Parallel Rendering Overview

An overview of parallel rendering.

Threading Overview

Originally, the renderer ran in the Render Thread, and commands for that thread were enqueued by the Game Thread to be run later in the frame. These commands made calls into the Render Hardware Interface (RHI) layer, which serves as our cross-platform interface into the different graphics APIs on the platforms we support.

In order to improve the efficiency of this process and take advantage of the capabilities of our supported platforms, the Render Thread now serves as a frontend that enqueues platform-agnostic graphical commands into the renderer's command list. Then, a new thread, the RHI Thread, translates (executes) them via the appropriate graphics API on the backend. This separation enables independent backend parallelization on platforms that support it, such as game consoles, DX12, and Vulkan. In general, anything generated in parallel on the frontend is translated in parallel on the backend.

There are certain commands which are carried out without the use of the command list system, such as Lock and Unlock operations. These commands are issued by the Render Thread directly. In these cases, the Engine will either flush the RHI Thread and wait for the operations to finish, or it will copy the data and queue it. This is implemented differently depending on the operation and the platform.

Graphical Commands and the Command List

The commands enqueued by the Render Thread are instances of structs derived from the `FRHICommand` template. They override the `Execute` function, which is called during the translation process, and store any data that they require. Commands can be submitted to the GPU differently on different platforms (for example, submitted multiple times per frame, all at once at the end of the frame, and so on) based on heuristics for that platform's optimal performance, or they can be submitted by issuing the `FRHISubmitCommandsHint` command.

The main interface used in translation is the `IRHICommandContext`. There is a derived `RHICommandContext` for each platform and API. During translation, the `RHICommandList` is given an `RHICommandContext` to operate on, and each command's `Execute` function calls into the `RHICommandContext` API. The command's context is responsible for state-shadowing, validation, and any API-specific details necessary to perform the given operation.

Synchronization

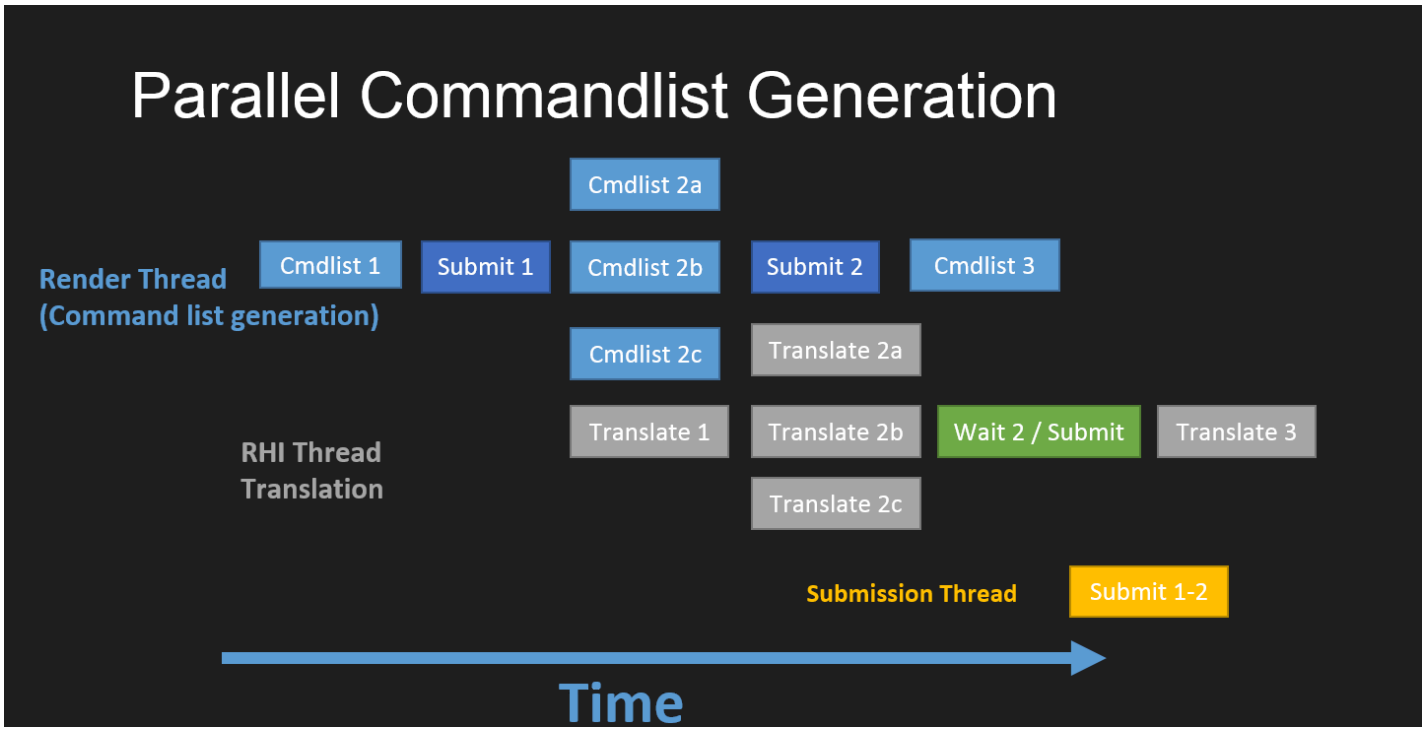
Synchronization of the renderer between the GameThread, RenderThread, RHI Thread, and the GPU is a complex topic. At the highest level, Unreal Engine 4 is normally configured as a "single frame behind" renderer. This means that the RenderThread is processing Frame N while the GameThread processes Frame N+1, except in cases where the RenderThread runs faster than the GameThread. The addition of the RHI Thread complicates this slightly, in that the RenderThread is able move ahead of the RHI Thread by completing visibility calculations for Frame N+1 while the RHI Thread is processing Frame N. The end result is that, while the GameThread is processing Frame N+1, the RenderThread may be processing commands for Frame N or Frame N+1, and the RHI Thread may also be translating commands from Frame N or Frame N+1, depending on execution times. These guarantees are arbitrated by

`FFrameEndSync` and the `FRHICommandListImmediate` function called `RHIThreadFence`. It is also guaranteed that, regardless of how parallelization is configured, the order of submission of commands to the GPU is unchanged from the order the commands would have been submitted in a single-threaded renderer. This is required to ensure consistency and must be maintained if rendering code is changed.

Debugging

There are several console variables that you can use to control this behavior. Because many of these stages are orthogonal, they can be independently disabled for testing, and new platforms can be brought up in stages as time allows. For example:

Command	Description
r.rhicmdusedeferredcontexts	If set to 0, will disable parallelization of the backend. Default value is 1.
r.rhicmduseparallelalgorithms	If set to 0, will disable parallelization of the frontend. Default value is 1.
r.rhithread.enable	If Set to 0, will disable the RHI Thread completely. Command lists will still be generated, but they will be translated directly on the Render Thread at certain points.
r.rhicmdbypass	If set to 1, will completely disable command list generation and make the renderer behave like it originally did, directly calling the RHI commands on the Render Thread without a command list. Note that this will be ignored unless the RHI Thread is disabled.



Showing how parallel command list generation works in Unreal Engine.