进程管理：电梯调度模拟——说明文档

> 题目：电梯调度
>
> 姓名：孟宇
>
> 学号：1951477
>
> 指导教师：张慧娟

一、 项目简介

本项目设计的是一个电梯调度系统。在本项目实现的电梯调度系统中：共有 5 部电梯，楼层总共高度为 20 层（数量均可自行设置），每层楼设有一个“向上”按钮和一个“向下”按钮。每一部电梯内设置有各楼层的数字键、开门键、上行建、下行键和报警键。

电梯调度算法支持多电梯独立运行，联合调度。对于每一个单独电梯采用LOOK算法进行调度，对于楼层按钮产生的请求通过联合调度电梯来响应任务，通过将请求根据算法计算分配给一个电梯，从而使得在最短的时间内响应楼层请求，并提供直观的调度日志。

项目具有清晰的可视化界面。项目支持通过点击楼层按钮或电梯内部按钮来模拟实际电梯中的功能，并通过UI的变化来展示电梯的具体调度过程。

二、 开发环境

- 开发平台：Windows 11 10.0.22000
- 开发工具：Visual Studio Code 1.67.0
- 开发语言：Python 3.10.4
- 第三方库：PyQt6 为主要UI库，其他依赖详见 `requirements.txt`

三、 项目功能

- 多楼层多电梯的调度：项目实现了多个电梯在多个楼层之间的调度，并且可以同时通过外部按钮和内部按钮（包括楼层按钮、开关门、警报）双重控制电梯的状态
- 电梯信息的显示：电梯的楼层以及当前运行状态会显示在电梯的上部。
- 相关参数的更改：应用启动时可以自行设置电梯和楼层数量
- 警报暂停功能：可以手动停止某一电梯运行，或将暂停的电梯重新启动

四、 运行方式与使用说明

1. 文件结构

```
├─main.exe                          // 可执行程序
├─requirements.txt                  // 环境配置文本
├─项目文档.pdf                      // 项目文档
├─src                               // 源代码文件夹
|   ├─app_frame.py
|   ├─app_views.py
|   ├─customized_widget.py
|   ├─elevator.py
|   ├─main.py
|   └utils.py
├─resources                         // 资源文件夹
|      ├─checked.png
|      ├─elevator_closed.png
|      ├─elevator_down.png
|      ├─elevator_open.png
|      ├─elevator_stop.png
|      ├─elevator_up.png
|      └release.ico
```

2. 运行方式

a. 直接运行

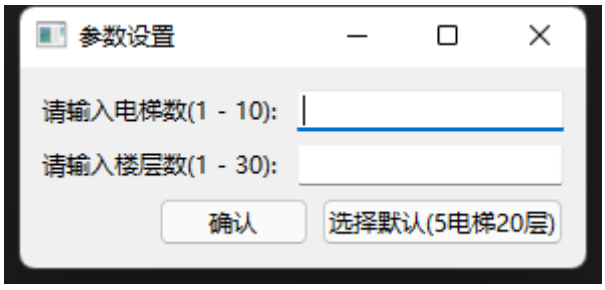
- 双击 `main.exe` 运行即可（需确保 `main.exe` 与 `resources` 文件夹在同级目录下，启动可能较慢，需要等待一段时间）

b. 构建（需要python环境）

```
pip install -r requirements.txt
python src/main.py
```

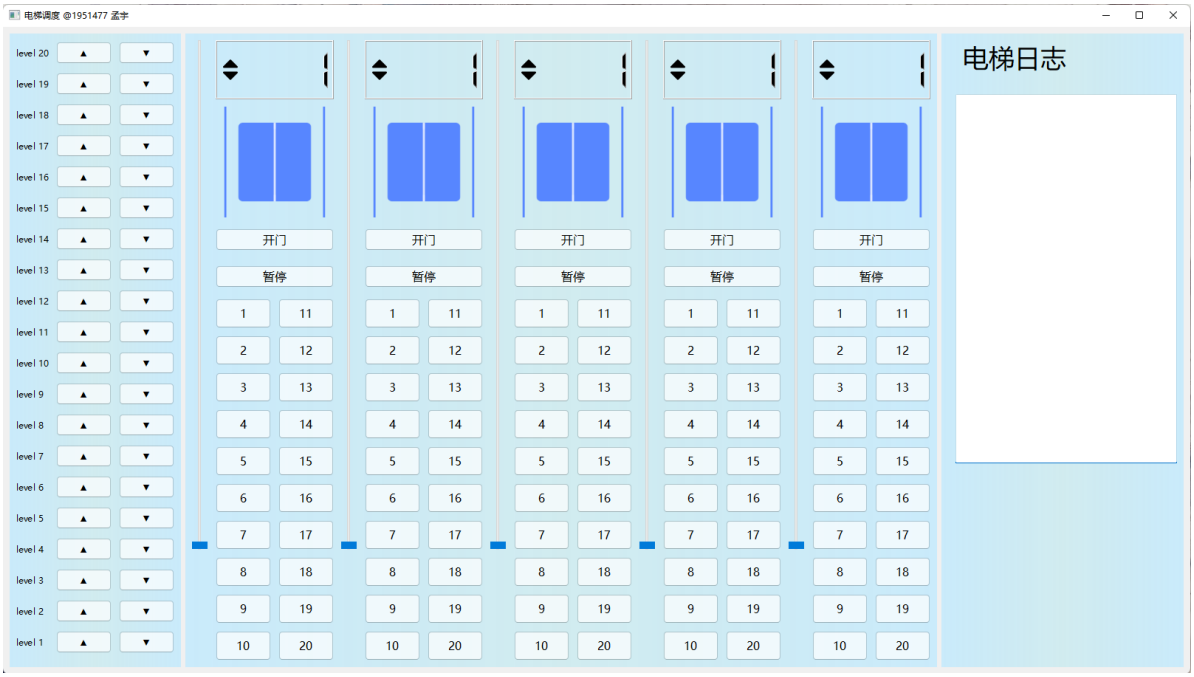
3. 使用说明

启动项目时将要求对电梯数与楼层数进行设置：



输入后点击确认即可，也可直接选择默认 5 电梯 20 楼层，输入有误时直接按默认处理。（10 与 30 并非上限，但数值过大可能使项目 ui 界面尺寸过大，因此不建议设置太大）。

设置后主程序启动，界面如下：



项目主界面见上，其中电梯的功能组件与可视化组件有:

- **电梯图标与电梯滑块**: 电梯的可视化，包含开关门的状态图标，和指示电梯所在高度的滑块
- **LCD显示器**: 电梯楼层的可视化，实时显示电梯所在楼层
- **电梯日志**: 电梯调度命令的可视化，输出用户命令与指令响应的记录日志
- **电梯内部楼层按钮**: 电梯内上下楼的操作按钮，电梯内调度主要控件。
- **电梯内部开门按钮**: 手动打开电梯门操作所需按钮，当电梯运行时无法使用
- **电梯工作暂停按钮**: 电梯故障时暂停工作。任何正常运行状态下都可使用
- **电梯外部上楼按钮**: 走廊上楼请求按钮，电梯外调度主要控件

用户可使用的主要功能有：

- **电梯内部楼层请求的内调度**：用户在电梯内按下楼层按钮， 所在电梯通过内调度安排楼层抵达的先后顺序
- **电梯外部上下楼请求的外调度**：用户在电梯外按上下楼按钮，系统通过外调度方法计算出可选择的最佳电梯并调度其前往，若无可用电梯时给出提示
- **电梯内部开门请求**：在电梯停止时进行手动开关电梯门，开门关门状态可以立即互相打断覆盖，由于演示开门时间较短，故不再单独设计关门按钮。
- **电梯暂停请求**：用户在电梯正常运行时按下暂停按钮，电梯被停用，电梯门关闭。
- **查看电梯调度情况**：通过文本框实时查看历史调度命令与调度系统的响应情况

五、 实现方案

1. 全局布局

`App` 为本应用的核心类，其中维护了与调度相关的所有变量。本项目中通过让一组电梯线程访问和修改与之一一对应的一组状态数组，实现电梯的调度。这些状态数组包括

变量名	类型	功能
elevatorGoal	Dest	标识电梯当前的运行目标，包括楼层和方向（向上、向下）
state	int	标识电梯移动状态，包括向上、向下和停止
shouldSleep	int	标识电梯门的状态，值为 1 表示暂停并开门
pause	int	标识电梯工作状态，用于警报暂停
floor	int	标识各个电梯当前所在的楼层

`Elevator` 类继承自 `QThread`，每一个电梯线程在运行中不断检查上述状态数组中本线程对应的状态值，以检查状态->处理状态->更新状态的过程在 `Elevator.run` 中进行循环，根据这些状态值调整和修改其他状态，并在循环的每一过程中同步对 UI 进行控制。

2. 调度策略

电梯内调度：LOOK

实现代码在 `App.setInternalDest` 中。

> LOOK算法是扫描算法的一种改进。扫描算法(SCAN)是一种按照楼层顺序依次服务请求的算法，它让电梯在最底层和最顶层之间连续往返运行，在运行过程中响应处在于电梯运行方向相同的各楼层上的请求。扫描算法较好地解决了电梯移动的问题，在这个算法中，每个电梯响应乘客请求使乘客获得服务的次序是由其发出请求的乘客的位置与当前电梯位置之间的距离来决定的，所有的与电梯运行方向相同的乘客的请求在一次电梯向上运行或向下运行的过程中完成，免去了电梯频繁的来回移动。对LOOK算法而言，电梯同样在最底层和最顶层之间运行。但当LOOK算法发现电梯所移动的方向上不再有请求时立即改变运行方向，而扫描算法则需要移动到最底层或者最顶层时才改变运行方向。

电梯内发出请求的调度基本采用 LOOK 算法，区别是在电梯上行时，不接受电梯内发出的目标为电梯当前所在楼层以下的楼层的请求；在电梯下行时，不接受电梯内发出的目标为电梯当前所在楼层以上的楼层的请求，这样的请求，应在电梯到达内部请求的最高（低）层之后再发出。

电梯外调度

实现代码在 `App.setExternalDestUp` 和 `App.setExternalDestDown` 中。

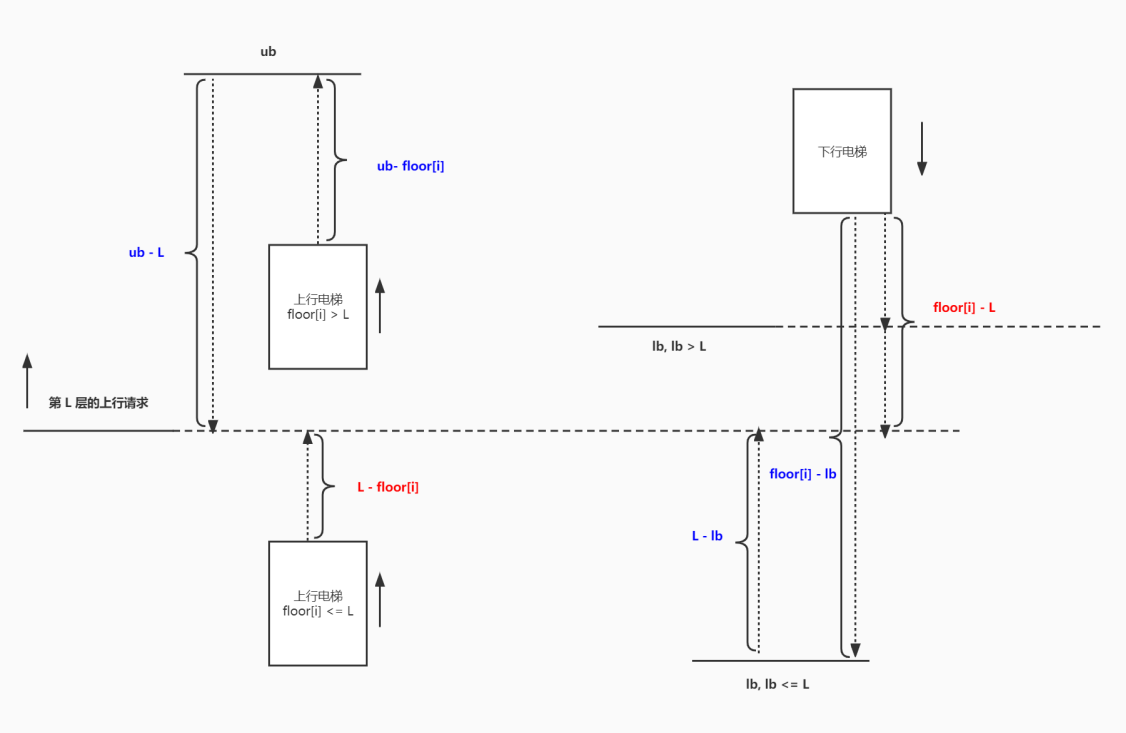
处理外部向上的请求和向下的请求是对称的，本文档以处理上行请求为例。当走廊中第 `L` 层发出了上行的请求时，做以下处理：

1. 检查是否存在电梯 `i` 满足：
    - 电梯 `i` 未处于警报停运状态，即 `pause[i] == 1`
    - 电梯 `i` 在第 `L` 层，即 `floor[i] == L`
    - 电梯 `i` 处于静止状态，即 `state[i] == STOP`

    若存在，则直接为本次请求分配电梯 `i` ，若不存在，进行步骤 2。
2. 检查每一部电梯 `i` ，求出电梯 `i` 与 `L` 层在上行前提下的距离 `dist` ，距离的定义为：
    - 若电梯 `i` 处于上行状态，设 `ub` 为电梯本次上行的最高目标
        - 若电梯 `i` 此时所在楼层高于 `L`，即 `floor[i] > L`，此时 `dist = 2 * ub - L - floor[i]`
        - 若电梯 `i` 此时所在楼层不高于 `L`，即 `floor[i] <= L`，此时 `dist = L - floor[i]`

- 若电梯 `i` 处于下行状态，设 `lb` 为电梯本次下行的最低目标
  - 若 `lb <= L`，此时 `dist = floor[i] + L - 2 * lb`
  - 若 `lb > L`，此时 `dist = lb - L`

直观的距离计算依据如下图：



找出使得 `dist` 值最小的电梯 `i`，将 `L` 层的上行请求加入电梯 `i` 的请求队列即可。

3. 其他功能

- **开门**：按下对应电梯的开门按钮，对应电梯的 `shouldSleep` 值置为 1 ，可以打开电梯门。此功能只能在电梯状态为 `STOP` 时使用，运行中点击开门将没有任何效果。
- **报警暂停**：按下对应电梯的暂停按钮，对应电梯的 `pause` 值值为 0，电梯停止工作，此后无法从内部添加请求，所有的外部请求在选择电梯时都会跳过此电梯。
- **电梯日志**：电梯在报警暂停、添加请求、分配请求、到达目的等情况下，会触发信号对象 `logger` 发送一条消息到电梯日志文本框，产生一条电梯日志。

六、运行展示

七、总结

遇到的问题：

1. Qt 的信号发收机制与直接调用方法在多线程异步场景下表现区别较大，尝试中遇到很多问题。
2. `QMovie` 在多线程场景的创建、播放和暂停比较复杂，项目中没有实现电梯开关门的动画播放。
3. QtGui 的 `imageio` 模块在读取 png 文件时发生 libpng warning，原因暂未知。

展望

- 改进调度算法提高效率。
- 将电梯容量纳入调度策略的考虑要素中。
- 增加随机乘客测试。