分布式自主智能无人系统实践

Project for the Practice of Distributed Autonomous Intelligent Unmanned System

（Tiny SimWorld）

小组名单

序	学号	姓名	职责与实际承担工作
1	1950083	刘智宇(组长)	数字世界服务端； 手机 APP 传感器数据获取； 设备间通信
2	1950089	季马泽宇	协助设计框架； 3 维可视化
3	1953907	齐晓燕	传感器数据处理； 手机位置与姿态解算
4	1951580	赵元德	仿真车建模； MATLAB 控制仿真
5	1954163	图尔荪托合提	Matlab 通信

本小组自评成绩	优
自认为是否完成全部必做要求	是
闪光点（不超过 3 句话）	仿真世界功能完善，且利用抽象类和接口等面向对象思想，方便后续功能扩展。 可视化界面使用 OpenGL 技术，3D 可视化更加和清晰美观。
好人分送给如下同学（0-2 分，不允许出现小数）	图尔荪托合提 2 分，再报告撰写提供帮助 成员获奖情况见附录七

2022.09

同济大学人工智能/自动化专业综合设计与实践课程项目报告
Team Project Report

目录

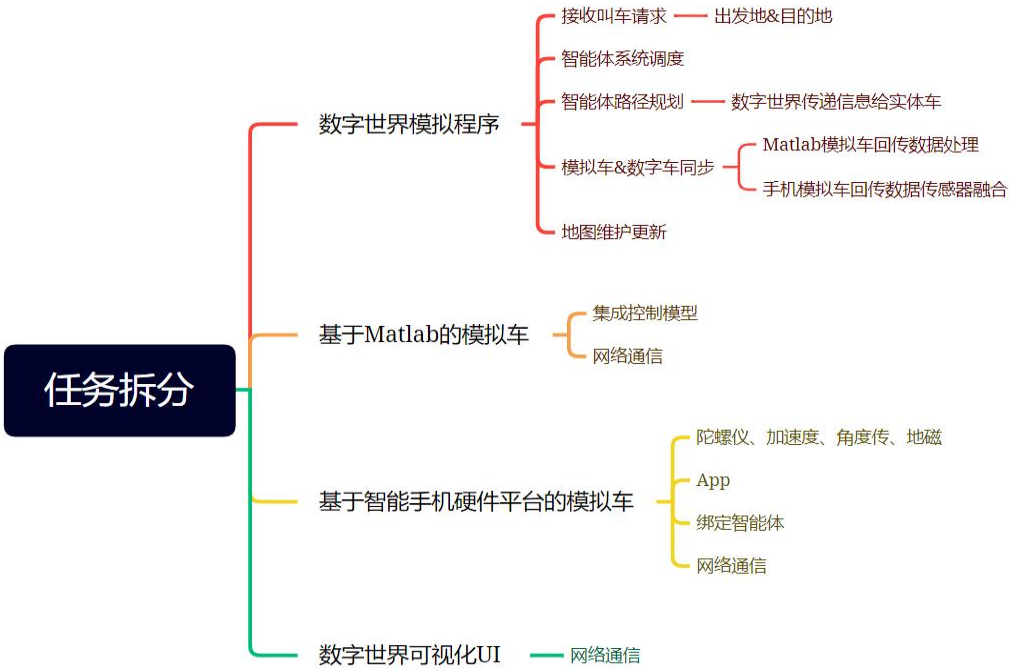
报告正文

一、任务及开发路径选择

任务	开发路径选择
数字世界服务端	Java+多线程+定时任务线程池
手机模拟车	Android+多线程
Matlab 仿真模拟车	Matlab
可视化	OpenGL+imGUI
网络通信模块	Socket+JSON+TCP

二、任务架构

任务架构与张伟老师课程所要求的保持一致，并且基本功能都进行了实现：

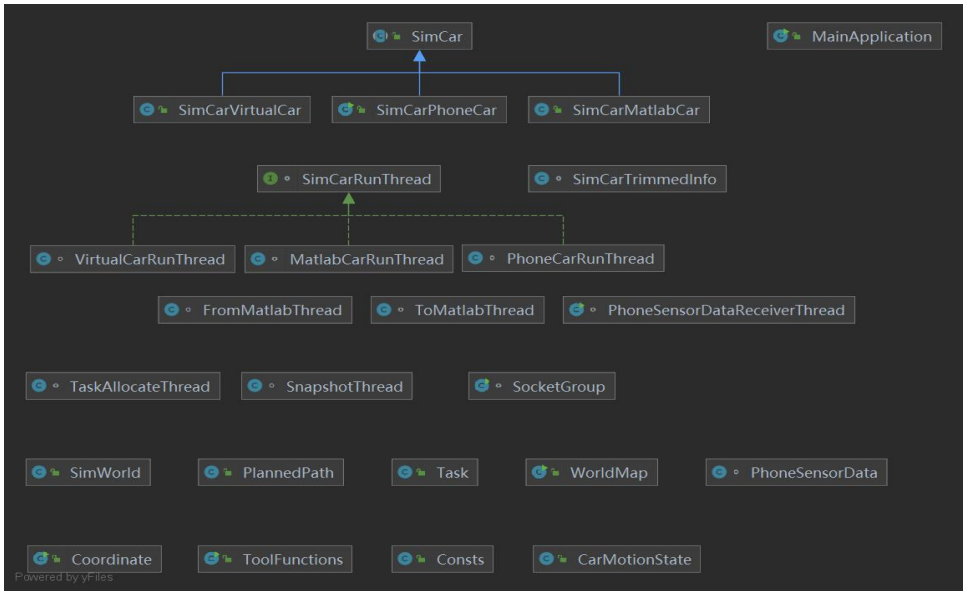


三、任务具体实现

1. 数字世界服务端

1.1 数字世界服务端全部类总览

下图是数字世界服务端所运用到的全部类的一个总览图。

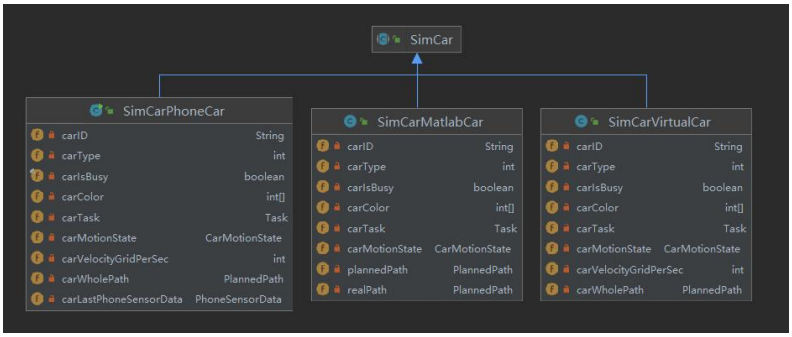


1.2 智能体相关类简述

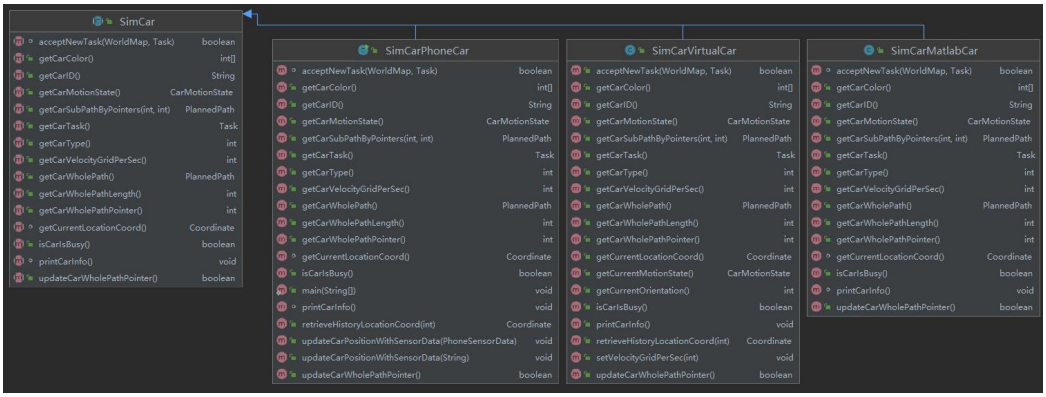
在实现智能题的时候，我采用了较多的面向对象思想。
首先，在服务端运行三种仿真车辆，都继承 SimCar 抽象类

1. SimCarVirtualCar，非手机 APP 车辆和 Matlab 车辆，存粹由服务端管理
2. SimCarPhoneCar，与手机端保持同步的模拟车
3. SimCarMatlabCar，与 Matlab 端保持同步模拟车

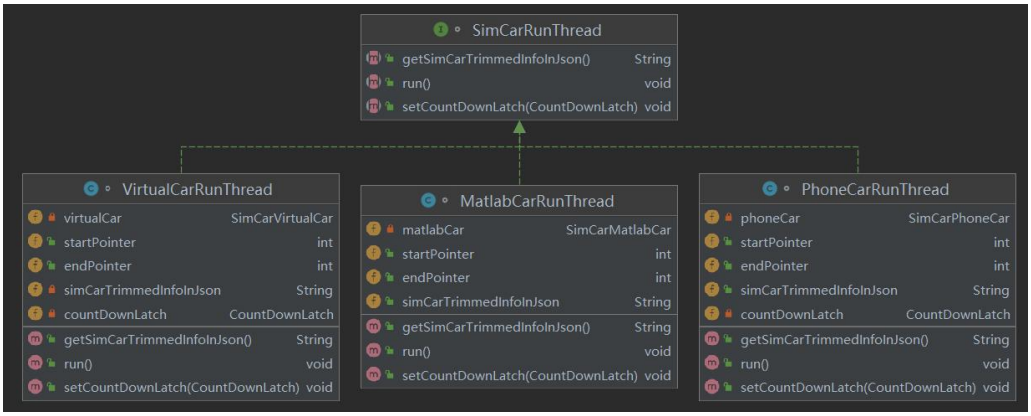
仿真车辆的成员变量：主要包含车辆的各个基本信息，



仿真车辆的成员函数：



其次在实现仿真车辆运行线程时，也都是让三种车辆的运行线程实现同一个自己定义的运行线程接口，如下图所示：



通过这样的面向对象思想，我在完成一个类型的车辆及其对应运行线程后，在实现其他类型的车辆时都较为的便捷，也让我理解了面向对象思想的优势所在。相信未来如果要拓展更多类型的车辆，也会更加便捷。

1.3 智能体优化决策模块简述

我在仿真世界中对各个智能体的全局路径规划最终采用四领域搜索方式、曼哈顿距离作为启发函数的 A*算法。当然我也实现了八领域搜索、欧式距离、迪杰斯特拉算法等额外方式，最终为了展示效果没有启用，在服务端中的配置文件（Consts.java）中进行修改即可。

```
// ================ 路径搜索相关 ================
public static final int CHOSEN_DIRECTIONS_NUM = Consts.FOUR_DIRECTIONS_NUM;
public static final int CHOSEN_DISTANCE_CALCULATE_METHOD = Consts.USE_MANHATTAN_DISTANCE;
```

A*算法、迪杰斯特拉算法都较为常见，这里不再赘述。在未来我们可能会尝试更多高级的多智能体路径规划算法。

1.4 数字世界多线程简述

我们的数字世界一大主要的亮点就说使用了大量的多线程技术，不仅仅有普通线程、还有定时任务线程池等待技术，这里进行简单展示。

首先在主线程中，SimWorld 对象会先按照配置文件（Consts.java）进行仿真世界的初始化：包括地图创建、与客户端的连接（通信 socket 的获取）、初始任务池、各个类型的仿真车辆以及车辆对应的运行线程和通信线程的创建工作。

```
========================开始进行SimWorld初始化    时间：Fri Sep 09 16:42:49 CST 2022========================
地图创建...

创建的地图JSON信息如下
{"mapRows":36,"mapCols":36,"mapMaxHeight":255,
 "mapMat
 ":"111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
 000000000000011111000000000000000000000000000011111000111111111111111111111110001111100011111111111
 000000000001111000000000000111111000000000000011110000000000001111100000000000011110000000000001111110
```

```
车量创建：共5辆...
创建第  0 个车辆
    carID                   :   PhoneCar-0
    carType                 :   0

创建第  1 个车辆
    carID                   :   VirtualCar-1
    carType                 :   2

创建第  2 个车辆
    carID                   :   VirtualCar-2
    carType                 :   2
```

```
任务创建：共10个...
创建第  0 个任务
    taskID                  :   0
    taskOriginCoord         :   (32,25)    taskDestinationCoord    :   (20,24)
    taskPathCost            :   46

创建第  1 个任务
    taskID                  :   1
    taskOriginCoord         :   (30,31)    taskDestinationCoord    :   (19,5)
    taskPathCost            :   74
```

```
连接所需的服务端
[SOCKET INFO] : ServerSocket is Ready. Waiting for Phone and Matlab to Connect...
[SOCKET INFO] : Let PhoneSocket Connect.
[SOCKET INFO] : PhoneSocket is Connected.
```

随后，在完成所有初始化后，主线程会开启一个定时任务线程池，主要按一定周期进行任务分配、快照拍摄工作。通过使用任务线程池的方式，我们可以以一个十分精准的频率完成上述任务。

```
// 运行任务分配线程的API
public void runTaskAllocateThread()
{
    TaskAllocateThread allocateTaskTimerTask = new TaskAllocateThread( outerSimWorld: this);
    // 任务创建完毕后加入线程池
    this.scheduledThreadPool.scheduleAtFixedRate(allocateTaskTimerTask,
            Consts.ALLOCATE_TASK_THREAD_TIMER_DELAY, Consts.ALLOCATE_TASK_THREAD_TIMER_PERIOD, TimeUnit.MILLISECONDS);
}

// 运行快照线程API
public void runSnapshotThread()
{
    SnapshotThread snapshotThread = new SnapshotThread(this.simCarsList);
    this.scheduledThreadPool.scheduleAtFixedRate(snapshotThread,
            Consts.SNAPSHOT_THREAD_TIMER_DELAY, Consts.SNAPSHOT_THREAD_TIMER_PERIOD, TimeUnit.MILLISECONDS);
}
```

两个定时任务的工作时输出信息如下图所示。

```
=========================第 0 次任务分配线程启动    时间：Fri Sep 09 08:30:26 CST 2022=========================
PhoneCar-0 is busy!
PhoneCar-0 refuses Task-0
VirtualCar-1 is not busy!
添加去往任务起点的路径     :  原始路径长度 : 1  ||  新增路径长度 : 38  ||  合计路径长度 : 39
添加任务路径              :  原始路径长度 : 39  ||  新增路径长度 : 24  ||  合计路径长度 : 63
VirtualCar-1 accepts Task-0
VirtualCar-2 is not busy!
添加去往任务起点的路径     :  原始路径长度 : 1  ||  新增路径长度 : 50  ||  合计路径长度 : 51
添加任务路径              :  原始路径长度 : 51  ||  新增路径长度 : 38  ||  合计路径长度 : 89
VirtualCar-2 accepts Task-1
VirtualCar-3 is not busy!
添加去往任务起点的路径     :  原始路径长度 : 1  ||  新增路径长度 : 33  ||  合计路径长度 : 34
添加任务路径              :  原始路径长度 : 34  ||  新增路径长度 : 27  ||  合计路径长度 : 61
VirtualCar-3 accepts Task-2
VirtualCar-4 is not busy!
添加去往任务起点的路径     :  原始路径长度 : 1  ||  新增路径长度 : 20  ||  合计路径长度 : 21
添加任务路径              :  原始路径长度 : 21  ||  新增路径长度 : 38  ||  合计路径长度 : 59
VirtualCar-4 accepts Task-3
```

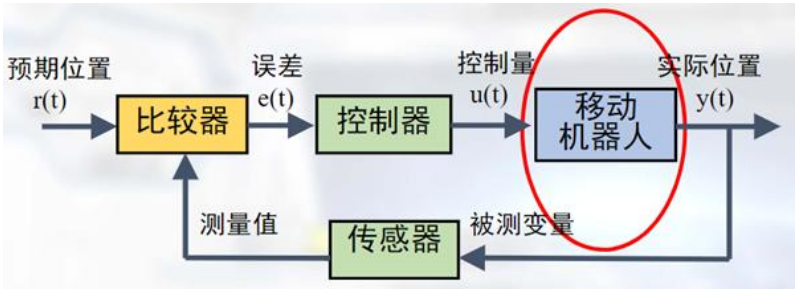
```
=========================第 4 次快照拍摄线程启动    时间：Fri Sep 09 08:30:39 CST 2022=========================
[SNAPSHOT INFO] : 等待各个车辆拍摄快照中....
[SNAPSHOT INFO] : PhoneCar-0快照拍摄中
开始运行
[SNAPSHOT INFO] : VirtualCar-1快照拍摄中
开始运行
[SNAPSHOT INFO] : VirtualCar-2快照拍摄中
开始运行
[SNAPSHOT INFO] : VirtualCar-3快照拍摄中
```

剩下的就是一些车辆单独运行的线程，整体逻辑较为简单，它们会将车辆的路径进行追加到一个车辆自身的 Path 中。等到快照线程进行访问时，将上次快照到现在的全部路径信息打包，传递给快照线程。快照线程整合所有车辆信息后，会发送给 GUI 模块。

2. Matlab 仿真车

2.1 任务描述

根据服务端的规划模块输出的路径信息，利用 MATLAB 实现轨迹跟踪和速度控制。本次任务的要求假设中，小车被简化为一个质点，在地图中占据一个 cell，且为简单起见，我们认为不存在运动半个 cell 的情况，要求 cell 足够小。对于此控制问题，首先确定整体模型如下：



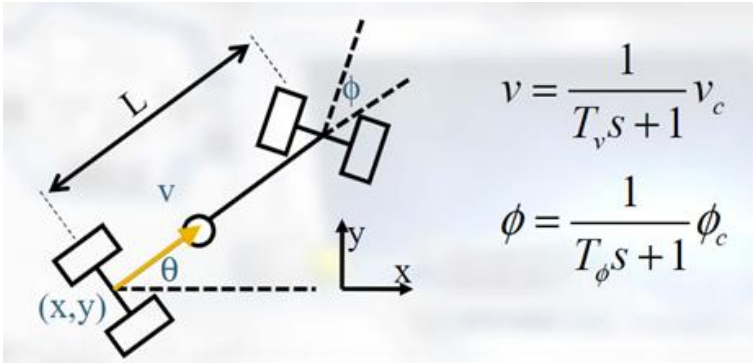
其中预期位置给定（由服务端输出或规划模块输出），由传感器获取相应的机器人信息（位置、速度等）反馈计算得误差进行控制。

2.2 仿真车建模

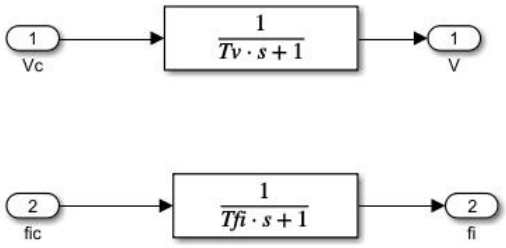
建立移动机器人模型参考 Car-like 小车建模，车长为 1 个单位，代表小车占据一个 cell，分为两个部分：动力学模型和运动学模型。

2.2.1 动力学建模

不考虑 RoboTaxi 的体积、几个轮子、动力，因此不需要考虑电机减速驱动等因素的影响。一般情况下，可以假设控制变量（V_c, ϕ_c）到系统输出（V, ϕ）满足一阶惯性关系，如下图所示。



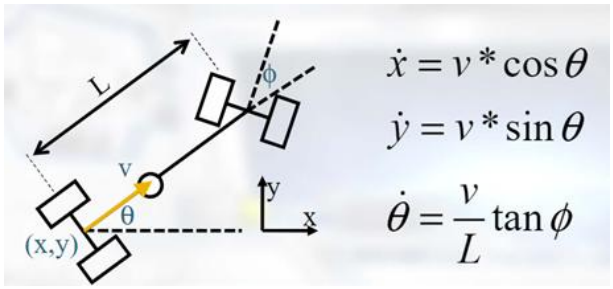
在 simulink 中搭建仿真结构并封装为线性动力学模块：



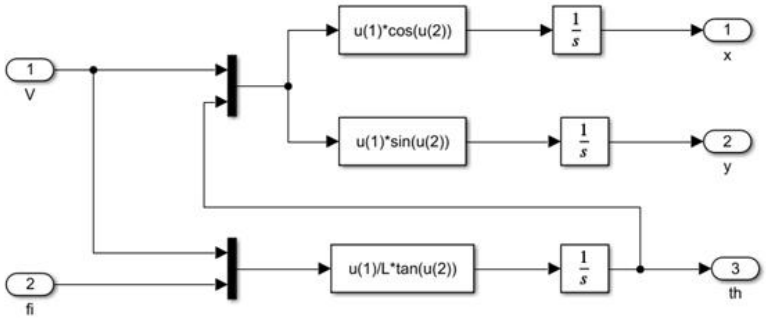
其中 T_v, T_{fi} 分别为速度驱动时间常数和转向驱动时间常数，可根据实际情况进行修改。

2.2.1 运动学建模

建立运动学模型，先写出运动学方程：$\begin{cases}\dot{x}=v\cos(\phi)\\ \dot{y}=v\sin(\phi)\\ \dot{\theta}=\frac{v}{L\tan\phi}\end{cases}$，其中 L 为小车的车长，在仿真时可以将其设置为 1 代表小车占据一个 cell，如下图所示：

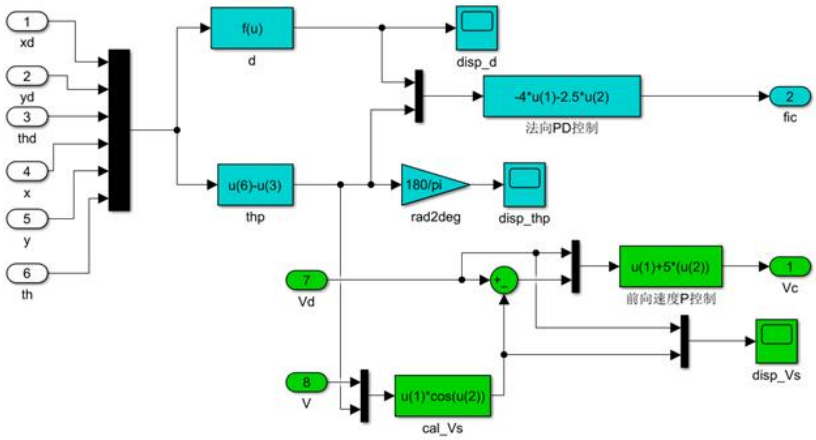


在 simulink 中搭建仿真结构并封装为运动学模块：



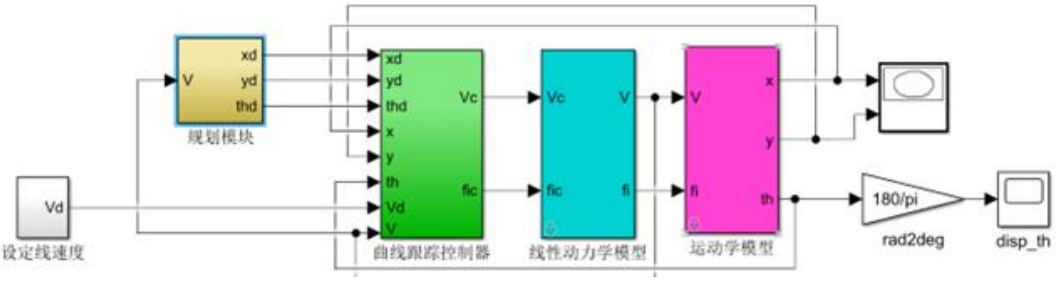
2.2 跟踪误差方程与 PID 控制

跟踪误差方程 $\begin{cases} \theta_p = \theta - \theta_d \\ v_s = v_d \cos\theta_p \\ \dot{d} = v_s \tan\theta_p \end{cases}$，其中 θ_p 为方位角误差，v_s 为前向速度，d 为法向位置误差。控制的目标在于使得方位角误差等于零，前向速度等于预设速度，方向位置误差等于零。此次控制方法采用 PID 控制，将方位角误差和法向位置误差的 PID 控制简化为 PD 控制，实现速度方向以及轨迹的控制，将控制前向速度与预设速度的误差的控制简化为 P 控制，实现速度大小控制。

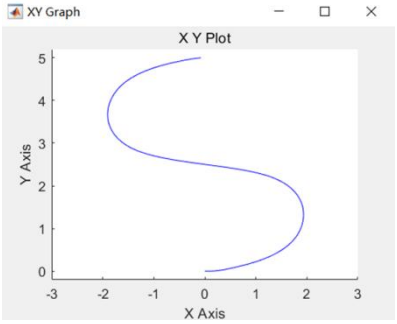


2.3 控制系统构建与测试

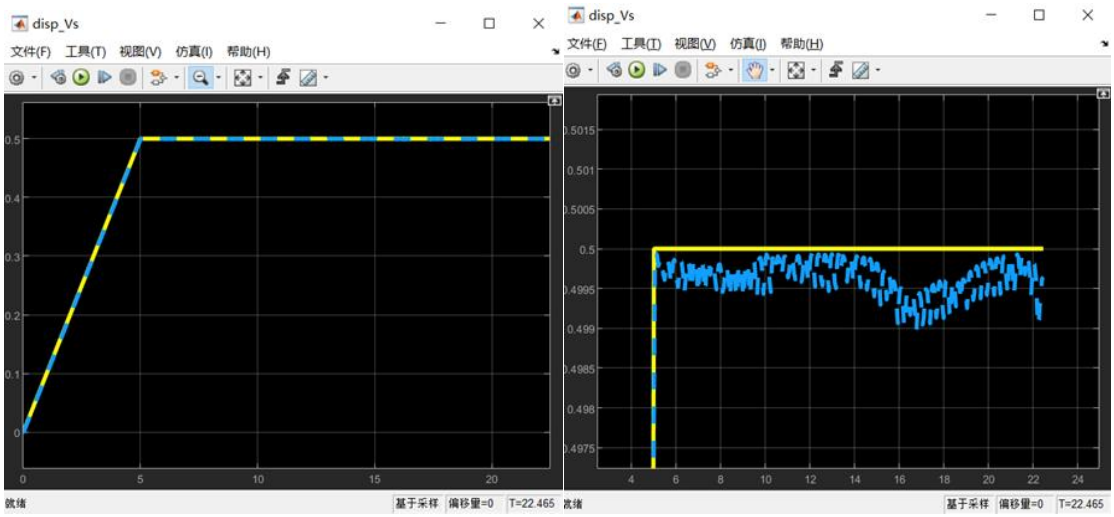
将这些模块串联起来，并加入简单的规划模块，进行控制部分的测试，如下图：



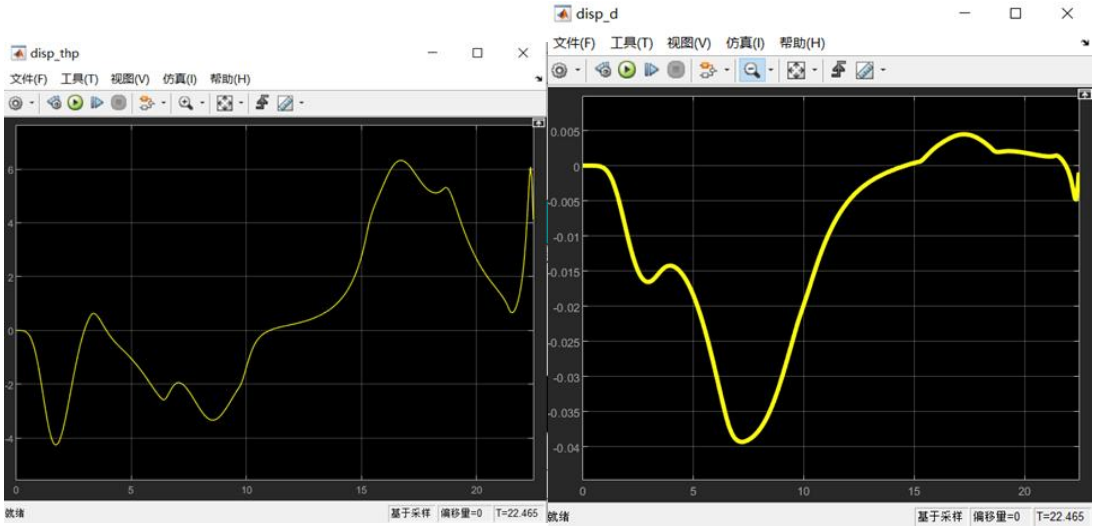
其中，在规划模块中用过起点和终点的 B 样条曲线函数计算，输出规划的路径信息。在设定好预期线速度后进行测试，输出规划路径如下图：



输出预设速度与实际速度的图像进行对比如下，可以认为基本实现速度控制。（右侧图为局部放大图）



再分别输出方位角误差和法向位置误差，可以认为基本实现方向控制，如下图：



从图像中可以看到，方位角误差较小，可见小车实际运行方向与规划路径方向差距不大；法向误差很小，可以认为小车完全在规划路径上运动。

3. 手机 App 仿真车

我们的手机 APP 仿真车采用的 Android 系统，使用 Android Studio 进行开发。

3.0 手机传感器界面展示

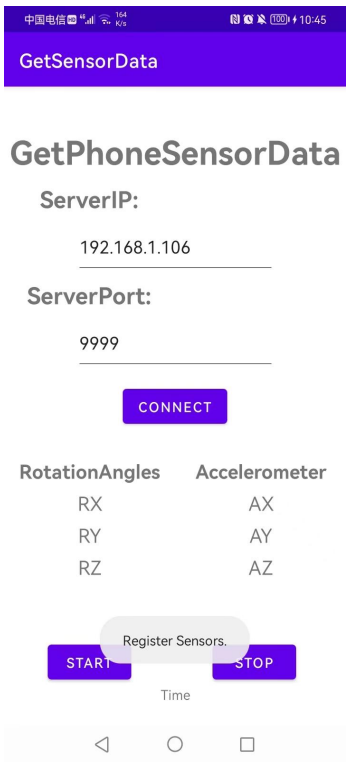


图 1：APP 启动界面

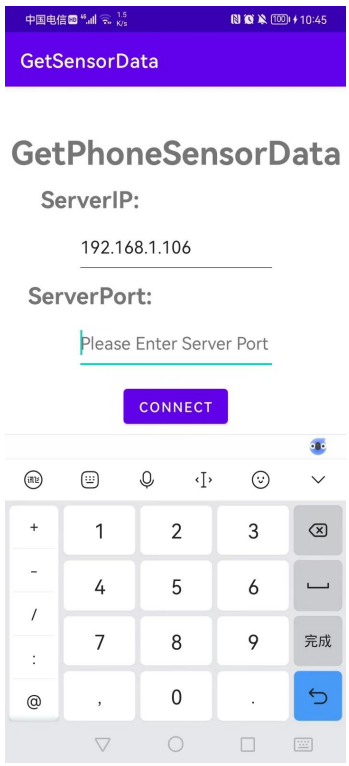


图 2：修改服务端 IP 和端口

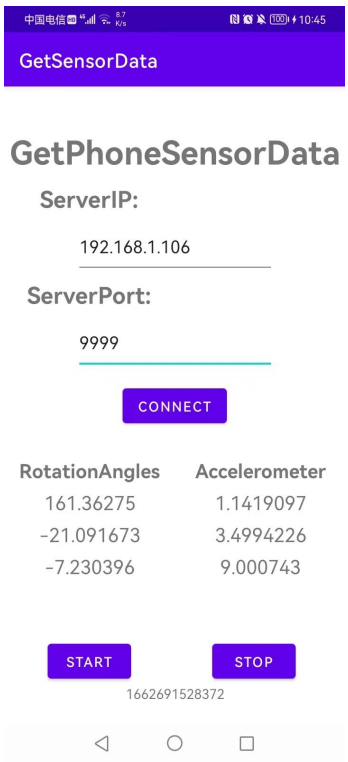
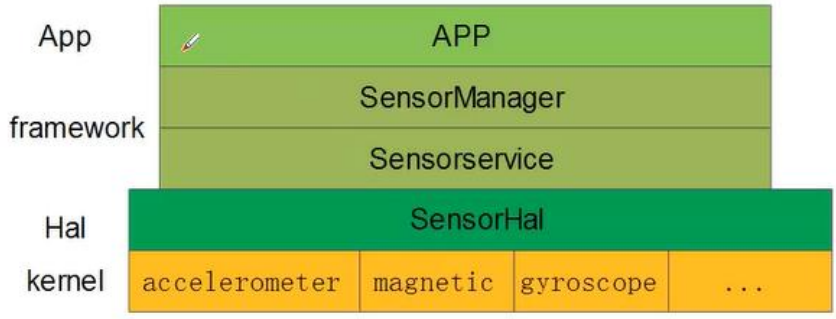


图 3：传感器数据获取

3.1 传感器数据采集

安卓系统的 Sensor 框架如下：



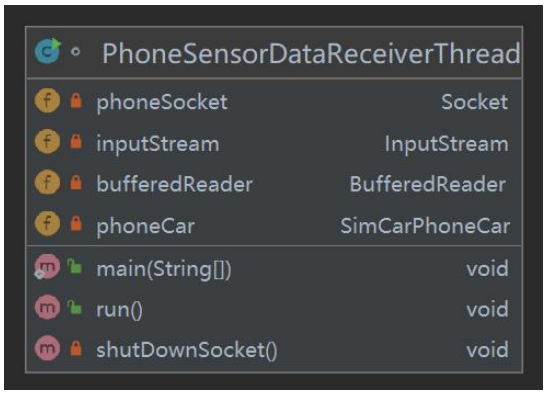
1. 应用层（APP）
2. 中间层（framework hal）
   SeneorManager 向上提供接口，支撑 APP
   SensorService 向 SensorManager 提供服务
   SensorHAL 是动态库，根据不同平台 load 不同动态库
3. 底层（kernel）
   设备的驱动（实体 Sensor），生成设备节点

根据以上框架，在获取手机传感器信息时，首先要获取手机的 SensorManager，通过 SensorManager 获取相关传感器资源，同时用于注册传感器时间监听器。

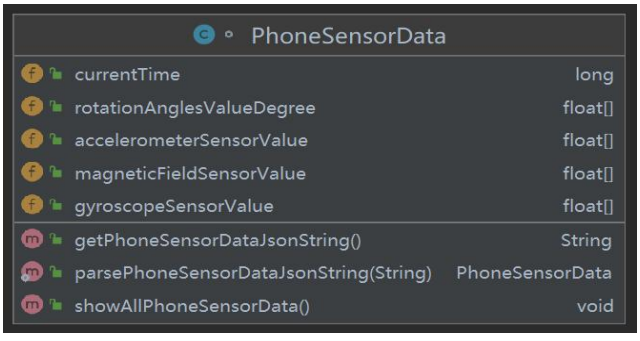
为了实现所需的相关功能，我实现了自己的 SensorEventListener 接口。在 SensorManager 中注册自己的传感器事件监听器。

3.2 手机数据回传

手机数据回传主要采用的时 Socket 技术，通过 TCP 协议与服务端进行通信。服务端首先开启 SeverSocket，等待客户端连接。连接后，服务端开启一个会相应的开启 PhoneSensorDataReceiverThread 线程进行通信。PhoneSensorDataReceiverThread 类的各个成员变量和成员函数如图所示。



通信时，使用 Android 原生的 JSON 打包方式对数据进行封装，并通过 Socket 传递给数字世界服务端。数字世界服务端接收客户端传来的各个传感器数据，用于更新车辆位姿信息。



具体传递的数据形式会在后续的网络通信部分进行阐述。

3.3 姿态解算分析

3.3.1 基于卡尔曼滤波的欧拉角法

惯导姿态角的初始值是由加速度数据和磁场数据计算得到的，加速度数据计算得到滚转角和俯仰角，由滚转角，俯仰角以及磁场数据经过计算再得到初始的偏航角。下面是具体的计算和推导过程。航坐标系为东北天坐标系（ENU），载体坐标系为右、前，上（x,y,z）,姿态角的旋转顺序为：z,x,y(偏航、俯仰，滚转)。

3.3.1.1 计算滚转角和俯仰角

对于加速度计有以下公式：

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = C_n^{b^T} * \begin{bmatrix} 0 \\ 0 \\ |g| \end{bmatrix}$$

导航系到载体系的转换矩阵为C_n^b。设偏航、俯仰，滚转的初始值分别为z_0, y_0, x_0。

$$C_n^b = \begin{bmatrix} c(y0)c(z0) - s(y0)s(x0)s(z0) & -c(x0)s(z0) & s(y0)c(z0) + c(y0)s(x0)s(z0) \\ c(y0)s(z0) + s(y0)s(z0) & c(x0)c(z0) & s(y0)s(z0) - c(y0)s(x0)c(z0) \\ -s(y0)c(x0) & s(x0) & c(y0)c(x0) \end{bmatrix}$$

并且满足：$|g| = \sqrt[2]{a_x^2 + a_y^2 + a_z^2}$

经过计算得到:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} -g * c(x0)s(y0) \\ g * s(x0) \\ g * c(x0)c(y0) \end{bmatrix}$$

所以：

$$\mathrm{x0} = \mathrm{asin} \frac{a_y}{\sqrt[2]{a_x^2 + a_y^2 + a_z^2}}$$

$$\mathrm{y0} = \mathrm{atan}\,2 \frac{-a_x}{a_z}$$

3.3.1.2 计算偏航角

这块可以想象一下，假设一开始平稳放置磁力计，则偏航角的计算公式如下：

该平稳状态设定为 a 状态，即俯仰角和滚转都为 0,对于磁力计放置的任意状态，有以下公式：

$$\begin{bmatrix} m_x^a \\ m_y^a \\ m_z^a \end{bmatrix} = C_a^b * \begin{bmatrix} m_x^b \\ m_y^b \\ m_z^b \end{bmatrix}$$

$$C_a^b = \begin{bmatrix} \mathrm{c(y0)} & 0 & \mathrm{s(y0)} \\ \mathrm{s(x0)s(y0)} & \mathrm{c(x0)} & \mathrm{-c(y0)s(x0)} \\ \mathrm{-c(x0)s(y0)} & \mathrm{s(x0)} & \mathrm{c(x0)c(y0)} \end{bmatrix}$$

$$\begin{bmatrix} m_x^a \\ m_y^a \end{bmatrix} = \begin{bmatrix} m_x^b c(y0) + m_z^b s(y0) \\ m_x^b s(y0)s(x0) + m_y^b c(x0) - m_z^b * c(y0)s(x0) \end{bmatrix}$$

根据上式求出和之后，再根据公式就可求出偏航角。

$$\mathrm{z0} = \mathrm{atan\,2}\left(m_y^a, m_x^a\right)$$

欧拉角算法通过求解欧拉角微分方程直接计算航向角、俯仰角和横滚角,欧拉角微分方程关系简单明了,概念直观,容易理解,解算过程中无需作正交化处理,但方程中包含有三角运算,这给实时计算带来一定困难。而且当俯仰角接近 90°时方程出现退化现象。

3.3.1.3 卡尔曼滤波进行多传感器融合

卡尔曼滤波整体体现为：预测+更新

预测：通过状态转移函数计算当前值

更新：融合当前观测状态：计算两个高斯分布融合后的均值和协方差矩阵

卡尔曼滤波的经典五步：

状态一步预测：$\hat{X}_{k/k-1} = \Phi_{k,k-1}\hat{X}_{k-1}$

状态估计：$\hat{X}_k = \hat{X}_{k/k-1} + K_k(Z_k - H_k\hat{X}_{k/k-1})$, 其中$K_k$就是卡尔曼增益

滤波增益：$K_k = P_{k/k-1}H_k^T(H_kP_{k/k-1}H_k^T + R_k)^{-1} = P_kH_k^TR_k^{-1}$

一步预测均方误差：$P_{k/k-1} = \Phi_{k,k-1}P_{k-1}\Phi_{k,k-1}^T + \Gamma_{k-1}Q_{k-1}\Gamma_{k-1}^T$

估计均方误差：$P_k = (I - K_kH_k)P_{k/k-1}$

3.3.1.4 程序分析

```
void Fusion(float* angles, float* accelerometer, float* compass, float*
gyroscope)
{//1.读取传感器数据
//2. 陀螺仪与地磁算出欧拉角
accel_compass2angle(am, cp, am_angle);
//3.加速度计得到欧拉角
gyro2angle(gy, gyro_angle);
//4.将两次计算欧拉角值带入卡尔曼滤波，数据融合
KalmanFilter(am_angle_mat, gyro_angle_mat);
```

融合结果测试：

```
data is
时间间隔：0ms
手机角度传感器：yaw:0 pit:0 rol:0
其他传感器融合：yaw:0 pit:-0.0536119 rol:0.0111282
时间间隔：2ms
手机角度传感器：yaw:-2.49307 pit:-0.080027 rol:0.12241
其他传感器融合：yaw:-2.49259 pit:-0.0802845 rol:0.12241
时间间隔：23ms
手机角度传感器：yaw:-2.48954 pit:-0.103962 rol:0.139946
其他传感器融合：yaw:-2.48836 pit:-0.104528 rol:0.488373
时间间隔：1ms
手机角度传感器：yaw:-2.58487 pit:-0.103962 rol:0.139946
其他传感器融合：yaw:-2.58382 pit:-0.104537 rol:-0.184426
时间间隔：41ms
手机角度传感器：yaw:-2.58487 pit:-0.103962 rol:0.139946
其他传感器融合：yaw:-2.58382 pit:-0.104529 rol:0.0385902
时间间隔：13ms
手机角度传感器：yaw:-2.62937 pit:-0.103962 rol:0.139946
其他传感器融合：yaw:-2.62842 pit:-0.104527 rol:0.318746
时间间隔：0ms
手机角度传感器：yaw:-2.45132 pit:-0.114216 rol:0.0778747
其他传感器融合：yaw:-2.44983 pit:-0.114997 rol:-0.482932
时间间隔：1ms
手机角度传感器：yaw:-2.45132 pit:-0.114216 rol:0.0778747
其他传感器融合：yaw:-2.44983 pit:-0.114962 rol:-0.201716
时间间隔：59ms
```

3.3.2 基于 PID 控制的四元数法

3.3.2.1 四元数

- 定义：

域 F 上的一个矢量空间 V 叫做域 F 上的代数;如果除数乘、加法外还定义叉乘,如果 V 是 F 上的有限维空间,称 V 为 F 上的有限维代数;如果乘法满足结合律,称 V 为结合代数;实数是一维结合代数,复数是二维结合代数,四元数是四维结合代数,都是可除代数;

以i,j,k表示四元数的基元,四元数的一般形式为:$A=\omega+xi+yj+zk$（ω,x,y,z为实数），基元i,j,k的运算规则为：$i\cdot j=k;i\cdot k=j;j\cdot k=i;$

i,j,k可代表沿x,y,z轴的单位矢量,四元数是标量ω与矢量$V=xi+yj+zk$之和。

- 优势：

四元数法只需求解四个未知量的线性微分方程组,计算量比方向余弦法小,且算法简单,易于操作,是较实用的工程方法。但四元数法实质上是旋转矢量法中的单子样算法,对有限转动引起的不可交换误差的补偿程度不够,所以只适用于低动态运载体(如运输机等)的姿态解算。而对高动态运载体,姿态解算中的算法漂移会十分严重。

3.3.2.2 一阶龙格库塔法

利用龙格库塔公式求四元数的 q0~q3，需要这个周期的角速度 gx，gy，gz 和周期时间 deltaT

```
qa = q0;
qb = q1;
qc = q2;
q0 += (-qb * gx - qc * gy - q3 * gz);
q1 += (qa * gx + qc * gz - q3 * gy);
q2 += (qa * gy - qb * gz + q3 * gx);
q3 += (qa * gz + qb * gy - qc * gx);
```

3.3.2.3 四元数微分

令 q(t) 是一个单位四元数函数，ω(t) 是由 q(t) 确定的角速度。则 q(t) 的导数为：

$$\dot{q} = \frac{1}{2}\omega q$$

在 t+Δt 时刻，旋转可以描述为 q(t+Δt)。在 Δt 过程中，物体坐标系在经过了 q(t) 旋转的前提下，又经过了额外的微小旋转。这个额外的微小旋转的瞬时旋转轴为 ω^=ω/∥ω∥，旋转角度为 Δθ=∥ω∥Δ，可以用一个单位四元数描述：

$$\begin{aligned} q(t+\Delta t) - q(t) &= (\cos\frac{\|\boldsymbol{\omega}\|\Delta t}{2} + \hat{\boldsymbol{\omega}}\sin\frac{\|\boldsymbol{\omega}\|\Delta t}{2})q(t) - q(t) \\ &= -2\sin^2\frac{\|\boldsymbol{\omega}\|\Delta t}{2}q(t) + \hat{\boldsymbol{\omega}}\sin\frac{\|\boldsymbol{\omega}\|\Delta t}{2}q(t) \end{aligned}$$

等式右边第一项是高阶项趋近于 0，可省略。因此

$$
\begin{aligned}
\dot{q}(t) &= \lim_{\Delta t \to 0} \frac{q(t+\Delta t) - q(t)}{\Delta t} \\
&= \hat{\boldsymbol{\omega}} \lim_{\Delta t \to 0} \frac{\sin(\|\boldsymbol{\omega}\| \Delta t/2)}{\Delta t} q(t) \\
&= \hat{\boldsymbol{\omega}} \frac{d}{dt} \sin(\frac{\|\boldsymbol{\omega}\| t}{2})|_{t=0} q(t) \\
&= \hat{\boldsymbol{\omega}} \frac{\|\boldsymbol{\omega}\|}{2} q(t) \\
&= \frac{1}{2} \boldsymbol{\omega}(t) q(t)
\end{aligned}
$$

3.3.2.4 程序分析

- 程序设计流程如下：

```
vector3f v_gravity, v_error, V_error_l;
acc.normalize();//1.重力加速度归一化
vector_gravity(v_gravity);//2.提取四元数的等效余弦矩阵中的重力分量.
v_error = acc % v_gravity;//3.向量叉积得出姿态误差
v_error_l += v_error * Ki;//4.对误差进行积分
Gyro += v_error *Kp +v_error_l;//5.互补滤波，姿态误差补偿到角速度上，修正角速度
Q.  Runge_Kutta_1st(Gyro, deltaT);//6.一阶龙格库塔法更新四元数
Q.normalize();//7.四元数归一-化
Q.to_euler(&angle.x,&angle.y,&angle.z);
```

4. 基于 OpenGL 的可视化人机交互

4.1 开发工具

库	功能
GLEW	OpenGL 扩展库，用于跨平台编译
GLFW	GLFW 是一个 OpenGL 的应用框架,主要用来处理特定操作系统下的特定任务。
imgui	轻量级的 C++开源跨平台图形界面框架，多使用于游戏
jsonxx	C++11 轻量级 config 库，用于 JSON 解析和序列化功能，并和 C++ 输入输出流交互。

4.2 OpenGL 介绍

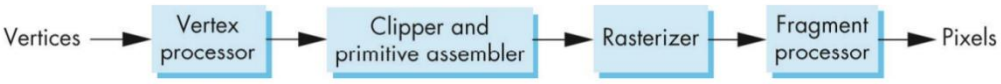
OpenGL 是用于渲染 2D、3D 矢量图形的跨语言、跨平台的应用程序编程接口（API）。这个接口由近 350 个不同的函数调用组成，用来绘制从简单的图形比特到复杂的三维景象。OpenGL 常用于 CAD、虚拟现实、科学可视化程序和电子游戏开发。

4.3 imGUI 介绍

ImGUI 又称为 Dear ImGui，它是与平台无关的 C++轻量级跨平台图形界面库，没有任何第三方依赖，可以将 ImGUI 的源码直接加到项目中使用，也可以编译成 dll, ImGUI 使用 DX 或者 OpenGL 进行界面渲染，对于画面质量要求较高，例如客户端游戏，4k/8k 视频播放时，用 ImGUI 是很好的选择，当然，你得非常熟悉 DirectX 或者 OpenGL，不然就是宝剑在手，屠龙无力。相对于 Qt、MFC、DuiLib、SOUI 等，ImGUI 的拓展性更好，也更轻量级，当然对于开发者的要求也更高。

4.4 OpenGL 关键技术

4.4.1 渲染基本流程



Vertex processor：对顶点进行预处理，原始点都是三维的，最终我们看到的点都是二维的。这个映射过程和点的实际位置，和我们从哪个视角去看都有关系。需要通过一系列矩阵调整点的坐标。另外，还需要确定点的颜色，这个和点到镜头的距离和光照情况都有关系。总之，最终得到的是我们视图上每个点的位置以及它的颜色。

Cipper：我们的视图是有限的，在视图以外的点可以通过算法预先处理掉，不需要进行后续计算。

primitive assemble：图元装配。就是使用一个个小的网格去将表示。

Rasterization：栅格化。就是把一个个几何图形变成像素格形式。

Fragment processor：通过纹理和插值算法给每个像素上色。

4.4.1.1 管线渲染详细步骤

图形渲染管线的第一个部分是顶点着色器(Vertex Shader)，它把一个单独的顶点作为输入。顶点着色器主要的目的是把 3D 坐标转为另一种 3D 坐标（后面会解释），同时顶点着色

器允许我们对顶点属性进行一些基本处理。

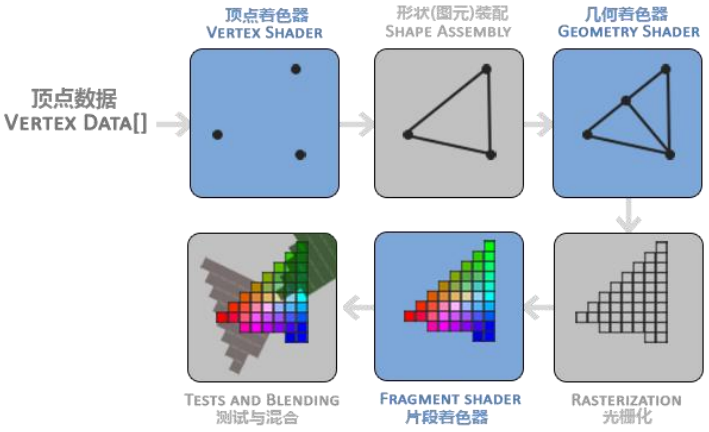
图元装配(Primitive Assembly)阶段将顶点着色器输出的所有顶点作为输入（如果是GL_POINTS，那么就是一个顶点），并所有的点装配成指定图元的形状；本节例子中是一个三角形。

图元装配阶段的输出会传递给几何着色器(Geometry Shader)。几何着色器把图元形式的一系列顶点的集合作为输入，它可以通过产生新顶点构造出新的（或是其它的）图元来生成其他形状。例子中，它生成了另一个三角形。

几何着色器的输出会被传入光栅化阶段(Rasterization Stage)，这里它会把图元映射为最终屏幕上相应的像素，生成供片段着色器(Fragment Shader)使用的片段(Fragment)。在片段着色器运行之前会执行裁切(Clipping)。裁切会丢弃超出你的视图以外的所有像素，用来提升执行效率。

OpenGL 中的一个片段是 OpenGL 渲染一个像素所需的所有数据。

片段着色器的主要目的是计算一个像素的最终颜色，这也是所有 OpenGL 高级效果产生的地方。通常，片段着色器包含 3D 场景的数据（比如光照、阴影、光的颜色等等），这些数据可以被用来计算最终像素的颜色。



4.4.1.2 GLSL（OpenGL 着色语言）

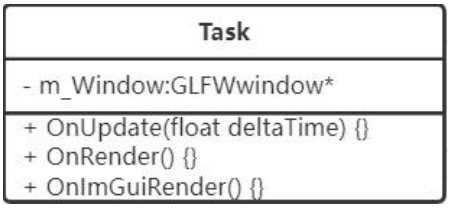
OpenGL 着色语言是用来在 OpenGL 中着色编程的语言，也即开发人员写的短小的自定义程序，他们是在图形卡的 GPU 上执行的，代替了固定的渲染管线的一部分，使渲染管线中不同层次具有可编程性。比如：视图转换、投影转换等。GLSL（GL Shading Language）的着色器代码分成 2 个部分：Vertex Shader（顶点着色器）和 Fragment（片断着色器），有时还会有 Geometry Shader（几何着色器）。负责运行顶点着色的是顶点着色器。它可以得到当前 OpenGL 中的状态，GLSL 内置变量进行传递。GLSL 其使用 C 语言作为基础高阶着色语言，避免了使用汇编语言或硬件规格语言的复杂性。

4.5 程序设计

任务:

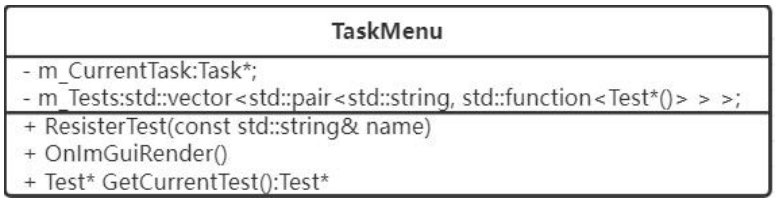
每个任务分为三部分：首先是初始化参数，接着是每隔一段时间更新参数，最后再将图

形渲染出来。渲染的过程分为 OpenGL 本身的渲染和 imGUI 的渲染



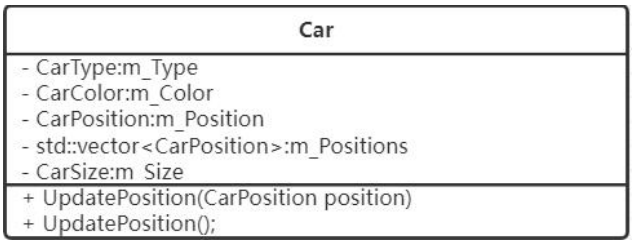
任务菜单：

继承任务，每个任务类型都可以注册在任务菜单中，通过 GUI 交互进行运行。



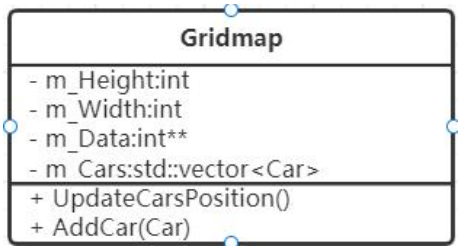
车辆：

每辆车有不同的类型、颜色、大小、ID。UpdatePosition 函数会更新车辆位置，传入具体的 position 参数可以直接修改车辆位置，不传入参数则代表和服务器进行通讯获得新的位置。



地图：

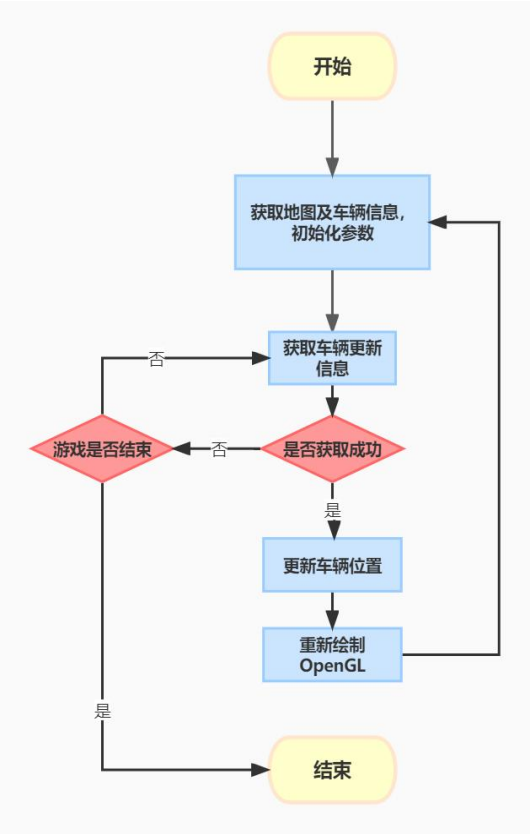
使用 m*n 的栅格地图进行表示，每个单元格的数字表示此处的高度。一张栅格地图关联了多辆车，在初始化阶段对 json 文件进行解析获得车辆信息。每次 opengl 进行重新渲染时会自动调用 UpdateCarsPosition 函数，更新每一辆车的位置。



4.6 程序流程

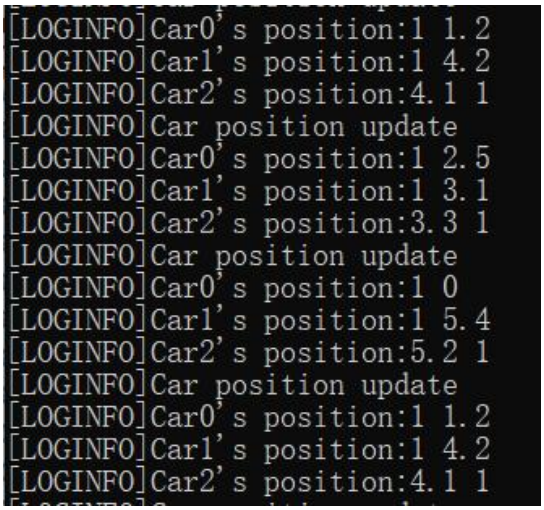
通过鼠标点击任务开始，获取地图信息及车辆信息并初始化参数。每隔 delta 时间，重

新获取车辆的信息，如成功则更新车辆信息，如失败则判断任务是否结束。如任务未结束，继续尝试获取车辆信息。获取车辆信息后，更新车辆位置并重新进行绘制，以达到实时播放动画的效果。

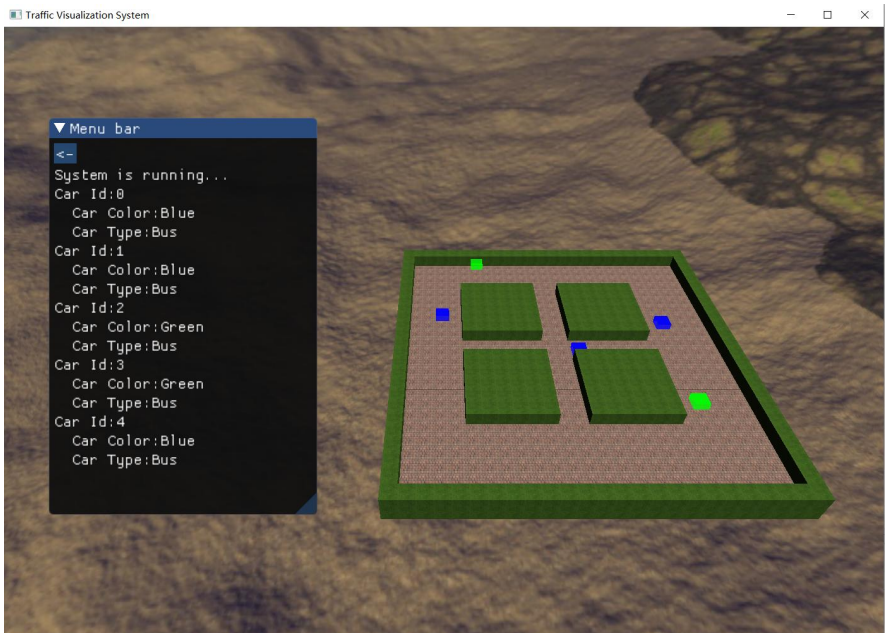


4.7 运行过程截图

后台打印日志：



前端显示可视化地图，以及车辆的基本信息：



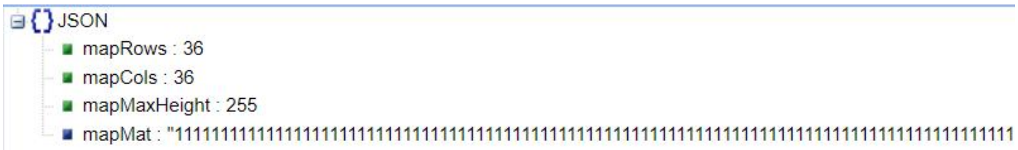
5. 网络通信

网络通信上总体采用的是 Socket 技术+TCP 协议+JSON。

5.1 与可视化 GUI 部分通信

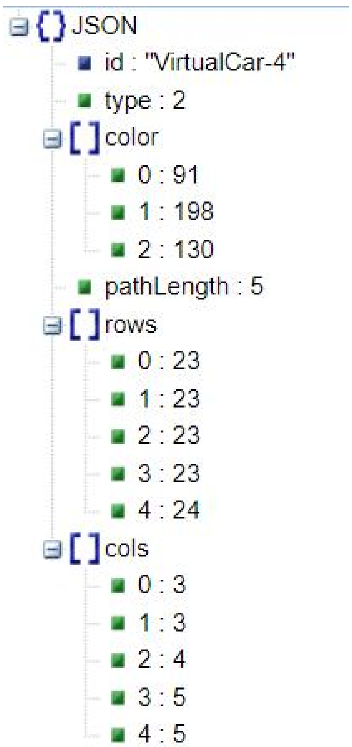
首先需要将地图信息发送给服务端，让 GUI 部分知道一个完整的环境信息。地图的 JSON 消息结构如下图所示：

{"mapRows":36,"mapCols":36,"mapMaxHeight":255,"mapMat":"1111111(中间省略)1111"}



随后就是由快照拍摄线程获取一段时间的各个车辆的信息。再统一将这些信息存储在本地的文件中，同时不断发送给 GUI 进行车辆运动的绘制。一辆车的一次快照信息如下图所示。

{"id":"VirtualCar-4","type":2,"color":[91, 198, 130],"pathLength":5,"rows":[23, 23, 23, 23, 24],"cols":[3, 3, 4, 5, 5]}



5.2 与 Matlab 通信

服务端和 Matlab 进行的是一个双向的数据传递过程。服务端首先会将为 Matlab 模拟车

规划的路径发送给 Matlab，Matlab 通过 Simulink 中实现控制算法控制车辆的运动，使其能够在较短的时间内到达目的地，不与障碍物碰撞。随后 Matlab 完成该段任务后会将实际路径进行回传。

这里传输的 JSON 数据和上面所属的快照在形式上是一致的，这里不再赘述。

5.3 与手机 APP 通信

在与手机进行通信时，我们设计的仿真世界主要是作为一个数据的接收方，被动的接受手机 APP 所传递的各个传感器数据，并在服务端进行数据融合。

每当手机端的传感器事件监听器被触发，则会将当前手机的传感器数据进行发送。发送数据的 JSON 结构如下图所示。

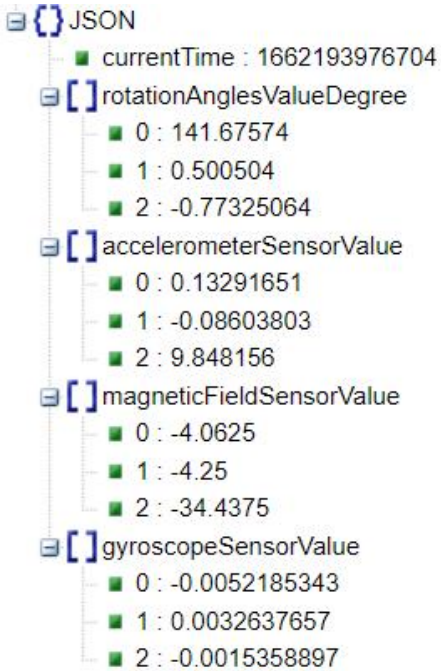
{currentTime:1662193976704,

rotationAnglesValueDegree:[141.67574,0.500504,-0.77325064],

accelerometerSensorValue:[0.13291651,-0.08603803,9.848156],

magneticFieldSensorValue:[-4.0625,-4.25,-34.4375],

gyroscopeSensorValue:[-0.0052185343,0.0032637657,-0.0015358897]}



四、改进思路

本次综合实践设计总体来说，我们组完成的都还可以。但是在答辩后听取老师意见和建

议后还是有很多不足之处需要改进的。

1. 添加物理实体交互，让仿真世界与实际世界中的智能体进行实时交互
2. 更好的人机交互，编写更加友好的交互界面
3. 添加更多的任务种类，以及更多的车辆种类，使得仿真世界更加贴近真实世界的情况
4. 让智能体可以适应更复杂的地图环境，对动态障碍物进行处理
5. 完善仿真世界服务端，让其在避障、多智能体路径规划、任务分配中更加优化
6. 利用更多的传感器（摄像头等）实现目标检测与识别，使得基于智能手机硬件平台的模拟车更加贴近真实
7. 将 SLAM、建图等更加高级的技术与仿真世界进行结合

五、总结

纸上得来终觉浅，通过这次小组合作，我们完成了理论学习到工程实现的跨步，从最开始看到任务时觉得是天方夜谭的造火箭，到学会任务分解、小组合作。不仅是工程能力的提升，也是工程思维的培养，我们 5 个人都收获良多。

感谢学校提供学习机会，也感谢各位老师的精彩授课与耐心解答。

附件 1：需求分析

1.1 需求清单

功能需求

编号	一级需求	二级需求（功能点）
F1	数字世界模拟需求	接收交车请求、智能体系统调度、路径规划
F1.1		模拟车与数字车同步
F1.2		地图维护更新
F2	网络接入需求	能够实时与 matlab 模拟车回传数据
F2.1		能够实时与手机 APP 回传数据
F2.2		

技术需求

编号	一级需求	二级需求（功能点）
T1		能够支持部少于 5 辆模拟车的并发模拟
T2		数据自模拟车（Matlab 模拟或 Android 模拟）到界面上实时显示的延迟不大于 1 秒

1.1 用例设计与场景说明（可选）

附件 2：架构设计

2.1 系统硬件结构图

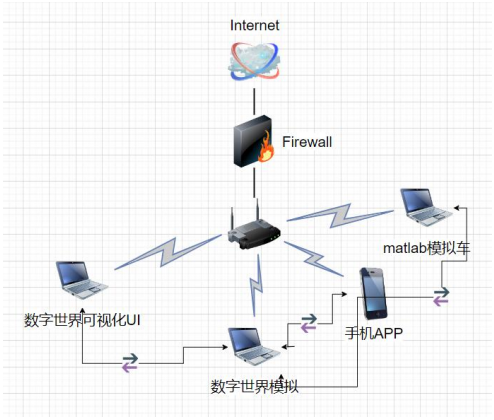


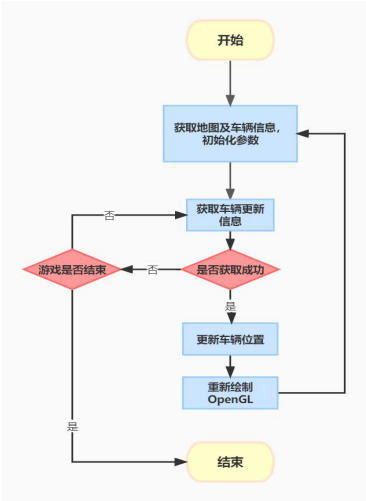
图 2.1 硬件结构图

附件 3：软件概要设计

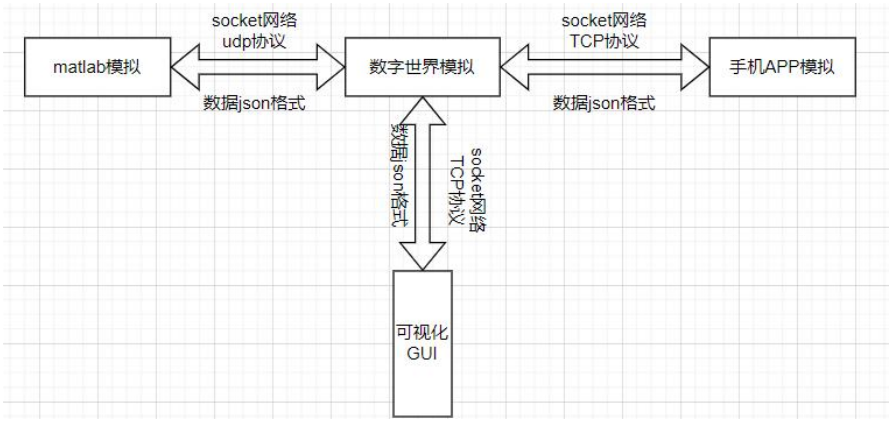
3.1 模块清单及模块功能说明

编号	模块名称	模块功能
1	模拟	接收交车请求，路径规划，系统调度，同步模拟车与数字车，形成并更新维护地图，绑定智能体
2	网络通讯	服务端与 matlab 模拟车传输数据，服务端与手机硬件平台模拟车传输数据，服务端与可视化 GUI 部分传输数据
3	UI	将数字模拟世界可视化

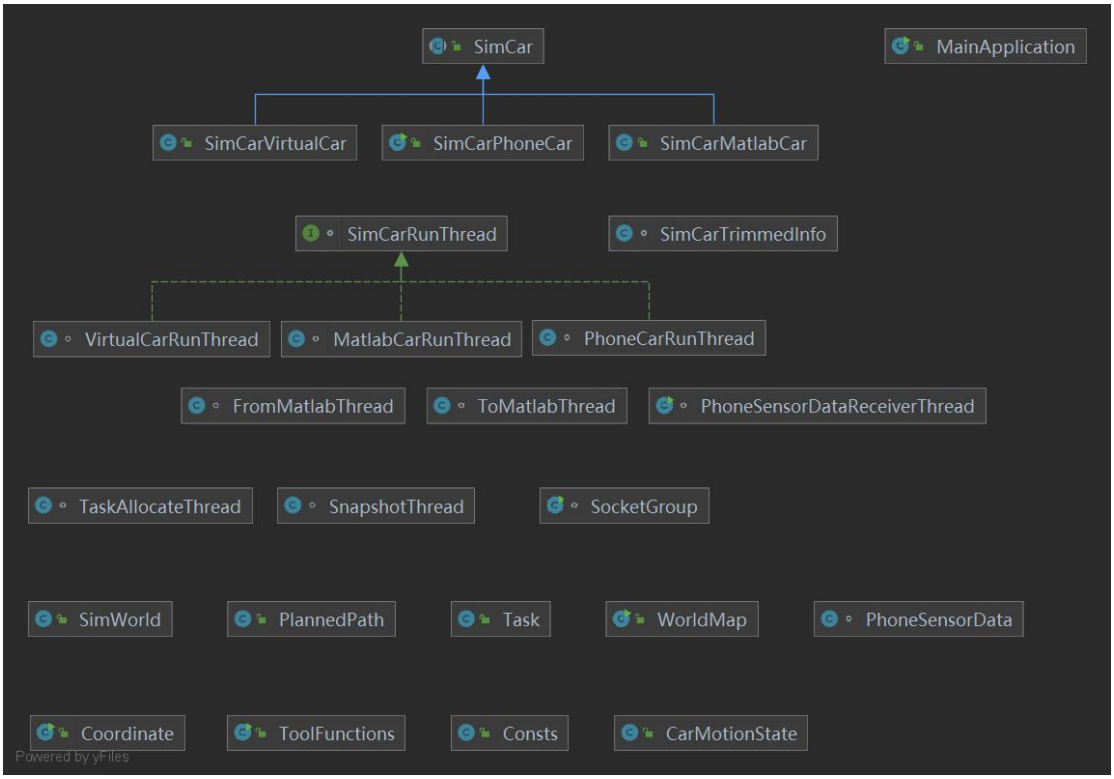
3.2 人机交互设计（UI 设计）



3.4 通信协议设计



附件 4：软件详细设计



附件 5：安装启动和使用说明

手机 APP 部分：

①直接安装 APK 文件即可。

②在 APP 输入部分填入服务器的 IP、Port，点击 Connect 进行连接。
③连接成功后，点击 start 即可发送传感器数据，点击 stop 停止发送。

仿真世界客户端部分：
①运行 MainApplication.java。
②程序会阻塞等待客户端连接。
③待所有客户端连接完成后，程序自动运行。

附件 6：提交物检查清单

所有提交物请收集到一个文件夹下，并压缩成 zip 格式文件提交或通过 email 超大附件方式提交。Zip 包以小组组长学号和姓名命名。如未特别说明，每小组总起来提交 1 份即可。请按照本表格开列的资料清单逐一在表格中打勾确认，没有的条目请标记无。

序号	说明	是否已提交
1	课程设计报告，其中应包含本小组全体同学学号和姓名（即本文档）	是
2	用于演示的 PowerPoint 幻灯片文件，其中应包含本小组全体成员姓名和学号	是
3	报告和演示 PPT 中用到的短视频 Video 文件	是
4	源代码：服务端数字世界模拟引擎、接口以及网络接入服务（请删除不必要的中间临时文件）	是
5	源代码：GUI 监控前端（请删除不必要的中间临时文件）	是
6	源代码：Matlab 模拟车（请删除不必要的中间临时文件）	是
7	源代码：Android 模拟车 （请删除不必要的中间临时文件）	是
8	竞赛奖证书电子版（尽量附竞赛报告，这一项由同学单独提交给任课教师）	是
9	其它你认为有必要提交的资料	无

附件 7：小组获奖情况


第十七届全国大学生
智能汽车竞赛
- 获奖证书 -
同济大学 齐晓燕 同学在2022年第十七届全国大学生智能车竞赛中荣获华东赛区讯飞智慧服务组比赛
二等奖
竞赛组委会:
第十七届全国大学生智能汽车竞赛
组织委员会
吴澄院士
吴启迪
主办单位:
中国自动化学会
周杰
协办单位:
英飞凌科技（中国）有限公司
深圳国芯人工智能有限公司
百度在线网络技术（北京）有限公司
上海灵动微电子股份有限公司
南京沁恒微电子股份有限公司
科大讯飞（苏州）科技有限公司
航天恒材科技（北京）有限公司
上海睿赛德电子科技有限公司
思智浦（中国）管理有限公司
承办单位:
东北赛区 - 沈阳航空航天大学
华北赛区 - 太原工业学院
华东赛区 - 南京信息工程大学
华南赛区 - 哈尔滨工业大学（深圳）
西部赛区 - 乐山师范学院
安徽赛区 - 安徽国际商务职业学院
山东赛区 - 山东科技大学
浙江赛区 - 湖州学院
总 决 赛 - 南京信息工程大学
SMART CAR RACE 2022
2022
第十七届全国大学生
智能汽车竞赛