Overview

Below are some important code and data changes that you should be aware of when moving to 3.8.4.

Emissive Materials

As part of our ongoing transition to physical light units, emissive materials have recevied some updates to simplify their setup and improve consistency with scene lighting:

- Emissive lighting is now always additive and is added on top of diffuse and specular lighting contributions for a surface
- Emissive intensity is now specified in kilonits (kcd/m2)
- Emissive color acts as a multiplier for the emissive map and does not have any effect if Emissive Intensity is 0
- New separate texture slot for emittance map; diffuse and decal slot are no longer used for emittance
- New gamma control for emittance map to allow for an increasing range of values
- Loadtime conversion for basic backwards compatibility:
  - If glow was enabled, diffuse map gets copied to the emittance slot during loading
  - Existing glow intensity gets converted to new physical units

Customizable Binary Output Folders

As of 3.8.4 the project's binary output folder location can be modified. Prior to 3.8.4 the location was fixed for e.g. to *<root>/Bin64* for *Windows x64* builds.

Furthermore, when compiling the engine code using WAF all required 3rd party libraries will be copied into the binary output folder as part of the build step.

The final binary output location can be changed via the *cry_waf.exe -> WAF Options -> Output Folder* e.g. *out_folder_win64 -> bin/win_x64*

A new options variable "output_extension_<configuration>" has been added to allow users to target different output folders for different configurations for e.g. *win x64* | *release -> bin/win_x64_release*

The following rules apply for the final output folder destination:

1. The binary output path is relative to the CryEngine root path. The path can be the root path itself.
2. The root path is identified by containing a folder called "Engine" or "engine"

Experimental Support for Visual Studio 2013 and 2015 Compilers

Many of our customers have been asking for improved support for other Windows compiler tool-chains, rather than/just Visual Studio 2012.
Previously, it was possible to make WAF pick up those tool-chains, but it required changes to the WAF scripts to make it work.

In 3.8.4 we have improved the support for Visual Studio 2013 and 2015 compilers. The primary supported compiler tool-chain for Windows (for 3.8.4) remains Visual Studio 2012 compiler, but we have included experimental support for the other tool-chains. You may have to change some WAF options to customize which tool-chain you want to use, please read the "Visual Studio supported versions_old" page for information on how to configure this feature.

For CRYENGINE 3.8.4 - when using Visual Studio 2015 compiler then the following features will not work:

- PerforcePlugin for SandBox, the 3rdParty library it depends on does not support Visual Studio 2015.
  This project must be removed from the WAF spec file when using Visual Studio 2015.
  To do this, edit `<root>\_WAF_\specs\gamesdk_and_tools.json` and delete the line mentioning `PerforcePlugin.`

These limitations do not apply when compiling with Visual Studio 2012.
These limitations do not apply to EaaS versions of CRYENGINE, since these tools do not need to be compiled in that version.

Changes to `assert`, and `CRY_ASSERT` Helpers

In 3.8.4, we have made some changes to the `assert` and `CRY_ASSERT` files in preparation of allowing more control over assert behavior. However, as part of this work the (undocumented) `FORCE_ASSERTS_IN_PROFILE` definition is no longer available. Similar behavior can be obtained by customizing `CryCommon/CryAssert.h` to suit your purposes (ensure USE_CRY_ASSERT ends up being defined before line 26).

For CRYENGINE 3.8.4, we do not recommend enabling asserts in profile.