Game Development and beyond – About save states, modern processors and online multiplayer

Summary:

While working on code for the semester projects, we tend to focus a lot on gameplay programming and getting the game to run in the first place. As valuable as this is, there are parts of game development that usually take on a lesser role during our studies. This goes from saving the game state across play sessions to optimizing the game beyond the PCs it was developed on to making a game extendable in the future.
In this elective we take a look at some of these techniques from a programming perspective and apply them to real world game development scenarios. We will make use of C++ and a small custom engine but the principles of this course are applicable to programming in general and are agnostic of the language or framework you are using.

(Remote) organization:

I will create a Discord server where I will hold lectures, share resources and answer questions. The elective will be held twice a week from the beginning of November until the winter break (6 weeks total).

Unlike the elective of last year, this time there will not be a 'goal' project as it was the case for the ECS. Thus, if there is more interest in a specific topic over another one, we can allocate more time towards that. The topics all make use of the same underlying principles but are used in different areas related to game development:

- **Serialization and application states**
    - We will take a look at how games in particular make use of writing their state to a file via the use of save states. We will identify the issues with naïve approaches to this topic and how to make one's life a lot easier by carefully planning a save/load feature.
- **Multithreading**
    - Instead of achieving higher and higher clock speeds, modern CPUs prefer to contain multiple processor cores to parallelize code and speed up applications by splitting their instructions up into multiple parts. Commercial game engines so far were quite hesitant to expose multithreading to game code but under the hood it has been in use for end user PCs for well over 10 years. We take a look at its use and its most common pitfalls.
- **Online multiplayer**
    - Interestingly enough, when combining the technology behind save states and multithreading, we get the basic framework of an online multiplayer framework. Thus, we will see how to synchronize a game's state across multiple PCs and, if time allows, also take a quick look at how to avoid exploits and cheating behavior.

Prerequisites:

- **Rudimentary understanding of coding in general**
  - You should be reasonable comfortable writing code. I will assume that you know about basic data types ranging from integers to strings, about object orientation and how to use them.
- **Prior experience with C++ is highly advised**
  - This course utilizes C++ as its language of choice but it's not a C++ introduction course. I will assume that you know about the basic differences between a manually managed language like C++ and a language like C# that makes use of garbage collection.
- **Basics of memory layouts should be known**
  - Before entering this elective you should make yourself familiar with the inner workings of your code. As an example, I will assume that you know what the difference between a 32-bit precision floating point number and a 64-bit double precision number is, what a pointer does or why memory can be corrupted.

If you are unsure whether your current skill level would be sufficient for this elective, you can always write me a message at thomask@crytek.com and ask.