电梯调度设计方案

2253893 苗君文

1. **项目分析**
   1. **基本任务**

某一层楼20层，有五部互联的电梯。基于线程思想，编写一个电梯调度程序。

* 1. **功能描述**

每个电梯具有一些按键，如：数字键、关门键、开门键、上行键、下行键、报警键等。

有数码显示器指示当前电梯状态。

每层楼、每部电梯门口，有上行、下行按钮、数码显示。

五部电梯相互联结，即当一个电梯按钮按下去时，其它电梯相应按钮同时点亮，表示也按下去了。

所有电梯初始状态都在第一层；每个电梯在没有相应请求的情况下，则应该在原地保持不动。

1. **开发环境**

开发环境：Windows 11

开发软件：QtCreator (Desktop Qt 5.14.2 MinGW 32-bit)

开发语言：C++

运行方式：打开demo文件，双击elevator.exe文件。

1. **系统架构**

系统设计采用面向对象的四项，将电梯、控制器和界面等功能模块分别设计为类，通过封装、继承和多态等面向对象特性实现系统的灵活性和可扩展性。

* 1. **Container类**

Container类表示电梯的货箱，用于管理电梯的运行状态和乘客请求信息。

**成员变量：**

\_status：电梯当前的运行状态，包括上行、下行和空闲状态。

\_floor：当前电梯所在的楼层。

\_Floors[MAX\_FLOORS]：表示需要到达的楼层。要到达楼层设为TRUE。

\_extend：表示是否需要考虑另一方向的按钮请求。

\_closeDoor：表示电梯门是否关闭。

**成员函数：**

addFloor(int floor)：添加电梯需要到达的楼层。

clearFloor(int floor)：清除电梯到达指定楼层的请求。

setStatus(int status)：设置电梯的运行状态。

getStatus()：获取当前电梯的运行状态。

getFloor()：获取当前电梯所在的楼层。

checkFloor(int floor)：检查当前电梯是否被按下到达指定楼层的按钮。

setExtend(bool extend)：设置是否需要考虑另一方向的按钮请求。

getExtend()：获取是否需要考虑另一方向的按钮请求。

setFloor(int floor)：设置电梯当前所在的楼层。

getCloseDoor()：获取电梯门的关闭状态。

setCloseDoor(bool flag)：设置电梯门的关闭状态。

* 1. **Elevator类**

Elevator类继承自QMainWindow，表示整个电梯调度系统的主要控制类，管理界面和电梯调度逻辑。

**成员变量：**

\_isDamage[ELEVATOR\_NUM]：表示五部电梯的损坏情况。

\_elevator[ELEVATOR\_NUM]：表示五部电梯对象。

\_upWaitFloors[MAX\_FLOORS]：上行等待队列。

\_downWaitFloors[MAX\_FLOORS]：下行等待队列。

**定时器对象：**

realTime：用于检查电梯状态的定时器。

colorTime：用于调整按钮颜色的定时器。

wakeUp：用于唤醒电梯的定时器。

updateElevatorOne 到 updateElevatorFive：分别用于更新五部电梯状态的定时器。

closeDoors：定时关门的定时器。

**成员函数：**

openDoor(int i)：电梯开门动画函数。

closeDoor(int i)：电梯关门动画函数。

**定时器槽函数：**

checkState()：周期性检查电梯状态和乘客请求，处理开关门等操作。

checkColor()：定时调整按钮颜色的槽函数。

updateWake()：唤醒电梯的槽函数，根据乘客请求调整电梯运行状态。

updateEle1() 到 updateEle5()：分别用于更新五部电梯状态的槽函数。

updateDoors()：定时关门的槽函数。

**电梯按钮槽函数和楼层按钮槽函数**：处理用户操作，例如乘客按下电梯内部按钮或楼层按钮的响应。

* 1. **类之间的关系**

Elevator类作为系统的主要控制类，包含多个Container类的实例，用于管理多部电梯的运行状态和乘客请求。

Elevator类通过定时器槽函数调度多部电梯的运行，并根据乘客请求调整电梯状态和执行相应的动画效果。

1. **核心算法**
   1. **电梯调度算法**

电梯调度算法由updateWake函数实现，用于根据乘客请求和电梯状态，决定如何分配空闲电梯的运行方向和楼层。

**算法思路：**

对每一部电梯进行遍历，首先判断是否为损坏状态。如果电梯未损坏且处于空闲状态（FREE），则尝试唤醒该电梯以响应乘客请求。

首先，检查电梯内部是否有乘客按下楼层按钮，若有，则根据楼层位置确定电梯运行方向（UP 或 DOWN）。

如果电梯内部无乘客请求，再检查外部同层楼按钮的状态，根据楼层位置决定电梯的运行方向。

若外部同层楼按钮没有请求，则根据外部非同层的按钮请求，寻找距离最近且方向适合的空闲电梯，将其唤醒。

**具体操作：**

**（1）电梯内部唤醒**

首先遍历所有电梯，检查每个电梯的状态和请求：

如果电梯没有损坏且处于空闲状态 (\_elevator[i]->getStatus() == FREE)，则根据当前电梯所在楼层和电梯内部的请求来唤醒电梯：如果电梯内部有上行请求（即有乘客要去更高的楼层），则设置电梯状态为上行 (\_elevator[i]->setStatus(UP))，并且电梯会前往最高的请求楼层。如果电梯内部有下行请求（即有乘客要去更低的楼层），则设置电梯状态为下行 (\_elevator[i]->setStatus(DOWN))，并且电梯会前往最低的请求楼层。

**（2）外部同层楼唤醒**

如果电梯仍然处于空闲状态，表示电梯没有内部请求，然后检查同层楼外部是否有乘客等待电梯：如果当前楼层有上行请求 (\_upWaitFloors[\_elevator[i]->getFloor()])，则设置电梯状态为上行，表示电梯要前往请求的楼层。如果当前楼层有下行请求 (\_downWaitFloors[\_elevator[i]->getFloor()])，则设置电梯状态为下行，表示电梯要前往请求的楼层。

**（3）外部非同层唤醒**

如果电梯仍然处于空闲状态且没有在同层楼内找到请求，则根据外部请求来寻找最适合的电梯。

对于上行请求，遍历所有楼层，寻找最近且满足条件的电梯：如果有正在上行且比请求楼层低的电梯，则选择这个电梯；如果没有符合条件的上行电梯，则选择空闲状态且离请求楼层最近的电梯。

对于下行请求，同样遍历所有楼层，寻找最近且满足条件的电梯：

如果有正在下行且比请求楼层高的电梯，则选择这个电梯；如果没有符合条件的下行电梯，则选择空闲状态且离请求楼层最近的电梯。

**（4）更新电梯状态**

根据上述步骤得到的最适合的电梯和请求楼层，设置电梯的运行状态：如果找到了适合的电梯并且需要换向，则将电梯状态设置为对应方向，并标记该电梯已经换向；如果找到了适合的电梯并且不需要换向，则根据电梯当前状态来确定电梯的运行方向。

* 1. **更新电梯运行状态。**

在电梯调度系统中，针对每部电梯的运行状态更新是系统中的关键操作，通过监测乘客请求和电梯当前状态，决定电梯的运行方向和楼层位置。此操作通过updateEle1-5函数实现。

**算法思路：**

首先判断电梯是否损坏或者门是否关闭，若是，则不进行任何操作，直接返回。

如果电梯处于上升（UP）状态，检查上方是否有乘客请求或者等待乘客的楼层。

如果有乘客请求或者等待乘客的楼层，则继续上升。

如果没有请求，则判断是否需要切换方向（根据电梯当前的运行状态和乘客请求确定是否需要切换方向）。

如果电梯处于下降（DOWN）状态，类似地检查下方是否有乘客请求或者等待乘客的楼层，然后继续下降。

如果电梯处于空闲（FREE）状态，可以根据电梯内外部的乘客请求决定运行方向。

**具体操作：**

**（1）检查电梯损坏状态**

检查电梯是否处于损坏状态 (if(\_isDamage[0]))，如果是，则直接返回，不进行任何操作。

**（2）检查电梯门状态**

检查电梯的门是否处于关闭状态 (if(!\_elevator[0]->getCloseDoor()))，如果门没有关闭，则返回，不执行任何操作。

**（3）根据电梯当前状态执行相应逻辑**

根据电梯的当前状态（上升、下降、空闲），执行相应的逻辑：

如果电梯正在上升 (\_elevator[0]->getStatus()==UP)：

检查上升过程中是否有内部或外部请求需要停靠或换向。

如果需要停靠或换向，则调整电梯状态和位置，并更新楼层显示。

同时执行电梯上升的动画效果，通过调整门的位置实现视觉效果。

如果电梯正在下降 (\_elevator[0]->getStatus()==DOWN)：

检查下降过程中是否有内部或外部请求需要停靠或换向。

如果需要停靠或换向，则调整电梯状态和位置，并更新楼层显示。

同时执行电梯下降的动画效果，通过调整门的位置实现视觉效果。

* 1. **电梯状态的检查与处理**

这是保证系统正常运行的关键环节，用于处理电梯的上升和下降状态下的楼层请求和电梯楼层按钮的状态，此功能由函数checkState实现。

**算法思路：**

对每一部电梯进行遍历，检查电梯的状态（UP、DOWN），并根据当前状态处理相应的乘客请求和电梯楼层按钮的状态。

如果电梯处于上升（UP）状态：检查当前楼层是否有乘客请求（上行等待队列）或者电梯内部的楼层按钮被按下。如果有乘客请求或者电梯内部有楼层按钮被按下，则开启电梯门（如果门是关的），处理请求，清除对应楼层的等待状态，并将电梯门的状态设置为开启。

如果电梯处于下降（DOWN）状态：类似地，检查当前楼层是否有乘客请求（下行等待队列）或者电梯内部的楼层按钮被按下。如果有乘客请求或者电梯内部有楼层按钮被按下，则开启电梯门（如果门是关的），处理请求，清除对应楼层的等待状态，并将电梯门的状态设置为开启。

**具体操作：**

**（1）循环遍历每个电梯**

**（2）检查电梯状态为非损坏状态**

**（3）处理上升状态**

如果电梯处于上升状态 (\_elevator[i]->getStatus() == UP)，则判断当前楼层是否有上行请求或内部请求需要处理：

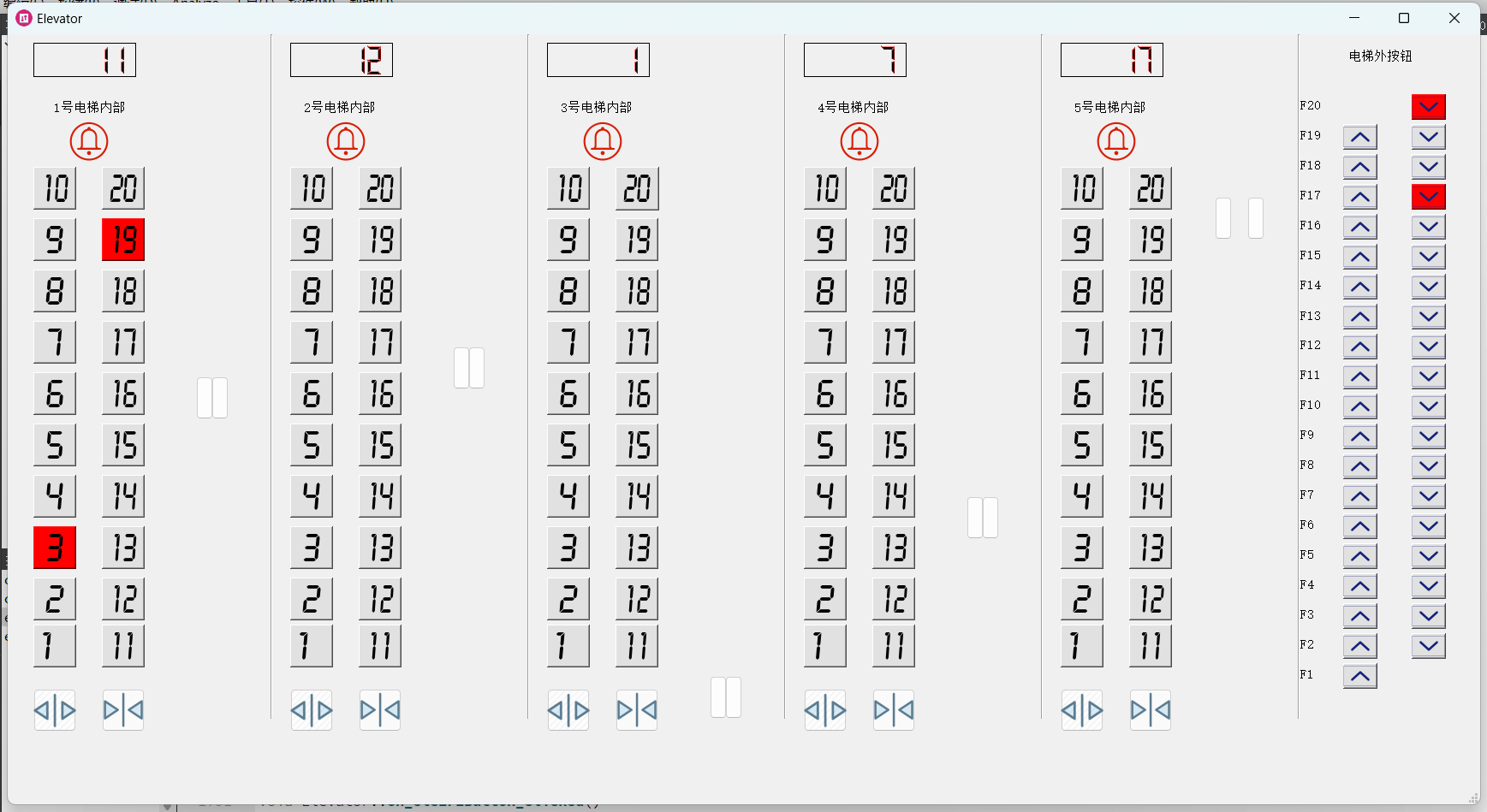
如果当前楼层有上行请求 (\_upWaitFloors[\_elevator[i]->getFloor()]) 或者电梯内部有乘客请求该楼层 (\_elevator[i]->checkFloor(\_elevator[i]->getFloor()))，则执行以下操作：若电梯门是关闭的 (\_elevator[i]->getCloseDoor())，则调用 openDoor(i) 函数打开电梯门。而后清除该楼层的上行请求标记，并清除电梯内部的乘客请求，设置电梯门为打开状态。

**（4）处理下降状态**

如果电梯处于下降状态 (\_elevator[i]->getStatus() == DOWN)，同样判断当前楼层是否有下行请求或内部请求需要处理：

如果当前楼层有下行请求 (\_downWaitFloors[\_elevator[i]->getFloor()]) 或者电梯内部有乘客请求该楼层 (\_elevator[i]->checkFloor(\_elevator[i]->getFloor()))，则执行以下操作：若电梯门是关闭的 (\_elevator[i]->getCloseDoor())，则调用 openDoor(i) 函数打开电梯门。而后清除该楼层的下行请求标记，并清除电梯内部的乘客请求，设置电梯门为打开状态。

1. **用户界面设计**



用户界面采用Qt实现，界面设计简介直观，考虑电梯内部与电梯外部的情况，界面主要涵盖以下部分：

**（1）电梯实时状态显示**：数字显示当前电梯所在的楼层，有电梯门表示电梯的位置移动与开关门状态。若电梯故障，则电梯内部的所有楼层按钮被禁用。

**（2）按钮**：电梯内部按钮均用清晰的数字表示。每个楼层都有上行和下行的按钮，而考虑实际情况最底层无下行按钮，最高层无向上按钮。每部电梯内部设置报警按钮，再按一次可以取消报警状态。开门关门按钮可以在电梯不移动时使用，开关门的效果可以通过电梯门动画看出。

1. **心得体会**
   1. **Qt Creator的学习与使用**

在本次项目中，我深入学习和使用了Qt Creator作为开发工具。Qt Creator提供了强大的图形化界面设计工具，可以通过拖拽和布局设置快速设计界面，降低了界面开发的复杂度和工作量。信号与槽机制是其核心特性之一，通过信号与槽的连接，实现了组件间的通信和交互。这种方式简洁而灵活，使得界面和逻辑之间的耦合度大大降低。Qt Creator提供了便捷的多线程编程支持，可以通过QThread等类实现多线程操作，提高了程序的并发性和响应速度。

* 1. **算法设计**

电梯调度算法是电梯系统的核心部分，设计了唤醒空闲电梯和处理电梯状态的算法，以实现高效的电梯调度和请求处理。通过合理的调度算法，可以最大限度地减少乘客等待时间和电梯运行时间，提升系统的效率和用户体验。在实现电梯调度算法的过程中，我不断优化算法逻辑，考虑到电梯的状态和乘客请求的优先级，确保系统的稳定性和高效性。同时，通过Qt Creator提供的调试和测试工具，验证和调整算法的实际效果。