长安大学

二〇一九届毕业设计

交通拥堵信息的传播与车辆规避仿真平台

学　　院：信息工程学院

专　　业：计算机科学与技术（卓越工程师）

姓　　名：刘发源

学　　号：201524020411

指导教师：安毅生 刘建书（校外）

完成时间：2019 年 6 月 11 日

二〇一九年六月

摘 要

道路发生交通异常后，如果处理不当会对道路通行带来很大的影响。交通异常发生后事件消息往往不能及时得到传播，车辆会源源不断地汇入事故路段进而造成堵塞，进一步导致外界（例如救护车和警察车辆的救援）无法进入事故地点实施救援。由此可见，交通事故消息的传播对于提高道路通行效率极其重要。车辆使用 V2V 来共享信息已成为一项研究热点并且技术日趋成熟。这使得车辆之间实现短距离无线通信的普及指日可待。因此，对道路中车辆之间的信息交互仿真必不可少。具体的研究内容如下：

1．规划可运行的道路，实现车辆在道路上自主的运行，实现自动寻路的功能。同时可以对拥堵路段做出停车等反应。

2．对于出现交通拥堵的路段，要在路口标识拥堵信息，将该信息传递给经过该路口的车辆，以免更多的车辆进入该路段；知道拥堵信息的车辆与驶出拥堵路段的车辆要将拥堵信息传递给不知道拥堵信息的车辆。

3．所有知道拥堵信息的车辆要在经过拥堵道路时，主动避开该拥堵道路，以免加重拥堵。

4．该仿真平台的交互界面要具有直观简洁了，易于操作，方便使用的特点，同时界面要始终保持和用户的沟通

关键字：智能车辆协调，V2V，交通拥堵，微观交通仿真。

ABSTRACT

If traffic is abnormal on the road, improper handling will have a great impact on road traffic. If the event message cannot be transmitted in time after the traffic anomaly occurs, the continuous flow of the vehicle into the road section will cause the road to be blocked, which further causes the outside world (such as the rescue of ambulances and police vehicles) to enter the accident site to carry out rescue, thereby increasing the congestion. It can be seen that the broadcasting of traffic accidents is important in improving the efficiency of road traffic. The use of V2V communication to share information has become a research hotspot and technology is becoming more mature. This makes it possible to realize the popularity of wireless communication between vehicles in a short distance. Therefore, it is essential to simulate the interactive simulation of information between vehicles on the road.Specific research contents are as follows:

1. Planning the runnable road, realizing the autonomous operation of vehicles on the road, and realizing the function of automatic routing. At the same time, it can respond to congestion by parking.
2. For the section where traffic congestion occurs, the traffic information should be marked at the intersection and transmitted to the vehicles passing through the intersection so as to avoid more vehicles entering the section; the vehicles who know the traffic information and those who drive out of the congestion section should transmit the traffic information to the vehicles who do not know the traffic congestion information.
3. All vehicles who know the congestion information should avoid the congested road actively when passing through the congested road to avoid aggravating the congestion.
4. The interactive interface of the simulation platform should be intuitive, concise, easy to operate and easy to use. At the same time, the interface should always maintain communication with users.

KEY WORDS: Intelligent Vehicle Coordination, V2V, Traffic Congestion, Micro Traffic Simulation.

目 录

1 绪论

1.1 选题背景

随着人民生活水平的不断提高，私家车已经步入了寻常百姓家，与此同时，每座城市的城市化进程也在不断的加快，道路拥堵成为了生活在城市里的每个不可以避免的问题[1]。城市拥堵不仅对于普通人来说是一种时间上的浪费，对于救护车、警车、消防车来说更是一种致命的延误，因交通事故或者意外事故处置不及时，从而造成人身伤害，财产损失[2]。因此，车辆拥堵不仅浪费了运输资源，并使运输效率降低，消耗社会成本会，更会使人产生烦躁焦虑、沮丧、愤怒等情绪从而诱发交通事故的产生[3]。

而现在普通的车载导航软件对于路况的反馈是基于出租车、私家车对于导航软件的使用的反馈，将数据上传到服务器后，再由服务器进行判别是否有堵车的产生，并将结果传输给用户。这种大数据的分析方法，虽然准确，对于离堵车地点距离较远的汽车来说是非常有用的[4]。但对于离堵车地点较近的车辆来说，这种有延时的反馈是非常不及时，等到车辆收到拥堵的信息后，可能汽车已经驶入的拥堵地段。因此使用 V2V 的方法进行信息的传递，让知道拥堵的车辆可以及时将拥堵信息传递给其他车辆，这种泛洪法的信息传播方式可以让距离堵车地点较近的车辆及时知道路面的情况从而避开事发路段，减轻事发路段拥堵的压力，同时配以现有地图导航对于路况的信息反馈，可以减轻拥堵地段的塞车压力[5]。

对于复杂的道路、交通环境通过重复的动画仿真，可以直接从交通流现象寻找影响通流的主要症结。通过仿真实验分析，对比多个优化方案，可以在建设项目实施之前寻找出最优方案，避免了个人主观经验的随意性。借助于交通仿真技术，通过良好的用户输入输出界面，软件的运算结果可方便地与用户交互，增强了软件的实用性和方便性．仿真结果的动画演示的直观性使得即使是非专业人员也能很容易理解[6]。

Java 作为现在主流的开发语言，其面向对象的方法，可以将车辆封装成对象，给予其初始化条件，同时在 Java 线程的帮助下，让汽车在可以自主的对路况产生判断，并做出相应的反应，这样更加容易实现系统的仿真，使得整个系统看起来更加的直观，代码更加容易通俗易懂。同时由于 Java 可移植的特性，使得 Java 可以跨平台运行，符合本次设计的要求。

1.2 国内外相关研究动态

随着计算机技术的不断发展与完善，计算机在交通与公路方面的仿真得到了不断地普及，交通仿真对于城市建设成本控制、预期效果等方面都起重要。依托交通仿真对现有系统或者未来规划的交通运行情况进行场景再现或者提前预知，从而对复杂的交通现象进行分析、评测、预估找出未来实施时可能会出现的问题，最终对研究的交通系统进行综合或者特定方面的优化。

在国外，交通仿真研究已经取得了有效并且更加成熟的工作成果，并研制了众多的交通仿真平台，其中许多平台已经实现产品化和商业化。国外的仿真软件从上世纪 60 年代的起步阶段发展到现在，已步入成熟。摈弃了传统 Fortran 语言或者专用仿真软件语言，开始逐步使用面向对象的开发语言，软件开发使用的途径也有了更加多元化的选择，开发方式更是多种多样，让人们有了更多的选择。同时因为计算机语言的发展，各种仿真平台的性能也得到了充足的提高。当前国外比较主流的交通仿真软件包括 Vissim、Paramics、TransModeler，尤其是 Vissim 在交通

模拟方面已经成为模拟软件的标准，其投入的深入研发力量和世界范围内的大批用户保证了VISSIM 在同类软件中处于领先地位[7]。

在国内，我国整体的交通行业起步比较晚，对于交通仿真工具的发展更是较为落后。我国交通仿真软件发展初期，只是针对较为简单的问题进行仿真与分析。在 2000 年初，我国交通行业内的专家开发了一套比较系统的道路交通环境模拟仿真系统[8]。近些年，我自主研发设计的城市交通需求分析系统软件 TraSluion 在我国交通建设方面做出了突出的贡献，也体现出 TraSluion 的优越性。东南大学也开发出了一套交通仿真软件交运之星 TranStar，该软件得到了我国公安部、建设部等相关部门的高度认可并在我国对这款软件做了相应的大力推广[9]。到了 2010 年之后，我国的互联网得到了飞速的发展，在交通仿真方面的软件越来越成熟与先进。但现有的导航软件的路况分析中，均是基于服务器端的信息收集，有一点的延迟对于将要驶入拥堵路段的汽车来说这种延迟是非常要命的。因此在这方面交通仿真软件的进入空间还是非常大的。

1.3 课题的研究目的与意义

随着生活水平的不断提高，私家车开始步入了寻常百姓家，因此路面状况也变得和我们息息相关。由于现有的地图软件的路况信息是基于软件服务器对与终端的实时反馈，这种反馈不仅反应时间比较长，并且对于服务器来说的因为要辨别路段是否出现堵车的情况，因此延迟会比较的大，对与靠近拥堵路段的车辆来说，获取前方的路况信息较迟，不以避开拥堵路段[10]。因此本次课题旨在模拟车辆在道路上的运行，做到车辆在一定距离内可以实时传递拥堵信息，以达到即使避开拥堵路段的目的。在现实交通环境中，一些方面需要更多的人力与财力的投入，一些领域还包含着许多不安全因素，这样寻求最佳方案的设想变得很渺小，更有可能无法实现。这样，运用计算机技术进行车辆以及交通的模拟成为了一种更加有效的手段。交通仿真平台时计算机技术在交通工程领域的一个重要的应用，它不仅可以将当前的交通状态数字化，为交通调度与疏导提供直接可靠的依据，而且还可以对各种参数进行比较与评估，以及车辆影响的评价等[11]。因此，交通仿真平台成为的交通工程方面测试和优化各种道路规划、设计方案、描述复杂道路交通现象的一种直观、方便、灵活、有效的交通分析工具[12]。

1.4 本系统的研究方法

本系统主要时通过 Java 语言分别模拟汽车拥堵信息的传递以及对于事发路段的规避。主要使用的方法为面向对象程序设计。面向对象程序设计(简称 OOP)是当今主流的程序范型。面向对象的程序是由对象组成的，每个对象包含对用户公开的特定功能部分和隐藏的实现部分。程序中的很多对象来自标准库，还有一些是自定义的。从根本上说，只要对象能够满足要求，就不必关心其功能的具体实现过程[13]。在 OOP 中，不必关系对象的具体实现，只要能够满足用户的需求即可。

交通拥堵信息传递的模拟主要是使用 Java 的类与类之间的信息传递。每一辆汽车用一个线程进行代替，并通过其他类给该线程一定的初始化条件，通过线程的执行方法使得可以模拟每辆车的运动以及对外界信息的处理，通过对 JPanel 的不断刷新来表示车辆移动的位置[14]。

车辆规避的模拟主要是通过对于当前位置的检测，是否将要抵达事发路段，通过这种检测以达到重新规划路线，规避拥堵区域，实现汽车顺畅的运行与减轻拥堵路段压力的目的[15]。

总的来说，本仿真系统使用的方法以 Java 界面为主要的呈现形式，以类与该类中的成员来模拟每一辆单独车辆以及这些车辆的行为，通过另一个类中的算法计算该车辆的形式路径，同时

与其他辅助类相结合，帮助车辆与道路的定义，最终实现整个交通拥堵信息传递与规避的仿真系统。

1.5 本文的主要工作

（1）道路规划与车辆行驶。通过 Java 语言的对于图形的绘制功能，在 JFrame 绘制出合适的道路，规划相应的路线，可以让车辆在随机的入口进入并在随机的出口驶离该路段。寻找并编写合适的算法，让车辆可以在规划好的线路上随机选择道路运行。并可以对拥堵路段做出一定的反应，当遇到拥堵时可以模仿现实情况中排起长队时的样子，进行排队等待。

（2）车辆拥堵信息传递。选择并编写合适的算法，当车辆已知拥堵路段后，通过信息的传递，可以将拥堵路段传递给与其在一定范围内的其他车辆，通过这种车辆与车辆之间的不断传递，使得整个全局路段内所有的车辆均知道事故的发生地点，当有新车驶入该路段后，也可以做到提前的知晓拥堵路段，已做好规避的准备，重新规划线路。当拥堵路段消失后，当有车辆接收到拥堵信号消失后，可以将该信息以同样的方式传递给其他车辆实现该路段的正常行驶。

（3）车辆拥堵路段规避。挡车辆已知事故发生路段后，首先要对堵车地段进行判别，路段为十字路口或者是直行路段，这两种路段可以分别采取不同的绕行方式，以达到绕开拥堵路线的目的。

（4）编写明确简洁的人机友好界面，给予用户充足的提示，可以让用户更加直观的使用该仿真平台，同时在道路中每辆车运行的情况要尽量贴近实际，做到对于车辆的真是仿真。

2 系统分析

2.1 需求分析

2.1.1 用户需求

本仿真平台在界面方面需要有简洁的操作界面，以最少的按钮完成所有基本的仿真平台的操作，过少的按键与操作可以增加系统的健壮性，能够正确的完成对现实情况的模拟与仿真，有利于问题的分析和解决，同时简洁的操作界面可以减少系统的复杂程度，提高用户的上手难度，节约时间，并有利于该仿真平台的推广。用户需求的用户交互按钮主要包括开始与结束按钮，添加与消除拥堵等按钮，可以根据世界情况增加或者减少按钮。同时至少要存在信息控制窗口，可以显示或者提示当前用户所做的动作。

选取的仿真场景尽量符合真实的情况，地图内车辆、道路的大小与数量选取要适度，车辆的数量过少不仅会让整个场景看起来比较空旷，而且对于实现本仿真平台的信息传递与拥堵规避的效果会更加不明显，完成不了仿真平台对于现实情况的仿真。车辆的数目也不宜过多，过多的车辆会导致整个系统看起来杂乱无章，不易于观察目标车辆，对于实际问题的分析与解决提供不了帮助。同时在拥堵路段要符合实际拥堵的场景的情况，将拥堵路段发生拥堵后可以用塞车的场景进行仿真，以较明显的特征提醒拥堵此地发生拥堵，或者发生过拥堵。同时拥堵消失后，也应以比较明显的方式提示用户此地拥堵已消失。这些信息的表现可以帮助用户对车辆状态的分析，便于交通拥堵问题的解决。

对于主要功能——拥堵信息的传递与规避尽量做到细节处理，比如交互前与交互后每辆车的状态变化，能够做到用户可以直观的观察出该变化。对于拥堵路段的规避也要尽可能的有所体现，这种体现不仅可以在要进行规避的车辆上进行体现，也可以使用信息窗的方式，以文字直接显示给用户。给予用户多种获取信息的途径。同时不仅要体现在拥堵发生时，知道拥堵的车辆与不知道拥堵的车辆之间的交互；也需体现在拥堵路段消失之后，知道拥堵消失的车辆与不知道拥堵消失信息车辆之间的交互，这些都要通过图像与文字表现出来[16]。

内部代码的编写要尽量符合规范，重要的函数方法与接口要有明确的注释说明，类与类之间尽量设计为松耦合，减少全局变量的使用，每个类尽量减少对于全局变量的引用，引用时要有详细的注释声明。同时变量的起名方式也要合乎规范，不能随意起名，起无意义的变量名，每个变量名要有实际意义，不断优化无意义的变量名，方便程序在今后的维护与升级。

2.1.2 系统需求

该仿真平台主要 Java 语言进行编写，Java 语言有丰富的 UI 界面可以满足对该仿真平台对于界面的要求，同时 Java 作为面向对象的语言，采取每个类单独封装的方式，可以更加简单与容易的编写该仿真平台，同时也有利于项目的后期修改与维护。从下面三个方面对该仿真系统进行需求分析。

（1）功能需求

功能需求是有关软件系统的最基本的需求表述，用于说明系统应该做什么[17]。该仿真系统主要是实现对交通拥堵信息的传递和对该拥堵路段的规避的仿真。因此需要对该仿真平台要有真实的地图或者路况信息，并可以在地图上添加拥堵，可以让经过这个路段的车辆携带该信息运行，

同时遇到其他车辆时可以实现信息的传递，让更多车辆知道该信息的存在。对于拥堵路段的规避要在前面功能的实现基础上进行实现，根据实际的路况，选择不同点的规避方式绕过拥堵路段，实现顺畅的运行。

要实现上述功能，主要的两个类为寻路类和车辆运行类，也可以将车辆运行类看车车辆类，车辆类要预留出与寻路类进行交互的接口，寻路类初始化参数为起始、终止、不可到达区域这三个参数，同时寻路类中内置寻路方法，可以在以上三个条件均初始化完成后，自动找到可以通行的路径，该方法的返回值为该路径。车辆类根据寻路类的路径传递，可以实现自主的运行。对于车联如何得知拥堵路段的地段，可以通过将拥堵路段的地段设置为全局变量的方法，虽然使用全局变量破坏了类的封装，但是这种方法可以简化系统所需的空间，减少系统的出错概率，更加容易实现该仿真平台。

仿真平台内要存在一定的异常处理的方法，如果需要进行异常处理，比如用户在执行程序时出现错误，则需有异常处理给予用户一定的提示，帮助用户可以使该软件恢复正常。

（2）数据需求

该仿真平台只是对车辆中的一部分进行仿真与模拟，同时只需要模拟一段道路的对拥堵信息的仿真，因此车辆的信息与道路的信息量不会大，用不到数据库对数据的存储。对于各个信息的存储完全可以使用一个类，在该类中编写静态方法，实现对一些在整个程序运行过程中不会改变的数据的存储。仿真平台主要是通过按键来实现对整个系统的控制，因此，初始化时，无需进行输入数据。给车辆类的数据可以是坐标集合，也可以是道路的集合。对于寻路类的输入数据可以是坐标类的集合，也可以是输入一个 Java API 中的面板参数。

（3）其他需求

为保证界面的整洁性，因此可以选择将按钮设置为小方格图片的样子，鼠标悬浮后可以提示该按钮能够实现什么功能，地图与运行区域尽量宽广，同时变现形式简洁，易于用户直接抓住主要对象进行问题的分析。同时也需要信息交互窗口，可以完成对各个车辆信息的观察。

2.2 可行性分析

当我们研究一个项目时，首先应该解释我们的目标和任务，然后我们应该仔细研究该项目，并将开发阶段进入项目开发的第一阶段。 我们从以下几个部分进行正确性的研究。

2.2.1 技术可行性分析

面向对象具有维护简单、质量高、效率高、可扩充性等特点，其实质是以建立模型体现出来的抽象思维过程和面向对象的方法。模型是用来反映现实世界中事物特征的。任何一个模型都不可能反映客观事物的一切具体特征，只能对事物特征和变化规律的一种抽象，且在它所涉及的范围内更普遍、更集中、更深刻地描述客体的特征[18]。通过建立模型而达到的抽象是人们对客体认识的深化。

同时 Java 语言是纯面向对象的语言，并且提供了多种内置的 API，可以帮助我们更好的进行面向对象的开发，尤其是其跨平台的特点，可以让仿真平台在多种计算机平台上运行，增加了平台的兼容性。

Java 语言提供了多种 UI 界面与 UI 风格

因此将车辆运用面相对象的开发，对于开发来说更加容易维护，同时采取 Java 线程进行开发，可以更加突出每辆小车单独运行的效果，并提高开发的效率[19]。

2.2.2 运行可行性分析

随着生活水平的提高，现在的私家车也变得越来越多了起来，同时对于路况的反应需求也更加的严格，而信息技术和计算机网络技术的快速发展为这种对于路况的实时反馈提供了条件。车辆拥堵信息的传递与地图导航软件对于路况分析的相结合，可以让广大车主更加及时、准确的获取自己所需要的路况信息[20]。这种准确性对于所有的车主是急切所需的。

2.3 系统功能分析

本仿真平台主要模拟车辆在道路上行驶的情况下，车辆与车辆之间的传递与对于堵车路段的规避，同时点击车辆后可以查看车辆进入整个街区的起始位置和驶出街区的位置，并查看该车辆是否知道事故发生的地点。并且用户可以自己手动添加车辆，添加完成后可以实现车辆的自行寻走。系统总体功能结构图如图 2.1 所示。

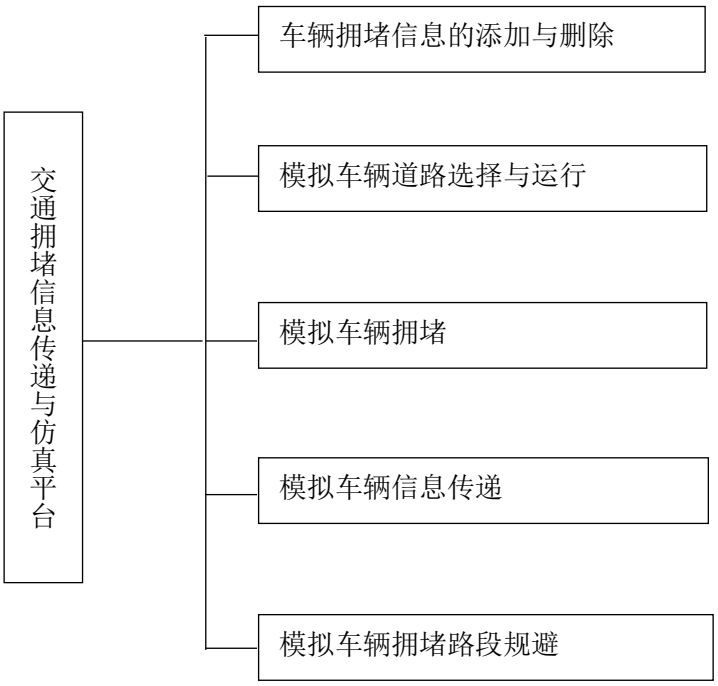

图 2.1 系统总体结构

2.3.1 车辆拥堵信息的添加与删除功能分析

点击按钮添加拥堵信息后，在拥堵路段行驶的车辆会会自动停下，排队等待通过堵车发生路段，同时设置全局堵车标识位为 true，可以让所有的汽车均可以知道堵车是否发生。

点击按钮后将堵车路段取消，每辆车都可以顺利通过堵车路段，并给携带拥堵信息的车辆发送拥堵已消失的信号，是的本区域内每辆车都更新自己所携带的信息。

2.3.2 模拟车辆道路选择与运动功能分析

模拟车辆的运行通过在窗口中绘制坐标信息，通过坐标来定位车辆的位置，并以此作为小车运动的依据。

模拟车辆道路选择是针对现有的车辆起始和驶出位置，通过自动寻路算法，自主的选择合适的路线，生成的线路为一组连续的坐标，将路线传递给“车辆”类，不断刷新汽车在坐标系中的位置以达到运行的效果。

2.3.3 模拟车辆拥堵状态的功能分析

车辆在行驶的过程中遇到堵车时，可以像现实状态一样，在仿真平台中让车辆排起长队，等待一定的时间后，待拥堵状态清楚后，该堵车路段的车辆可以离
开该区域正常行驶。

2.3.4 模拟车辆信息传递功能分析

当车辆遇到堵车的状况时，该车辆马上将信息传递给路口的塞车信号接收装置，每辆经过路口的汽车均可以收到来自该路口传递过来的塞车信息，接收到信息的车辆可以将此路段已堵车的信息传递给距离本车一定范围内的车辆。所有知道堵车信息的车辆在遇到其他不知道堵车信息的车辆时，自动将这条信息发送给对方，逐渐使整个区域内所有的车辆均知道拥堵发生的路段，为避开该路段做好准备。

当该路段的拥堵消失后，堵在塞车路段的车辆在逐渐将道路畅通的信息传递给其他获取到拥堵路段信息的车辆，逐步使整个路段恢复正常。

2.3.5 模拟车辆规避拥堵路段功能分析

如果车辆在已知拥堵信息的情况下，通过检测自己行驶的路段中是否包括堵车路段，如果包括堵车路段，则重新规划街道的选择，以达到绕开堵车路段的目的。

3 系统设计

3.1 系统设计目的与原则

交通拥堵信息传递与规避仿真平台主要目的是可以呈现实际交通环境的特征、分析交通环境在拥堵情况下对于信息的传递，和车辆对于拥堵路段的规避，尽可能模拟交通拥堵的真实情况和汽车在道路上运行的真实情况[21]。同时在系统中增加信息提示窗口，可以让用户看到每辆车的动态，和自己所做一系列操作的提示。自定义按键可以使用户自由暂停和继续本系统的仿真。因此本仿真平台重点实现的是对道路拥堵与信息传递进行仿真，以实现对于现实情况中如何减轻拥堵压力的最优解。依托 Java 的语言的面向对象的设计，可以合适的完成对这套系统的模拟，在使用 Java 面向对象设计时，应精确把握面向对象的概念，把握应当设计的每个对象的功能与意义，做好类与类之间松耦合，尽量减少全局变量的使用，增加每个类之间的独立性[22]。同时对每个类内部的尽可能的进行保护，在没有访问特定的方式时，减少对于本类中变量的访问。设计的每个类尽可能实现功能的独立，不要出现交叉功能的现象。

交互界面设计的友好、简洁、易懂，基于用户尽可能多的提示，防止用户的操作不当而造成的仿真结果的错误，达不到用户预期的要求。

3.2 面向对象的设计

3.2.1 面向对象的程序设计思想

面向对象有以下的几种特点，同时也是面向对象设计的关键所在[23]：

（1）封装。所谓的封装就是一种信息隐藏的技术，就是把属性进行私有化，简单点来说就是把对象的行为和属性看作是一个密不可分的整体，或者就是指把不需要把外界知道的信息进行隐藏。

（2）继承。继承就是子类继承父类的过程。举个简单的例子就是有一个 Animal 类和一个 Dog 类，显而易见，Animal 是一个父类，Dog 类该父类的子类，父类处于上层，子类在父类的下层可以扩展出来一些特有的属性或者方法。一般在类中，一个子类只允许继承一个父类，即单继承。Java 通过接口来弥补其不能实现多继承而带来的子类不能共享多个父类的缺点。

（3）多态。所谓的多态就是“一种定义，多次实现”，多态允许程序中出现重名的现象，当父类的某个方法被子类重写后，将不同的子类对象赋值给父类对象的引用，可以产生不同的功能。Java 语言中含有方法重载与对象多态两种形式的多态：

方法重载：在一个类中允许多个方法使用同一个名字，但方法的参数不同，完成的功能也不同。

子类对象可以和父类对象实现交互功能，根据使用的子类不同，完成的功能也不同。

（4）抽象。抽象就是把现实中的事物抽象出共同的、本质的属性的过程。抽象过程就是比较的过程，通过比较找出事物之间的共同属性，通过比较区分本质。

3.2.2 面向对象的类的定义

相对于更加复杂的交通仿真软件来说，由于主要实现的是对交通拥堵与信息穿的模拟，所以在本仿真平台中，需要进行定义的实体比较少包括汽车、道路、十字路口的检测信号、行驶路

线、地图模块、堵车点、坐标点等，通过分析上述各个实体类的关系，可以将这些现实情况分别抽象成汽车类（Move）、道路类（RoadDataBase）、十字路口类（cross）、路线类（AutoFindWay）、事故类（Acc_Panel）、坐标类（FangKuaiPosition）和地图（BasePanelGround）。然后根据实现类的实际需求和他们之间的相互联系，对每个的成员变量和方法进行定义。

（1）汽车类（Move）。汽车类主要实现，汽车的移动、信息传递、交通拥堵规避的功能。确保汽车类的正常运行是保证整个仿真平台能否正常运行的重要保障，同时对于对汽车的类的访问也是获取仿真平台实时动态的重要依据。定义的汽车类中的成员变量要包括车辆编号、路线、事故信号、汽车速度等。运行的方法为 run()方法，通过比较当前汽车位置和下一个位置的 x、y 坐标的大小，修改汽车在画面中的坐标位置，以达到汽车的移动。同时在移动过程中不断检索与其邻近的其他汽车的位置和临近汽车是否知道拥堵路段的信息，以实现对自身事故信号的修改。移动过程中也不断检索自己是否到达了十字路口，并接收十字路口传递的信息，查看四周的道路是否出现拥堵。上述两种对于信息的传递可以做到对拥堵路段的避让，避免驶入拥堵路段。

若汽车在不知道拥堵路段时，如果驶入拥堵路段，则排队等待通过事故路段，或者等到拥堵结束再通过该路段。

（2）道路类（RoadDataBase）。由坐标类 FangKuaiPosition 组成，每条道路均是由多个左边点构成，最终所有的坐标点构成公共静态 List 数组。每辆车只能在生成的 List 数组中行驶。

（3）事故类（Acc_Panel）。事故类由坐标点 x、y 构成，以及事故路段的背景图片 Icon 构成，最终事故类实例化的对象均为 public static 的访问限权。让所有的车辆类均可得知事故地点的出现。

（4）十字路口类（Cross）。十字路口类由该路口的坐标（FangKuaiPosition）、是否知道事故发生路段（ifHavaAcc）以及发生在哪个方向上（east、west、south、north）这些成员变量构成，同时分别有设置、获取这些数值的方法构成。

（5）坐标类（FangKuaiPosition）。由于在背景之下的坐标地图由一个个的方块构成，因此每一个方块即为一个坐标类（FangKuaiPosition），坐标类由这个方块在 Frame 上的为像素点 x、y，以及方块长与宽构成。

（6）地图类（BackGroundPanel）。地图类均由一个个的坐标类（FangKuaiPosition）构成。

（7）寻路类（AutoFindWay）。寻路类中包括两个方法：getWayList()方法，用来生成汽车要行驶的路径的坐标信息，将此坐标信息最终整合成为一个集合，返回值为此集合；aroundFK 方法，此方法在道路坐标集生成的过程中，用来探测当前坐标周围是否存在存在可以到达的位置。

3.3 系统功能模块设计

3.3.1 车辆拥堵信息的添加与删除设计

设置全局标志变量 Acc，默认值为 false，及未有堵车的发生。当点击按钮后在某一路段添加拥堵，Acc 变为 true，并在道路上出现堵车的 Panel 图层用来显示事故的发生，同时将进入该路段的两端十字路口的堵车标志位也设置为 true。JPanel 的图形也为全局变量，易于汽车可以在运行的过程中不断检测自己是否到了堵车路段，做出堵车排队的动作。同时检测自己是否到达

十字路口并对十字路口的堵车标识位进行检索，察看该路段是否堵车，为该路段的规避做好准备。

设置标识为 ,默认值为 true，当拥堵路段取消后，将该标识设置为 true,并告知在堵车路段等待的车辆，在堵车路段等待的车辆驶出事发路段时，将该地区拥堵与消失的信号传递给其他汽车，同时也将堵车路段十字路口的堵车标识位更改为 false。具体功能设计如图 3.1 所示。

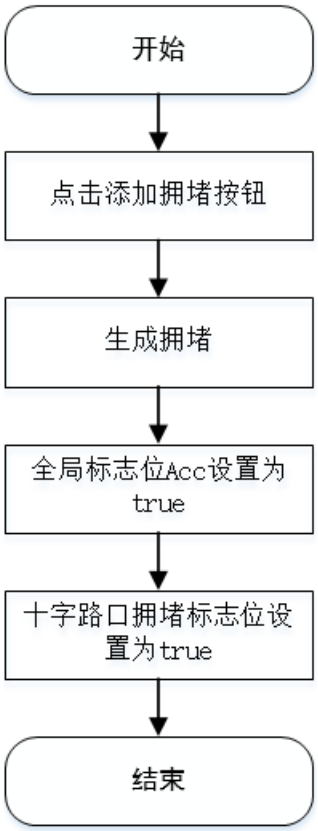

图 3.1 车辆拥堵信息的添加与删除设计

3.3.2 模拟车辆道路选择与运行设计

模拟车辆道路选择设计。为了突出演示的效果，选用随机道路选择的算法，使车辆在可运行的街区中停留更多的时间。使得车辆之间的交互效果更加的明显。

随机道路选择的算法。根据图 3.2 所示，从起始坐标（红色方块）开始遍历周围的坐标，若此点坐标处在非十字路口时，只有前进与后退两个方向上的坐标可以选择，将来时的已经走过的道路加入到 closedList 集合中，每走过一个坐标，需要查找新的坐标时，不会检测 closedList 集中的方块，因此保证随机道路的选择不会出现车辆来回进退的情况出现，只需要将前方的坐标加入 wayList 集即可，如果周围存在多条可以选择的方向，则随机选择的某一方向的坐标加入 wayList 集中,并且该方向上第一方块没有再 closedList 集合当中。为了保证车辆在当前的街区有充足的运行时间，在汽车到达路口时，标注自己到达与出口最近的十字路口的次数，只有当次数到达一定次数时，汽车才驶入出口路段，最终离开本街区。此时得到的所有的 wayyList 集为从起点到终点的线路。

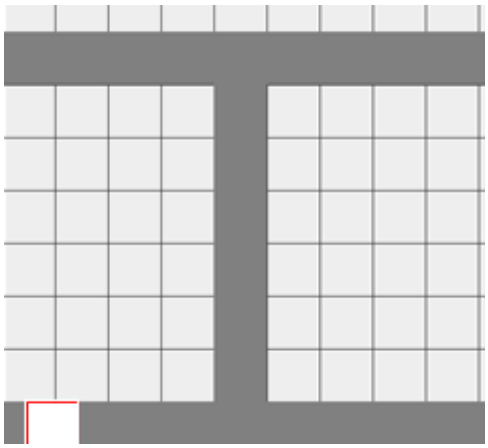

图 3.2 随机道路选择

模拟车辆的运行设计。由于每辆车在平面的运动轨迹已经由寻路算法所找到，所以每辆车只需要根据已生成的线路进行运动即可。每条线路由一个个的坐标点组成，不断的查看汽车本身所在的平面位置与下一个前后左右的坐标关系，以此控制汽车向四周的移动。此时的运行流程图如图 3.3 所示.

3.3.3 模拟车辆信息传递设计

模拟车辆的信息传递主要是查看本车辆与其他车辆之间的位置关系，如果两辆汽车之间的距离小于或等于一个坐标点时，检测自身是否知道车祸地点，并查看将要与其交互的车辆是否也知道车祸信息，同理拥堵已消失的路段的信号传递也以此方法进行。

设置全局位置信息变量 position 数组，position 中的数组成员均为坐标类，用来记录每一辆汽车当前的位置。每辆车移动一个坐标点后，检索 position 数组，看是否有其他车辆与其邻近一个坐标位。同时查看这辆车是否知道车祸已经发生及 knowAcc 的状态值，若知道及 knowAcc = true，则改变车的状态，将自身的 knowAcc 也设置为 true。

当车辆到达十字路口时，检测此十字路口的路段的拥堵位是否已经设置为 true，如果此路段已经拥堵，则也将此车的 knowAcc 也设置为 true，并避开这个拥堵路段。过程如图 3.4 所示。

3.3.4 模拟车辆拥堵路段规避功能设计

一般情况下，当车辆遇到堵车路段时，主要采取的手段为绕过该拥堵路段，绕过该路段之后，回到原始路线继续行走。

当车辆即将抵达拥堵路段之后，查看自己的线路，看自己将要往哪个方向行走如果将要行走的方向为拥堵路段，则查看其他方向是否可以绕过该拥堵路段。

以如图 3.5 所示的示意图，汽车模型 A 运行到十字路口 B 时，路段 L 出现拥堵，而汽车模型 A 需要通过路段 L 到达点 C，因此可以查看其它方向是否可以绕过该路段，则汽车模型 A 可以先向南行走 7 个坐标单位，再向东行走 7 个坐标单位，最后向北行走 7 个坐标单位，到达目的点 C，同时更新自己的路段信息，原本的下一个在 wayList 中要比较的坐标单位为 D 点，绕行之后将下一个将要比较的坐标单位设置为 E 点。此时完成对于拥堵路段的规避。流程图如 3.6 所示。

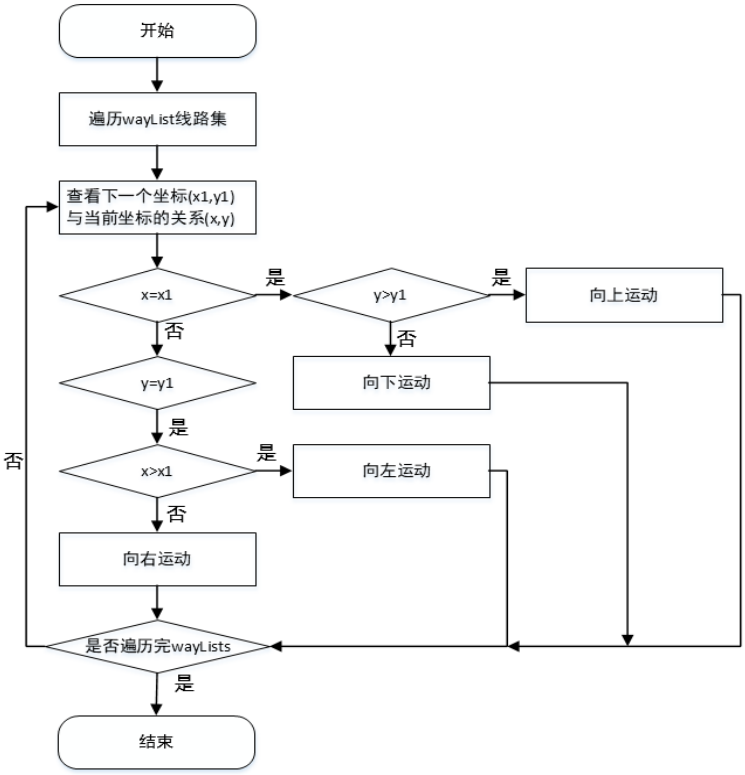

图 3.3 车辆道路选择与运行设计

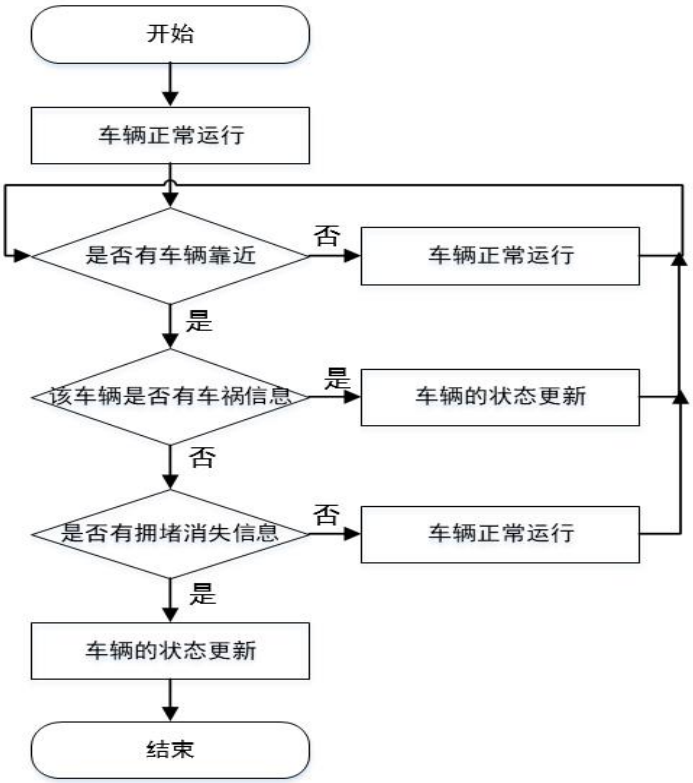

图 3.4 车辆信息传递设计

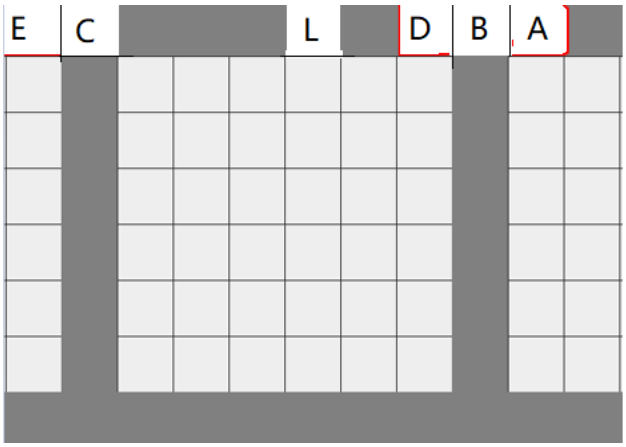

图 3.5 汽车绕行示意图

3.3.5 模拟车辆拥堵样式功能设计

在现实情况中，当出现堵车时，该堵车路段会排起长龙车队等待缓慢的通过，因此该仿真平台，需要模拟汽车因拥堵所造成的长龙现象。

在设计初始拥堵路段时，会设置拥堵地区的大小范围，及事故面板区域 Acc 的大小。当拥堵路段发生在南北区域时，向南行驶的小车接近拥堵地点后，Acc 事故面板区域的起始坐标要以新到达的车辆的停下后所在的坐标为起始点，同时 Acc 事故面板区域的南北长度增加；向北行驶的车辆抵达事故路段时，Acc 事故面板区域的起始坐标不变，南北长度增加。当拥堵路段发生在东西路段时，向西行走的车辆到达拥堵发生地点时，Acc 事故面板的起始坐标不会发生改变，东西长度增加；向东行驶的车辆到达拥堵路段发生地点时，Acc 事故面板的起始坐标为新到达车辆的坐标，东西长度增加。所有在拥堵路段的车辆在离开事故露路段后要将 Acc 事故面板的起始坐标和长度恢复到原来的位置与方向。流程如图 3.7 所示。

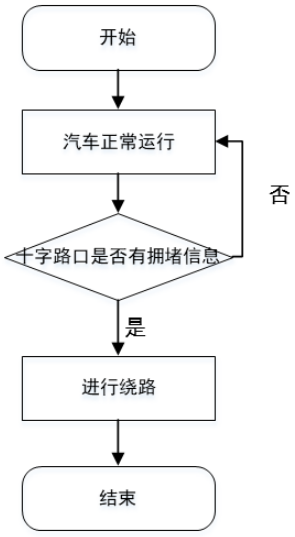

图 3.6 车辆拥堵路段规避功能设计

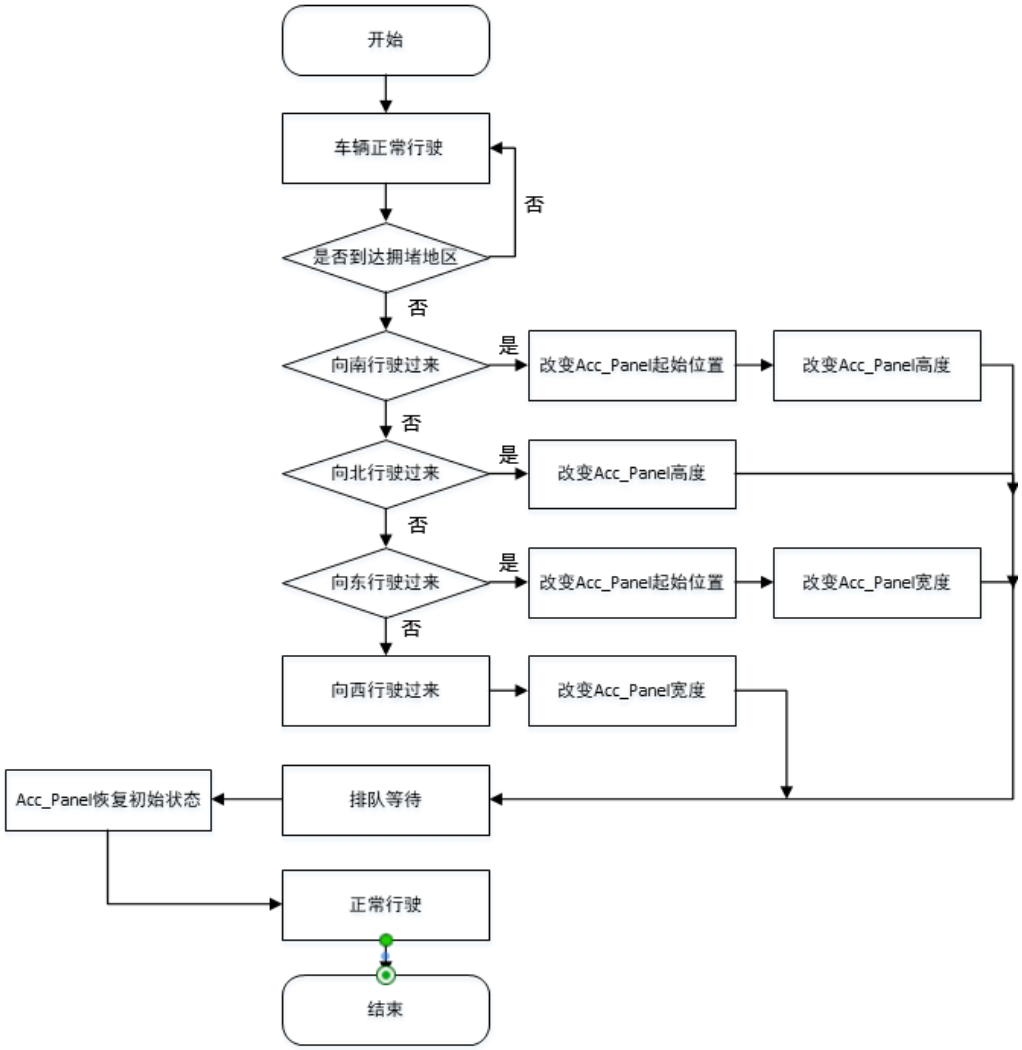

图 3.7 模拟车辆拥堵样式功能设计

4 系统的实现

4.1 系统实现技术

4.1.1 界面的实现

Java.awt 控件：Java 的软件包之一，包含用于创建用户界面和绘制图形图像的所有分类。在 AWT 术语中，诸如按钮或滚动条之类的用户界面对象称为组件。Component 类是所有 AWT 组件的根。当用户与组件交互时，一些组件会激发事件。AWTEvent 类及其子类用于表示 AWT 组件能够激发的事件。有关 AWT 事件模型的描述，请参见 AWTEvent。容器是一个可以包含组件和其他容器的组件。容器还可以具有布局管理器，用来控制容器中组件的可视化布局。AWT 包带有几个布局管理器类和一个接口，此接口可用于构建自己的布局管理器[25]。

Javax.swing 控件。Swing 允许选择程序的图形界面风格常用的有 java 风格，windows 风格等。顶级 Swing 容器为其它 Swing 组件在屏幕上的绘制和处理事件提供支持，常用的顶级容器：

JFrame（框架）：表示主程序窗口。

JDialog（对话框）：每个 JDialog 对象表示一个对话框，对话框属于二级窗口。

JApplet（小程序）：在浏览器内显示一个小程序界面。

一个框架包括边界、菜单栏、工具栏、状态栏，以及中间占主要部分的窗格，窗格也可以看作是一种面板，但它是框架的一个组成部分。组件不会直接放到框架上，而是放在若干个面板上，这些面板再放到窗格上。用框架对象的 getContentPane()函数来获得窗格，再调用窗格的 add()函数放置面板。Swing 用纯 Java 写成，所以同 Java 本身一样可以快平台运行。

4.1.2 控制程序的实现

面向对象的程序设计：面向对象的思想是通过软件模拟实际的对象。面向对象这种方法是根据现实世界的对象进行软件开发的。并逐步成为现在开发的主流模式。因此可以将汽车、道路全部抽象成为类，使得交通仿真更加的容易理解，对与代码更加容易理解，并且为今后对该平台的维护与扩充也变的更加容易与轻松。同时，面向对象的编程可以有效和实际地对交通拥堵和交通流量等各种现象进行仿真，深入分析车辆、道路以及交通流的交通特征，有效的进行交通道路的选择、拥堵路段规避等方面的研究。

4.2 面向对象类的实现

（1） 坐标类（FangKuaiPosition）的实现

坐标类主要是实现对于像素位置的封装，和当前位置的获取与设置。构造函数为当构造函数的参数为 MyPanel 类或者 Acc_Panel 时，Panel 的坐标除以我们 Panel 的大小，即为在我们绘制的地图中的坐标位置。主要的成员变量有 x(int)——横轴坐标、y（int）——纵轴坐标；以及成员函数对于横纵坐标的获取。

（2） 汽车类(Move)的实现

汽车类（Move）要继承自 Thread（线程类），每个线程在传递完参数之后，可以实现汽车类的自由运行，同时实现 Thread 的 run( )方法。

汽车类要实现车辆的识别、定位、运动、规避、信息传递、速度的控制等工作，因此需要的成员变量有 MyPanel（继承自 JPanel）——实现车辆在界面上的显示、num（int）——当前车辆的识别号、wayList（List 数组）——车辆预先设定的路线、knowAcc（boolean）——是否知道车祸、knowSmooth（boolean）——车辆是否知道道路顺畅、count（int）——交互次数。

成员函数要分别实现上述成员变量的 set（设置）、get（获取方法）以及线程的 run( )方法的重写。Run( )方法内为一个 while(true)的不断循环，检测汽车当前位置与 List 数组中下一个位置的位置关系控制汽车的启动。同时在 while 循环中，不断检测是否发生车祸，BasePanel.acc == true

BasePanel.acc 中的 acc 为主类中的 static 全局变量，用来使所有的类均知道车祸的是否发生。同时在 Windows 类（界面类）中的 Acc_Panel 也为 static 全局变量，用来告知当前界面内所有车辆事故发生地区。

部分代码：

```
While（true）{
    for(int i=0;i<wayList.size;i++){
        Move();
        Detection(){
    if ((BasePanel.acc&&fk.getY()*50==Windows.accp.getY()&&fk.getX()== Windows.accp.getX())
    //detetion code write here
        }
    }
}
```

（3）道路类（roadDataBase）的实现

道路类的实现主要是将特定的坐标（FangKuaiPosition）输入到 List 数组中，分别利用循环函数实现对于横向和纵向路段的加入

（4） 十字路口类（Cross）的实现

十字路口类（Cross）类中，构造函数为对十字路口位置坐标的封装和车祸信息的初始化。主要的成员变量为（x，y）——十字路口位置、position（int）——车祸位置、ifHavaAcc（boolean）——车祸信号。成员函数主要为对于上述成员变量的 set(设置)与 get（获取）。

（5）路线类（AutoFindWay）的实现

路线类为随机算法所生成的路线。

随机道路选择方法所要调用的成员变量主要包括 cat（MyPanel）——起始位置、fish（MyPanel）——终点、closedList(List 数组)——存放已经走过的方块、wayList(List 数组)——存放生成的线路。主要的成员函数为 public List<FangKuaiPosition> getWayLine(MyPanel cat){}，其方法主要实现对路线的生成，方法为：①选取起始位置方块，②调用 aroudFk( )方法将起始位置的方块的周围方块加入到 closedList 集中，③利用随机函数，随机选取一个可以行走的方向，将下一个方向的坐标加入到 wayList 中，④选取 closedList 中的最后一个坐标，重复步骤②。arounFk(){}方法，主要看一个坐标的周围坐标有哪些道路可以选择，周围坐标不能为不可到达坐标，且此坐标不能出现在 closedList 集中，并在 arounFk 方法中计算每个符合要求的坐标，返回 List 数组供 getWayLine（）方法使用。

aroudFk( )代码如下所示：

```
public List<FangKuaiPosition> aroundFk(FangKuaiPosition fk) {
        if (fk.getX() == 10 && fk.getY() == 11) {
        }
        List<FangKuaiPosition> list = new ArrayList<FangKuaiPosition>();
        //判断上面的方块是否符合条件
        //判断是否超过越边界
        if (fk.getY() - 1 >= 0) {
            FangKuaiPosition tmpFk = new FangKuaiPosition(fk.getX(), fk.getY() - 1, fk);
            //判断是否是障碍物/已计算过的方块
            if (!BasePanel.zhangaiList.contains(tmpFk)
                    && !BasePanel.closedList.contains(tmpFk)) {
                list.add(tmpFk);
            }
        }
        //判断下面的方块是否符合条件
        if (fk.getY() + 1 < BasePanel.heightLength) {
            FangKuaiPosition tmpFk = new FangKuaiPosition(fk.getX(), fk.getY() + 1, fk);
            if (!BasePanel.zhangaiList.contains(tmpFk)
                    && !BasePanel.closedList.contains(tmpFk)) {
                list.add(tmpFk);
            }
        }
        //判断左面的方块是否符合条件
        if (fk.getX() - 1 >= 0) {
            FangKuaiPosition tmpFk = new FangKuaiPosition(fk.getX() - 1, fk.getY(), fk);
            if (!BasePanel.zhangaiList.contains(tmpFk)
                    && !BasePanel.closedList.contains(tmpFk)) {
                list.add(tmpFk);
            }
        }
        //判断右面的方块是否符合条件
        if (fk.getX() + 1 < BasePanel.widthLength) {
            FangKuaiPosition tmpFk = new FangKuaiPosition(fk.getX() + 1, fk.getY(), fk);
            if (!BasePanel.zhangaiList.contains(tmpFk)
                    && !BasePanel.closedList.contains(tmpFk)) {
                list.add(tmpFk);
            }
        }
        //将中心方块添加到已处理过的集合中
        BasePanel.closedList.add(fk);
        return list;
}
```

可以看出 aroudFk( )方法，不断对要处理的方块的上下左右进行检测，查看要处理的方块是否越界或者是已经处理过的方块，如果这些方块均符合下一步要处理的条件，用于后面比较选择哪个方块才是到达终点的最佳路线。同时计算过上述三个值得方块要加入 closedList 集合，避免重复处理。

4.3 核心模块功能实现

4.3.1 地图信息模块的实现

地图的创建对于后面车辆的定位运行，路径的选择都起到了至关重要的决定，地图信息不仅规定了道路的样式，哪些地方可以走，哪些地方不可以走，而且也要定义坐标信息，坐标信息的定义对于实现车辆的定位、运行、信息的传递都起到了至关重要的作用。

第一步需要实现的是对地图底层坐标的实现。如图 4.1 所示，整个地图的底层由 30 行 18 列的方格组成，每一个方格代表一个坐标单元。在后面的程序中，所有车辆的运行就依靠这些方格坐标进行定位。同时，在底层地图中标记道路信息，在图 4.1 中灰色的部分即为道路的坐标。规定每辆在该区域运行的车辆，只能在灰色地段运行，周围的白色地区为不可到达区域。坐标的绘制主要是使用 Java 语言中 java.awt.Graphics 包中的 drawLine 函数画出图中所有的直线。道路的生成主要是在 List 数组集中加入符合要求的坐标（方块）。并通过用坐标初始化的方式，将路径显示在底层地图上。

最后，在底层地图 BackGroundPanel 上利用 awt 的 drawImage 函数为，底层地图附上实际地图图片，如图 4.2 所示。

部分代码如下所示

路段生成的部分代码：

```
public static void getWay(){
     for(int i = 4; i <= 25; i++){
         way.add(new FangKuaiPosition(i, 1));
         way.add(new FangKuaiPosition(i, 8));
         way.add(new FangKuaiPosition(i, 15));
     }
     for(int i = 2; i <= 8; i++){
        way.add(new FangKuaiPosition(4,i));
        way.add(new FangKuaiPosition(11,i));
         way.add(new FangKuaiPosition(18,i));
         way.add(new FangKuaiPosition(25,i));
         way.add(new FangKuaiPosition(4,i+7));
         way.add(new FangKuaiPosition(11,i+7));
         way.add(new FangKuaiPosition(18,i+7));
         way.add(new FangKuaiPosition(25,i+7));
      }
  }
```

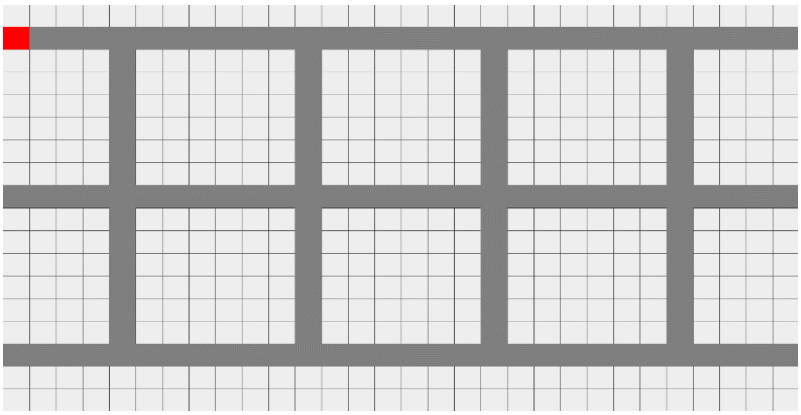

图 4.1 底层地图

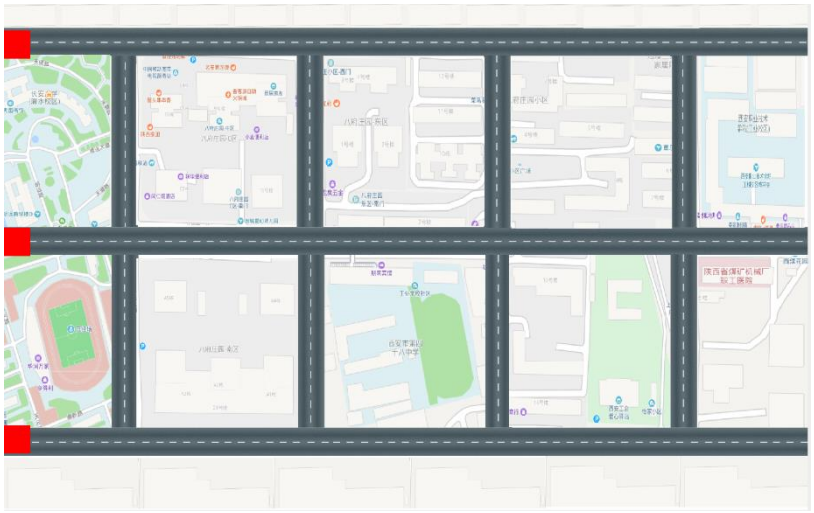

图 4.2 实际效果图

4.3.2 模拟车辆道路选择与运行模块的实现

车辆的道路选择主要是通过 AutoFindWay 类实现的，给 AutoFindWay 起始和初始位置后，在选择路线模式后，AutoFindWay 类中的 getWayList 方法会自动返回路径的集合，在 Move 类中的线程运行的 run( )方法中比较当前位置与下一个位置的东西南北方向关系，不断更新自己的位置。

部分代码如下所示：此处代码为向北运行的代码，东西方向上的运动只要比较 X 轴上的坐标即可，向南行驶的只要 cat（当前位置）比 fk（下一个位置）的纵坐标小即可，增加 cat 的 Y 坐标即可。

```
while (cat.getY() > fk.getY() * MyPanel.size) {
    cat.setBounds(cat.getX(), cat.getY -speed, MyPanel.size, MyPanel.size);
    try {
        sleep(10);
    } catch (Exception e) { e.printStackTrace();}
}
```

4.3.3 车辆拥堵信息的添加与删除模块的实现

通过按钮的点击拥堵添加的按钮，可以在道路上随机生成一个堵车地点，当到达此路段的车辆数量超过一定数量时，此时会在该路段排起长队，等待一段时间后，才可以慢慢通过该路段。在道路上生成的事故地点用 JPanel 显示在具体的道路上，同时该道路两端的十字路接收到该信号提醒路过路口的车辆进行绕道。如图 4.3 所示，在该路段 2 辆汽车会排起长队，最前面扭曲的汽车为因车祸而造成的的车辆停止。

部分代码如下：

```
BasePanel.smooth = false;
Acc_Panel accp = new Acc_Panel(14, 8);
jl.setText("已设置车祸地点：和平路(" + accp.getX() + "," + accp.getY() + ")");
```

图 4.3 拥堵现场

4.3.4 模拟车辆信息传递模块的实现

车辆之间信息传递的实现主要依靠模拟来进行实现，及假定两辆车之间的距离不超过一个坐标点的距离，若其中一辆车知道拥堵路段的信息（knowAcc = = true），或者知道拥堵已消失的信息（knowSmooth = = true），而另一辆车不知道的话，则将另一个汽车中的 knowAcc 或 knowSmooth 设置为相应的属性。为了可以使用户可以看出信息已经传递，或者该车已经知道其他车辆给他传递的信息，将得知拥堵路段的车辆设置为蓝色，正常运行的车辆或知道拥堵已消失的车辆设置为白色，效果图如图 4.4 所示。。

在主类 BasePanel 中存在全局静态变量 FangKuaiPosition 类数组 P，数组下标代表每辆汽车的唯一标识，同时 P 数组存放的为当前数组下标所代表的汽车的位置。在车辆类运行时，每当移动一个坐标单位，就利用循环查看 P 数组中每一辆车的位置，当其中有一辆车的位置横坐标、纵坐标均与本车辆的横纵坐标小于一个坐标单位（50 像素），且检索到的那辆车辆的知道事故的发生，则本车辆颜色变蓝；或知道道路已变成顺畅，则将本车辆的颜色变白。效果图如图 4.5 所示。

部分代码如下所示：

```
for (int j = 0; j < BasePanel.p.length; j++) {
    if (j == num) {
        continue;
    }
    if (!this.ifknowAcc() && BasePanel.m[j].ifknowAcc() && this.count == 0
```

```
        && Math.abs(BasePanel.p[j].getX() - BasePanel.p[num].getX()) <= 50
        && Math.abs(BasePanel.p[j].getY() - BasePanel.p[num].getY()) <= 50) {
            cat.add();cat.repaint();
            knowAcc = true;
            knowSmooth = false;
            this.count = 1;
            break;
        }
}
```

如上的代码所示，BasePanel.m[j].ifknowAcc()为序号为 j 的车辆是否知道有事故的发生，true 为知道，false 为该车不知道事故的发生，count 的值为 0 时代表该车没有被传递过信息，count 的值为 1 时，表示该车被传递过一次拥堵发生的信息。Count 的值为 2 时，表示该车被传递过拥堵路段已经发生过的信息。因此接受信息的为未接收过任何信息的车辆

图 4.4 汽车正常运行效果

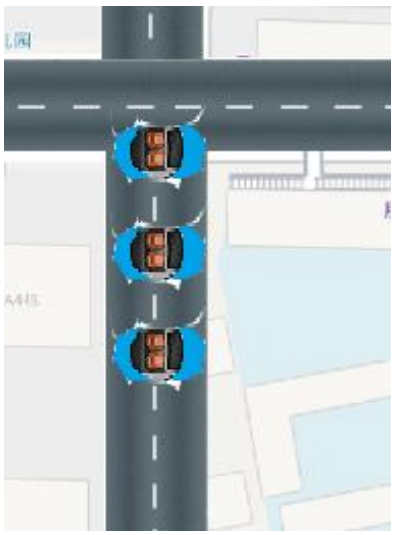

图 4.5 汽车得知拥堵后

4.3.5 模拟车辆拥堵路段规避功能模块的实现

当拥堵区域发生拥堵之后，由于和该路段链接的十字路口的信号接收装置中的 ifHaveAcc 变量设置为 true，每个经过该十字路口的车辆会与十字路口进行交互，交互方式也为模拟交互。每辆车在运行了一坐标单位后就会在十字路口库中检索所有的十字路口信号，当检索到有信号为 true 的十字路口时，则查看该路口的 position（int）变量，position 变量为 1 时代表该路口东边发生事故， position 变量为 2 代表该十字路口西边发生车祸，position 变量为 3 代表该十字路口南边发生车祸，position 变量为 4 代表该十字路北边发生车祸。当该车辆得知该路口的哪个方向发生交通拥堵后，判断自己的原路线是否要经过该十字路口，不经过则不需要改变自己原路径，若要经过拥堵路段，则走绕行路线，例如图 4.5 所示当 position 为 1 时，先查看是否可以向南走，可以则向南走 7 个坐标单位，再向东走 7 个单位，最后向北走 7 个单位，最后到达既定的通过十字路口的位置，已达到绕行的目的。

图 4.6 蓝车绕行图

5 系统测试

5.1 系统测试方法

本次系统测试采用面向对象软件的测试方法。由于面向对象软件开发引入了包括类与对象、继承、消息传递和动态链接等许多不同的特性。随着面向对象软件的开发方法的广泛使用。尽管例如白盒测试、黑盒测试、各种测试活动等软件测试技术对于面向对象软件同样适用的，但针对面向对象软件的特征研究相应的测试方法，能更好的保证面向对象软件的质量[26]。

本次面向对象软件的测试分别采用类的测试和交互测试两种方法[27]。

类是面向对象的基本类型，对类实例化得到的对象才能运行。因此，对类进行测试首先需要开发驱动程序，负责创建类的实例，然后测试驱动程序根据测试用例的定义，向类的实例（对象）发送一个或多个消息（调用类的方法），根据响应值、实例对象的属性状态的变化判断测试用例的执行结果是否正确。测试驱动程序的主要功能时运行可执行的测试用例并记录运行的结果。对于面向对象软件驱动程序可用下面的方式编写。实现一个 main( )函数作为测试驱动程序，使得测试程序可以编译、运行。在 main( )函数中执行测试用例，独一类进行实例化，给类发送响应的测试消息，并输出测试执行的结果。

交互测试，面向对象软件的运行主要是通过各种对象之间的消息传递来完成。因此，即使单个类中的每个方法在独立测试中时是正确的，多个类的实例通过消息传递交互的完成某个任务时，如果交互过程不正确，软件仍然可能是错误的。在交互测试中，测试驱动程序负责对参与交互的多个类进行实例化，并向相应的对象发送启动交互的消息或交互所需的信息序列。在判断交互测试的结果是，一方面要看交互完成后的输出，另一方面要看各个对象在参与交互之后所处的状态是否正确。

交互测试中不但要考虑信息的顺序，设计测试用例时，不仅要检查是否可以争取接受正常的消息序列，还要检查是否能够恰当处理不正确的消息序列。

5.2 系统测试数据及过程

主要进行测试的类为 Move 类与 AutoFindWay 类，这两个类是实现本仿真系统的关键类，其他所有类均为这两个类提供服务的，比如 FangKuaiPosition 类为 Move 类与 AutoFindWay 类定位坐标，本身无任何意义。因此这种类无需参加测试，只需要在人工检查环节检查代码的正确性，保证代码的所提供的服务无误即可。

5.2.1 随机路径选择算法模块的测试

本仿真平台将随机路径选择算法模块打包成类 AutoFindWay，因此针对这个类写出测试驱动程序。

主要设置的初始参数只有起始位置、终点位置以及可运行的区域，因此测试驱动只需给出起始于终止的参数坐标和可到达区域的 List 坐标数组集即可。驱动程序如下所示。

```
public static void main(String[] args) throws InterruptedException {
        getZhangAiList();
        AutoFindWay afw = new AutoFindWay(cat, fish);
        List<FangKuaiPosition> wayList = afw.getWayLine(cat, fish);
```

```
        System.out.println(wayList);
    }
```

此驱动程序中 cat 与 fish 分别为测试的起始于终止位置，getZhangAiList()为不可到达的区域，及其余地方均设置为可到达的区域，完成设置后实例化 AutoFindWay 类，并获取 wayList 集合中的信息，测试数据分别给出终点在不可到达区域的情况与可到达区域的情况，测试如表 5.1 所示。

表 5.1 路径选择算法测试

起始位置	终止位置	信号	路径
（13,12）	（2，12）	有路	(12,12)、(11,12)、(10,12)、(9,12)、(8,12)、(7,12)、(6,12)(5,12)、(4,12)、(3,12)、(2,12)
（15,11）	（1，2）	有路	(15,10)、(14,10)、(14,9)、(14,8)、(14,7)、(14,6)、(14,5)、(14,4)、(14,3)、(14,2)、(13,2)、(12,2)、(11,2)、(10,2)、(9,2)、(8,2)、(7,2)、(6,2)、(5,2)、(4,2)、(3,2)、(2,2)、(1,2)
（8，2）	（0，0）	无路	

5.2.2 汽车类的测试

在汽车类中，主要包含了对于车辆的运行控制，遭遇堵车时的拥堵处理，以及车辆与车辆之间的信息交互与传递，因此对于本类的测试应当将本类实例化后，分别对运行方法、信息传递方法、堵车处理方法分别进行测试，确保该类的正常运行。测试驱动在实例化 Move（车辆类）后，分别调取该类中的方法，在运行方法中需要传递的参数为路径选择算法返回的 wayList 集合；拥堵方法需要得知整个地区的车祸发生地点的全局变量；信息交互的方法需要给予每辆车唯一的信号标识，与该车临近坐标单位中的其他车辆的坐标和其他车辆所知道的信息。

因此给予运行方法的 WayList 集合中可以到达的路径，给与该方法不可到达的路径，不给予其路径三种测试样例。测试结果如表 5.2(a)所示。

对于拥堵处理的测试样例，分别给予无拥堵路段的情况；在东西或者南北道路上的拥堵坐标并且车在该路段；在东西或者南北道路上的拥堵坐标并且车未在该路段。测试结果如表 5.2(b)所示。

信息传递的方法中，测试用例分别设置为周围无车的状况，周围有车但不知道拥堵地区，周围有车且知道拥堵地区，本车知道拥堵地区与其交互的车辆知道拥堵地区已消失，这四种测试用例。测试结果如表 5.2(c)所示。

表 5.2(a) 汽车运行方法的测试

起点	终点	wayList.size	结果
(13,12)	(2,12)	15	可以正常到达
(13,12)	(0,0)	0	路径不可达
无	无	null	未设置路径

表 5.2(b) 拥堵行为的测试			
是否有拥堵	拥堵路段	车辆是否在该路段	车辆行为
否	无	——	正常行驶
有	东西/南北路段	是	不更新
有	东西/南北路段	否	简单绕行

表 5.2(c) 信息传递的测试			
交互对象	交互对象信息	本车状态	交互后状态
无	无	无信息	——
有	无信息	无信息	不更新
有	有堵车信息	无信息	得知拥堵区域
有	堵车路段消失	有拥堵信息	去除拥堵信息

5.2.3 交互测试

交互测试将类与类之间存在联系的类全部进行实例化，并对相应的对象传递相应的数据 ，完成实例化后检查每个类的状态和最终的输出结构。

本仿真平台中存在信息交互的为 AutoFindWay 与 Move 类之间的信息传递，AutoFindWay 将计算好的路径信息传递给 Move 类，在运行过程中的一切操作均有 Move 类自身进行判定，无需外界的干预。同时本仿真系统中的其他类不参与信息的交互，只是作为辅助类为上述两个类提供服务。因此他们无需进行交互测试。

测试驱动如下所示：

```
Main(){
  Move m = new Move();
  AutoFindWay afw = new AutoFindWay(openList, closedList);
  wayList0 = afw.getWayLine(new MyPanel(15, 11));
  p = new sePosition();
  m = new Move(new MyPanel(15, 11));
  m.setList(wayList0);
  m.setNum(0);
}
```

如上驱动程序所示，给 AutoFindWay 类两个集合，分别为未处理的集合和处理过的集合，完成 AutoFindWay 的初始化方式，同时是给 Move 类及汽车类初始化的条件，如车辆信号、起始位置等。Move 类中各变量状态测试结果如下表 5.3(a)所示，AutoFindWay 中各变量如表 5.3(b)所示。

表 5.3(a) Move 类中变量状态		
名称	类型	值
Cat	MyPanel	(15, 11)
Fish	MyPanel	(1,2)
wayList	ArrayList	Size = 24
num	Int	0
knowAcc	boolean	false
knowSmooth	boolean	True

表 5.3(b) AutoFindWay 类中变量状态		
名称	类型	值
wayList	List<FangKuaiPosition>	Size = 24
beginFk	FangKuaiPosition	(15, 11)
endFk	FangKuaiPosition	(1,2)
closedList	ArrayList	Size = 142
openList	ArrayList	Size = 16

测试结果如上面两个表单所示，在 Move 类与 AutoFindWay 类进行信息交互后，每个类的输出正确，各个类中的状态均符合要求，可以实现该仿真平台的正常运行。

5.3 测试结果与分析

通过对主要类的测试与类与类之间的关联的测试，每个类进行单独的测试，均可以输出正确的结果，且可以实现预设置的功能。类与类之间关联的测试，输出完全正确，且各个类交互之后所处的状态也完全正确，测试结果均正常。以上的测试结果说明，本仿真系统的正常使用，并保证功能可以正常的提供给用户使用，该仿真系统已经基本实现了交通拥堵与规避的仿真平台的目标和功能，并且考虑到了现实情况中可能会出现的各种问题，并对这些问题做了基本的设计与解决，可以保证在使用过程中满足对相关问题的分析与解决，为交通拥堵的解决提供了一定的参考，类与类之间的独立性，与松耦合，可以方便后续开发与维护人员更加简单的对该系统进行升级和维护，每个类内注释清晰，详细说明关键变量和方法的作用与意义。 且系统具备以下特性：

（1）系统界面友好，方便用户操作，程序的可靠性高。

（2）系统实现简单，开发时间短，开销小，使用方便。

6 结论与展望

6.1 结论

随着计算机技术的发展与成熟，计算机与各行各业的结合也变得更加密切，对计算机的应用也变得更加的频繁。在中国城市化的发展与经济进步的今天，城市交通问题越来越成为一个关键并且重要的问题，交通部门对于交通道路的畅通也更加关注，因此交通仿真平台对于这一类问题的解决方案会更加多种多样。而本仿真平台只是对其中一种方案的模拟，为拥堵问题的解决提供了一种新的思路。

通过这段时间的设计与开发，总体上完成了以下几项工作：

1.研究了交通仿真软件，参考了一些相关的文献资料，并使用了一些交通仿真系统，对交通仿真系统有了一定的认识和了解。

2.研究了 Java 技术对于实现交通仿真系统的可行性，温习可能会在编程中用到的 Java 的 API 函数接口和 Java 线程相关的知识，为系统原型的实现提供解决方案。

3.在对技术有了一定的掌握后，对系统的总体实现有一定的初步大纲，并在大纲的基础上尽可能考虑到各种现实情况和解决方案，完善设计说明。

4.在研究设计的基础之上对系统进行实现，该系统大体上实现了总体设计方案中所提到的拥堵状态、信息传递、线路规避等模块的功能。

5.对系统进行测试，尽管系统可以无误的运行，但在功能设计方面还有一些不足与稚嫩，在今后的软件编程中可以随着自己经验的增长和知识的积累逐步做到完善。

6.2 展望

交通拥堵信息传递与规避的仿真平台的设计与实现涉及了多方面的理论、方法和技术。但是这种仿真的实现对于解决设计车辆拥堵和车联网的使用提供了一种新的思路，尤其是随着车联网技术的发生，这种信息的传递方式对于无人车的线路规有着重要的作用[28]。

虽然该仿真平台只是在一台主机上模拟了车辆与车辆之间对于拥堵信息的传递，同时采用较为简单的策略实现了车辆对于交通拥堵路段的规避。但是随着互联网技术发展与进步，车联网会变得越来越普及与简单，每辆车之间信息的传递会很快得到实现，并且这种技术会最终有所体现和应用。从车联网的终局来看，是实现无人驾驶，是智能交通的核心部分[29]。车联网是未来的 ITS（智能交通系统）的核心信息通信平台，发挥越来越多作用[30]。在保障道路行驶安全方面，即使的信息传递可以提前告知车辆事发路段，避免二次灾害的发生。对于防止交通堵塞方面，如果对该平台进行一定的扩展，可以实时监控了路况，提前预警，效缓解交通堵塞压力。

致 谢

在论文即将完成之际，非常感谢我的指导老师安毅生老师对我工作的指导，安毅生老师治学严谨，知识渊博，充满工作热情，这些优点值得我终身学习。在系统的开发和论文的撰写过程中，安老师不辞辛劳，不厌其烦的一遍又一遍的帮我修改并提出意见。对于我提出的问题，安老师总会非常及时的给予我解答，并提出修改意见。大学 4 年，安老师作为我的导师，不仅在学习上对我提供了许多帮助，这对于我的生活也给予我了一些关键性提议，再次对老师表示由衷的感谢，感谢老师对我大学四年的照顾。也反省自己的在本次毕业设计中的不足，向老师学习。

还要感谢刘建书老师对我毕业论文的指导。刘老师对于我的论文和毕设提出了许多有效的建议，让我的论文更加的严谨。刘老师严肃的科学态度，一丝不苟的学术精神，求同存异的工作作风不断激励着我。同时感谢学姐和其他同学在我系统开发中给我提出的建议，感谢他们的帮助。

参考文献

[1]宋博,赵民.论城市规模与交通拥堵的关联性及其政策意义[J].城市规划,2011.
[2]姬浩,李佩,苏兵,徐阳.事故车辆影响下的城市道路交通仿真研究[J].中国安全生产科学技,2019.
[3]国家统计局.中国统计年鉴[J].北京:中国统计出版社,2017.
[4]高德地图交通大数据. 2017 年度中国主要城市公共交通大数据分析报告.高德地图,2017.
[5]赵津,张博,张庆余,赵鹏超.基于城市先进道路交通系统的智能交通仿真平台设计[J].汽车工业研究,2019
[6]郭凤香,熊坚,王朝英。交通分配模型研究及其应用[J].交通信息与安全,2004.
[7]刘洋.城市微观交通仿真系统研究[D].西北工业大学,2006.
[8]赵玉平.微观交通仿真技术及其应用研究[D].电子科技大学,2014.
[9]Vittorio Astarita,Vincenzo Pasquale Giofré. From traffic conflict simulation to traffic crash simulation: Introducing traffic safety indicators based on the explicit simulation of potential driver errors[J]. Simulation Modelling Practice and Theory,2019.
[10]张白一,崔尚森,张辰.面向对象程序设计——Java(第三版)[M],西安电子科技大学出版社,2016.
[11]戈茨.java 并发编程实践[M],电子工业出版社, 2007.
[12]Bernat Goñi-Ros,Wouter J. Schakel,Alexandros E. Papacharalampous,Meng Wang,Victor L. Knoop,Ichiro Sakata,Bart van Arem,Serge P. Hoogendoorn. Using advanced adaptive cruise control systems to reduce congestion at sags: An evaluation based on microscopic traffic simulation[J]. Transportation Research Part C,2019.
[13]魏明,杨方廷,曹正清.交通仿真的发展及研究现状[J].系统仿真学报,2003.
[14]耿彦斌,张雪莲.关于我国交通仿真技术发展战略的思考[J].肩带交通技术,2010.
[15]任其亮,刘博航.交通仿真[M].人民交通出版社.2013.
[16] 杭佳宇,周溪召.基于 VISSIM 仿真的学府路与谷阳路交叉口优化设计[J].物流科技,2017.
[17]丹尼斯·舍伍德.系统思考.机械工业出版社.2008.
[18]Joshua Bloch.Effective Java.机械工业出版社.2009.
[19]方丽琴,段满珍,李铮,张敬.城市交叉口交通信号控制仿真模块设计[J].华北理工大学学报,2018.
[20]J Barceló.Fundamentals of traffic simulation[M].New York:Springer,2010.
[21]刘小兰.交通仿真技术在道路交通工程中的应用研究[D].湖南:湖南大学, 2005.
[22]罗伯特·马丁.Clean Code(评注版)[M],电子工业出版社,2012.
[23]科内尔,Java 核心技术卷Ⅰ:基础知识[M].电子工业出版社.2011.
[24]Thomas H.Cormen,Charles E.Leiserson.算法导论[M].机械工业出版社.2006.
[25]Bruce Eckel.Java 编程思想(第 4 版)[M].机械工业出版社.2007
[26]陈明.软件工程[M].中国铁道出版社.2011.
[27] Kent Beck,孙平平 译.Test-Driven Development[M].中国电力出版,2004.
[28]徐晓齐.车联网[M].化学工业出版社.2015.
[29]吴建平.无人驾驶未来交通与仿真研究[J].中国建设信息化,2018.
[30]罗征宇.人工智能技术在城市智能交通方面的应用研究[J].通讯世界,2019.