Task 24 Spike: Performance Measurement

CORE

Context

Performance can be a very important consideration for games, as many games try to create an immersive experience and push the boundaries of technology to calculate and present details. When developers write code, they should be aware of performance issues related to the structure of their code. There are often trade-offs between different considerations.

Knowledge/Skill Gap:

Developers need to be able to collect and analyse software performance data. Sometime this can be done very simply, and it is often a quick way to test and decide on a code design choice. Once measurements have been made, the developer can identify performance issues and make improvements, where needed, supported by data.

Goals

Your overall goal is to create working code, and a simple spike report, that you could share with a colleague so that they would know how to measure code performance directly (in your code, not with IDE or other tools at this stage).

To help you do this, and to get used to using the spike problem-solving approach, we have provided sample code for this task. See the unit website for the supporting files and resources.

Demonstrate the following performance and measurement concepts (in any order that matches your work, not the code order necessarily).

1. **Single Tests**: Demonstrate how to measure both single and multiple function execution time.
2. **Ramp-up Test**: Execute and show, with numbers and **a chart**, both linear and exponential ramp-up testing of function execution time. Is there a difference to ramp-down tests?
3. **Repeatability**: Show, with numbers and **a chart**, how ***repeatability*** will vary depending on test conditions.
4. **Function Comparison**: There are two "char in string" counting functions provided (code sample 1). Clearly show the difference in performance (if any), and check if the execution time difference is linear with respect to string length (size).
   (Note, you will probably want to create random strings of the various size to test with.)
5. **IDE Settings**: Show what, if any, is the difference in execution time between ***debug*** settings and ***release*** settings. (Remember to have a task that runs for long enough that it matters.)
6. **Compiler Settings:** Turn down/off compiler optimisation and demonstrate a difference. (Make a note of how to do this so that a team member would be able to reproduce your result.)

Expected Output

Repository

1. Code
2. Spike Report

Canvas

1. Spike Report

Notes

Note: You must create visual charts for some of these points. We recommend you do this in MS Excel or similar. For this simple task, add the chart images to your spike report (not a separate report document). If you create an accompanying .xlsx file, make sure that is added to your repository and show it to your tutor.

- Assume you are writing steps to help a colleague to recreate your results. Make sure your notes make enough sense!

- As a review, show your notes to another student. Afterwards, update with the suggestions or omissions so that your notes are valuable. Your repo commit comment would be something like “spike report improvements based on feedback from …” or similar and include who you showed your notes to.
- Learn about std::chrono!