照片墙

照片墙的实现较为简单，将图片放置在窗口上，排列成特别的形状。

实现

由于图片一共有259张，比较多，原图是180x260的，因此需要对尺寸进行处理。缩小后就可以进行放置操作了。
需要用的模块如下：

```
from PyQt5.QtWidgets import QWidget, QApplication, QLabel, QGridLayout
from PyQt5.QtGui import QPixmap
from PIL import Image
import os.path
import sys
```

函数说明

本程序写成一个类，该类继承了QWidget。因此初始化为

```
def __init__(self):
    super().__init__()
    self.filepath = r"./传播智客/"    # 旧路径
    self.outpath = r"./传播智客小图/"  # 新路径

    self.setWindowTitle("照片墙")
    self.pre_process()   # 预处理图片，缩小尺寸，执行一次即可
    self.load_photo()      # 载入图片
    self.lay_photo()       # 放置图片
    self.show()            # 显示窗口
```

该程序包含了图片的预处理，即尺寸缩小。

pre_process(self)

该函数主要用来将原图缩小，并将新图片存储起来以备后面使用。

```
for each in self.photo_list:
    img = Image.open(each)
    img = img.resize((25, 36), Image.BILINEAR)
    img.save(os.path.join(self.outpath, os.path.basename(each).split('\\')
[-1]))
```

load_photo(self)

载入图片真正要做的就是收集图片名，这里用了一个列表当容器，方便后续的处理。

```
def load_photo(self):
    """
    载入图片
    图片名存储到新的列表中: self.newPhotoList
    """
    self.newPhotoList = os.listdir(self.outpath)
    self.newPhotoList = [os.path.join(self.outpath, file) for file in
self.newPhotoList]  # 完整路径
```

接下来就是最关键的放置部分。

lay_photo(self)

考虑到一共259张图片，将**259**分解因式可得**259 = 37 x 7**。 那么最简单的方法就是放置7行，每行37张。
使用栅格布局。

```
def lay_photo(self):
    gLayout = QGridLayout()
    for each in self.newPhotoList:
        picture = QLabel(self)
        picture.setPixmap(QPixmap(each))
        gLayout.addWidget(picture, self.newPhotoList.index(each) // 37,
self.newPhotoList.index(each) % 37)
        # 这里地板除是为了得到行数，取余是为了得到列数
    self.setLayout(gLayout)
```

记为布局一。 设想将图片排列成一棵树的形状。由于图片很多，考虑等差数列。

$$sum = n \cdot a + \frac{n \cdot (n-1) \cdot d}{2}$$

为了简洁，令$n = 7$，那么可以得到

$$259 = 7a + 21d$$

因为259有7这个因子，所以

$$37 = a + 3d$$

令$a = 13, d = 8$，则满足此式。考虑到a较大，则再拆分成两个较小的数，看上去应该会美观一些。 则需要将上面的 `gLayout.addWidget(picture, self.newPhotoList.index(each) // 37, self.newPhotoList.index(each) % 37)` 这一句更改为

```
if i < 5:
    gLayout.addWidget(picture, 0, self.newPhotoList.index(each) % 5 + 24)
elif 5 <= i < 18:
    gLayout.addWidget(picture, 1, self.newPhotoList.index(each) % 13 + 20)
elif 18 <= i < 39:
    gLayout.addWidget(picture, 2, self.newPhotoList.index(each) % 21 + 16)
elif 39 <= i < 68:
    gLayout.addWidget(picture, 3, self.newPhotoList.index(each) % 29 + 13)
elif 68 <= i < 105:
    gLayout.addWidget(picture, 4, self.newPhotoList.index(each) % 37 + 10)
elif 105 <= i < 150:
    gLayout.addWidget(picture, 5, self.newPhotoList.index(each) % 45 + 5)
elif 150 <= i < 203:
    gLayout.addWidget(picture, 6, self.newPhotoList.index(each) % 53 + 1)
elif 203 <= i < 259:
    gLayout.addWidget(picture, 7, self.newPhotoList.index(each) % 56)
```

记为布局二。
没想到更好的处理方法，只能暴力一点了。

实例化

```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = PhotoWall()
    app.exec_()
```

结果展示

布局一

布局二