郑州大学
创新创业基础与工程设计实践项目

VisionVoyage-基于鱼眼相机与其他感知技术的自动驾驶仿真系

概要设计报告

公司名称： IngenuityDrive-创智行科技有限公司

小组编号： 21 级计算机类 09 组

团队成员： 徐梓航 郭顺 徐梦蝶 郑辰乐 陈自豪

赵柏茗 郭晓卿 蔡从轩 华勇 李景尧

指导老师： 程楠

所属学院： 计算机与人工智能学院

编订日期： 2024 年 3 月

郑州大学 ZHENGZHOU UNIVERSITY

目录

1.引言

1.1 项目背景

现对目标系统开发做出总体架构说明，具体项目及要求如下：

项目名称：VisionVoyage

用户：对自动驾驶感兴趣的用户

开发者：IngenuityDrive-创智行科技有限公司

运行环境：Ubuntu 20.04.6 LTS Desktop、Windows 10/11

建议软件最短寿命：3 年

开发工具：前端：PySide6、PyQt6 后端：Python3、C++

项目经理：徐梓航

项目成员：郭顺、郑辰乐、李景尧、郭晓卿、陈自豪、华勇、蔡从轩、徐梦蝶、赵柏茗

开发人员要求：有计算机专业知识的人，且对鱼眼相机以及自动驾驶有所了解。

目标系统采用 C/S 结构。

1.2 文档架构说明

本阶段是在需求分析的基础上，对 VisionVoyage 做概要设计，主要解决了实现该系统需求的程序模块设计的问题。包括如何把系统分成若干个模块、决定各个模块之间的接口、模块之间传递的信息，以及数据结构、模块结构的设计。作为详细设计及系统实现的依据。

1.3 术语和定义

鱼眼相机：指带有鱼眼镜头的相机，是一种焦距极短并且视角接近或等于 180°的镜头。16mm 或焦距更短的镜头。 它是一种极端的广角镜头，“鱼眼镜头”是它的俗称。为使镜头达到最大的摄影视角，这种摄影镜头的前镜片直径且呈抛物状向镜头前部凸出，与鱼的眼睛颇为相似，“鱼眼镜头”因此而得名。鱼眼镜头属于超广角镜头中的一种特殊镜头，它的视角力求达到或超出人眼所能看到的范围。因此，鱼眼镜头与人们眼中的真实世界的景象存在很大的差别，因为我们在实际生活中看见的景物是有规则的固定形态，而通过鱼眼镜头产生的画面效果则超出了这一范畴。

语义分割：语义分割是计算机视觉领域的一种图像分割技术，其目标是将一张图像中的每个像素分配给预定义的类别。与传统的图像分割技术不同，语义分割不仅仅将图像分成若干个区域，而是对每个像素进行分类，从而能够获得更加精细的图像分割结果。它在许多领域中都有广泛的应用，如自动驾驶、医学影像分析、机器人视觉等。

实例分割：实例分割技术可以看做是物体检测技术与语义分割技术的结合。与物体检测技术只能检测出某张图片中某一物体的大概位置不同，实例分割可以将每一个物体的边缘一一划分出来，而且还能够为每一个物体进行分类。

深度：在计算机视觉领域，深度通常指的是一个物体在三维空间中的深度位置，也就是物体到摄像头或者视点的距离。深度图像或深度地图是一种特殊的图像，其像素值代表了图像中每个像素点到摄像头的距离。深度感知能力使得计算机视觉系统可以理解物体的三维空间位置，对于许多

应用如自动驾驶、机器人导航、3D 建模等都是至关重要的。

DVS：DVS（Dynamic Vision Sensor）是一种基于事件的动态视觉传感器，也称为神经形态传感器。与传统的图像传感器不同，DVS 仅在场景中发生变化时才产生输出，而不是连续地捕获整个图像。这种工作方式使得 DVS 能够在高速、低功耗和低延迟的条件下进行操作，并且对光照变化和噪声具有很强的鲁棒性。DVS 常用于机器人、自主驾驶汽车、无人机等智能设备中，可以实现高效的运动检测和目标跟踪，以及快速反应和决策。更新：对数据库的一种操作，用于更改数据库中的数据信息。

光流：光流是空间运动物体在观察成像平面上的像素运动的瞬时速度，是利用图像序列中像素在时间域上的变化以及相邻帧之间的相关性来找到上一帧跟当前帧之间存在的对应关系，从而计算出相邻帧之间物体的运动信息的一种方法。一般而言，光流是由于场景中前景目标本身的移动、相机的运动，或者两者的共同运动所产生的。

逻辑结构图：可用 Visio 框图表述，也可用 UML 的包图和类图表述，包表示在体系结构设计中可以再分解的元素，类表示在体系结构设计中不可再分的元素。

数据持久化：把数据保存到可掉电式存储设备中以供之后使用。大多数情况下，特别是企业级应用，数据持久化意味着将内存中的数据保存到硬盘上加以“固化”。

1.4 参考资料

表 1-1 参考资料

文件编号	文件名称	发表日期	出版单位

ISBN978-7-302-33098-1	《软件工程导论（第 6 版）》张海藩	2013 年 8 月第 6 版	清华大学出版社

2. 设计目标

2.1 关键功能

表 2-1 模块关键功能表

编号	名称	核心功能	必做功能	高风险功能	独特功能
001	普通图像鱼眼化	是	是	否	是
002	拍摄鱼眼数据集	是	是	是	是
003	上传资源进行分割处理	否	是	否	否
004	仿真传感器	否	否	否	否
005	驾驶仿真	是	是	否	是

2.2 关键质量属性

在进行系统架构设计的时候，要注意到一种架构风格，很大程度上与设计者如何满足质量要求的对策有关。需求的功能和非功能两方面都可能有质量要求，具有归纳如下：

①与功能性有关的属性

正确性：是指软件按照需求正确执行任务的能力，这无疑是第一重要的软件质量属性。在软件设计之初，我们花大量的精力讨论业务分析与建

模，讨论产品业务的定义，都是为了保证正确性这一最重要的质量指标。

健壮性：指的是在异常情况下，软件能够正常运行的能力。正确性与健壮性的区别在于，前者是在需求之内描述问题，后者是在需求之外描述问题。健壮性一般有两层含义：首先是容错能力，其次是恢复能力。容错指的是发生异常情况不出错误的能力，而恢复指的是软件发生错误以后能恢复到没有发生错误前的状态的能力。

可靠性：是一个与时间相关的属性，指的是在一定的环境下，在一定的时间段，系统不出现故障的概率。通常用平均无故障时间（MTTF，mean-timetofault）来衡量。

②与非功能性有关的属性

性能：性能是指软件的“时间-空间”效率，而不仅仅是指软件运行速度。换句话说是速度要快而占用资源要少。有人认为随着机器越来越好（CPU、内存），性能优化的必要性下降了，这是不全面的，因为随着机器的升级，软件系统也越来越庞大和复杂，性能优化的压力将更大。

易用性：指的是用户使用软件的容易程度，由于现代人的生活节奏加快，对软件易使用性提出越来越苛刻的要求是无可非议的。关键是易使用是站在用户的角度来说的，开发人员感觉易使用的软件，对用户来说未必。

清晰性：意味着工作成果易读、易理解。一个臃肿不堪的软件系统迟早要出问题。所以简洁是人们“精益求精”结果，而不是潦草应付的结果。

安全性：它的目的是系统应该具备防止非法入侵的能力，这既属于技术问题也属于管理问题。随着软件规模越来越大，出现安全漏洞的机率也就随之升高。

可扩展性：这反映软件适应“变化”的能力，包括需求、设计的变化、算法的改进和变化。当软件规模很大的时候，可扩展性就成为一个非常重要的质量属性，也是系统设计的时候必须着重考虑的问题。

可移植性：指的是软件不经修改（或者稍加修改）就可以在不同软硬件环境中使用的能力。在互联网时代，这个要求所占的重要性越来越大。

（“+”表示行促进列，“—”表示行影响列，“ ”表示行列两种质量属性之间影响不明显）

	持续可用性	性能	可扩展性	安全性	可互操作性	可维护性	可移植性	可靠性	可重用性	鲁棒性	可测试性	易用性
持续可用性								+		+		
性能			—		—	—	—	—		—	—	—
可扩展性		—		—		+	+	+			+	
安全性		—			—				—		—	—
可互操作性		—	+	—			+					
可维护性	+	—	+					+			+	
可移植性		—	+		+	—			+		+	—
可靠性	+	—	+			+				+	+	+
可重用性		—	+	—	+	+	+	—			+	
鲁棒性	+	—						+				+
可测试性	+	—	+			+		+				+
易用性		—								+	—	

图 2-2 质量属性关系矩阵

在考虑关键质量属性的时候，不同角色对质量属性的关注点不同，客户偏向于可靠性、性能、安全性方面，而开发组织则更偏向于可维护性、可移植性、可扩展性等，对不同的关注点都需要引起重视，并不仅仅是满足客户的需求。

在 VisionVoyage 中，在满足客户的需求下，结合开发人员的需求，

确定关键质量属性为安全性，可维护性，可靠性，性能这四个属性。前三个属性受其他属性影响较小，性能几乎要收到各个属性的影响，因此在本软件设计中，主要要确保两个功能的性能，鱼眼数据集的高质量和驾驶仿真的高质量，提高响应速率，保证用户较好的使用感受。

2.3 设计约束

1.业务环境因素约束

①本系统应当可以使客户 24 小时进入并且允许使用各类传感器、管理图像资源以及查看个人信息。

②关注相应的法律法规、专利限制。为本系统设计的算法思想申请专利，保障公司的权益。

2.使用环境约束

①软件提供给全国下载软件使用的用户。

②软件仅限中国使用，并不遍及多个国家。

③使用期间的环境与硬件因素有关，暂不提供云算力支持。

3.构建环境约束

①一个项目的开发需要整个团队的共同努力，因此如果开发团队的技术水平有限、磨合程度不高或者分布在不同城市都会影响整个项目的进度。在项目开发的过程中不仅需要提高个人的能力水平，还需要提高每一位成员的团队合作意识与能力。

②需要顾及开发管理方面、源代码保密工作。

4.技术环境约束

本系统前后端的集成开发环境采用 PyCharm 与 VSCode，编程语言采用 Python3、C++。Python 语言简洁、开发效率高、可移植性强，经过多年的生态建设，Python 有非常成熟的包管理，有利于业务的开发，并且 Python 入门简单。前端开发是基于 PySide6 及 PyQt6 框架用来做软件页面。PySide6 作为基于 Qt 框架的 Python 绑定库，具有出色的可移植性，能够轻松地在多个操作系统上运行。由于其开源的特性，开发人员可以自由地使用、修改和分发 PySide6，而无需支付额外费用或受到法律限制。此外，PySide6 与 Qt 框架高度兼容，使得开发者能够充分利用 Qt 框架的各种功能和特性。它提供了清晰简洁的 API 和丰富的文档和示例，以帮助开发人员快速上手并编写高效的代码。PySide6 还拥有广泛的 Qt 模块和类，包括窗口管理、布局管理、图形渲染、事件处理和数据持久化等功能，能够满足各种应用程序的开发需求。最后，PySide6 拥有庞大的开发者社区和持续的技术支持，能够帮助开发人员解决问题并获得持续的更新和维护。

3.设计原则与决策

3.1 设计原则

该系统旨在解决目前自动驾驶领域存在的一些挑战和问题，例如鱼眼数据集稀缺、鱼眼图像畸变处理、多摄像头组合的语义分割算法等方面的技术瓶颈。

特色功能为：提供普通图像转鱼眼图像、仿真环境下各种传感器的图像获取与处理、虚拟驾驶体验和自动驾驶仿真等功能，我们的软件可以帮助研究人员和工程师更好地理解和应用鱼眼相机和其他传感器数据，从而

提升自动驾驶系统的感知能力和安全性。

3.2 对后续工作的限制

用户若开发良好的自动驾驶算法，可以及时向我司投稿，我司会对良好的自动驾驶算法作者进行奖励并集成到我司提供的自动驾驶算法进行优化，达到一个交互的效果。引进更多的活跃用户去优化自动驾驶算法，保持新颖，保证及时更新。

4.总体架构

VisionVoyage 主要包括五个子模块：主页、上传、获取数据集、驾驶仿真、我的。

其中，主页主要是 VisionVoyage 的主要信息。

上传负责普通图像转语言图像、普通图像拼接成鱼眼图像、上传图像/视频进行分割处理。

获取数据集主要起到获取鱼眼图像及获取其他非畸变传感器图像的功能。

驾驶仿真给用户提供模拟驾驶和自动驾驶两功能。

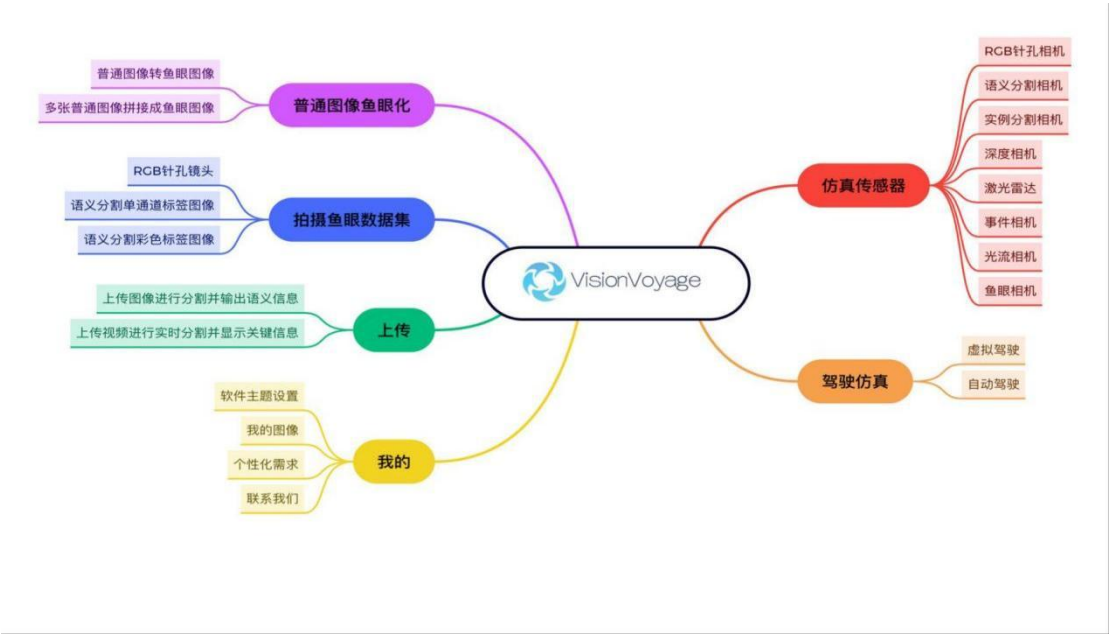

图 4.1 系统功能模块图

5.系统技术架构

本项目将采用 C/S 架构，前后端分离的方式进行开发。

本系统前后端的集成开发环境采用 PyCharm、VSCode 与 pyside6-designer，编程语言采用 Python3、C++。Python 语言简洁、开发效率高、可移植性强，经过多年的生态建设，Python 有非常成熟的包管理，有利于业务的开发，并且 Python 入门简单。前端开发是基于 PySide6 及 PyQt6 框架用来做软件页面。PySide6 作为基于 Qt 框架的 Python 绑定库，具有出色的可移植性，能够轻松地在多个操作系统上运行。由于其开源的特性，开发人员可以自由地使用、修改和分发 PySide6，而无需支付额外费用或受到法律限制。此外，PySide6 与 Qt 框架高度兼容，使得开发者能够充分利用 Qt 框架的各种功能和特性。它提供了清晰简洁的 API 和丰富的文档和示例，以帮助开发人员快速上手并编写高效的代码。PySide6

还拥有广泛的 Qt 模块和类，包括窗口管理、布局管理、图形渲染、事件处理和数据持久化等功能，能够满足各种应用程序的开发需求。最后，PySide6 拥有庞大的开发者社区和持续的技术支持，能够帮助开发人员解决问题并获得持续的更新和维护。

6.逻辑架构

6.1 逻辑结构图

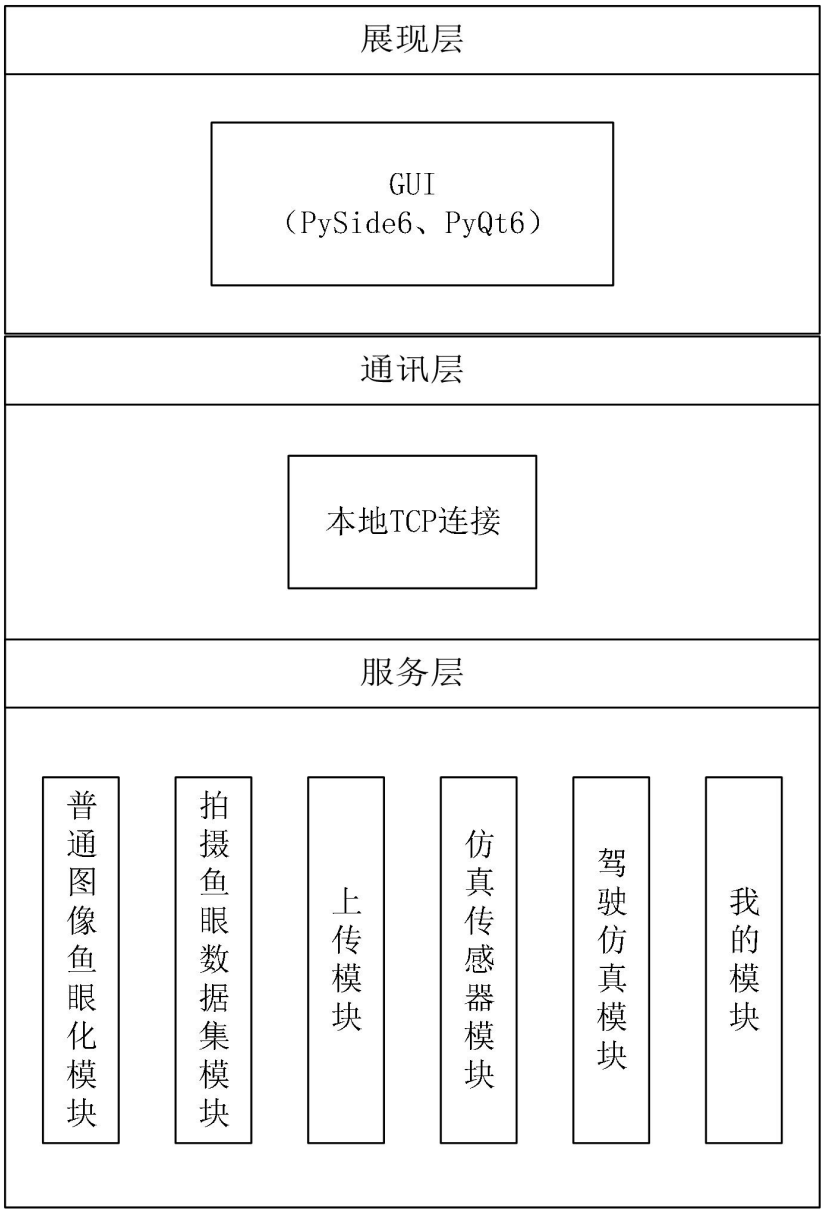

图 6-1 VisionVoyage 逻辑结构图

6.2 各主要组成部分概要描述

整体系统逻辑架构可以分为三个层级，通过层级划分，可以全面展现整体应用系统的设计思路。

(一)　服务层

服务层提供整个系统最核心服务功能，是系统架构的实现部分。分为普通图像鱼眼化、拍摄鱼眼数据集、仿真传感器、驾驶仿真、我的这五个主要模块。

①普通图像鱼眼化：我司设计了一个投影变换算法来将普通图像转为鱼眼图像，一个立方体贴图算法将前后左右上五个视角的图像拼接成一个鱼眼图像，供用户大致了解鱼眼图像和普通真空相机图像的区别，为VisionVoyage 后续功能的使用铺路。

②拍摄鱼眼数据集：VisionVoyage 为研究人员和开发者提供了更多的实验和模拟环境，用户可以根据自己的需求选择需要拍摄的鱼眼数据集类型。

③仿真传感器：VisionVoyage 提供了 RGB 针孔相机、语义分割相机、实例分割相机、深度相机、激光雷达、事件相机、光流相机、鱼眼相机等众多自动驾驶感知技术所需传感器，供用户选择自己所需传感器来进行学习研究。

④驾驶仿真：VisionVoyage 提供了仿真环境，用户可以选择用键盘控制汽车行驶或者自动驾驶，汽车压线或者碰撞时，VisionVoyage 会根据激光雷达和相机返回的数据来判断并提醒用户，汽车时速、帧率、坐标等关

键信息也会展示在窗口上，也提供天气转换、转换摄像头视角、更换车辆等功能。

⑤我的：“我的”模块是软件中的个性化定制中心，为用户提供定制化服务和个性化体验。首先，用户可以根据自己的喜好选择适合自己的软件主题，实现暗色和亮色两种风格的切换。其次，用户可以通过“我的图像”功能查看经过软件图像转鱼眼操作的图片以及原始图片，带来全新的视觉体验。此外，用户还可以提交个性化需求，我司会使用 UE4 Editor 进行需求定制所需的仿真地图和车辆模型等 UE 资产，以获得独特的自动驾驶仿真体验。最后，用户可以通过“联系我们”功能方便快捷地与我们取得联系，提出需求、问题或意见，我们将及时回复并为用户提供所需的支持和帮助。

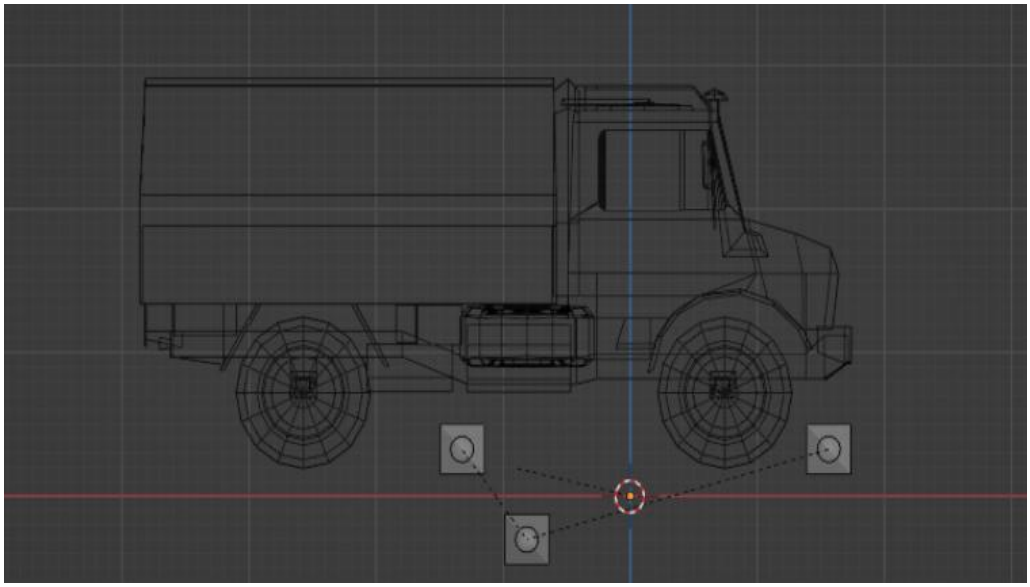

图 6-2 定制车辆模型

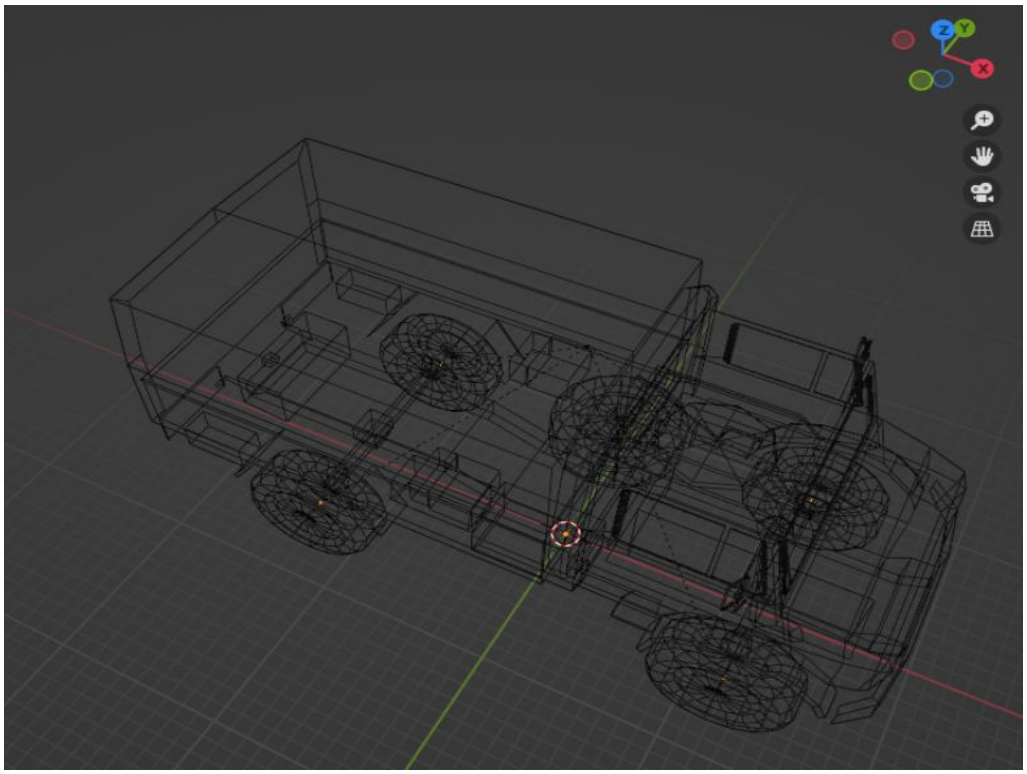

图 6-3 定制车辆模型

(二)　通讯层

TCP 通信与 HTTP 等应用层协议不同，它更加底层，提供可靠的、面向连接的数据传输。

在本地 TCP 通信中，服务器端负责监听指定的端口，并接收来自客户端的连接请求。而客户端则主动发起连接请求，与服务器建立连接。这种连接是全双工的，允许双方进行双向的数据传输。

在 UE4 中，我们使用 Socket 类库实现本地 TCP 通信。服务器端程序创建一个监听套接字，并指定一个可用的端口号，以便等待客户端的连接请求。一旦有客户端连接成功，服务器端就可以通过该连接与客户端进行数据交换。

客户端程序需要指定服务器的 IP 地址和端口号，然后创建一个套接字

并发起连接请求。一旦连接建立成功，客户端就可以通过该连接向服务器发送数据或者接收服务器发送的数据。我们使用异步套接字操作来实现非阻塞式的通信。这样通信过程不会阻塞游戏主线程，从而提高程序的性能和响应速度。

(三)　展现层

系统最终展现在用户面前的，是程序显示。具有多种环境匹配的功能，适合多种操作系统的操作。

7.关键质量属性设计

7.1 安全性设计

在设计基于 UE4 本地 TCP 连接的项目时，我们深入考虑了系统的安全性需求，尤其关注用户数据的隐私保护以及 TCP 连接过程中的网络安全风险。具体措施如下：

（1）用户数据安全：

①权限控制：对于任何可能涉及用户信息交互的部分，我们采用严格的权限管理和操作日志记录机制。限制非必要的信息访问，所有涉及用户数据的操作均需记录操作员 ID、操作时间及操作内容，确保任何操作可追溯。

②通信加密：本地 TCP 连接过程中，我们拟采用可靠的加密算法，如 AES 等，对传输的数据进行加密处理，确保即使在本地网络环境中，用户数据也能得到充分保护，防止未经授权的访问和篡改。

（2）网络安全设计：

①通道安全：在 UE4 中构建 TCP 连接时，我们设定专有的身份验证流程，

确保客户端与服务端之间的连接只能由合法实体建立。同时，采用分层通信架构，限制低安全级别的进程无法访问高安全级别的通信内容。

②防火墙规则：我们强调在同一设备上的不同应用间的访问控制。遵循最小权限原则，仅为必要的应用赋予监听和建立 TCP 连接的权限。

③病毒防护：集成可靠的防病毒解决方案，确保在项目运行环境中具备实时病毒扫描能力，定期更新病毒库，防止潜在的恶意代码通过 TCP 连接传播。同时，对项目内的资源和数据进行定期完整性校验，一旦发现异常立即采取应对措施。

7.2 可靠性设计

系统可靠性定义是系统在规定的时间内及规定的环境条件下,完成规定功能的能力,就是系统无故障运行的概率。具有 4 个主要子特征：

- 成熟性：指系统避免因错误的发生而导致失效的能力
- 容错性：在系统发生故障或违反指定接口的情况下,系统维持规定的性能级别的能力
- 易恢复性：系统发生失效的情况下,重建规定的性能级别并恢复受直接影响的数据的能力
- 依从性：系统依附于可靠性相关的标准、约定和规约的能力。

通常,提高系统的可靠性采用冗余技术、软件容错技术、双机容错技术和集群技术，具体可靠性设计如下：

(1) 容错设计技术对于软件失效后果特别严重的场合

- 恢复块设计：选择一组操作作为容错设计单元,从而把普通的程序

块变成恢复块。一个恢复块包含若干个功能相同、设计差异的程序块文本,每一时刻有一个文本处于运行状态。一旦该文本出现故障，则用备份文本加以替换，从而构成“动态冗余”。

- N 版本程序设计:N 版本的核心是通过设计出多个模块或不同版本，对于相同初始条件和相同输入的操作结果,实行多数表决，防止其中某一模块/版本的故障提供错误的服务,以实现软件容错。

- 冗余设计：冗余设计技术实现原理是在一套完整的软件系统之外,设计一种不同路径，不同算法或不同实现方法的模板或系统作为备份，在出现故障时可以使用冗余的部分进行替换，从而维持软件系统的正常运行。

(2) 检错设计

- 检测对象：即检测点和检测内容。在设计时应该考虑把检测点放在容易出错的地方和出错对软件系统影响较大的地方，检测内容选取那些有代表性的、易于判断的指标。

- 检测延时：在软件检测设计时要充分考虑到检测延时，如果延时长到影响故障的及时报警,则需要更换检测对象或检测方式。

- 实现方式：最直接的一种方式是判断返回结果，如果返回结果超出正常范围，则进行异常处理。计算运行时间也是一种常用的技术，如果某个模块或函数运行超过预期的时间，可以判断出现故障。另外还有置状态标志位等多种方法，自检的实现方式要根据实际情况来选用。

- 处理方式：大多数检测采用“查出故障-停止软件系统运行-报警”的处理方式，但也有采用不停止或部分停止软件系统运行的情况,这一般由故障是否需要实时处理来决定。

(3) 降低复杂性设计

- 在保证实现软件功能的基础上，简化软件结构，缩短程序代码长度，优化软件数据流向，降低软件复杂度从而提高软件可靠性。