**设计方案报告**

**——电梯调度**

**1项目背景及需求**

1.1项目背景

在一个20层高的建筑中，有五部电梯供用户使用。为了有效地管理这些电梯，需要设计并实现一个电梯调度程序。这个程序将基于线程的思想，利用合适的算法来调度电梯，以提高乘客的乘坐体验，减少等待时间，同时确保电梯的运行效率和安全性。

1.2项目需求

首先，为简化难度电梯调度存在以下两个前提假设：一是五部电梯门口的按钮是互联结的。当一个电梯按钮按下去时，其他电梯的相应按钮也就同时点亮，即外部调度的对象是所有的电梯。二是所有电梯初始状态都在第一层，且如果每个电梯在它的上层或者下层没有相应请求情况下，则应该在当前楼层保持不动。

在上述前提假设下，项目需求主要有以下三个方面：一是要动态调节建筑楼高和电梯数量。二是我们要设计必要按键和显示界面，如电梯内部的数字键、开门键、关门键和报警键，电梯外部的上行键和下行键，以及显示当前所在的楼层数和状态（DOWN、UP、STOP、OPEN、CLOSE）。其中电梯内外均需要显示当前所在楼层和当前电梯状态，因此将其合并在一起。三是设计合理的电梯调度算法。

**2程序类设计**

2.1 MainWindow类

该类主要实现输入楼层数和电梯数。通过UI界面，实现两个数值以步长为1进行调节或是直接输入。具体UI界面如下。

图 主界面UI

2.2 elevator类

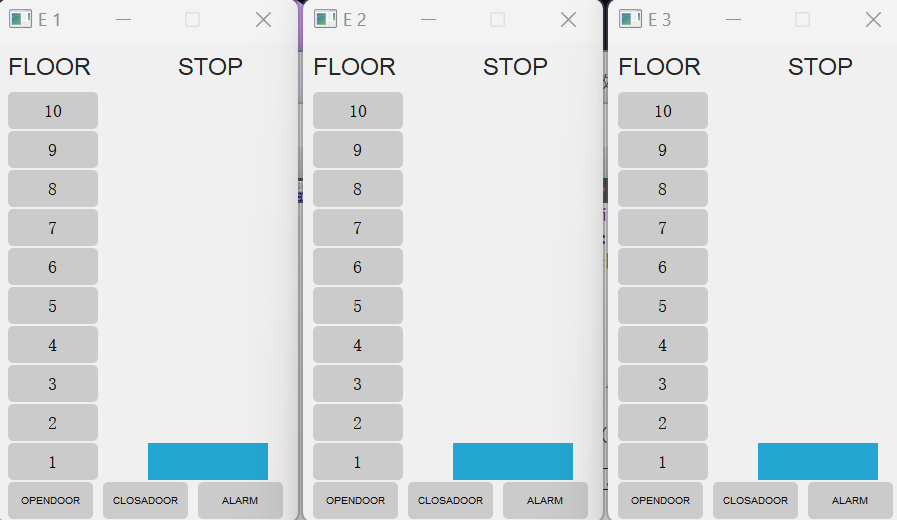
该类主要实现电梯内部按键操作、当前楼层数显示和当前状态显示。楼层数根据上一步的输入生成每一层的按键。同时，由于每一步电梯的按键显示模块类似，可以根据上一步的电梯数打开相应的新窗口，即每一个窗口代表一个电梯。具体UI界面如下。

图 2 电梯界面UI

其中包括电梯内部的数字键、、开门键、关门键和报警键，以及显示当前所在的楼层数和状态（DOWN、UP、STOP、OPEN、CLOSE）。（蓝色代表电梯，其左侧的数字按钮代表当前所在的楼层）

2.3 controller类

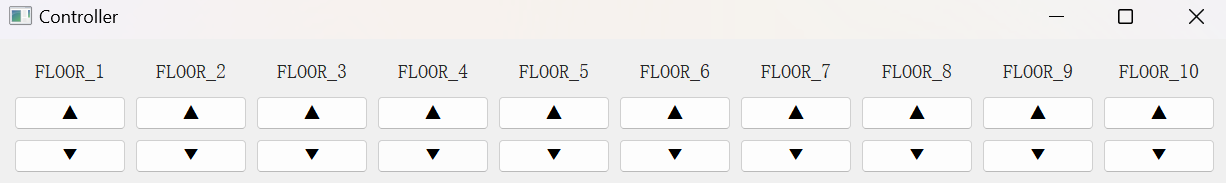
该类主要实现电梯外部的按键操作。

图 3 控制界面UI

其中包括电梯外部的上行键和下行键。电梯外部楼层显示和状态显示可以与电梯内部的显示合并，因此就不再重复显示。

**3功能介绍**

首先，主界面中可以输入或调节楼层数和电梯数，程序会根据输入或调节的值打开相应的界面。

然后，E 1至E n代表相应的电梯。按下FLOOR中对应的数字代表在电梯内部按下相应楼层的按钮。FLOOR右侧的标签会实时更新电梯的状态，状态包括DOWN、UP、STOP、OPEN、CLOSE五类。整个界面下部有三个按钮，分别是OPENDOOR、CLOSEDOOR、ALARM。在电梯处于停止状态时可以按下OPENDOOR键，电梯会立即打开门随后再次关闭。在电梯处于开门状态时，可以按下CLOSEDOOR键，电梯就会立即关闭（关门的时间固定为3秒）。按下ALARM键表示为电梯出现故障，本项目直接关闭该电梯在后续调度过程中不予考虑。

最后controller中的上下箭头，对应每层楼电梯外侧的上行、下行按键，按下按键后，会综合考虑所有未点击警报按钮的电梯，并进行选择。

**4调度算法**

首先说明，每部电梯内部按钮和外部按钮按下的信息分别存储在不同的数组中，以便进行调度操作。

4.1内部调度算法

电梯内部调度算法又分为从停止状态开始调度和从运行状态开始调度两种情况，默认每隔1秒更新一次电梯状态，即电梯上行下行的速度是每秒一层。电梯会根据当前的状态对楼层数进行加1或者减1的操作。在此基础上，我们的重点变为确定电梯的状态。

对于从停止状态开始调度，首先计算所有已经按下楼层中距当前楼层最近的楼层。如果该楼层大于当前楼层，则改变当体状态为上升；如果该楼层小于当前楼层，则改变当电梯状态为下降；如果该楼层等于当前楼层，则进行开门操作。除此之外，我们还要注意每次进行完如上操作后，检查是否完成某项任务，如果完成，则将该楼层从任务列表中删除。

对于从上行或下行状态开始调度，首先要检查当前所处楼层上面或下面的楼层是否有调度的任务，并进一步结合上前电梯状态，确定下一步电梯的状态。具体状态变化如下表所示。

表 2 电梯状态变化表

|  |  |  |  |
| --- | --- | --- | --- |
| **电梯当前状态** | **上方是否有任务** | **下方是否有任务** | **电梯下一步状态** |
| ↑ | 有 | 有 | ↑ |
| ↑ | 有 | 无 | ↑ |
| ↑ | 无 | 有 | ↓ |
| ↑ | 无 | 无 | 停止 |
| **电梯当前状态** | **上方是否有任务** | **下方是否有任务** | **电梯下一步状态** |
| ↓ | 有 | 有 | ↓ |
| ↓ | 有 | 无 | ↑ |
| ↓ | 无 | 有 | ↓ |
| ↓ | 无 | 无 | 停止 |

如上表所示，标红的是电梯前后状态变化的特殊情况，也是我们需要重点考虑的情况。

对于调度时电梯正在处于开门和关门状态的情况，程序不会改变其状态，我们无需特殊处理。直到电梯状态变为停止、上升、下降中的一种状态时，我们做上述相同处理即可。

4.2外部调度算法

外部调度的总体规划是一次见检查各电梯是否符合加入任务的条件，如果符合加入条件，则将对应楼层加入相应的数组，否则不加入。如果没有电梯接收该楼层，则等待一定时间后再次遍历电梯判断是否可以加入，直至将该信息加入相应的数组中。

上述电梯是否接收该楼层的判断依据如下表所示。

表 3 电梯是否接收外部调度的判断依据

|  |  |  |  |
| --- | --- | --- | --- |
| **电梯当前状态** | **请求状态** | **当前VS请求楼层** | **是否调度** |
| ↑ | ↑ | ＞ | 是 |
| ↑ | ↑ | ＜ | 是 |
| ↑ | ↓ | ＞ | 是 |
| ↑ | ↓ | ＜ | 否 |
| ↓ | ↑ | ＞ | 否 |
| ↓ | ↑ | ＜ | 是 |
| ↓ | ↓ | ＞ | 是 |
| ↓ | ↓ | ＜ | 是 |

基于内部调度的算法进行分析。当前电梯状态和请求状态都是上升，且当前楼层大于请求楼层，如果加入该请求，则表明当前楼层上方和下方均有任务，电梯状态保持上升不变。当前电梯状态和请求状态都是上升，且当前楼层小于请求楼层，如果加入该请求，电梯内部调度算法会综合考虑加入后的楼层进行调度。当前电梯状态上升、请求状态为下降，且当前楼层大于请求楼层，如果加入该请求，则表明当前楼层上方和下方均有任务，电梯状态保持上升不变。以上三种情况都是合理的。但当当前电梯状态上升、请求状态为下降，且当前楼层小于请求楼层时，如果加入该请求，则意味着电梯有可能会先在请求楼层停下然后下行，从而忽视请求楼层之上按钮信息。

举个简单的例子。电梯当前楼层为5且为上升，目的是去相应10层的按钮信息，而我们在7层电梯外侧按下向下的按钮。我们希望的是电梯上升到10层后，在下降到7层。但是如果即可将当前楼层加入电梯按钮信息数组中，电梯会现在7层停下，再在10层停下。因此，为避免这种情况，该电梯拒绝加入外部按钮的信息，我们继续判断下一个电梯是否符合条件。

相应的，表中后四种情况的分析类似。最终电梯需要拒绝上表中标红的两种情况。

**5测试**

5.1内部调度测试

先点击10层按钮，在电梯未到达7层之前点击7层的按钮，电梯会现在7层停下，然后到达10层。符合预期结果。

先点击10层，在电梯超过5层后点击5层按钮，电梯会先到达10层，然后下降到5层。符合预期结果。

下降过程同理，均符合预期结果。

5.2外部调度测试

当当前电梯处于上升状态、按下外部按钮下降，且当前楼层小于外部按钮楼层，该电梯拒绝请求。电梯1在内部按下10层使其处于上升状态，然后按下7层的下降按钮，电梯2相应该请求。符合预期结果。

另一过程同理，符合预期结果。

5.开门、关门、和警报测试

按下OPENDOOR按钮,电梯状态变为OPEN。按下CLOSEDOOR按钮，电梯状态立刻变为CLOSE。符合预期结果。

按下警报按钮后，该电梯直接关闭不在相应任何请求。符合预期结果。