**简介：**

本作业基于qt完成，选择本文件夹下的可执行文件Intersection.exe快捷方式即可打开。

（由于源码使用了std::thread，请在windows系统下打开exe）

**文件说明：**

Intersection文件夹下为源码及qt项目文件：

1. Intersection.h及Intersection.cpp为十字路口类（Intersection）的具体实现的源码。其中对创建进程较为关键的几个函数为Tick、GoVehicle、GoEmergency、GoVehiclelater。开启进程的函数为：GoThread。
2. mainwindow.ui为构建ui界面的源码。
3. mainwindow.h及mainwindow.cpp为联系窗口各组件与Intersection类中成员函数的源码。
4. main.cpp为主函数。

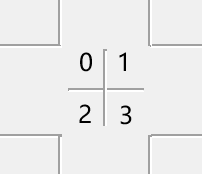
Intersection\_Release文件夹下为qt的exe运行所需库及可执行文件Intersection.exe

**内容说明：**

线程管理逻辑：

紧急车辆的调度采用非抢占式，采用设定优先级（南北向优先）的方式避免死锁的发生。

将路口交叉部分的四个象限设为一个size=4的std::mutex互斥量组road\_[4]



设置size=4的std::mutex互斥量组queue\_RW\_[4]表示队列是否被读写

创建存储Vehicle类的vector(vehicle\_queue\_[4])存储四个方向即将进入路口的车辆

1. 添加车辆：如果车辆为普通车辆，则从队列队尾进入等待队列。如果车辆为紧急车辆，则从队列队头+当前路口紧急车辆数目位置进入等待队列（紧急优先，同优先级遵循先进先出）。
2. 信号灯管理：由Tick函数创建的线程，红灯为8秒（如有需要，允许直接添加改变信号灯持续时间或人工介入实时改变信号灯颜色的功能），绿灯加黄灯时间等于红灯，且绿：黄=3:1。以及车辆通过路口两段的时间都为1s。由于黄灯的存在，普通车辆的判断不需要特殊处理即可避免死锁。
3. 车辆调度总管理：每个方向由GoVehicle函数创建的线程持续管理车辆进入路口，在每辆车进入路口后再创建由GoVehiclelater函数创建一个线程持续判断是否继续前进，若成功前进驶出路口，则该线程返回。（两个线程均每50ms进行一次判断）
4. 普通车辆的进入路口调度：判断条件：对向信号灯为绿灯、左右路口没有紧急车辆在队列中或在路上（因为若只有对向有紧急车辆，则不会影响，可以平行行驶）、即将进入的路口未被占用。
5. 紧急车辆的进入路口调度：南北方向车辆只要即将进入的路口未被占用即可行驶，东西方向车辆还需要南北向没有紧急车辆行驶或等待才可行驶。
6. 车辆的后半段调度：前方路口未被占用即可继续行驶驶出路口。

**功能说明：**

本作业能够较好地实现：

1. 信号灯、车辆(线程)数、路口互斥量是否占用等信息的实时显示。
2. 对全部为直行行驶车辆的十字路口交叉路口的模拟与交互，能够避免大部分的死锁问题。
3. 根据信号灯及紧急车辆等待与行驶情况对普通车辆的车辆调度。
4. 根据路面占用情况及紧急车辆等待与行驶情况对紧急车辆的车辆调度。

本项目不足之处：

1. 由于QtPropertyAnimation组件使用不熟练，可能导致未知原因的闪退或动画界面出错。
2. 若南北向一直有紧急车辆行驶，则东西向紧急车辆无法行驶，因此存在饥饿问题。

感谢您的阅读与使用。