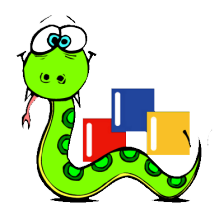
**我用370行代码写了一个wxPython的任务托盘程序：实用的屏幕录像机**

[[](https://blog.csdn.net/xufive/category_9418763.html)wxPython专栏收录该内容](https://blog.csdn.net/xufive/category_9418763.html)

**1. 前言**

最近有同学咨询如何用wx写任务托盘程序，也有同学咨询怎样创建wx的异形窗口。恰好，我也正需要一个可以将屏幕显示或者操作录制成gif文件的工具。于是乎，结合同学们的问题，我用wx写了一个屏幕录像机代码，既包含任务托盘的实现，也用到了异形窗口，还使用了DC绘制录像区域边框。这段代码，可以很方便地打包成exe程序。程序启动后，栖身于任务托盘。你需要的时候，可以随时召唤它。录像区域可以调整大小，生成gif的参数也可以调整，此外还提供了启动/停止的热键（Ctr + F2）操作，使用起来非常方便。

**2. 设计思路**

程序启动后，创建一个全屏的异形窗口，除了10个像素宽的录像区域边框外，其余部分全部透明。全屏窗口位于最顶层，因为录像区域边框外其他区域透明，所以不会影响我们操作其他窗口。当鼠标进入录像区域边框时，可以拖动边框以改变录像区域的大小。启动录像后，使用pillow的ImageGrab定时捕捉录像区域内的内容，保存在一个列表中；停止录像后，使用imageio模块的mimsave()函数，将保存在列表中的PIL图像序列转存为gif文件。

**3. 源码**

代码比较简单，我在关键位置都有注释，就不再具体分析了，直接贴出源码。运行代码需要一个图标文件，保存在和脚本文件同级的res目录下。请自备图标文件，或者去GitHub上下载，地址在文末。

# -\*- coding:utf-8 -\*-

import os

import wx

import wx.adv

import wx.lib.filebrowsebutton as filebrowse

from win32con import MOD\_CONTROL, VK\_F2

from threading import Thread

from datetime import datetime

from configparser import ConfigParser

from PIL import ImageGrab

from imageio import mimsave

class MainFrame(wx.Frame):

"""屏幕录像机主窗口"""

MENU\_REC = wx.NewIdRef() # 开始/停止录制

MENU\_SHOW = wx.NewIdRef() # 显示窗口

MENU\_HIDE = wx.NewIdRef() # 窗口最小化

MENU\_STOP = wx.NewIdRef() # 停止录制

MENU\_CONFIG = wx.NewIdRef() # 设置

MENU\_FOLFER = wx.NewIdRef() # 打开输出目录

MENU\_EXIT = wx.NewIdRef() # 退出

def \_\_init\_\_(self, parent):

wx.Frame.\_\_init\_\_(self, parent, -1, "", style=wx.FRAME\_SHAPED|wx.FRAME\_NO\_TASKBAR|wx.STAY\_ON\_TOP)

x, y, w, h = wx.ClientDisplayRect() # 屏幕显示区域

x0, y0 = (w-820)//2, (h-620)//2 # 录像窗口位置（默认大小820x620，边框10像素）

self.SetPosition((0, 0)) # 无标题窗口最大化：设置位置

self.SetSize((w, h)) # 无标题窗口最大化：设置大小

self.SetDoubleBuffered(True) # 启用双缓冲

self.taskBar = wx.adv.TaskBarIcon() # 添加系统托盘

self.taskBar.SetIcon(wx.Icon(os.path.join("res", "recorder.ico"), wx.BITMAP\_TYPE\_ICO), "屏幕录像机")

self.box = [x0, y0, 820, 620] # 屏幕录像窗口大小

self.xy = None # 鼠标左键按下的位置

self.recording = False # 正在录制标志

self.saveing = False # 正在生成GIF标志

self.imgs = list() # 每帧的图片列表

self.timer = wx.Timer(self) # 创建录屏定时器

self.cfg = self.ReadConfig() # 读取配置项

self.SetWindowShape() # 设置不规则窗口

self.Bind(wx.EVT\_MOUSE\_EVENTS, self.OnMouse) # 鼠标事件

self.Bind(wx.EVT\_PAINT, self.OnPaint) # 窗口重绘

self.Bind(wx.EVT\_ERASE\_BACKGROUND, self.OnEraseBG) # 擦除背景

self.Bind(wx.EVT\_TIMER, self.OnTimer, self.timer) # 定时器

self.taskBar.Bind(wx.adv.EVT\_TASKBAR\_RIGHT\_UP, self.OnTaskBar) # 右键单击托盘图标

self.taskBar.Bind(wx.adv.EVT\_TASKBAR\_LEFT\_UP, self.OnTaskBar) # 左键单击托盘图标

self.taskBar.Bind(wx.adv.EVT\_TASKBAR\_LEFT\_DCLICK, self.OnTaskBar) # 左键双击托盘图标

self.taskBar.Bind(wx.EVT\_MENU, self.OnRec, id=self.MENU\_REC) # 开始/停止录制

self.taskBar.Bind(wx.EVT\_MENU, self.OnShow, id=self.MENU\_SHOW) # 显示窗口

self.taskBar.Bind(wx.EVT\_MENU, self.OnHide, id=self.MENU\_HIDE) # 隐藏窗口

self.taskBar.Bind(wx.EVT\_MENU, self.OnOpenFolder, id=self.MENU\_FOLFER) # 打开输出目录

self.taskBar.Bind(wx.EVT\_MENU, self.OnConfig, id=self.MENU\_CONFIG) # 设置

self.taskBar.Bind(wx.EVT\_MENU, self.OnExit, id=self.MENU\_EXIT) # 退出

self.RegisterHotKey(self.MENU\_REC, MOD\_CONTROL, VK\_F2) # 注册热键

self.Bind(wx.EVT\_HOTKEY, self.OnRec, id=self.MENU\_REC) # 开始/停止录制热键

def ReadConfig(self):

"""读取配置文件"""

config = ConfigParser()

if os.path.isfile("recorder.ini"):

config.read("recorder.ini")

else:

out\_path = os.path.join(os.path.split(os.path.realpath(\_\_file\_\_))[0], 'out')

if not os.path.exists(out\_path):

os.mkdir(out\_path)

config.read\_dict({"recoder":{"fps":10, "frames":100, "loop":0, "outdir":out\_path}})

config.write(open("recorder.ini", "w"))

return config

def SetWindowShape(self):

"""设置窗口形状"""

path = wx.GraphicsRenderer.GetDefaultRenderer().CreatePath()

path.AddRectangle(self.box[0], self.box[1], self.box[2], 10)

path.AddRectangle(self.box[0], self.box[1]+self.box[3]-10, self.box[2], 10)

path.AddRectangle(self.box[0], self.box[1]+10, 10, self.box[3]-2\*10)

path.AddRectangle(self.box[0]+self.box[2]-10, self.box[1]+10, 10, self.box[3]-2\*10)

self.SetShape(path) # 设置异形窗口形状

def OnMouse(self, evt):

"""鼠标事件"""

if evt.EventType == wx.EVT\_LEFT\_DOWN.evtType[0]: # 左键按下

if self.box[0]+10 <= evt.x <= self.box[0]+self.box[2]-10 and self.box[1]+10 <= evt.y <= self.box[1]+self.box[3]-10:

self.xy = None

else:

self.xy = (evt.x, evt.y)

elif evt.EventType == wx.EVT\_LEFT\_UP.evtType[0]: # 左键弹起

self.xy = None

elif evt.EventType == wx.EVT\_MOTION.evtType[0]: # 鼠标移动

if self.box[0] < evt.x < self.box[0]+10:

if evt.LeftIsDown() and self.xy:

dx, dy = evt.x-self.xy[0], evt.y-self.xy[1]

self.box[0] += dx

self.box[2] -= dx

if self.box[1] < evt.y < self.box[1]+10: # 左上角

self.SetCursor(wx.Cursor(wx.CURSOR\_SIZENWSE))

if evt.LeftIsDown() and self.xy:

self.box[1] += dy

self.box[3] -= dy

elif evt.y > self.box[1]+self.box[3]-10: # 左下角

self.SetCursor(wx.Cursor(wx.CURSOR\_SIZENESW))

if evt.LeftIsDown() and self.xy:

self.box[3] += dy

else: # 左边

self.SetCursor(wx.Cursor(wx.CURSOR\_SIZEWE))

elif self.box[0]+self.box[2]-10 < evt.x < self.box[0]+self.box[2]:

if evt.LeftIsDown() and self.xy:

dx, dy = evt.x-self.xy[0], evt.y-self.xy[1]

self.box[2] += dx

if self.box[1] < evt.y < self.box[1]+10: # 右上角

self.SetCursor(wx.Cursor(wx.CURSOR\_SIZENESW))

if evt.LeftIsDown() and self.xy:

self.box[1] += dy

self.box[3] -= dy

elif evt.y > self.box[1]+self.box[3]-10: # 右下角

self.SetCursor(wx.Cursor(wx.CURSOR\_SIZENWSE))

if evt.LeftIsDown() and self.xy:

self.box[3] += dy

else: # 右边

self.SetCursor(wx.Cursor(wx.CURSOR\_SIZEWE))

elif self.box[1] < evt.y < self.box[1]+10: # 上边

self.SetCursor(wx.Cursor(wx.CURSOR\_SIZENS))

if evt.LeftIsDown() and self.xy:

dx, dy = evt.x-self.xy[0], evt.y-self.xy[1]

self.box[1] += dy

self.box[3] -= dy

elif self.box[1]+self.box[3]-10 < evt.y < self.box[1]+self.box[3]: #下边

self.SetCursor(wx.Cursor(wx.CURSOR\_SIZENS))

if evt.LeftIsDown() and self.xy:

dx, dy = evt.x-self.xy[0], evt.y-self.xy[1]

self.box[3] += dy

if self.box[0] < 0:

self.box[2] += self.box[0]

self.box[0] = 0

if self.box[1] < 0:

self.box[3] += self.box[1]

self.box[1] = 0

w, h = self.GetSize()

if self.box[2] > w:

self.box[2] = w

if self.box[3] > h:

self.box[3] = h

self.xy = (evt.x, evt.y)

self.isFullScreen = self.GetSize() == (self.box[2],self.box[3])

self.SetWindowShape()

self.Refresh()

def OnPaint(self, evt):

"""窗口重绘事件处理"""

dc = wx.PaintDC(self)

dc.SetBrush(wx.RED\_BRUSH if self.recording else wx.GREEN\_BRUSH)

w, h = self.GetSize()

dc.DrawRectangle(\*self.box,)

def OnEraseBG(self, evt):

"""擦除背景事件处理"""

pass

def OnTaskBar(self, evt):

"""托盘图标操作事件处理"""

menu = wx.Menu()

menu.Append(self.MENU\_REC, "开始/停止(Ctrl+F2)")

menu.AppendSeparator()

if self.IsIconized():

menu.Append(self.MENU\_SHOW, "显示屏幕录像窗口")

else:

menu.Append(self.MENU\_HIDE, "最小化至任务托盘")

menu.AppendSeparator()

menu.Append(self.MENU\_FOLFER, "打开输出目录")

menu.Append(self.MENU\_CONFIG, "设置录像参数")

menu.AppendSeparator()

menu.Append(self.MENU\_EXIT, "退出")

if self.recording:

menu.Enable(self.MENU\_CONFIG, False)

menu.Enable(self.MENU\_EXIT, False)

else:

menu.Enable(self.MENU\_CONFIG, True)

menu.Enable(self.MENU\_EXIT, True)

self.taskBar.PopupMenu(menu)

menu.Destroy()

def OnShow(self, evt):

"""显示窗口"""

self.Iconize(False)

def OnHide(self, evt):

"""隐藏窗口"""

self.Iconize(True)

def OnRec(self, evt):

"""开始/停止录制菜单事件处理"""

if self.recording: # 停止录制

self.StopRec()

else: # 开始录制

self.StartRec()

def StartRec(self):

"""开始录制"""

self.OnShow(None)

self.recording = True

self.timer.Start(1000/self.cfg.getint("recoder", "fps")) # 启动定时器

self.Refresh() # 刷新窗口

def StopRec(self):

"""停止录制"""

self.timer.Stop() # 停止定时器

self.recording = False

self.OnHide(None)

# 启动生成GIF线程

t = Thread(target=self.CreateGif)

t.setDaemon(True)

t.start()

# 弹出模态的等待对话窗

count, count\_max = 0, 100

style = wx.PD\_APP\_MODAL | wx.PD\_ELAPSED\_TIME | wx.PD\_ESTIMATED\_TIME | wx.PD\_REMAINING\_TIME | wx.PD\_AUTO\_HIDE

dlg = wx.ProgressDialog("生成GIF", "共计%d帧，正在渲染，请稍候..."%len(self.imgs), parent=self, style=style)

while self.saveing and count < count\_max:

dlg.Pulse()

wx.MilliSleep(100)

dlg.Destroy() # 关闭等待生成GIF结束的对话窗

self.OnOpenFolder(None) # 打开动画文件保存路径

def OnOpenFolder(self, evt):

"""打开输出目录"""

outdir = self.cfg.get("recoder", "outdir")

os.system("explorer %s" % outdir)

def OnConfig(self, evt):

"""设置菜单事件处理"""

dlg = ConfigDlg(self,

self.cfg.getint("recoder", "fps"),

self.cfg.getint("recoder", "frames"),

self.cfg.getint("recoder", "loop"),

self.cfg.get("recoder", "outdir")

)

if dlg.ShowModal() == wx.ID\_OK:

self.cfg.set("recoder", "fps", str(dlg.fps.GetValue()))

self.cfg.set("recoder", "frames", str(dlg.frames.GetValue()))

self.cfg.set("recoder", "loop", str(dlg.loop.GetValue()))

self.cfg.set("recoder", "outdir", dlg.GetOutDir())

self.cfg.write(open("recorder.ini", "w"))

dlg.Destroy() # 销毁设置对话框

def OnExit(self, evt):

"""退出菜单事件处理"""

self.taskBar.RemoveIcon() # 从托盘删除图标

self.Destroy()

wx.Exit()

def OnTimer(self, evt):

"""定时器事件处理：截图"""

img = ImageGrab.grab((self.box[0]+10, self.box[1]+10, self.box[0]+self.box[2]-10, self.box[1]+self.box[3]-10))

self.imgs.append(img)

if len(self.imgs) >= self.cfg.getint("recoder", "frames"):

self.StopRec()

def CreateGif(self):

"""生成gif动画线程"""

self.saveing = True # 生成gif动画开始

dt = datetime.now().strftime("%Y%m%d%H%M%S")

filePath = os.path.join(self.cfg.get("recoder", "outdir"), "%s.gif"%dt)

fps = self.cfg.getint("recoder", "fps")

loop = self.cfg.getint("recoder", "loop")

mimsave(filePath, self.imgs, format='GIF', fps=fps, loop=loop)

self.imgs = list() # 清空截屏记录

self.saveing = False # 生成gif动画结束

class ConfigDlg(wx.Dialog):

"""录像参数设置窗口"""

def \_\_init\_\_(self, parent, fps, frames, loop, outdir):

"""ConfigDlg的构造函数"""

wx.Dialog.\_\_init\_\_(self, parent, -1, "设置录像参数", size=(400, 270))

sizer = wx.BoxSizer() # 创建布局管理器

grid = wx.GridBagSizer(10, 10)

subgrid = wx.GridBagSizer(10, 10)

text = wx.StaticText(self, -1, "帧率：")

grid.Add(text, (0, 0), flag=wx.ALIGN\_RIGHT|wx.TOP, border=3)

self.fps = wx.SpinCtrl(self, -1, size=(80,-1))

self.fps.SetValue(fps)

grid.Add(self.fps, (0, 1), flag=wx.LEFT, border=8)

text = wx.StaticText(self, -1, "最大帧数")

grid.Add(text, (1, 0), flag=wx.ALIGN\_RIGHT|wx.TOP, border=3)

self.frames = wx.SpinCtrl(self, -1, size=(80,-1))

self.frames.SetValue(frames)

grid.Add(self.frames, (1, 1), flag=wx.LEFT, border=8)

text = wx.StaticText(self, -1, "循环次数")

grid.Add(text, (2, 0), flag=wx.ALIGN\_RIGHT|wx.TOP, border=3)

self.loop = wx.SpinCtrl(self, -1, size=(80,-1))

self.loop.SetValue(loop)

grid.Add(self.loop, (2, 1), flag=wx.LEFT, border=8)

text = wx.StaticText(self, -1, "输出目录")

grid.Add(text, (3, 0), flag=wx.TOP, border=8)

self.outdir = filebrowse.DirBrowseButton(self, -1, labelText="", startDirectory=outdir, buttonText="浏览", toolTip="请选择输出路径")

self.outdir.SetValue(outdir)

grid.Add(self.outdir, (3, 1), flag=wx.EXPAND, border=0)

okBtn = wx.Button(self, wx.ID\_OK, "确定")

subgrid.Add(okBtn, (0, 0), flag=wx.ALIGN\_RIGHT)

canelBtn = wx.Button(self, wx.ID\_CANCEL, "取消")

subgrid.Add(canelBtn, (0, 1))

grid.Add(subgrid, (4, 0), (1, 2), flag=wx.ALIGN\_CENTER|wx.TOP, border=10)

grid.AddGrowableCol(1)

sizer.Add(grid, 1, wx.EXPAND|wx.ALL, 20)

self.SetSizer(sizer)

self.Layout()

self.CenterOnScreen()

class MainApp(wx.App):

def OnInit(self):

self.SetAppName("Hello World")

self.frame = MainFrame(None)

self.frame.Show()

return True

if \_\_name\_\_ == '\_\_main\_\_':

app = MainApp()

app.MainLoop()

**4. 打包**

**4.1 打包成一个目录**

假定当前路径为脚本文件所在路径，图标文件已经保存当前路径下的res文件夹中。在当前路径下运行下面这个命令，即可生成一个dist文件夹，里面的ScreenGIF文件夹就是可以用来分发的屏幕录像机项目。

pyinstaller -D ScreenGIF.py -i res\recorder.ico -w --add-data “res;res”

**4.2 打包成一个文件**

要将代码打包成一个可执行文件，需要将图标等资源文件写到代码中。我已将将代码传至[GitHub](https://github.com/xufive/ScreenGIF)，感兴趣的同学，请自行下载。不过，打包成一个文件，启动的时候会非常慢，你得有足够的耐心才能接受。