天津大学

智能与计算学部

实习报告

班　　级：　计算机三班

姓　　名：　汪昌盛

学　　号：　3016216081

日　　期：　2018 年 9 月

实时动态的红绿灯系统

一、问题描述

请设计并实现一个实时动态的红绿灯系统。要求能够根据各个方向的等待车辆数（可以用随机数生成），实时更改红绿灯的等待时长。要求在图形界面上显示各个方向的车辆，以及红绿灯变化情况。

二、问题分析

2.1 问题理解

问题要求设计并实现一个实时动态的红绿灯系统。要求在图形界面上显示各个方向的车辆，以及红绿灯变化情况。考虑设计一个十字路口，一共四个方向，由于各个方向的车辆数可能很多，这里每个方向考虑只用一辆车代替。红绿灯的等待时长考虑将某个方向的车辆与十字路口的所有车辆作比较，按比例分配时间。由于车辆是实时变化的，红绿灯的等待时长应当每隔一段时间刷新一次。

2.2 解决方案说明

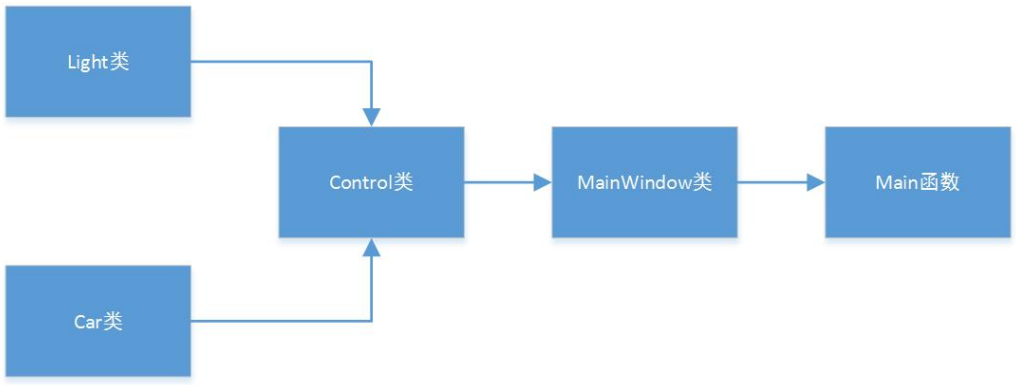

图 1 总体设计思路

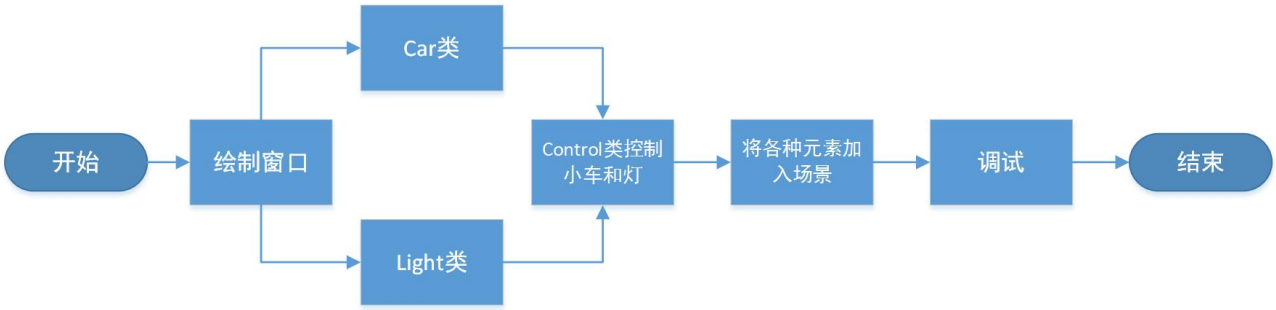

图 2 设计系统流程图

2.3 实验环境

Windows 10 操作系统
Qt 5.0

三、数据模型

3.1 Car 类

①枚举类型 (NoMove,MoveLeft,MoveRight,MoveUp,MoveDown) 小车的移动方向
②绘制小车形状的相关函数
③设置移动方向的函数，返回当前移动方向的函数
④小车的位置，移动方向
⑤Control 类的一个实例，便于后续操作
⑥小车运动函数（槽函数）
⑦小车向四个方向移动的函数

3.2 Light 类

①灯的位置、颜色（QColor）
②绘制灯形状的相关函数

3.3 Control 类

①计时器、时间计算值 (double)、颜色变量 (QColor)
②四辆小车，四个灯
③标示“上下左右”的四个标签
④动态刷新函数（槽函数）

⑤暂停和重新开始函数（槽函数）

3.4 MainWindow 类

①调整视图大小的函数
②初始化场景及背景的函数
③创建动作和菜单
③添加的动作 (QAction)

四、算法详细介绍

4.1 小车运动

```
1  void Car::advance(int step)
2  {
3      if (!step) {
4          return;
5      }
6      if (moveDirection == NoMove) {
7          return;
8      }
9      switch (moveDirection) {
10         case MoveLeft:
11             moveLeft();
12             break;
13         case MoveRight:
14             moveRight();
15             break;
16         case MoveUp:
17             moveUp();
18             break;
19         case MoveDown:
20             moveDown();
21             break;
22         case NoMove:
23             break;
24     }
25
26     setPos(location);
```

```
27 }
```

依据小车属性来 moveDirection 运行不同的移动函数，下面以 moveLeft() 为例，其他方向的移动函数类似。

```
1 void Car::moveLeft()
2 {
3       location.rx() -= 10;
4       if (location.rx() < 0) {
5          location.rx() = 500;
6       }
7 }
```

可以发现，moveLeft() 函数每调用一次，小车向左移动 10 个单位。下面给出控制小车每隔 50 毫秒运动一次的代码。

```
1 timer.start(50);
2 connect(&timer, SIGNAL(timeout()),&scene, SLOT(advance()));
```

可以发现，若小车的属性 moveDirection!= NoMove，每隔 50ms，advance() 将会被调用一次，小车向给定方向移动 10 个单位，这样就完成了动画的设计。

算法流程图如下：

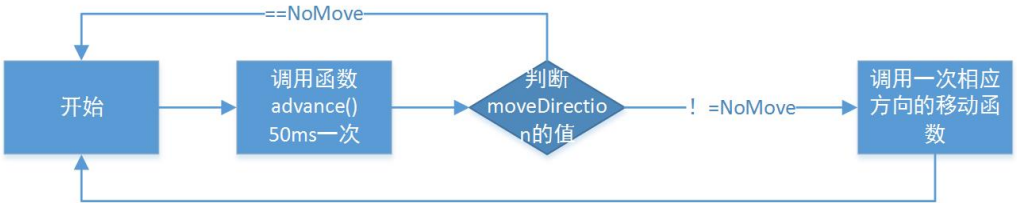

图 3 小车运动

4.2 视图动态刷新

下面给出各个方向车辆数量刷新和红绿灯等待时间刷新的函数 run()，其中函数 manycars1()，manycars2()、yellowlight1()、yellowlight2() 负责刷新屏幕。

```
1 void Control::run(){
2       double numleft=get_random_number(2)+1;//各个方向的等待车辆数
3       double numright=get_random_number(4)+1;
4       double numup=get_random_number(6)+1;
5       double numdown=get_random_number(8)+1;
```

```
6
7       numLeftLabel->setText(QString::number(numleft));//写入刷新的各方向车辆数
8       numRightLabel->setText(QString::number(numright));
9       numUpLabel->setText(QString::number(numup));
10      numDownLabel->setText(QString::number(numdown));
11
12      counttime=(numleft+numright)/(numleft+numright+numup+numdown);
13
14      manycars1();//运行一下(让车多的方向通过)
15
16      timer3->stop();
17      timer3->start(10000*counttime);
18      timer3->setSingleShot(true);
19      connect(timer3, SIGNAL(timeout()), this, SLOT(yellowlight1()));//黄灯
20
21      timer4->stop();
22      timer4->start(10000*counttime+1000);
23      timer4->setSingleShot(true);
24      connect(timer4, SIGNAL(timeout()), this, SLOT(manycars2()));
25
26      timer5->stop();
27      timer5->start(11000);
28      timer5->setSingleShot(true);
29      connect(timer5, SIGNAL(timeout()), this, SLOT(yellowlight2()));
30
31  }
```

下面给出 manycars1() 的部分核心代码。

```
1   void Control::manycars1(){
2
3     car1->location=QPointF(190,230);
4     car2->location=QPointF(300,270);
5     car3->location=QPointF(230,190);
6     car4->location=QPointF(270,300);
7
8     scene.removeItem(lig1);
9     scene.removeItem(lig2);
10    scene.removeItem(lig3);
11    scene.removeItem(lig4);
12    scene.removeItem(car1);
13    scene.removeItem(car2);
```

```
14      scene.removeItem(car3);
15      scene.removeItem(car4);
16
17
18      //左右
19      lig1->cl=QColor(Qt::green);
20      scene.addItem(lig1);
21      lig2->cl=QColor(Qt::green);
22      scene.addItem(lig2);
23      lig3->cl=QColor(Qt::red);
24      scene.addItem(lig3);
25      lig4->cl=QColor(Qt::red);
26      scene.addItem(lig4);
27
28      car1->setMoveDirection(Car::MoveRight);
29      scene.addItem(car1);
30      car2->setMoveDirection(Car::MoveLeft);
31      scene.addItem(car2);
32      car3->setMoveDirection(Car::NoMove);
33      scene.addItem(car3);
34      car4->setMoveDirection(Car::NoMove);
35      scene.addItem(car4);
36  }
```

可以发现由于 manycars1() 函数执行后，左右的车辆正常通行；与之对应，manycars2() 函数被调用后，上下的车辆正常通行。这说明函数 run() 第 14 行之后先令上下的车辆通行，变量 counttime 控制红绿灯的刷新时间，一秒黄灯后令左右的车辆通行。与小车的运行函数类似，下面给出控制函数 run() 多次调用的代码, 这样完成对视图的不断更新。

```
1  QTimer *timer2 = new QTimer(this);
2  connect(timer2, SIGNAL(timeout()), this, SLOT(run()));
3  timer2->start(10000);//以10s为一个周期
```

算法流程图如下：

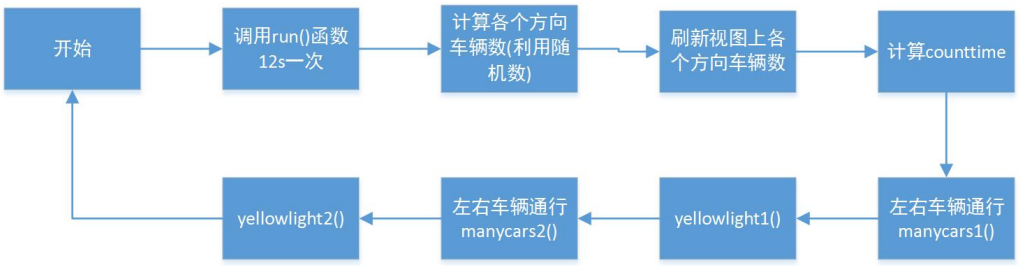

图 4 动态刷新

以上就是这个工程的两个核心算法，具体到如何建立坐标画图、添加灯和小车、生成随机数、暂停和重新开始函数等等这里就不再赘述。

五、实验结果分析

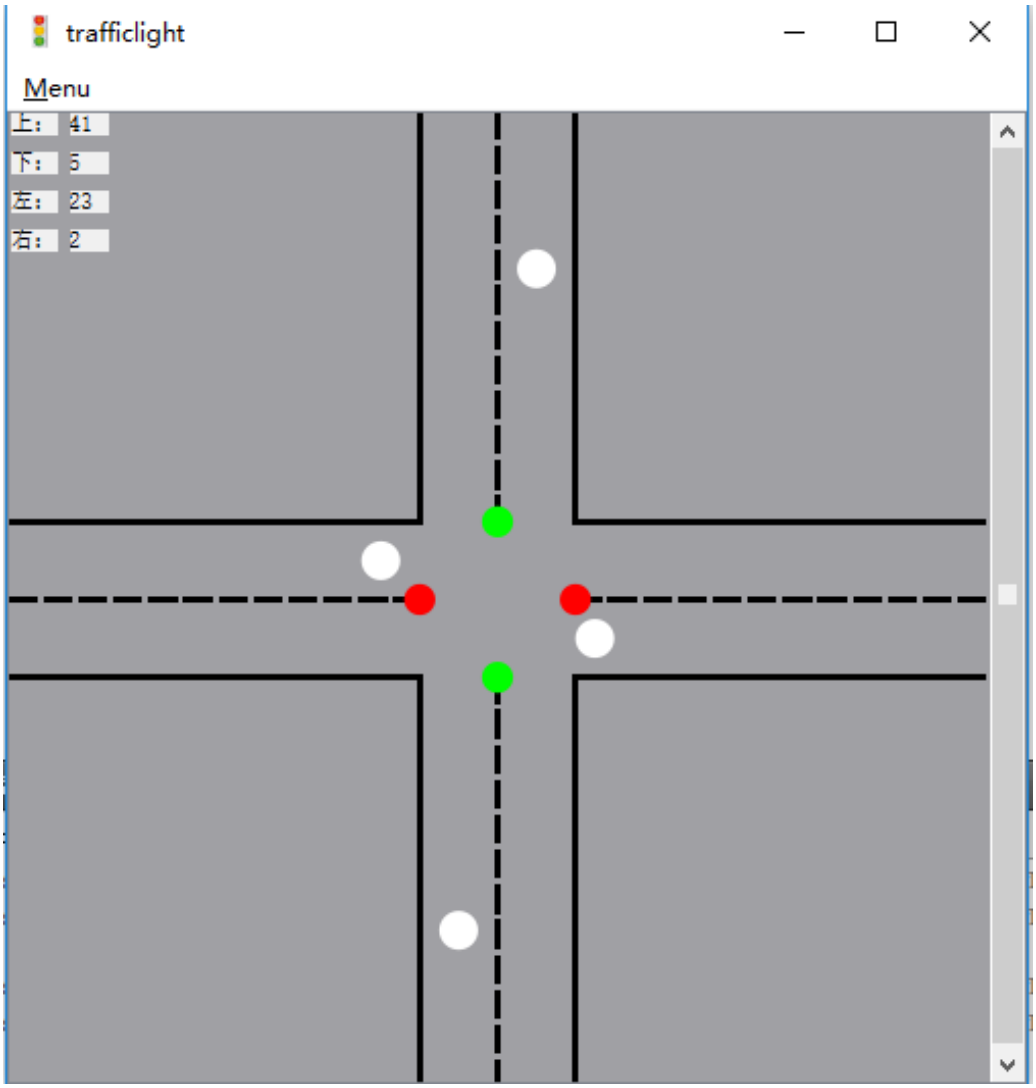

图 3 实验结果示意图

实验结果如上图。

左上角实时刷新上下左右四个方向的车辆数（用随机数生成）, 红绿灯的等待时长依据车辆数实时调整, 实验结果正确。考虑到车辆可能很多，由于屏幕较小绘制出很多车辆不太合适，此处一个方向只用一辆车代替该方向上的车辆运行情况。

六、实习心得

整个 C++ 软件实习还是很辛苦的：从熟悉使用 Qt 到一步步绘制出视图中的小车，再到编写小车运动、车辆数动态刷新的相关函数，每一步学习起来都相当麻烦。当然，通过这次软件实习也掌握了很多的知识，编程能力得到了提高，收获颇丰。

首先，当我拿到题目时，我肯定是一头雾水。因为从未使用 C++ 编写过图形用户界面，于是赶紧去学习 Qt，找到网上的教程学习了一波。然后开始构思界面，由于涉及到一个红绿灯的界面，我首先想到的就是先画出一个十字路口，这样主界面就设计出来了。于是开始设计红绿灯和小车，同时在小车类里编写了小车运动的相关函数。然后将小车和灯加到视图上，虽然小车是可以动了，但是题目要求设计一个实时动态的红绿灯系统，因此红绿灯的分配时间必须依据小车数的变化随时调整，于是我就编写了动态刷新函数，当然整个过程也是不断地出现各种 bug、各种调试。这样，主要功能基本实现。最后加了一个菜单（包括暂停、重新开始和退出），大功告成。

整个设计过程中，每一行代码都得仔细斟酌，尤其当出现了各种莫名其妙的 bug 时，总是需要不断地去学习、摸索，然后学以致用。不过最终还是完成了任务，还是非常开心的。总的来说，这门实习课还是相当不错的，感觉个人的代码实践能力得到很大提高。纸上得来终觉浅，绝知此事要躬行。编程还是的多实践！