FShaderCache

The FShaderCache provides mechanisms for reducing shader hitching in-game.

Overview

The FShaderCache provides mechanisms for reducing shader hitching in-game. It supports the OpenGLDrv and MetalRHI RHIs, and works on Mac, Linux, and Windows platforms.

There are a number of console commands that can be used to enable or disable FShaderCache functionality.

Console Command	Description
`r.UseShaderCaching [0/1]`	• Early submission during shader deserialisation rather than on-demand. • Tracking of bound-shader-states so that they may be pre-bound during early submission.
`r.UseShaderDrawLog [0/1]`	Tracking of RHI draw states so that each bound-shader-state can be predrawn.

Console Command	Description
`r.UseShaderPredraw [0/1]`	Predrawing of tracked RHI draw states to eliminate first-use hitches.
`r.PredrawBatchTime [Time in (ms)]`	Control over time spent predrawing each frame to distribute over many frames if required. Use -1 for all.
`r.UseShaderBinaryCache 0/1`	Accumulation of all shader byte code into a single cache file
`r.UseAsyncShaderPrecompilation 0/1`	Asynchronous precompilation of shader code during gameplay
`r.TargetPrecompileFrameTime [Time in (ms)]`	The target maximum frame time to maintain when r.UseAsyncShaderPrecompilation is enabled. Use -1 to precompile all shaders at once.
`r.AccelPredrawBatchTime [Time in (ms)]`	An option to temporarily accelerate predrawing when in a non-interactive mode such as a load screen. Use 0 to use `r.PredrawBatchTime`.
`r.AccelTargetPrecompileFrameTime [Time in (ms)]`	An option to accelerate asynchronous precompilation when in a non-interactive mode such as a load screen. Use 0 to use r.TargetPrecompileFrameTime.
`r.InitialShaderLoadTime [Time in (ms)]`	A maximum amount of time to spend loading the shaders at launch before moving on to asynchronous precompilation. Use -1 to load synchronously)

Use

The cache should be populated by enabling `r.UseShaderCaching` and `r.UseShaderDrawLog` on a development machine. Users/players should then consume the cache by enabling `r.UseShaderCaching` and `r.UseShaderPredraw`. Draw logging (`r.UseShaderDrawLog`) adds noticeable fixed overhead so avoid enabling it in shipped products if possible. The binary shader cache can either be accumulated during play through or (currently Mac targets only) during cooking by specifying the shader platforms to cache in the CachedShaderFormats array within the /Script/MacTargetPlatform.MacTargetSettings settings group of Engine.ini. For OpenGL the binary cache contains enough data about shader pipelines to construct fully linked GL programs or GL program pipelines (depending on availability of GL_ARB_separate_shader_objects) but not enough for pipeline construction on any other RHI. This can help reduce the amount of hitching on OpenGL without first playing through the game, though this is still advisable for maximum effect. Since the caching is done via shader hashes, it is also advisable to only use this as a final optimization tool when content is largely complete as changes to shader hashes will result in unused entries accumulating in the cache, increasing cache size without reducing hitches.

The code will first try and load the writable cache, then fall back to the distribution cache if needed.

Cache Type	Cache Location
Writable	`<Game>/Saved/DrawCache.ushadercache, <Game>/Saved/ByteCodeCache.ushadercode`
Distribution	`<Game>/Content/DrawCache.ushadercache, <Game>/Content/ByteCodeCache.ushadercode`

Integration Steps

While there are a number of console commands you can use to reduce hitching in your project, there is a recommended priority order to enable options to maximize performance while avoiding additional project work.

1. Enable `r.UseShaderCaching` & `r.UseShaderPredraw` in the project configuration for all users.
2. If possible, enable `r.UseShaderDrawLog` only for internal builds & ensure that shader draw states are recorded during final QA for each release. When not feasible (e.g. an

extremely large and/or streaming game), enable for all users.

Test whether this setup is sufficient to reduce hitching to an acceptable level, as the additional optimizations require more work and have significant side effects.

1. If the above is insufficient, then also enable `r.UseShaderBinaryCache` and configure the CachedShaderFormats to ensure population of the binary cache (currently Mac only).
2. All shader code will now be loaded at startup and all predraw operations will occur on the first frame, so if the loading times are too extreme also set `r.PredrawBatchTime` to a value greater than 0 in ms to spend predrawing each frame.
3. You may also wish to specify a larger value for `r.AccelPredrawBatchTime` that can be applied when showing loading screens or other non-interactive content.
4. To inform the shader cache that it may accelerate predrawing at the expense of game frame rate, call `FShaderCache::BeginAcceleratedBatching`, and when it is necessary to return to the less expensive predraw batching, call `FShaderCache::EndAcceleratedBatching`.
5. A call to `FShaderCache::FlushOutstandingBatches` will cause all remaining shaders to be processed in the next frame.
6. If the project still takes too long to load initially, then enable `r.UseAsyncShaderPrecompilation` and set `r.InitialShaderLoadTime` to a value > 0. As the initial shader load time increases, the work that must be done while the game is running decreases.
7. You can tweak the desired target frame time while asynchronously precompiling shaders by modifying the value of `r.TargetPrecompileFrameTime`.
8. As with predraw batching, a more aggressive value can be specified for precompilation by setting `r.AccelTargetPrecompileFrameTime` to a larger value and then calling `FShaderCache::BeginAcceleratedBatching` (and `FShaderCache::EndAcceleratedBatching` afterwards).
9. With `r.UseAsyncShaderPrecompilation` enabled, a call to `FShaderCache::FlushOutstandingBatches` will also flush any outstanding compilation requests.

Handling Updates/Invalidation

When the cache needs to be updated and writable caches invalidated the game should specify a new GameVersion. Call `FShaderCache::SetGameVersion` before initializing the RHI (which initializes the cache). This will cause the contents of a cache generated by a previous version to be ignored. At present you cannot carry over cache entries from a previous version.

Region/Stream Batching

For streaming games, or where the cache becomes very large, calls to `FShaderCache::SetStreamingKey` should be added with unique values for the currently relevant game regions/streaming levels (as required). Logged draw states will be linked to the active streaming key. This limits predrawing to only those draw states required by the active streaming key on subsequent runs.