Exercise 2 - Implement a multi-threaded jobsystem in C++ (40 pts)

Now that we are experienced with caches, multi-threading and the theory behind job systems, this assignment deals with a practical application of this. The goal of the assignment is to implement a jobsystem in C++ based on the theory we learned during the lecture.

Tasks

- ☐ Implement a scheduler-based jobsystem in C++
- ☐ Support job dependency configuration (Physics → Collision → Rendering, as stated in the assignment template). It should be allowed to re-configure dependencies when adding a job to the scheduler (no runtime switching necessary)
- ☐ Allow the jobsystem to be configured in regards to how many threads it can use and automatically detect how many threads are available on the target CPU. Choose an ideal worker thread count based on the available CPU threads and comment why you choose that number.
- ☐ Ensure the correctness of the program (synchronization, C++ principles, errors, warnings)

Bonus points

- ☐ Implement the work-stealing algorithm to use the available resources more efficiently (**+ 10 pts**)
- ☐ Implement the work-stealing queue (requires the above bonus task) with lock-free mechanisms (**+5 pts**, but please really only try this if you feel confident and if your solution is already **working** with locks, **don't** go for this for the first iteration)
- ☐ Allow configuration of the max. worker thread count via command-line parameters and validate them against available threads (capped) (**+2 pts**)

Facts

➔ You can work in groups of three students (as always add **all** names and UIDs to the file(s))

➔ Hand in the solution file (including all source files you used) on moodle (zipped)

➔ Hand in a **short executive summary** on how your jobsystem is supposed to work and which features it supports (work-stealing, lock-free, …)

This can be either max. 1 A4 page additionally, a readme or comments in the source code (but if they are comments, please be sure they explain your reasoning for doing something)

➔ Hand in on time - late assignments can only receive a maximum of 50% of the available points

➔ Ask questions in moodle or among your peers when unsure - don’t despair, you got this!