

COMP150: Game Development Practices

Optimisation and profiling

Today's class

- ▶ Optimisation and profiling
- ▶ General support
- ▶ Break
- ▶ UML worksheet

Reminder

- ▶ Student rep nominations for next year are open **now!**
- ▶ Nominate yourself on the FXU website

Optimisation

- ▶ **Optimisation** is the process of making a program **more efficient** in terms of speed, memory usage etc.
- ▶ **Macro-optimisation**: optimisation at the program design level, e.g. choosing algorithms or data structures with more efficient “big- O ” behaviour
- ▶ **Micro-optimisation**: optimisation at the code level, e.g. tweaking individual lines of code

To optimise or not to optimise?

Optimisation can **increase** or **decrease** software quality, depending on what measure of “quality” is being used.

- ▶ In pairs.
- ▶ Discuss for **3 minutes** the impact of optimisation on software quality.
- ▶ Log on to Socrative 6E8NSW3IN
- ▶ Suggest **one example** of how optimisation may **increase** the quality of your software.
- ▶ Now suggest **one example** of how optimisation may **decrease** the quality of your software.

“Rules of optimization:

Rule 1: Don't do it.

Rule 2 (for experts only): Don't do it yet.”

— Michael A. Jackson

“Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We *should* forget about small efficiencies, say about 97% of the time: **premature optimization is the root of all evil**. Yet we should not pass up our opportunities in that critical 3%.”

— Donald Knuth

“Measure twice, cut once.”

— Proverb

Profiling

- ▶ A **profiler** measures which parts of a program take the most time and/or use the most memory
- ▶ Profiling helps to identify **bottlenecks**: the parts of the software that are actually causing performance problems
- ▶ **Always** profile to find bottlenecks — don't try to guess where they are!

Profiling in Visual C++

- ▶ Visual Studio 2015 includes extensive profiling tools
- ▶ Google “Visual Studio 2015 diagnostic tools” for more info

Bottlenecks

- ▶ Where does the software spend most of its time?
- ▶ Code optimisation is most useful if your program is **CPU-bound**
 - ▶ Performing calculations or logical operations
- ▶ Programs can also be **GPU-bound**...
 - ▶ Rendering graphics
- ▶ ... or **memory-bound**...
 - ▶ Accessing data in memory
- ▶ ... or **IO-bound**
 - ▶ Waiting for disk, network, or other external devices
- ▶ E.g. don't waste time optimising a part of the program that is already limited by disk or network speed!

Optimisation by compiler

- ▶ When compiling in **release mode**, the compiler performs many micro-optimisations **automatically**
- ▶ **Common programmer mistake**: doing manual micro-optimisations that the compiler would do anyway
- ▶ Ideally, always profile your code in **release mode**