readme

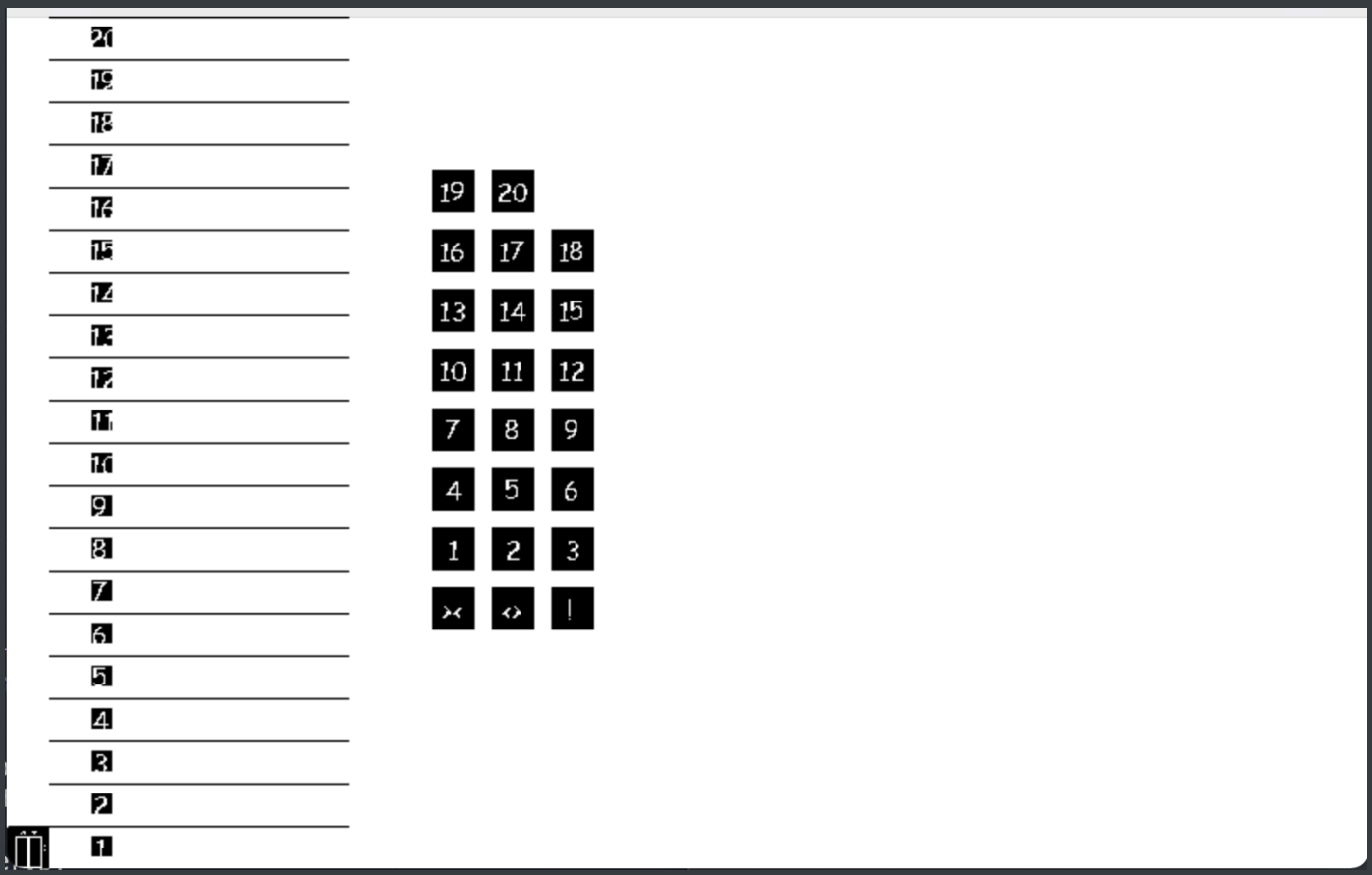
使用方式

首先进入工作目录

```
cd simpleSolution
```

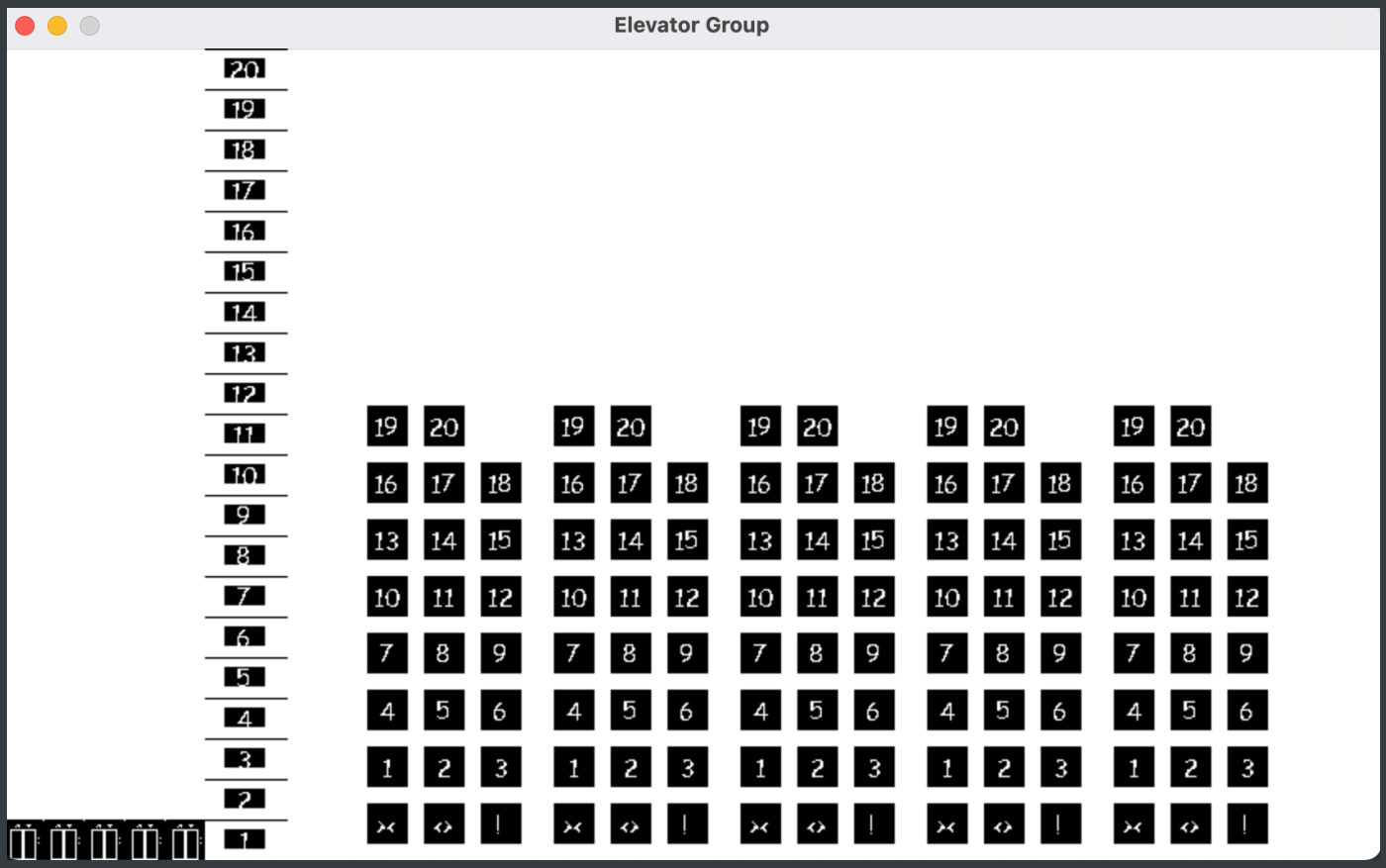
单个电梯

```
python SingleElevatorRender.py
```



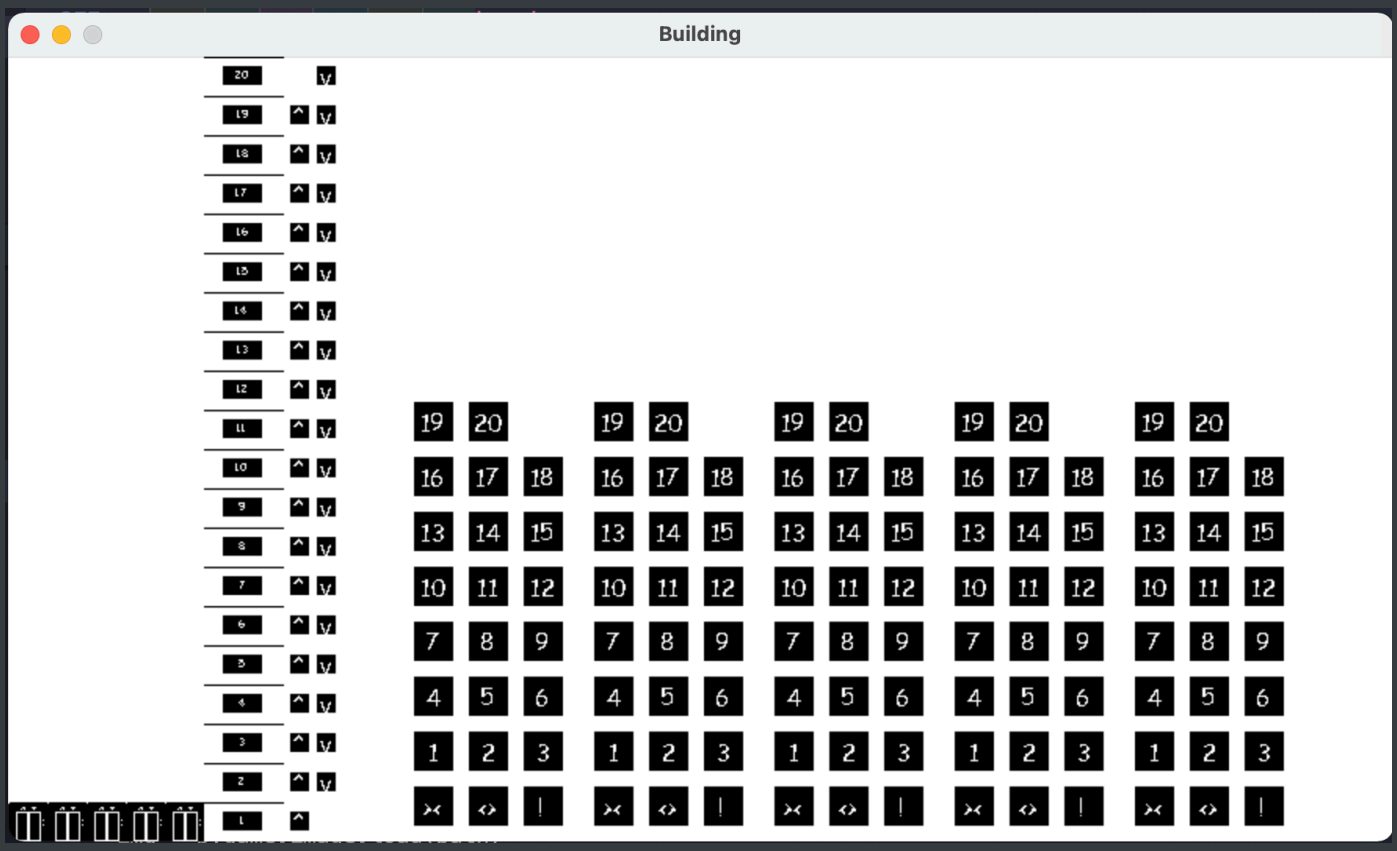
多个电梯

```
python ElevatorGroupRender.py
```



整栋楼的调度

```
python BuildingRender.py
```



调试方式

显示界面的参数都存在/simpleSolution/Global.py里

1. 更改窗口比例

   > 通过更改Global.py里的scale来调整
   >
   > 表示变成标准大小的1/scale倍

2. 更改楼层数

   > 通过更改Global.py里的max_Layer来调整

3. 更改电梯速度

   > 通过更改Global.py里的velocity来调整
   > 表示几个单位时间移动一层

4. 更改开门时间

> 通过更改Global.py里的waiting_Time来调整
>
> 表示停留几个单位时间

5. 更改电梯数目

> 通过更改Global.py里的default_Elevator_Number来调整

6. 更改每个显示东西的大小

> 下面的其他变量即可

类设计

```
- unit_Interval: int          #单位时间
- velocity: int               #几个单位时间移动一层
- waiting_Time: int           #在每个楼层停留多少单位时间
- max_Layer: int              #最大楼层数
- min_Layer: int              #最小楼层数
- default_Elevator_Number     #默认电梯数

Request
- layer: int                  #请求的层数
- is_Up: bool                 #是向上还是向下

Status
- state: int                  #0停止 1运行 2中停
- remaining_Time: int         #如果正在中停，即state==2，表示中停的剩余单位时间
                              #如果正在向上或者向下,即state==1，表示到下一层剩余
的单位时间

Elevator
- layer: int                  #所在层数，初始为0
- operation_Direction         #运行的方向 0代表没有方向 1代表向上 2代表向下
- status: Status          #状态
- button: {{layer_Num}:(T||F)}
- stop_Task: dict()
```

```
- add_Stop_Task() -> void   #电梯增加中停层，电梯已经运行起来了
- call() -> void
- button_Click() -> void
- step() -> Status          #走一步(走一个单位时间)，返回当前状态
- open_Click() -> void
- close_Click() -> void
- alarm_Click() -> void
- can_Stop() -> bool        #判断电梯当前是否能停下
- button_Restoration() -> void #复位
- can_Add_Task() -> bool
- get_Button_State -> list(bool)

Elevator_Group
- list: [Elevator]          #电梯的列表s
- wait_Queue: [Request]     #总的等待队列
- distribute() -> void      #每次循环的最后，开始为等待队列里的request分配电梯
- step() -> void
- add_Request() ->void

Floor
- layer: int                #所在层数
- up_Button_State: bool     #上行键状态
- down_Button_State: bool   #下行键状态

Building
- layers: [Floor]
- elevators: Elevator_List
```

调度设计

最简单的调度方式

> 哪个电梯有空，就让哪个电梯来

优化后的调度方式

> 对于每个楼层的任务，选择一个有空最近的电梯去接

认为还可以继续优化的方式

> 加入中断机制，在每一个单位时间，判断当前有空的电梯是否比当前正在运行的电梯位置更优，如果是，就中断正在运行的电梯，让更优的电梯去接。