D4.3 eStudio HDR Graphics Engine Final Version

Grant Agreement nr	761544
Project acronym	HDR4EU
Project start date (duration)	July 1st 2017 (36 months)
Document due:	31/3/2020
Actual delivery date	31/3/2020
Leader	Brainstorm Multimedia
Reply to	jmontesa@brainstorm3d.com
Document status	Submission version

Project funded by H2020 from the European Commission

Project ref. no.	761544
Project acronym	HDR4EU
Project full title	Enabling end-to-end HDR ecosystem
Document name	eStudio HDR graphics engine final version
Security (distribution level)	PU
Contractual date of delivery	31/3/2020
Actual date of delivery	31/3/2020
Deliverable name	D4.3 eStudio HDR graphics engine final version
Type	DEM
Status & version	Submission version
Number of pages	21
WP / Task responsible	WP4 – Real time Generation / Brainstorm Multimedia
Other contributors	-
Author(s)	Javier Montesa, Moises Ferrer
EC Project Officer	Mr. Ralph Dum, Ralph.Dum@ec.europa.eu
Abstract	Brainstorm tasks in WP4 were organised in order to obtain early and intermediate versions of the new HDR graphic engine to test the whole real time HDR chain as the team kept developing. This document goes through all the new features added to the final version, their implementations and their final result.
Keywords	HDR, real time, graphics engine, video in, video out
Sent to peer reviewer	Yes
Peer review completed	Yes
Circulated to partners	No
Read by partners	No
Mgt. Board approval	No

Table of Contents

1. Introduction

This document reports the final version of the eStudio HDR graphics engine developed according to the capabilities and features described in the HDR4EU proposal. The set of features implemented to achieve the final version can be distributed in six groups:

- **HDR Rendering Engine**
  This set of features is related to the render process, the engine does not treat colour as values from 0 to 255 any more, and the new representation modifies every single shading algorithm taking place in the rendering.

- **Color Managed Pipeline**
  There are a number of different colour spaces used by graphic tools, video signals, and image file formats to represent colour. In order to properly manage all types of inputs and outputs in the graphic engine, it is necessary to properly import and manage these contents in an adequate and unique manner. The features presented here allow users to determine the different colour spaces of input and output contents, and the required colour space to be used internally by the engine.

- **Physically Based Rendering**
  The new more precise way to represent colour inside the engine and the extended dynamic range allows for a much more complex and realistic way to calculate how objects are illuminated on a scene. The features presented here allow users to get advantage of this new way to render objects but also to compatibilize it with old elements or textures that still need to be treated based on the old colour standards.

- **Input/output system**
  The features presented here allow the user to manage the way contents are captured and provided. They can deal with specific hardware but also provide the option to output different formats when the system requires it, for example when the operator display is a common SDR device but the real time final output needs to be HDR.

- **Other developments**
  A part of the presented features there are other specific ones that cannot be grouped in a specific section, HDR compositing, Colour Correction, 3D luts, or Automatic ITM. These capabilities have also been implemented and are explained in a specific section of the document.

As we will see in the documents, based on all the previously implemented features, and given the new ones added in the last version the HDR4EU project has completed the transformation of the most important product of the company to be HDR ready and compatible with existing HDR inputs and outputs and those developed by other partners.

2. HDR rendering engine preparation

The first steps in order to adapt the previous engine into an HDR engine were to render the scene in a higher bit-depth per pixel, this allowed for two important improvements, first, a higher color range allowing to represent higher luminosity levels of the scene, and second, an increased precision per pixel. Starting from this, the following internal developments of the engine evolved in a similar way on different aspects (textures, video decoding, video input, video output, tone mapping), always taking into account the performance implications that the new algorithms can cause.

3. Color managed pipeline

The deliverable D4.2 already showed how the developments laid the basis of the color management approach, however, this has been a process continuously improved during this last year. Moving to a color managed pipeline is an effort which needs to be applied not only on the technical side of the product, but it also requires training people to understand brand new concepts that were not familiar until now.

This section describes the resulting pipeline after all the developments done. It involves work from the input to the output, so it is somehow transversal to the render engine.

3.1. Color space

In order to provide a proper color-managed pipeline, it is required that whenever color operations are involved, all the color sources are in a common and known color space. In the broadcast field, the color spaces to consider were mainly the Rec. 601 (SD) and Rec. 709 (HD). Those color spaces had similar RGB primaries, so a mixture of spaces was probably not noticed even for the trained eye.

With the new HDR formats, the reference color space is Rec. 2020, a much wider space than the current Rec. 709, and given that there is still a mixture of different spaces (CG assets are still sRGB), it is very important to take care of those conversions.

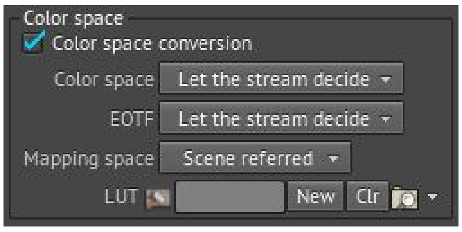
In order to perform such conversions, a suitable color workspace needs to be defined. That will be the color that every input in the system will be transformed to, so when it is used and finally blended in the framebuffer, the color information will be in that color space. Currently the software supports Rec. 709 and Rec. 2020, presented to the user as a setting[1].

As stated, the different inputs needed to be transformed in the working colorspace. How they are converted depends on the given source, but ultimately, they end up in the same space.

We can see for main types of inputs:
- Image files: most of the time those are in sRGB.
- SDI Video input: usually Rec. 709 or Rec. 2020. The colorspace is read from the stream, but it can be redefined by the user (per stream).

[1] Users can also opt to keep a project in a non-managed color, specially suitable for old projects that need to retain the previously designed look.

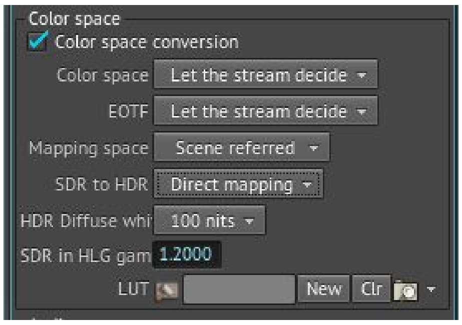
- Movie files: those come in all the above formats, plus jpeg space. The source colorspace is read from the file, but it can be overridden by a user defined one (per movie source).
- Editor colors: colors defined in the GUI.



Color space
✓ Color space conversion
Color space: Let the stream decide
EOTF: Let the stream decide
Mapping space: Scene referred
LUT

Image 1. Input color space conversion.

Finally, the color space in the output also requires to be defined, in order to perform the correct color transformations at the end of the pipeline. The output colorspace is defined for each output stream, since two simultaneous SDI outstreams can require different spaces. A typical case would be to have a system with HD and UHD outputs (Rec. 709 and Rec. 2020), both of them being feeded from the same generated content.



Color space
✓ Color space conversion
Color space: Let the stream decide
EOTF: Let the stream decide
Mapping space: Scene referred
SDR to HDR: Direct mapping
HDR Diffuse whi: 100 nits
SDR in HLG gam: 1.2000
LUT

Image 2. Output color space conversion.

3.2. Linear space

The color in digital images is typically stored in a non-linear way, resulting in a bigger distribution for darker values, which is where human vision is more sensitive. The reason behind this is that stored and transmitted data takes space, which depends on the bit-depth[2] of the image. Such codifications allow to store a wider range of intensities, without perceiving banding artifacts compared with a linear storage.

It is important for an application to care about data linearization before doing any image manipulation, working effectively in what is called a linear workflow. Not doing so will result in incorrect results, different areas, such as blending, shading, or light attenuation.

[2] Assuming uncompressed data for a typical image buffer.

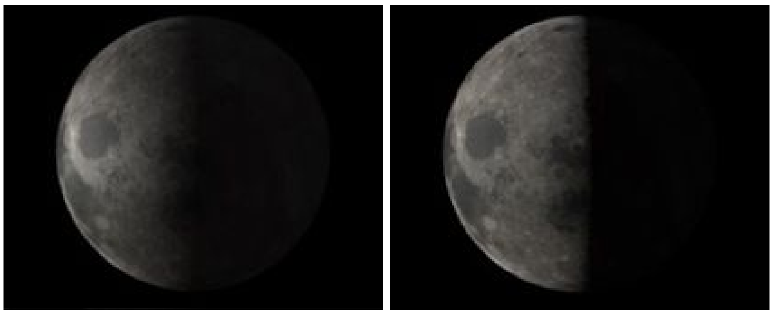


Image 3. Gray scale non-linear space shading (left) vs linear space shading (right).

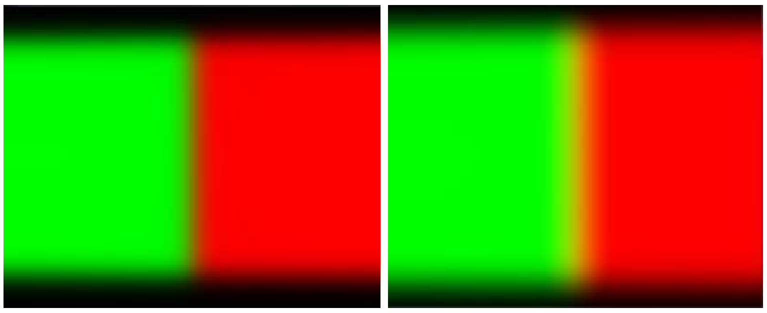


Image 4. Color non-linear space shading (left) vs linear space shading (right).

As part of the development of the engine, it has been added support for such a linear workflow. This is the preferred way to work, since it is the proper way to render and generate accurate images. That said, it is still possible to load old projects in compatibility mode, to prevent any undesired color change.

Textures are usually in sRGB color space, and are assumed to be in that color space. However, an algorithm has been developed to flag them as linear, which might be the case for some images encoding data, such as normal maps or data images.

Colors shown in editors on the GUI are also assumed to be in sRGB, since those are displayed on the user PC and not on a reference monitor. The required conversions are handled internally to get the same colors on the video output.

Video input and output systems show a much wider variety, since there is a wider set of transfer functions available (Rec. 709, HLG, PQ, etc...). As in the color spaces, those are read from the file or stream, but can be overridden by the user. Internally all required conversions are being carried out to work transparently in a linear or gamma workflow. As such, and in terms of HDR, the engine supports both HLG and PQ, the two most common transfer functions for the different existing HDR formats.

3.3. Scene/Display referred conversions

The deliverable D4.2 mentioned that the linearization process could be done in two different ways, which would yield different color results: scene mapping conversion or display mapping conversion. The final version of the graphics engine is capable of performing the linearization in any of the linear spaces, and again, it is configurable per stream. Also, it can be configured for the inputs and the outputs, so the user is in total control of the conversion process, with great flexibility.

3.4. SDR/HDR interconversion

Due to the nature of the broadcast projects, there is an additional issue that needs to be addressed: the interconversion of SDR and HDR formats. These conversions will happen both in the input and output video streams. This relates to the fact that there will be a mixture of HDR and SDR sources and potentially, and SDR and HDR outputs.

Multiple workflows can be foreseen that would need to deal with such conversions:

- An SDR scene (possibly an old project) embedded on an HDR stream
- An HDR scene with both HDR and SDR outputs
- SDR inputs used on an HDR scene

The developed features allow such mixture of options to be configured, providing for a wide range of operations. By default, the most common workflow is preconfigured, but users can fine tune conversion options both in inputs and outputs, allowing for the commented conversions. It is worth mentioning that when matching different sources, the idea is to make their white diffuse point match. The chosen level is the white level of an SDR stream (to simplify, 100 nits), but as mentioned in ITU-R BT.2390, for HDR streams it might be better to consider 200 nits. Given that recommendation, and having seen other tools using it, the development team opted to give the user the option to control it.

3.5. Color space validation

Validation of the whole color management pipeline is a cumbersome and error prone process. As such, the development team tested the project implementation with the help of an HDR hardware convertor, the FA9600, provided by ForA.

Image 5. ForA FA9600 HDR hardware converter.

An image comparison system was built where conversions on images were done both through the HDR4EU system and through the conversor. The resulting image from the frameconverter was again introduced to the graphic engine system, and compared with its implementation. Results proved that all the format conversions were in place (SDR/HDR and HDR/HDR conversions), while just some minor changes in colors could be perceived in some of the configurations.

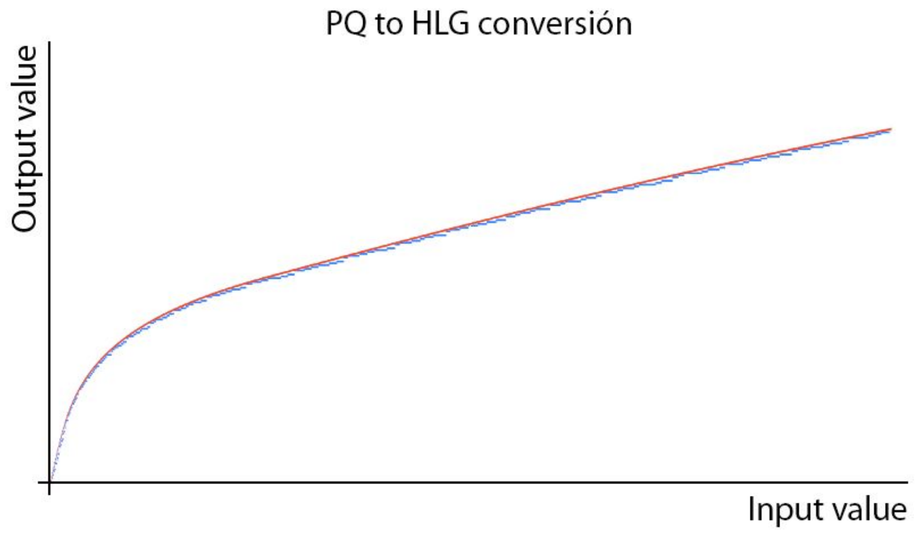


Image 6. Comparative between ForA FA9600 and Brainstorm engine

The detected slight color differences can be explained by the usage of different algorithms for Gamut mapping, which is left unspecified in the standards (at least broadcast ones), and is therefore implementation dependent.

3.6. Visualization tools

Introducing the concepts of color management to the Brainstorm Multimedia production team was a challenge in itself. While the film industry has a large trajectory on this, the broadcast field has lacked some of it, at least to some extent. Therefore a set of visualization tools has been developed on the graphic engine interface to explain the concepts of colorspace, transfer functions and the relevance behind all of this.

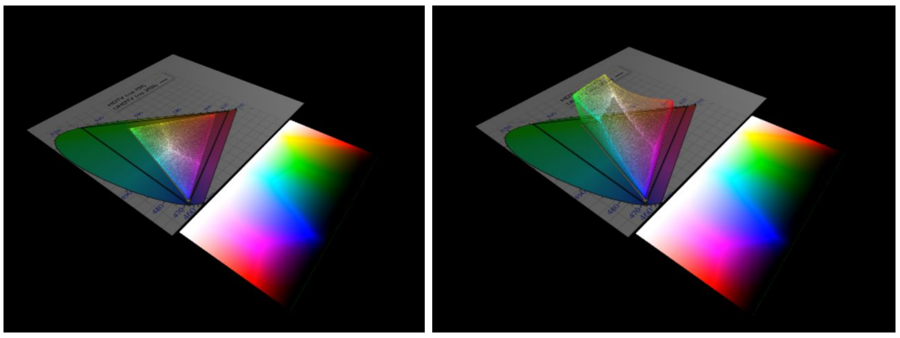


Image 7. Representation of an image gradient in the graphics engine and how it maps to the CIE xy chromaticity diagram.

4. Physically based rendering

Part of the development effort on the rendering engine has been directed to create a modern shading paradigm known as Physically Based Rendering (PBR). The shading method defines how the color of the materials is derived, usually considering both light and material properties.

Classic paradigms, such as Phong or Blinn, are limited to some extent. Often, material properties need to be tweaked when lighting conditions change. PBR shading, on the contrary, gets rid of most of those issues. Materials are defined with a different set of parameters, and the light is also based on HDR images, in a technique known as Image Based Lighting (IBL). The neat effect of these are materials that behave properly under changing light conditions, and are much more realistic.

Image 8. Different materials example with equal light.

At the current stage, the engine has support for both the classic and new PBR shading models, and they can be mixed in a project. Therefore, compatibility is fully supported, and extensibility is warranted. A set of predefined materials is provided in the form of a material library, so users can start using them without changing their workflow. Advanced users and graphists have the option, nevertheless, to fully customize the material with their own values or images.

4.1. <u>HDR cubemaps</u>

As mentioned above, PBR makes heavy use of Image Based Lighting, by using images to specify the light distribution in the scene. An High Dynamic Range image representing the whole environment is provided as the illumination source, it is then processed, other types of data are precalculated, and then, during shading calculations, the derived information is used to calculate diffuse and specular terms of the color.

In order to provide fast calculations per frame, the graphics engine used HDR cubemaps, in a technique developed by Epic Games for the Unreal Engine [(R10)]. At an early stage of our engine, when HDR cubemaps were still not available, our first implementation of PBR made use of spherical harmonics for diffuse light, and heavy use of texture sampling for the specular component. It was clear then that the cubemap approach, while requiring more GPU memory, was much better from the performance perspective. Currently, the former approach has been superseded by the cubemap one.

Image 9. Sphere mapped with an HDR cubemap, with different levels of exposure.

4.2. HDRe files

In order to use IBL in real time, the environment images need to be processed, whether it is to generate the cubemaps with prefiltered information, or calculate the spherical harmonics, it has an impact on load time. Those calculations, even done fully on the GPU, can take more than a second to complete, negatively impacting user experience. Moreover, it needs to be redone every time the project is reloaded.

In order to solve this situation, Universitat Pompeu Fabra (UPF) developed an open format, the HDRe files, for storing both source image and the preprocessed data directly in the file. This minimizes the load time, since all the precalculated times are removed. The only thing required is to create the graphic resources (ie the HDR cubemap textures) and fill into it the image data. As a result, the negative implications mentioned were leveraged, and still the light information is stored in a single file, easing asset management and the operation workflow.

5. Postprocess

One important part of a rendering engine is its postprocess effects module. Nowadays, many effects are produced at that stage: defocus, bloom or light effects. All those effects can benefit from the fact that an HDR render buffer provides more information, and makes use of it individually. This section depicts some of those improved effects and describes additional postprocess stages that have been implemented. While some of them are optional, other additional stages are required in order to produce a valid image for the output device. That is the case of the tonemapper, which is an essential part of the HDR rendering pipeline.

5.1. Bloom

One of the immediate graphic effects that can benefit from the HDR pipeline is the bloom effect, which consists of simulating halos around objects that are extremely bright. If the color information is constrained in a normalized range [0-1], which is the case for non-HDR buffers, the way to detect pixels that would contribute to the bloom effect is to get the brightest ones. However, it has the drawback that the white pixels exhibit a strong luminance value, even when they are not really bright. Sometimes it's needed to rely on tricks, such as bloom per object, in order to overcome such limitations.

The new render feature allows for a more robust and correct implementation, which can be based on really bright pixels. In addition, it has a better interface, fully integrated in the viewport modules, so it is even easier possible to achieve better results effortlessly.

5.2. Exposure

HDR provides much more information of the scene, both in the darker areas and in the brighter ones. However, it is not possible to appreciate all the range at once, very much as it happens in real life. The exposure control is a kind of multiplier that allows the user to make visible the darker and brighter areas, by increasing or decreasing the exposure.

This effect can be animated to achieve dynamic effects, if a drastic change in the illumination of the scene occurs.

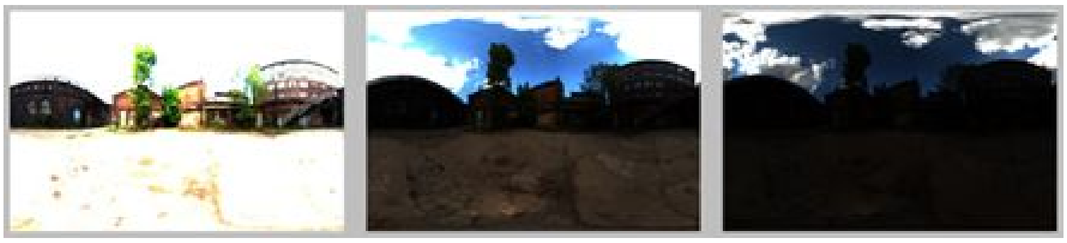


Image 10. Scene with different exposures comparative.

5.3. Eye adaptation

Related to the exposure control explained above, the human eye performs automatic adjustments when we move through areas of different light intensities. This effect can be simulated in a virtual world by measuring the overall light intensity of the scene and adjusting the exposure automatically.

The final version of the graphic engine implements such feature, where it is possible to have some degree of configuration on the effect. The user can control the adaptation speed for brighter and darker areas separately[3], and it can be further constrained not to go too far. The automatic exposure value is calculated by generating an histogram or the scene per frame, and is applied as an offset to the previously mentioned exposure.

[3] Typically it is faster to the human eye to adapt to brighter areas than darker ones.

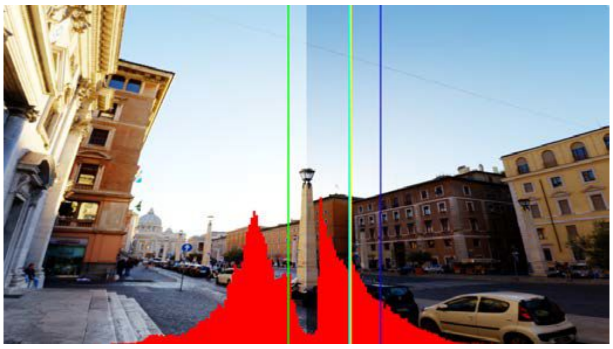


Image 11. Splitted scene applying eye adaptation effect and histogram.

5.4. <u>**Tonemapping**</u>

In order to really make the most from HDR rendering, we need to properly map the HDR information to the output device, which most of the times will be narrower. While the exposure gives us control to center around one area of brightness, the tone mapper is ultimately responsible for achieving the mapping from high to low dynamic range. The net effect of this is that an image can show areas that otherwise would be too dark or overlay saturated.

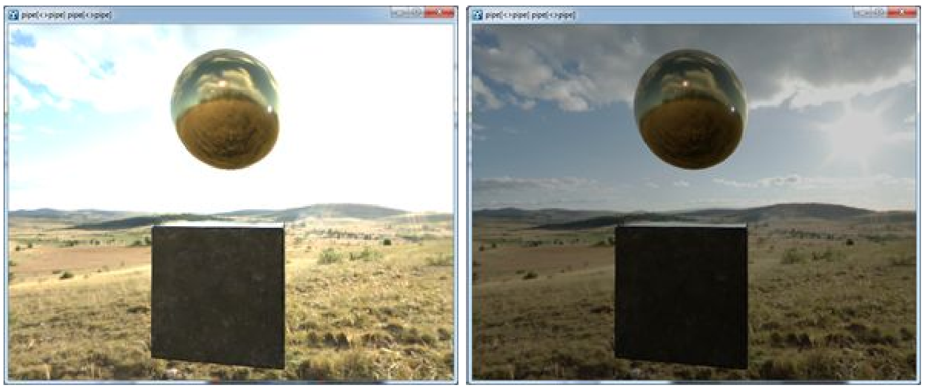
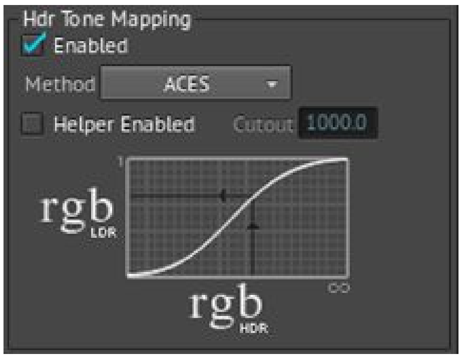


Image 12. Scene comparative between non tone mapped (left) and tone mapped (right).

The implemented features allow the setup of different tonemappers that can be easily selectable from a dropdown menu. One important thing to care about is that the correct mapping is done, since it is not the same to output a signal (normalized on the range [0-1]) for a 100 nits display than for a 1000 nits one. Using the wrong curve will end up spoiling the resulting image. For this reason, the engine does support curves for different output levels.


Hdr Tone Mapping

✓ Enabled

Method ACES

Helper Enabled Cutout 1000.0

Image 13. Brainstorm control curve for HDR Tone mapping.

6. Other developments

6.1. <u>HDR compositing</u>

On broadcast environments, in some specific workflows it is required that the SDI video input is composed with the CG elements, but yet its colors remain unmodified. Given the provided pipelines this inevitably breaks during the postprocess stage, because the tonemap affects the whole image.

While several approaches are feasible for handling the presented problem, the graphic engine implementation opted for the expansion of a layering system that was built at the start of the project. The system allows for different types of layers: video, 3D scene, mix of layers and stack of layers, and was augmented with those needs in mind, so tonemap can be enabled and applied per layer, allowing for configurations that respect the input video.

6.2. <u>Color correction</u>

The final engine version has improved the color correction tools, and has made them HDR aware. This means that the tools are not bound anymore to the [0-1] range that was considered before, and they are applied prior to the tone mapping. The benefit of doing the correction at this stage is that color will be more consistent for different outputs (ie with HDR and SDR outputs). It is also properly applied on linear data, so the control reacts in a much more reliable way, especially when dealing with highlights.

While this seems to be just another stage in the postprocess pipeline, it has been extended further into the system. This same functionality has been ported and is actually used as a tool for color correcting video inputs too in a very dynamic way. One common use case is to have a live video source that is shown over different backgrounds. The flexibility of changing the color settings dynamically allows for a better composition and fast corrections.

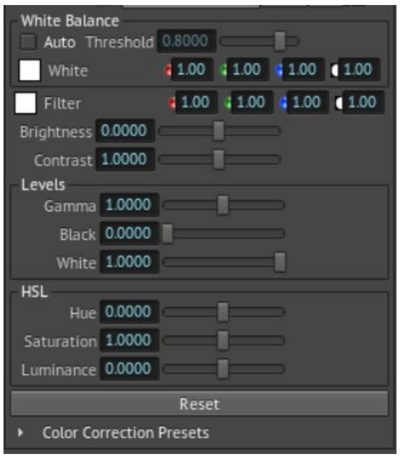

White Balance
Auto Threshold 0.8000
White 1.00 1.00 1.00 1.00
Filter 1.00 1.00 1.00 1.00
Brightness 0.0000
Contrast 1.0000
Levels
Gamma 1.0000
Black 0.0000
White 1.0000
HSL
Hue 0.0000
Saturation 1.0000
Luminance 0.0000

Image 14. Color correction tool interface on Brainstorm engine.

6.3. <u>3D luts</u>

Support for 3D LUTs has been added to the engine's last version. 3D luts are mostly used in two different ways, first they can be used to modify any input or output, allowing the engine to support color spaces or transfer functions that are not handled by the internal system. This can be used, for instance, to decode videos in the ARRI-log format and bring them to a Rec. 2020 space.

Secondly, the other use of 3D LUTs is related with color corrections. While we have improved the color correction tools and make them work in an unbounded linear space, it is very common to use 3D LUTS, especially in the film industry. Based on the inhouse production team, for this reason we added support for this feature too, which is being frequently used.

In addition, it is worth mentioning that the 3D LUTs are exposed to the shaders in the system as textures. This allows more advanced users to create custom effects or use them transparently through custom shaders.

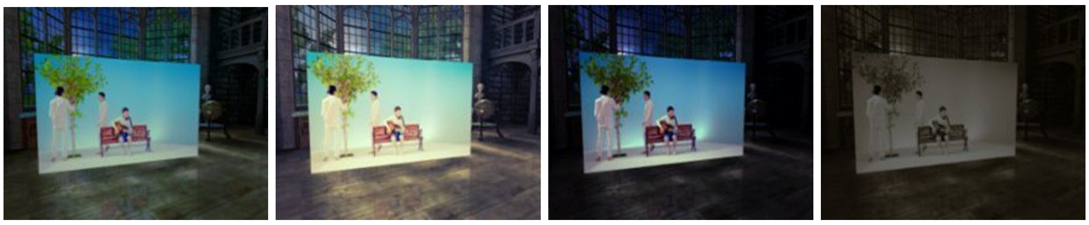


Image 15. The same scene with different 3D LUT applied

6.4. <u>Automatic Inverse Tone Mapping</u>

On HDR projects that incorporate SDR contents, the need to properly integrate this content is a must. There are several options to do this, which range from keeping the SDR look and feel, to modifying the original source attempting to recreate an HDR version. The first option is directly supported in the implemented system, while the second one can be achieved with a 3D LUT to some extent[4].

Following the "*Style aware tone expansion*" [(R11)] publication, we have implemented an automatic real time inverse tone mapper. It computes the histogram of a downsampled version of the image every frame, and then the correspondent stream derived from it. This way a contrast factor is applied to the image, which is actually used to get a look matching the HDR streams, and with the results shown in the mentioned paper.

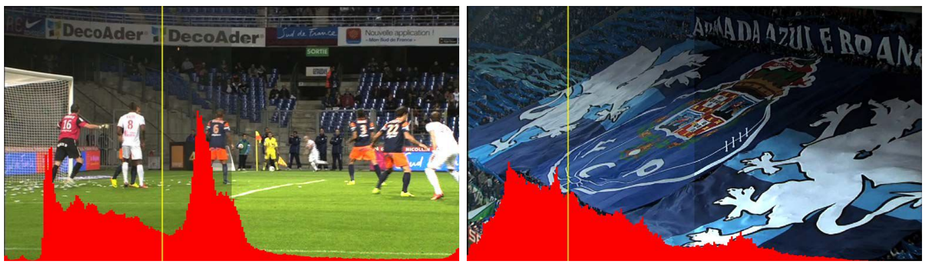


Image 16. Splitted view of video frames with the ITM applied to them (right side of the image). Overlaid on tom, the image histogram and the computed median.

This development is however not yet available to customers, given the advice of the company production department, the main reason being that in practice, not many projects need to deal with that intermix, and the perception of this need is not present now. Also, the fact that it affects performance and adds some complexity to the GUI, supposes an entry barrier for this feature.

7. Input / Output system

Transforming the engine to make it suitable for HDR rendering is tightly coupled with the input sources, given that they are the primary elements that will be used for generating content. Whether they are images, movies of video feeds, without them containing HDR values, the engine would be of limited use for that purpose. In the same way, being able to render HDR content and getting it at the output of the system as a suitable HDR format is required to fulfill the complete pipeline.

The importance of these two endpoints of the system lead to the inclusion of a specific section to list some of the most important features of the I/O developments.

7.1. Extended bit depth

When adding support for HDR inputs, it is important to realize that the textures bit depth use needs to be extended to store them. As this will increase GPU memory consumption and bandwidth, it could hurt performance, and for this reason it is needed to balance these tradeoffs and keep track of what items do really need an increased precision.

[4] The same color transform would be applied to all the frames of a video source, which would not yield the best results for the whole stream.

The internal system is able to work with textures of 10 bits per channel, half float and float[5]. Most of the time the first two representations are used. Notice that even if an input might be of 10 bits, linearization of the data requires it to be stored in 16 bits per channel, in order not to create banding artifacts.

An example of such optimizations can be found in the video input system, which has a queue per stream. For 10 bits elements, the queue holds a 10 bits texture per element in the queue, and just the frame currently presented is decoded into a 16 bits texture. Such optimizations are applied where necessary, in order to fulfill the requirements of a real time engine.

7.2. HDR images

One of the supported inputs are static images, which are loaded as simple textures internally. While the engine had support for the HDR format file from an early stage of the project, its lack of precision made it not very suitable depending on the task. Therefore, support for the EXR format has been added on this final version, providing a quality solution for those types of sources, and having both formats supported now.

HDR images are mostly used in IBL, such as in the HDR cubemaps that were mentioned in an earlier section of this document. And since not only loading, but also saving is supported, this allows the user to create HDR cubemaps from the engine itself, based on the generated virtual scenes.

7.3. HDR clips & video feeds

Movies and video sources are more difficult to treat than the static images counterpart, given the myriad of formats and possible conversions required. This complexity is brought down by reducing the internal supported formats to the most performant ones, and supporting the most convenient conversion paths only. That is nevertheless a task hidden to the user and handled transparently by the engine.

Image 17. Blackmagic HDR camera, used for development and testing.

[5] The 10 bits per channel is actually 10 bits for the red, green and blue channels, plus 2 bits for the alpha, which are mostly used as padding and byte alignment. The half float and float representations are of 16 and 32 bits per channel respectively.

In order to fully support HDR, enabling 10 and 16 bits per channel buffers was implemented, as commented on section 7.1. Also, the video system had to be extended in order to support encoding and decoding movies and video sources. Having lots of formats and layouts resulted in a great number of conversions and tests to be performed in order to comply with the wide range of video boards supported.

Regarding clips, extra developments were carried out in order to guarantee that performance was not going to be affected by the HDR formats. In this regard, new decoding options were enabled, including extended support for GPU decoding, which was extremely valuable for 4K HDR clips, leveraging CPU for other graphic engine tasks. Also, video conversion tools were extended in order to provide HDR video transcoding specially tailored for real time playback.

7.4. <u>Video boards</u>

Most of the time, broadcast workflows involve video boards, both for input and output. Since those are in SDI format, support for SDI video boards is an intrinsic need, to the point that video input/output is a fundamental part of our systems. As a result, many tasks have been directed to evolve this system during the project.

At an early stage, the graphic engine did not have support for flagging the data as HDR, as mentioned in D4.2, and it used external hardware to overcome this. Since then, it has been updated with new SDK's of several boards, and implemented the support for this capability. Now it is possible to inform the board both the color space and the transfer function used, so support for HLG and PQ has been achieved.

Video Setup, a high level tool provided with the graphics engine in order to configure the video systems, has been updated to ease all the configurations required. In house feedback was very relevant in order to hide the complexity of the configuration without losing usability, which is easy to get wrong given the huge configuration options.

7.5. <u>Barco´s projector integration</u>

While not being part of the committed tasks, Brainstorm and Barco worked on the integration of their systems. It was thought that using the real time graphic engine, Barco could quickly create CG elements and check the video output to test the HDR projector.

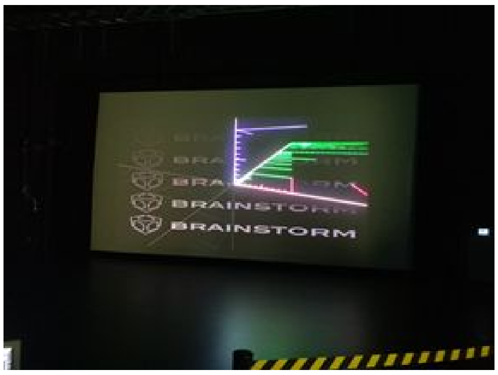
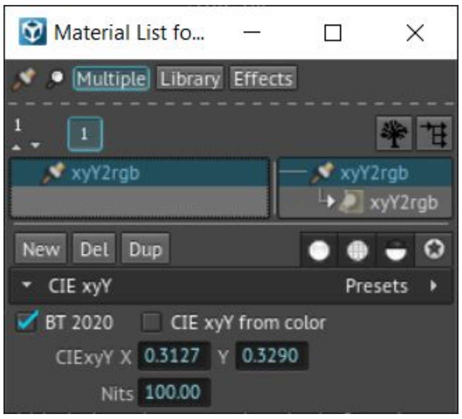


Image 18. Empty HDR scene seen on the Barco´s projector, during a test session.

In order to do that, some tasks were done on the video output in order to achieve the integration. Mostly, there was the need to support RGB444, which is a format not used in the broadcast field, but in the cinema industry. Brainstorm extended the video system and video boards with the required functions and conversions finally making the graphic engine compatible. The result is displayed in the image below.

Some tools to specify colors in a more convenient way for their tests were included. In that way, Barco could specify colors in CIExy format, rather than the usual RGB triplet, and Y in terms of nits, rather than normalized values.



☑ BT 2020 ☐ CIE xyY from color

CIExyY X 0.3127 Y 0.3290

Nits 100.00

Image 19. Custom material for Barco's color specification.

8. Conclusions

Probably the most important conclusion during the project and at its end, is how much the company products were lacking HDR features without even its clients noticing it. Now that the features are there, clients are using them and also proposing improvements. But summarizing four important conclusions were obtained after the final version of the engine was finalized:

- Color management has been a harder task than anticipated, and it required a lot of theory the development team lacked, but once implemented, the gained knowledge and the achieved results left the company in a much better position in terms of quality and competitive advantage in respect to other software solutions.
- Some of the developments are more related to the film industry than the broadcast area (3D luts, RGB444, full range for video…). However, the company also works with teams film-industry related (MR factory), and is getting very positive feedback on the new features. This has opened a new line of business, so the company is expecting this work to have real return from now on.
- The increase in photorealism PBR, together with the color managed pipeline, places us in a situation where integration with external packages is easier. Specially Unreal Engine, which is being demanded by customers, is easier to integrate. This is a great differentiation from the

competitors, which puts again the company in a good competitive position.

- The way HDR has been implemented makes the product capable of supporting many formats and providing very good results in terms of quality. The complexity that HDR involves is hidden to the user, by providing default configurations that enable transparently the usual workflows. But at the same time, it is possible to unlock the full power and flexibility of the system, and tweak it for any particular needs.
- Validation of all developments was carried out in-house with the production department and with some early adopter clients and their feedback on the tool has been invaluable to evolve things in the correct direction. The company understood very fast how HDR complexity needed to be hidden for common users and still possible to be unleashed for advanced ones. Some of the implemented features like Inverse Tone Mapping or Automatic Eye Adaptation were decided to be maintained as production code after the production team demonstrated that they are not really useful in broadcast environments and involve a little delay on calculations.
- Finally as the tool has been used with all the exposed features, a great effort has been made to organize the GUI regarding the proposals and GUI designs proposed by our in-house users.

9. References

- R1. ITU-R BT.2020: Parameter values for ultra-high definition television systems for production and international programme exchange
  - https://www.itu.int/rec/R-REC-BT.2020

- R2. ITU-R BT.2087: Colour conversion from Recommendation ITU-R BT.709 to Recommendation ITU-R BT.2020
  - https://www.itu.int/rec/R-REC-BT.2087

- R3. ITU-R BT.2100: Image parameter values for high dynamic range television for use in production and international programme exchange
  - https://www.itu.int/rec/R-REC-BT.2100

- R3. ITU-R BT.2390: High dynamic range television for production and international programme exchange
  - http://www.itu.int/pub/R-REP-BT.2390

- R5. CTA-861-G: A DTV Profile for Uncompressed High Speed Digital Interfaces
  - https://glenwing.github.io/docs/CTA-861-G.pdf

- R6. Perceptual Quantiser (PQ) to Hybrid Log-Gamma (HLG) Transcoding
  - http://downloads.bbc.co.uk/rd/pubs/papers/HDR/BBC_HDRTV_PQ_HLG_Transcode_v2.pdf

- R7. SL-HDR1 technical specification

    - http://www.etsi.org/deliver/etsi_ts/103400_103499/10343301/01.02.01_60/ts_10343301v010201p.pdf

- R8. HDR formats explained
    - https://www.cnet.com/news/dolby-vision-hdr10-advanced-hdr-and-hlg-hdr-formats-explained/
- R9. PBR and IBL approximations on OpenGL
    - https://learnopengl.com/PBR/Theory
- R10. Real shading in Unreal Engine 4
    - https://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_notes_v2.pdf
- R11. Style Aware Tone Expansion for HDR Displays
    - https://graphicsinterface.org/proceedings/gi2016/gi2016-8/