID_ANY,'Pannel son1', (25,25), (400,100))
    app.MainLoop()
```

Retour au menu Interface graphique avec wxPython

- Interface graphique avec WxPython