Git + AI + LaTeX Report

2D_Orange

2025 年 11 月 17 日

目录

1 Project Introduction

This is a project that develops a simple **2048 game** using **python**. In addition to the basic features, the game contains the following 3 new features.

- Variable game board size
- Game score record file
- Configuration file

Google Gemini and **Deepseek** were mainly used to generate genius ideas on game features and detailed code implementation. The report was written in **Overleaf**.

2 AI Prompts Used

Below are 6 AI prompts used during development, though they were written in Chinese.

Prompt 1: "现在我需要利用ai工具开发一个2048小游戏，并生成其python程序代码等。
不过在这里，我只想了解如何使用合适的提示词来引导ai生成尽可能完整的源码文件"

Prompt 2: "请用Python创建一个完整的2048游戏，要求：
1. 定义Game2048类，包含：4x4游戏板初始化、移动和合并逻辑、分数计算、
游戏状态检查
2. 实现以下方法：
__init__(): 初始化游戏
add_new_tile(): 在空白位置添加新数字
move(direction): 处理移动逻辑
is_game_over(): 检查游戏结束条件
print_board(): 显示当前游戏板
3. 主程序包含：游戏循环、用户输入处理(wasd控制方向)、胜利/失败提示
4. 额外功能：实时显示分数、游戏板彩色显示、移动有效性检查
请提供完整可运行的代码，包含所有必要的import语句"

Prompt 3: "游戏写得很好！现在我要加一项功能，在游戏开始前提示"输入棋盘大小:"，
要求玩家往命令行中输入一个整数，并存储于size变量中，代表棋盘的大小。限制
输入整数3<=size<=8，超出范围或格式不规范则要求重新输入，不键入则使用缺省
设置size=4。输出需要修改部分的代码，并确保不影响整个程序原功能的正常工作"

Prompt 4: "每次输入棋盘大小还是有点麻烦。现在需要创建一个和py文件同目录的
config.yaml文件，用以存储配置信息。在该yaml文件中填写board_size的值，
并用注释提示“range between 3 and 8, default is 4”。在python代码中将

yaml文件定义的board_size值传入，并实现限制范围和缺省设置的功能（输入为空也进行缺省设置），最终传给size变量。输出需要修改部分的代码，并确保不影响整个程序原功能的正常工作"

Prompt 5: "发现一个问题。如果读取发生错误，程序会打印提示信息，但马上会被打印棋盘的函数抹除，导致玩家看不到这些提示信息。修改一下程序在游戏开始前的行为，在打印提示信息后等待3秒再开始游戏并打印棋盘。输出需要修改部分的代码，并确保不影响整个程序原功能的正常工作"

Prompt 6: "最后再添加一个功能。使用record.csv文件记录每次游玩结束时的信息，文件第一列数据为游戏结束时的系统时间，格式为YYYY/MM/DD HH:MM:SS，第二列数据为游戏结束时的分数，第三列数据为游戏成功与否，成功则写入字符“+”，失败则不写入。如果没有record.csv文件，则先创建文件再写入信息；如果有record.csv文件，则在文件最后另起一行记录新的信息。输出需要修改部分的代码，并确保不影响整个程序原功能的正常工作"

3 Git Workflow

Firstly, a folder named **gqt_ws** was initialized as the repository and used to contain code files.

```
cd gqt_ws
git init
```

Add and commit the files every time after updating them.

```
git add .
git status
git commit -m "message"
```

After the first commit, create the branch **code**.

```
git branch code
git checkout code
```

After completion of game development, access the project repository on Github and fork it. Then push the branch **code** to the forked repository.

```
git remote add origin https://github.com/2D-Orange/Git_Quick_Task.git
git push origin code
```

Eventually, sync fork and finish the pull request on Github, so that the branch **code** is merged into **main**.

The git logs are as follows:

```
$ git log --oneline
5af0514 Add a feature to record game results
654b58a Fix not displaying error info before game
af6006d Add the config file
5ed306b Add a feature to change board size
f6e1556 Create the code file
```

4 Reflection

Through the project, I learned to:

- Use AI to assist in development, especially in code implementation
- Manage code with Git from scratch
- Write something and compile with LaTeX from scratch