

# RTS Final Report

組長:李家佑 資訊 113 F74096221

組員:楊育勝 資訊 113 F74094067、林昱崴 資訊 113 F74094025

## 一. 如何建置程式

環境與套件：

測試環境：Windows 11

程式語言：Python 3.10.6

Python 套件：Json5-0.9.8、PyQt5-5.15.7、keyboard-0.13.5

前置作業：

```
$ git clone https://github.com/Dannyyang0329/RTS-final-project.git
```

```
$ pip install -r requirements.txt
```

執行程式：

1) 以 GUI 形式執行（限 Windows）

雙擊 main\_window.exe 檔，即可運行程式。

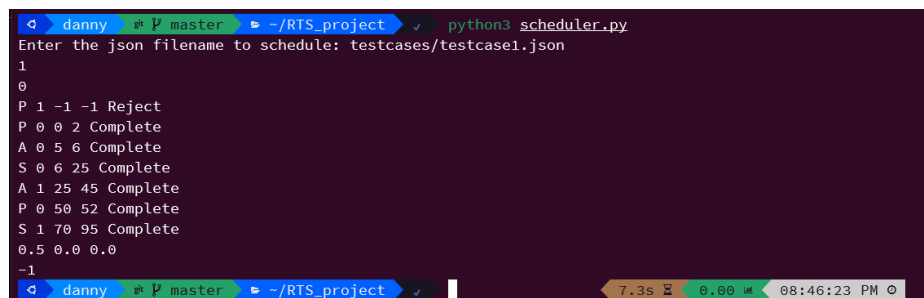
2) 以終端機形式執行

```
$ python3 scheduler.py
```

## 二. 測資的輸入與輸出

1) 終端機形式的輸入與輸出

當以 python 執行 scheduler.py 程式時，程式會請求使用者給定一個 JSON 檔作為輸入，等程式接收到檔案後，便會開始 schedule，並以正確的形式輸出到終端機畫面，如圖一。

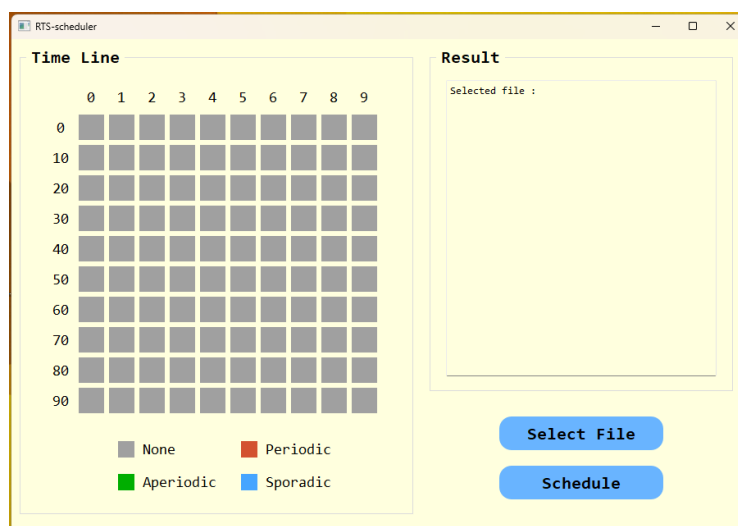


```
danny P master ~/RTS_project python3 scheduler.py
Enter the json filename to schedule: testcases/testcase1.json
1
0
P 1 -1 -1 Reject
P 0 0 2 Complete
A 0 5 6 Complete
S 0 6 25 Complete
A 1 25 45 Complete
P 0 50 52 Complete
S 1 70 95 Complete
0.5 0.0 0.0
-1
```

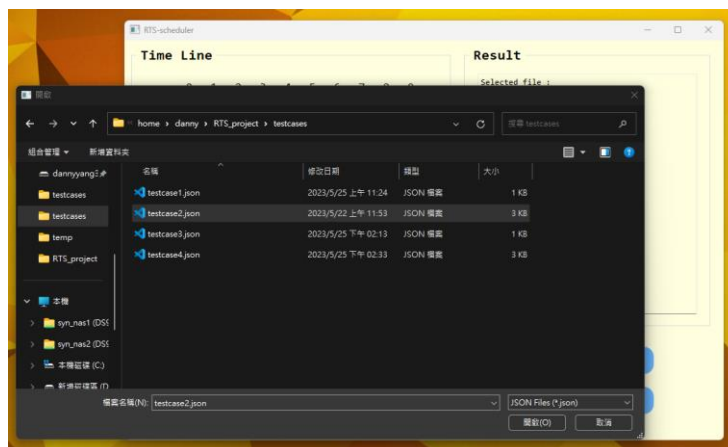
（圖一）終端機形式的輸入輸出範例

## 2) GUI 形式的輸入與輸出

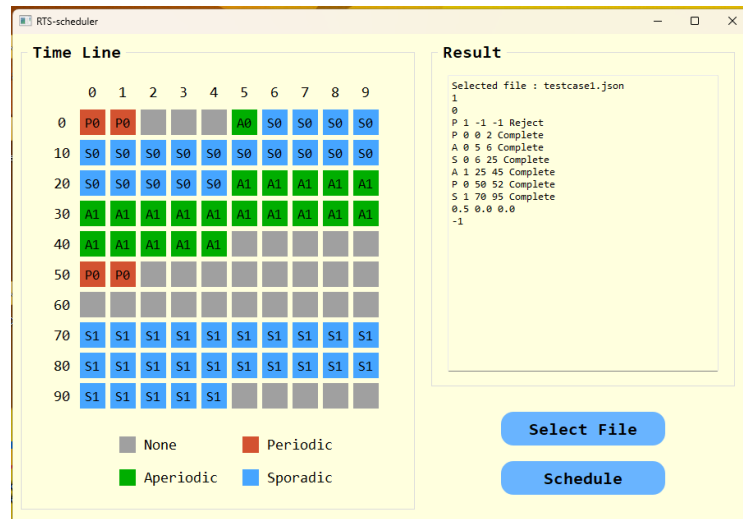
在成功運行 main\_window.exe 這個程式時，會看到圖二的畫面，此時需要透過點擊 Select File 這個按鈕來給定輸入(圖三)，之後便可以點擊 Schedule 的按鈕開始排程，排程的輸出會顯示在 Result 的文字方框中，除此之外，排程的結果也會以視覺化的方式一步一步呈現給使用者(圖四)。



(圖二) GUI 介面的程式



(圖三) GUI 介面的輸入



(圖四) GUI 介面的輸出

### 三. 排程的想法

就設計遊戲引擎排程器的角度思考，我們需要排的工作可以分三個類型：

- **Periodic** ：遊戲畫面刷新、更新角色狀態、計時器等。
- **Sporadic** ：按鈕事件、遊戲內玩家間的訊息傳遞等。
- **Aperiodic** ：自動存檔、優化畫面渲染、效能優化等。

對於三個類型而言，Periodic 的工作是否 miss deadline 會對玩家的遊玩體驗有最大的影響，若畫面沒有辦法及時更新，Sporadic 與 Aperiodic 的工作也無法達成預期的結果。例如，在遊戲中兩個玩家正在透過聊天室討論戰術(Sporadic job)，此時若是畫面沒有即時更新(Periodic job)，那麼此時雙方可能都沒有辦法看到訊息，嚴重影響遊戲體驗。此外，由於 Aperiodic 的工作主要是提升玩家的遊戲體驗，若是 miss deadline，產生的影響遠遠不及沒有完成 periodic 或 sporadic 的工作，因此我們給予三種類型的工作不同的優先級

- Periodic job > Sporadic job > Aperiodic job

因為有些工作本身就不適合被中斷(例如計時器)，也有可能因為工作的 interrupt 造成額外的資源消耗，因此我們把每個工作都視為 non-preemptive 的。也就是說每個工作都會在一段連續的時間區間內執行，不會中途被中斷。

#### 四. 程式的流程與排程的邏輯

1. 使用者給定 json 檔，使用 json5 這個套件幫助我們把讀取 json 檔的內容。
2. 將 json 檔讀取到的內容中，取出一筆 TaskGroup，解析並存到三個型態為 List 的物件中。
  - 例如，sporadic\_jobs = [[6, 25, 19, S0], [70, 95, 25, S1]]，每個元素對應到一個工作，每個工作的索引分別代表 arrival\_time、deadline、execution\_time 和 job\_name。
3. 將三種類型的工作排序。
  - periodic\_jobs：根據 deadline，由小到大排序。
  - sporadic\_jobs：根據 deadline，由小到大排序。
  - aperiodic\_jobs：根據 arrival\_time 由小到大排序。
4. 開始排程：根據 Earliest Deadline First
  - 遍歷 periodic\_jobs，若是無法在當前剩餘的時間執行完，拒絕該工作，反之，接受工作並在 timeline 預留時間。
  - 遍歷 sporadic\_jobs，若是無法在當前剩餘的時間執行完，拒絕該工作，反之，接受工作並在 timeline 預留時間。
  - 遍歷 aperiodic\_jobs，若是無法在當前剩餘的時間執行完，拒絕該工作，反之，接受工作並在 timeline 預留時間。
5. 輸出排程結果(reject 或 complete)，並根據接受與拒絕的比率，輸出三種類型排程的統計狀況。
6. 若是當前 json 檔還有尚未排程的 TaskGroup，回到第二步繼續執行，反之走到第七步。
7. 輸出結束符號 -1。

#### 五. 我們學到什麼

在這次的 Project 中，讓我們能夠不單單只是在上課中利用聽講的方式學習 Real-time System，而是能夠透過程式去實作出一款 Real-time System，而在實作的過程中，也才發現將理論化為實體並不是一件容易的事，在設計排程的時候其實都是要根據實際的情況做取捨的，因為時間跟資源都是有限的，沒辦法全部都做好。另外也除了將作業最基本的要求完成外，同時也思考了該怎麼樣的呈現結果才能讓人最一目了然，因此還另外設計了 GUI 介面，使我們在設計 Real-time System 的同時，也學習到了 GUI 的設計與利用，最後就是特過這個 Project，也學習到了小組之間的互相合作，才能合力完成這一份作業。

## 六. 參考資料

- Earliest Deadline First (EDF) :  
[https://en.wikipedia.org/wiki/Earliest\\_deadline\\_first\\_scheduling](https://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling)
- Game Loop :
- <https://www.oreilly.com/library/view/opengl-game-development/9781783288199/ch01s02.html>
- 老師的投影片

## 七. 補充

- Github 連結 : <https://github.com/Dannyang0329/RTS-final-project>