**What is multithreading?**

In a single-threaded computing system, one instruction goes in at a time, and one result comes out at a time. The time to load and complete programs depends on the amount of work you need the CPU to do.

Multithreading is a type of programming that takes advantage of a CPU’s capability to process many threads at the same time across multiple cores. Instead of tasks or instructions executing one after another, they run simultaneously.

One thread runs at the start of a program by default. This is the “main thread”. The main thread creates new threads to handle tasks. These new threads run in parallel to one another, and usually synchronize their results with the main thread once completed.

This approach to multithreading works well if you have a few tasks that run for a long time. However, game development code usually contains many small instructions to execute at once. If you create a thread for each one, you can end up with many threads, each with a short lifetime. That can push the limits of the processing capacity of your CPU and operating system.

It is possible to mitigate the issue of thread lifetime by having a [pool of threads](https://en.wikipedia.org/wiki/Thread_pool). However, even if you use a thread pool, you are likely to have a large number of threads active at the same time. Having more threads than CPU cores leads to the threads contending with each other for CPU resources, which causes frequent [context switching](https://en.wikipedia.org/wiki/Context_switch) as a result. Context switching is the process of saving the state of a thread part way through execution, then working on another thread, and then reconstructing the first thread, later on, to continue processing it. Context switching is resource-intensive, so you should avoid the need for it wherever possible.