Task 1

Writing Pattern Generator description

Writing Pattern Generator is an iterator class that generates writing patterns and creates a bin frame file for each pattern.

It stores a path to the YAML configuration file, an internal iterator that iterates patterns, and a dictionary containing information about the pattern currently being processed.

Writing Pattern Generator implements the following public functions:

- Init – used to initialize the generator with a YAML config file.
- Next – used to generate the next writing pattern and create a FRAMES.bin file by writing frames one by one into the file.
- Iter – used to get an instance of the Writing Pattern Generator.

See docstrings for a more detailed description of each function.

Additional questions

How to test the system?

- By looking at the contents of the FRAMES.bin file in a hex editor like HxD.
- By writing unit tests for each function of the Write Pattern Generator.
- By adding a functionality to get statistics of each simulation run and comparing the expected results with the actual simulation run statistics. For example (assuming THRESHOLD: 3) from comments in one of the input files:

```
# FAST SEQUENTIAL WRITE ABOVE THRESHOLD
# MEMORY WRITES: 4 (>THRESHOLD)
# TOTAL DURATION: 17 SECONDS BEFORE FAILURE (<DELTA)
# TOTAL FRAME COUNT IN FLASH: 13 (ONLY MW1 AND MW2 ARE FLUSHED TO FLASH)
# MEM0RY WRITE AVERAGE SPEED: 1.08 FPS
# AVERAGE SPEED WITH HTATs: 0.76 FPS
# STATUS: FAILURE
```

How would your implementation change in the case of multiple CPUs?

- I would implement a preprocessing script that runs multiple simulations with different configurations in separate processes in parallel. No inter-process communication is required. Furthermore, it is crucial to ensure that the simulations do not affect each other, are separated as much as possible, and share as few resources and context as possible.