

车载软件开发基础

课后实践3



**2022至2023学年第 1 学期**

|  |  |
| --- | --- |
| 学号 | 姓名 |
| E2021060 | 彭星宇 |
| 任课教师 | 刘骥 |
| 成 绩 |  |

|  |  |
| --- | --- |
| 任务书 | |
| 任务内容 | 请针对课后实践2的内容，补充多线激光雷达和底盘模块，使用订阅者模式，实现以下功能。   1. 多线激光雷达作为主题方，发布主题，并接受订阅 2. 底盘模块中增加观察者，向雷达模块进行订阅注册，并接收订阅通知。 3. 当雷达更新障碍物状态时，底盘模块通过订阅者模式接收通知并执行对应操作。 4. 障碍我状态为“前方“，则底盘执行”后退“执行 5. 障碍我状态为“右前方“，则底盘执行”左转“执行 6. 障碍我状态为“左前方“，则底盘执行”右转“执行   注意：  1、底盘执行指令，只需要在屏幕输出类似“左转。。。。”文字即可。  2、障碍物状态可通过命令行方式输入。比如1代表前方障碍，2代表左前方障碍，3代表右前方障碍。 |
| 程序规范 | （1）所有程序代码采用C++编写，使用git进行源代码管理；  （2）类名、变量名、函数名应符合C++的命名规范，并在代码中前后保持一致；  （3）涉及面向对象的程序，例如自定义的类，应符合面向对象的设计原则；  （4）正确使用头文件和源文件，自定义的头文件应符合头文件的编写原则，例如用条件宏定义确保头文件不被多次引用、不在头文件中进行类和函数的实现（模板除外）； |
| 报告要求 | （1）报告至少应该包括程序设计、程序效果展示、总结分析3个部分；  （2）程序设计描述组成程序的模块、类、函数以及他们之间的相互关系，若有算法，可以描述算法流程；  （3）程序效果展示除了程序运行效果截图之外，应该有必要的文字说明；  （4）总结分析可以分析实现的效果与理想情况的差异，分析导致这些差异的原因，切忌不要写成心得体会；  （5）报告应该格式规范、排版整洁、少语病和错误。 |
| 作业提交 | （1）含有git仓库（有.git目录）的完整源代码；  （2）任务报告。 |
| 评分标准 | 按照五级制打分，分为优秀、良好、中等、及格、不及格，各评分项占总成绩的比例为：  （1）任务完成情况占评分的60%；  （2）报告占评分的40%。  评分老师根据各部分的完成情况，直接给出总成绩。 |

**正文**

1. **程序设计**

（1）：程序采用面向对象设计，核心类和模块之间相互协作，主要类与函数如下：

|  |  |  |
| --- | --- | --- |
| 1. | Subject类实现主题代码 | Subject类是主题，它维护了一个观察者列表m\_observers。它有三个主要的方法：attach、detach和notify。attach方法用于添加一个观察者到列表中，detach方法用于从列表中移除一个观察者，notify方法用于通知列表中的所有观察者。 |
| 2. | Observe类实现订阅者抽象类 | Observer类是观察者，它有一个update方法，这个方法在接收到主题的通知时被调用。 |
| 3. | Lidar类继承Subject类 | 当它们的状态发生变化时，它们会通过Subject类的notify方法通知所有的观察者 |
| 4. | Chassis类继承Observe类 | 实现了update方法 |

**2.程序运行截图**

程序入口：文本

描述已自动生成 根据msg打印消息：

文本

描述已自动生成

**3.总结分析**

这个程序是观察者设计模式的一个典型实现。它允许我们在不修改观察者的情况下改变主题，使得主题和观察者之间的耦合度降低，提高了代码的可维护性。

然而，这个程序也有一些限制。例如，如果我们有大量的观察者，那么通知所有的观察者可能会花费很多时间。此外，如果观察者的update方法执行的操作很复杂，那么程序的性能可能会受到影响。

为了解决这些问题，我们可以考虑使用更高效的数据结构来存储观察者，或者在notify方法中使用多线程来并行通知所有的观察者。此外，我们也可以考虑将观察者的update方法的执行推迟到后台线程，以避免阻塞主线程。

[Github](https://github.com/pxy522/Cqu_EIE_projects/tree/main/Object-Oriented)