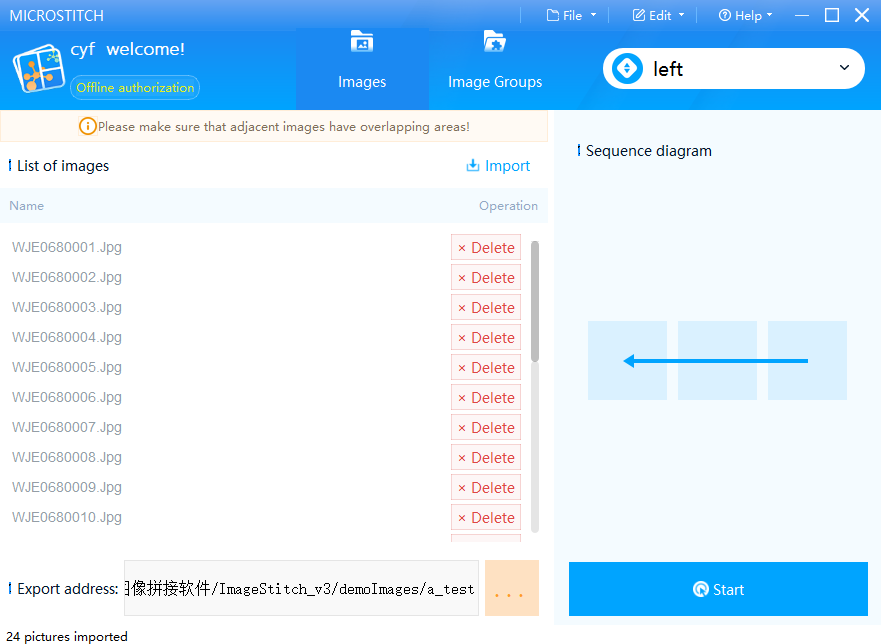
**pyqt软件开发之UI美化**

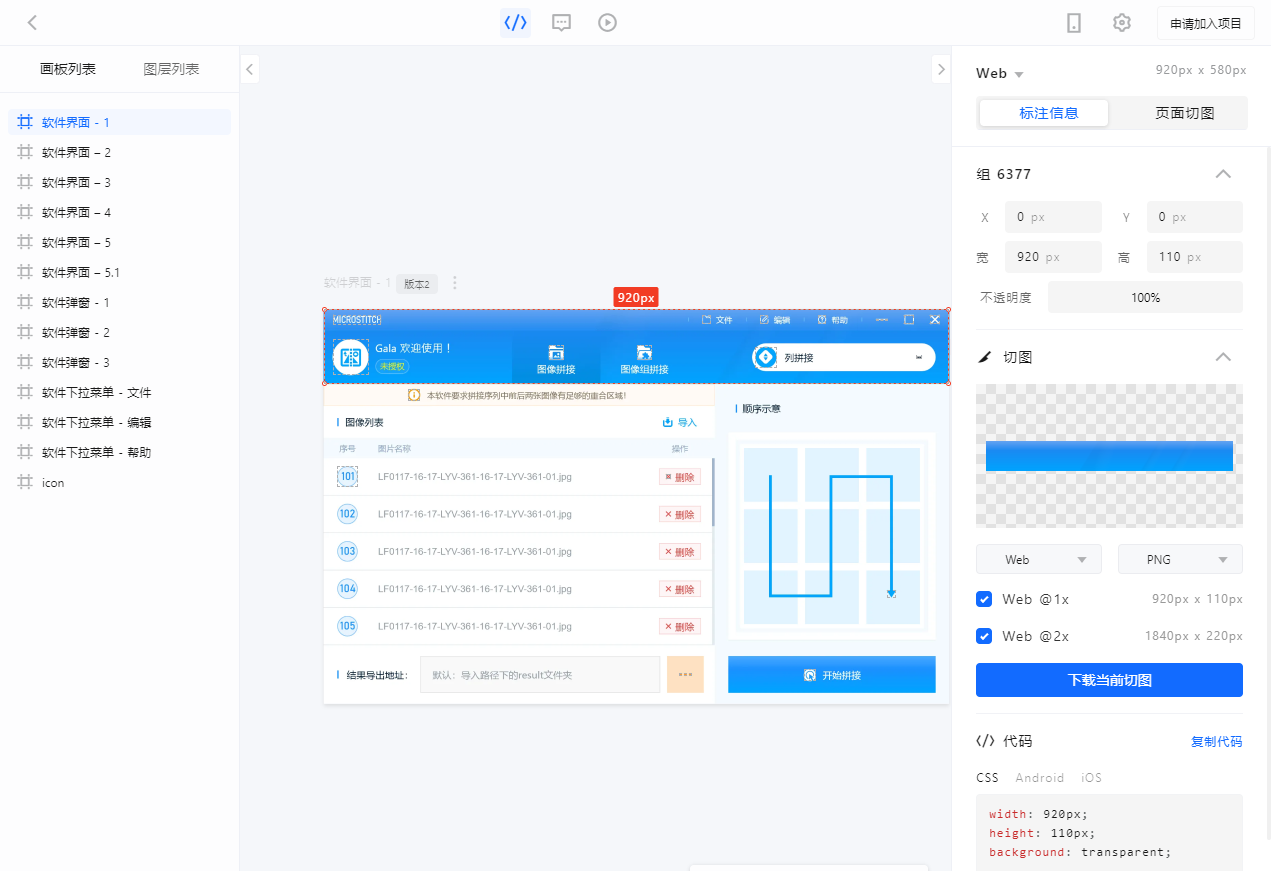
author：陈湧锋

**一、效果示例**

→

**二、工具**

1、设计图（腾讯CoDesign）



2、Qt Designer

注：使用Qt Designer比较直观看到效果，也可以直接通过代码设置样式

生成的ui文件需要使用PyUIC转换为py文件

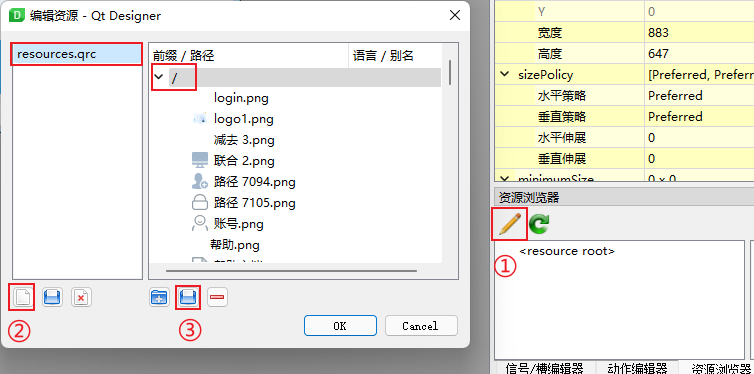


3、部分控件和样式使用代码实现

**三、实战**

1、添加资源文件

①图标等资源文件统一放置在一个文件夹下，在QtDesigner中创建qrc文件，并将图片添加到资源中。可按照类型创建不同的前缀，否则，只创建/前缀，所有文件添加到此路径下



2、设计界面与QSS样式

注：界面可设计为**可伸缩类型**，也可设为**不可伸缩类型**

**推荐设计为可伸缩类型！！！（善用布局）**

①创建窗口

窗口类型主要有两类：QMainWidget、QWidget

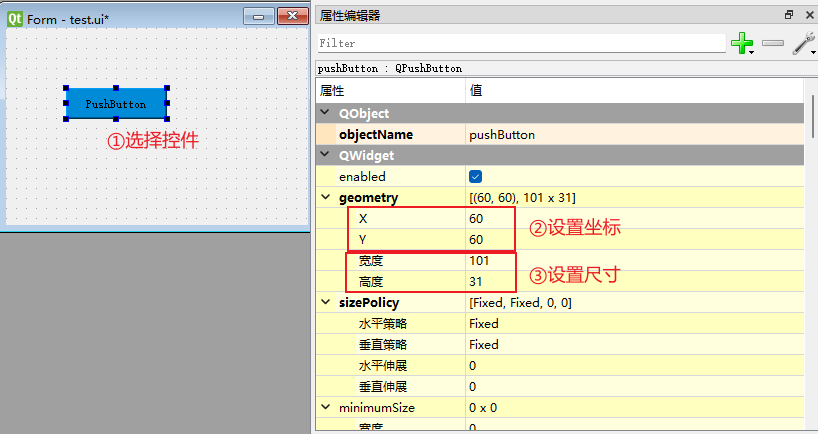
QMainWidget：带菜单栏、状态栏；

QWidget：只是空窗口

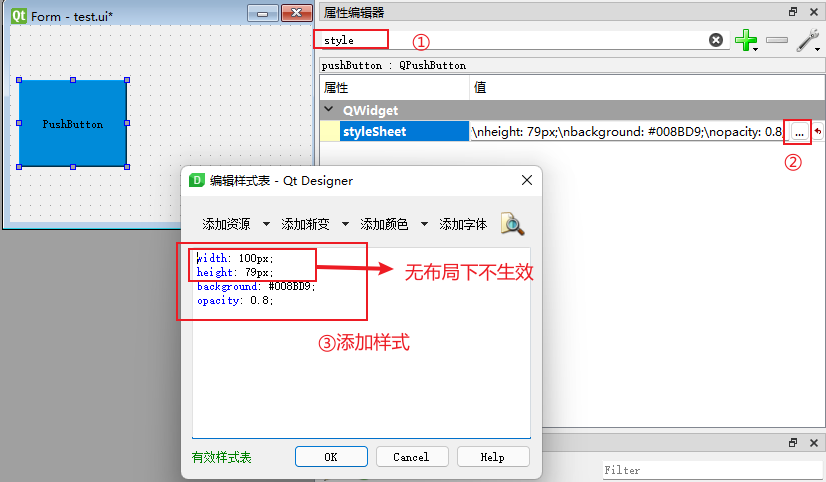
②拖拽控件到窗口中

A）不可伸缩窗口设计模式：

a、添加控件：直接将控件拖拽到指定的位置，选定控件后在属性窗口设置控件的坐标xy、尺寸



b、设置样式：设置颜色等style（此模式下qss的尺寸设置不生效，只能采用上方的方法设置尺寸）

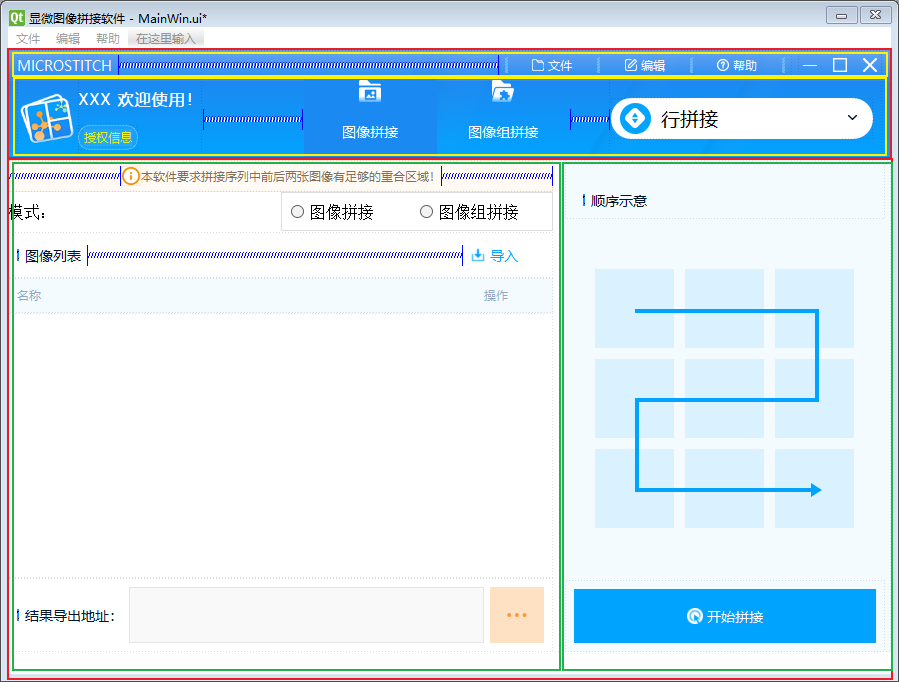


注：CSS的渐变颜色语法在QSS中不支持，若需要渐变需要渐变效果，需要安装qss的语法设置

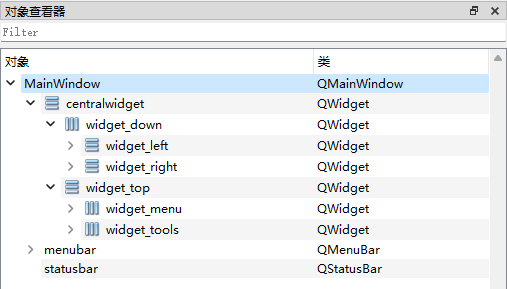
B）可伸缩窗口设计模式：

a、添加控件：添加根据控件的位置关系将整个窗口划分成几个模块，模块中多个控件通过布局控制大小和位置对齐，小模块嵌套在widget中。

添加顺序：从大到小（从外到内）依次放置Widget和控件



嵌套关系：



设置布局：从小到大（从外到内）依次设置布局



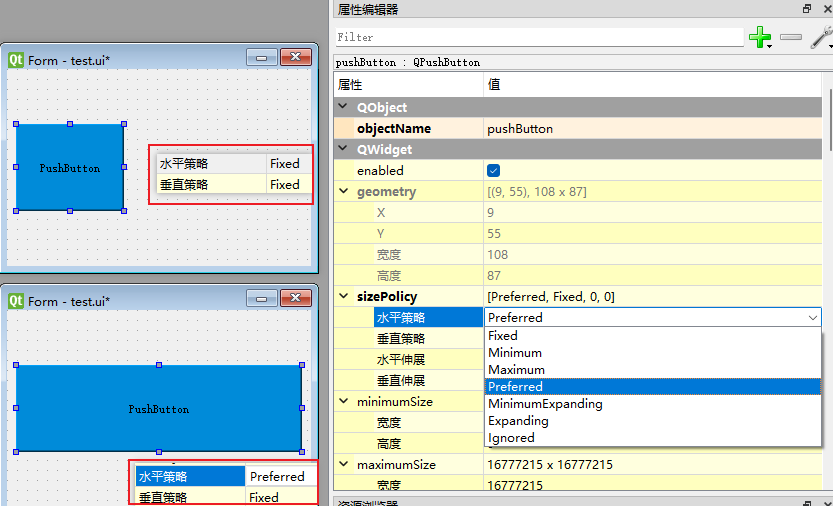
比较常使用的是**水平布局和垂直布局**（设置后按钮变灰色，倒数第二个为打破布局）

b、设置样式（尺寸、颜色等）

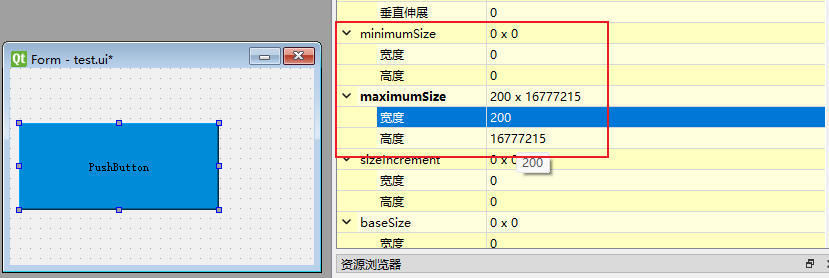
☆尺寸设置：

策略：fixed：自适应（自适应内容尺寸）、其他策略（伸缩尺寸）

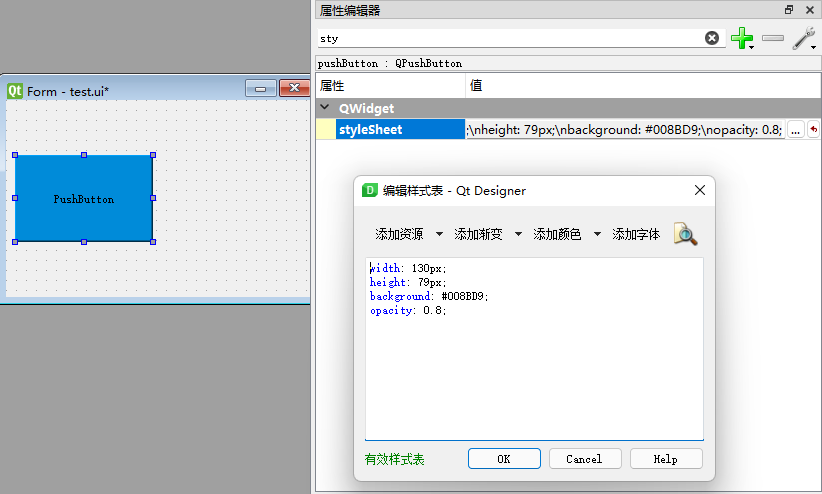
qss的尺寸设置仅在fixed、preferred、minimum、maximum策略下生效（大小固定），其他策略下尺寸会随窗口伸缩而改变



伸缩尺寸下需要改变控件的大小（最大尺寸/最小尺寸），需要通过属性修改（设置最大尺寸也会使得qss的尺寸属性失效！）



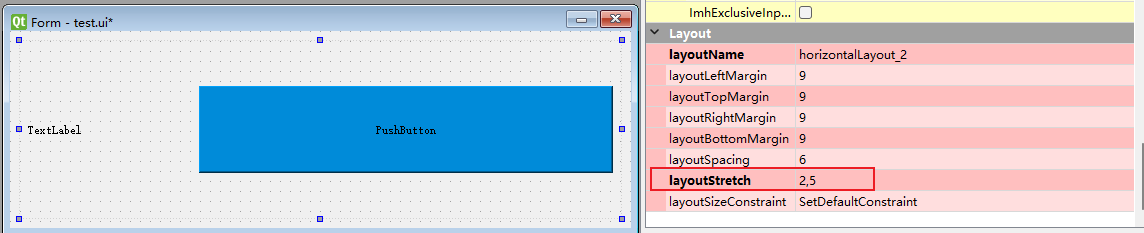
☆颜色、边框等其他样式：在stylesheet属性中设置



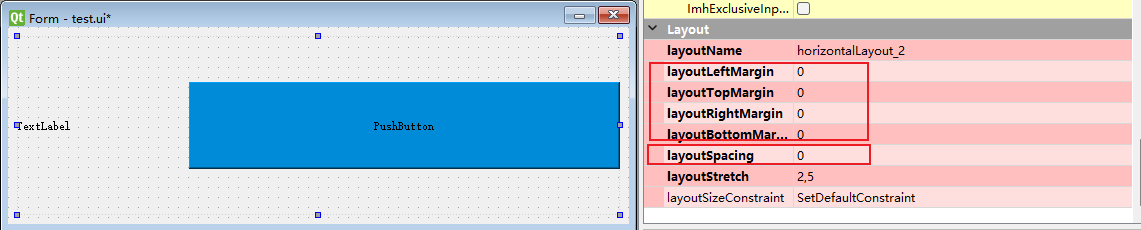
c、伸缩模式下设置控件尺寸比例：

设置伸缩策略：preferred

设置layoutStretch属性



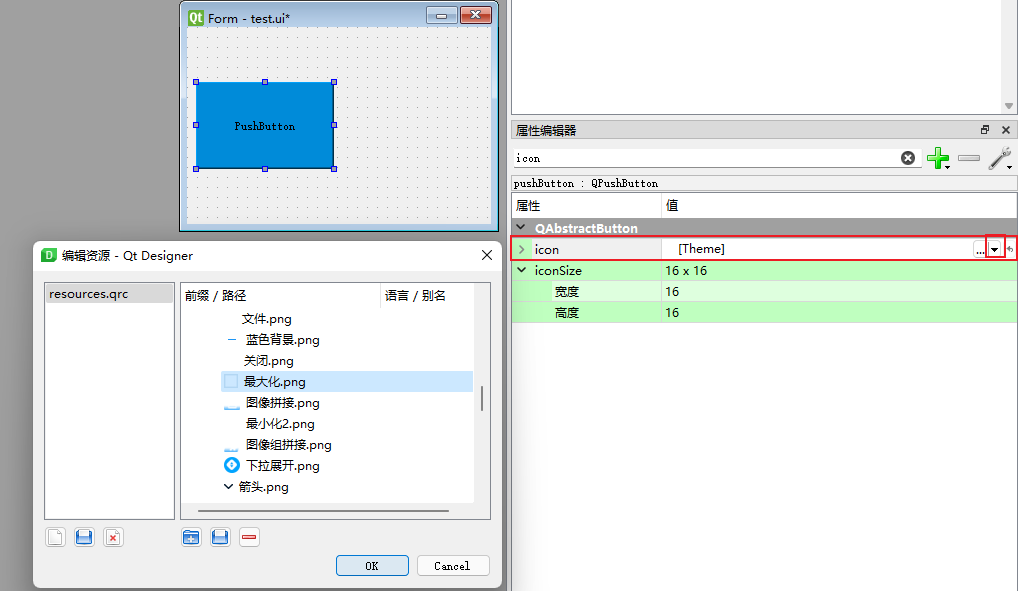
d、设置控件外边距为0，控件间的边距为0



3、设置图标

按钮：icon属性

标签：pixel属性



4、qss基本语法

注：多次执行setStyleSheet时，仅最后一次执行的效果生效！需要多个样式同时生效需要将样式字符串组合

①使用QSS选择器设置控件样式

qssStyle = '''

QPushButton[name='btn1']{

background-color:yellow;

}

'''

self.setStyleSheet(qssStyle) # 设置全局样式

②通过id设置指定控件样式

combobox=QComboBox(self)

combobox.setObjectName('myComboBox') # 设置对象名，相当于id

qssStyle = '''

QComboBox#myComboBox::drop-down{

image:url(下拉.png)

}

'''

self.setStyleSheet(qssStyle)

③设置子控件样式

注：多级子控件：QScrollBar::handle:vertical:hover、QMenu::item:selected…

qssStyle = '''

QComboBox#myComboBox::drop-down{

image:url(下拉.png)

}

'''

self.setStyleSheet(qssStyle)

5、窗口样式

# 关闭、最大化、最小化按钮

self.setWindowFlags(Qt.WindowCloseButtonHint | Qt.WindowMaximizeButtonHint|Qt.WindowMinimizeButtonHint)

# 隐藏标题栏、菜单栏、默认动画等

self.setWindowFlags(Qt.FramelessWindowHint)

# 窗口置顶

self.setWindowFlags(Qt.WindowMaximizeButtonHint)

6、代码设置控件

①设置样式

注：

* + 建议单独将所有样式放在一个py文件中，方便复用，也避免逻辑代码文件过于冗杂。
  + 控件样式遵从继承关系
  + 多次执行仅最后一次生效

a、全局设置

self.setStyleSheet(style)

b、控件设置

self.ui.btn.setStyleSheet(style)

②为标签、按钮添加背景图

from PyQt5.QtWidgets import \*

import sys

class LabelButtonBackground(QWidget):

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

super(LabelButtonBackground, self).\_\_init\_\_()

label1=QLabel(self)

label1.setToolTip('这是一个文本标签')

label1.setStyleSheet("QLabel{border-image:url('背景.jpg');}") # QLabel设置背景图

label1.setFixedWidth(476)

label1.setFixedHeight(259)

btn1 = QPushButton(self)

btn1.setObjectName('btn1')

btn1.setFixedWidth(200)

btn1.setFixedHeight(200)

style = '''

#btn1{

background-image:url('添加.png');

}

#btn1:Pressed{

background-image:url('已添加.png');

}

'''

btn1.setStyleSheet(style) # 为按钮添加背景图

lay = QVBoxLayout()

lay.addWidget(label1)

lay.addStretch()

lay.addWidget(btn1)

self.setLayout(lay)

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

app = QApplication(sys.argv)

win = LabelButtonBackground()

win.show()

sys.exit(app.exec\_())

③设置图片大小随控件缩放

from PyQt5.QtWidgets import \*

from PyQt5.QtGui import QImage, QPixmap

from PyQt5.QtCore import Qt

import sys

class LabelButtonBackground(QWidget):

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

super(LabelButtonBackground, self).\_\_init\_\_()

label1 = QLabel(self)

label1.setToolTip('这是一个文本标签')

label1.setFixedWidth(200)

label1.setFixedHeight(200)

filename = '添加.png'

img = QImage(filename)

# img\_scaled = img.scaled(label1.width(), label1.height(), Qt.IgnoreAspectRatio, Qt.SmoothTransformation)

img\_scaled = img.scaled(label1.width(), label1.height(), Qt.KeepAspectRatio, Qt.SmoothTransformation) # 保持长宽比例

label1.setPixmap(QPixmap.fromImage(img\_scaled))

lay = QVBoxLayout()

lay.addWidget(label1)

lay.addStretch()

self.setLayout(lay)

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

app = QApplication(sys.argv)

win = LabelButtonBackground()

win.show()

sys.exit(app.exec\_())

④窗口透明效果

self.setWindowOpacity(0.1)