D4.5 Demonstrator collaborative web tool

Interim demonstrator of collaborative web tool

Grant Agreement nr	761544
Project acronym	HDR4EU
Project start date (duration)	July 1st 2017 (36 months)
Document due:	30th June 2019
Actual delivery date	28th June 2019
Leader	UPF-GTI
Reply to	josep.blat@upf.edu
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	D4.5 Demonstrator collaborative web tool
Security (distribution level)	PU
Contractual date of delivery	30th June 2019
Actual date of delivery	28th June 2019
Deliverable name	Demonstrator collaborative web tool
Type	DEM
Status & version	Submission Version
Number of pages	15
WP / Task responsible	UPF-GTI
Other contributors	-
Author(s)	Alejandro Rodriguez, Javier Agenjo, Josep Blat
EC Project Officer	Mr. Rapolas Lakavicius, Rapolas.LAKAVICIUS@ec.europa.eu
Abstract	This documents describes briefly the interim demonstrator of the web collaborative tool (available at https://webglstudio.org/users/arodriguez/projects/HDR4EU/) . It discusses the main novel features of the demonstrator with respect to the previous web demonstrator: a web-based tool for assembling HDR images from LDR ones to be used as environment maps for photorealistic rendering using image based lighting; a collaborative repository of environment maps, which is also integrated in the demonstrator; and a HDR format to store both the HDR images and some pre-computations, to enhance the rendering performance without compromising the quality. It also presents the different improvements in the tools of the previous demonstrator.
Keywords	Web-based HDR tools and pipelines
Sent to peer reviewer	Yes
Peer review completed	Yes
Circulated to partners	No
Read by partners	No
Mgt. Board approval	No

Document History

Version and date	Reason for Change
Mid June 2019	Initial version
June 25th 2019	Version for internal review
June 28th 2019	Revision in response to review: final version submitted to Commission

Table of Contents

1 Executive Summary

This documents describes briefly the interim demonstrator of the web collaborative tool (available at https://webglstudio.org/users/arodriguez/projects/HDR4EU/). It discusses the main novel features of the demonstrator with respect to the previous web demonstrator: a web-based tool for assembling HDR images from LDR ones to be used as environment maps for photorealistic rendering using image based lighting; a collaborative repository of environment maps, which is also integrated in the demonstrator; and a HDR format to store both the HDR images and some pre-computations, to enhance the rendering performance without compromising the quality. It also presents the different improvements in the tools of the previous demonstrator.

2 Background and Introduction

This deliverable, **D4.5**, belongs to **WP4T6**, which is focused on the implementation of the web-based interactive 3D graphics application discussed in the DoA. In the task description, the different steps to be covered were stated as:

- Develop from prototype a renderer on the web achieving photorealistic results.
- Establish a process and a set of tools on the web to extract the environmental light, store that information and use it to match the rendered scene with the setting of a real environment.
- Real-time chroma keying algorithms on the web.
- Merge real-time images rendered in HDR with an input source of video streaming (LDR).
- Integration of the previous components within WebGLStudio
- Development from prototype of a collaborative web based tool.

In the previous deliverable **D4.4**, submitted in m12 of the project, an interactive demonstrator was produced, which essentially showed a proof of concept of a basic web-based rendering pipeline able to deal with HDR images, and, essentially, was able to display a PBR (Physically Based Rendering) rendered static model, which could be viewed from different angles, under a single Image Based Lighting (IBL) probe. The demonstrator addressed, in a basic way, the first five points indicated above, and the sixth one was not considered.

In this second demonstrator, we aim at showing that it is feasible to use a HDR rendering pipeline on the web, within its limitations. Indeed, this means that using the demonstrator should be possible to generate photorealistic images using light information captured from the real world (using affordable resources), and mixing real footage with synthetic content. It is a very enhanced version of the previous demonstrator, and it is available at the same link:
https://webglstudio.org/users/arodriguez/projects/HDR4EU/

In general terms, the second demonstrator has improved the five aspects mentioned above, going beyond the proof of concept stage, and a key addition is to provide ways for creators to fully use HDR content through the web, so that collaboration is enabled.

Taking into account the lack of easily available HDR content, one of the aspects discussed in this deliverable are the web-based tools to create HDR content (HDR images abbreviated HDRi in this deliverable) from standard dynamic range (SDR) images. A second aspect in the demonstrator is to enable a repository of HDRi, so that creators can easily capture and render HDR images on the web, and reach their viewers faster.

Due to the complexity of dealing with HDR within the web limitations, as most HDR processing approaches are very computational demanding, a major step to enable the interactivity has been to develop a novel format for HDR, which we call HDRE. This custom format involves a series of key pre-processing steps, which allow the rest of the web-based interactive rendering to take place in the web context. This allows extensive use of environments as IBL, and is able to provide realistic PBR. On the other hand, the range of tone mappers has been extended in this second demonstrator.

Before going into the detail of the deliverable, let us recall that the main point of this **WP4T6** is to enable HDR content on the web, as more media content is being moved to the web, and specifically HDR content represents a challenge taking into account the web constraints. Our approach has been based on analyzing the existing pipelines and tools to create HDR content. When translating current pipelines to the web environment, it is key to check their weak points from the point of view of the browsers limitations. The kernel of our approach is to create a proof of concept interactive web-based renderer designed to apply realistic light transmission equations, optimized data formats and proper image tone mappers to achieve the expected quality for HDR content.

How could browsers and networks handle HDR content from creators, from HDR capture, through HDR image/video file formats, to the generation and postprocessing of HDR images in real time and their display? Our experience in developing renderers and tools for the web has helped us to identify the weaker points in the current pipelines; problems include that current browsers do not support by default HDR image file formats, that there are not professionally tested renderers on the web that support current professional HDR standards, and the lack of tools to create HDR content for the web. Our main goal is to address these issues by creating a solid pipeline and the tools required around it, and making them available open source on the web.

The next section discusses the concept and implementation of the novel features, the assembling and repository of HDR images to be used as environment maps. Section 4 discusses the improvements of the existing tools in the first demonstrator and the integration of the novel features. A final section discusses the conclusion and further work.

3 HDR Images (HDRi)

Using HDR in current rendering pipelines is a big challenge. The entire pipeline is divided into sections, where each does not necessarily have a strong connection with the others. For example, the capture of HDR images and the render process concern different types of companies, as happens in the consortium of the project.

Within this overall challenge, when considering web based collaboration, one of the first issues creators come up against is that HDR resources are not easily available. Most of quality HDR files are privately owned, and hence it is difficult to work with them. That is why we propose a web based, low cost alternative to the creation of HDR images. The process is based in LDR assembling algorithms using bracketing. This section describes the concept, the implementation, the results, and the repository of HDR images created.

3.1 Creating HDRi on the web: Concept

Adobe Photoshop and similar raster graphics editors offer useful capabilities to create HDR images from low dynamic range (LDR) bracketed photos, using what is called the *multiple exposures technique*.

Contributing to HDR content creator pipelines in the web context, we have implemented an approach to perform HDR assembly completely on the web. Of course, the resulting quality is lower than the achievable one in desktop editors, but has the advantage for content creators of using a fully web based pipeline.

The implementation carried out is based in the algorithm of Ahmet Oguz Akyüz in the paper *High Dynamic Range Imaging Pipeline on the GPU [1],* which tries to generate HDR images merging bracketed sequences of LDR images only using the GPU. Essentially, the color of each pixel I_j in the merged HDR image can be computed using the following formula:

$$I_j = \sum_{i=1}^{N} \frac{f^{-1}(p_{ij})\omega(p_{ij})}{t_i} / \sum_{i=1}^{N} \omega(p_{ij})$$

where:

- N is the length of the LDR image sequence
- p_{ij} is the color (value) of the pixel j in the image i
- f is the camera response curve
- w is a weighting function used to attenuate the contribution of under/over exposed pixels
- t_i is the exposure time of the image i

LDR images are typically captured in the nonlinear sRGB color space. In this equation, we can use the inverse of the camera response function (f^{-1}) to linearize the LDR images, which can be recovered from the LDR sequence using response curve recovery algorithms, or assume that the images match the sRGB standard. In the latter case, it is possible to use the OpenGL **sRGB** texture support (also available in WebGL as *EXT_sRGB [2]*).

Although the result is logically improved when computing the camera response curve given the camera settings for the bracketed sequence, the algorithm becomes slower hence the performance is drastically affected.

The final implementation takes benefit of the sRGB texture support of the rendering API. Once the HDRI has been assembled using the GPU, the final step is to process it in order to generate a correct 3D environment.

3.2 Creating HDRi: Implementation

As discussed in the previous subsection, it is possible to assemble different LDR images into a single HDR one. Generally speaking, this process is carried out by weighting each image pixel depending on its luminance (remove under/over exposed pixels) and the color curve of the resulting image. The final result is hence an averaged sum of all weighted pixels. The formulation presented earlier to compute the final color of a HDR image from merging an LDR sequence, results into images that contain details in both bright and dark regions - thus increasing the dynamic range.

More precisely, in that formulation, it is assumed that the bracketed photos are taken 1 *f*-stop apart from each other following the multiple exposure technique. For the implemented algorithm, the sequence has to fulfill some requirements:

- Image exposures are 1 *f*-stop apart
- Images are sorted from underexposed to overexposed
- The minimum length of the sequence is 2 (images)
- The maximum length of the sequence is 6
- Since this algorithm uses the sRGB extension of WebGL, the images must be captured in nonlinear sRGB color space

And, of course, the images must be LDR, since the aim of this process is to create a HDR image from low dynamic range ones.

Once the sequence is loaded into the application, we perform a shader call to merge all images. It is necessary to use a screen shader to output all the vertices forming a quad. Before the shader call, we have to pass the sequence images as *uniform sampler2D* to the fragment shader together with the number of images.

Following the formula indicated to compute the final color of each pixel, Listing 1 shows how the merge code would appear:

```
for( int i = 0; i < 16; i++ )
{
        if( i < numImages )
        {
                vec3 ldr = samplers[i];
                vec4 lum = luminance( ldr );
                vec4 w = weight( lum ) + 1e-6;
                vec4 exposure = pow(2.0, vec4(i - refId));

                hdr.rgb += (ldr*exposure) * w;
                weightSum += w;
        }
}

hdr.rgb /= (weightSum + 1e-6);
hdr.a = log(luminance(hdr.rgb) + 1e-6);

gl_FragColor = hdr;
```

Listing 1: Fragment of code taken from the assembly shader, based on the algorithm in [1]

In this piece of shader code, *samplers* is a *vec4* array containing the image color in that specific pixel for all the LDR sequence. *refId* refers to the pixel color of the middle image in the image stack. Since we assume the image exposures to be separated 1 *f*-stop apart, this allows us to compute the exposure ratios using *refId,* instead of asking for them in the editor.

The *main* function calls two different functions, *luminance* and *weight,* to compute the pixel luminance and the contribution to the final HDR pixel value, respectively. Assuming the sRGB color space of the sequence, the computation of luminance is based on the ITU-R BT-709 primaries:

$$L(color_{RGB}) = color.r \cdot 0.2126 + color.g \cdot 0.7152 + color.b \cdot 0.0722$$

For the weighting function, we want to decrease the contribution of badly exposed pixels, while emphasizing the correctly exposed ones. For this purpose, we chose the tent function proposed by Debevec and Malik *[3]* since it is simpler than other methods and gives a good result. The code is provided in Listing 2.

```
float weight(float val) {
  if (val <= 0.5) return val * 2.0;
  else return (1.0 - val) * 2.0;
}
```

Listing 2: Weighting function used in the assembly process.

The output of this shader is the final HDR image, which has to undergo a post-processing step before being rendered on the screen. The objective of this step is to compress the dynamic range of the image, i.e., the HDR image needs to be tone-mapped to visualize it on the screen.

The tone-mapper applied is based on the photographic tone reproduction global operator (Reinhard et al 2002 [4]) . It computes an approximation of the overall subjective brightness of the scene by using the log-average luminance. This value is then mapped into a user-defined parameter in the 0-1 range depending on the brightness of the scene.

Once mapped, the computed value has to go through a dynamic range compression, in this case using a sigmoidal compression function. In the paper written by Ahmet Oğuz Akyüz, it is suggested to introduce a new term *Lwhite* for intentional color burning. This is expected to create a natural effect:

$$L_d(x,y) = \frac{L(x,y) \cdot (1 + \frac{L_d(x,y)}{L_w^2 hite})}{L(x,y) + 1}$$

Since $L_d(x, y)$ does not represent a color value but the display luminance, the last step uses the *XYZ* to *RGB* transformation to obtain the final color values.

In our implementation, the tools just described have been integrated within the web demonstrator previously developed, thus making the use of the pipeline easier to the user. The assembled HDRIs can then be used directly as environment maps, without the need of importing/exporting or processing files in external editors. We discuss more in detail the implementation in another section.

3.3 Creating HDRi: Results

As we can see in Figure 1, the merging of 3 LDR images with some under/over exposed zones has been performed successfully. In theory, the final image (a single HDRi) should fix any bad exposed areas of the LDR sequence. Looking again at Figure 1 it is possible to perceive the improvement.

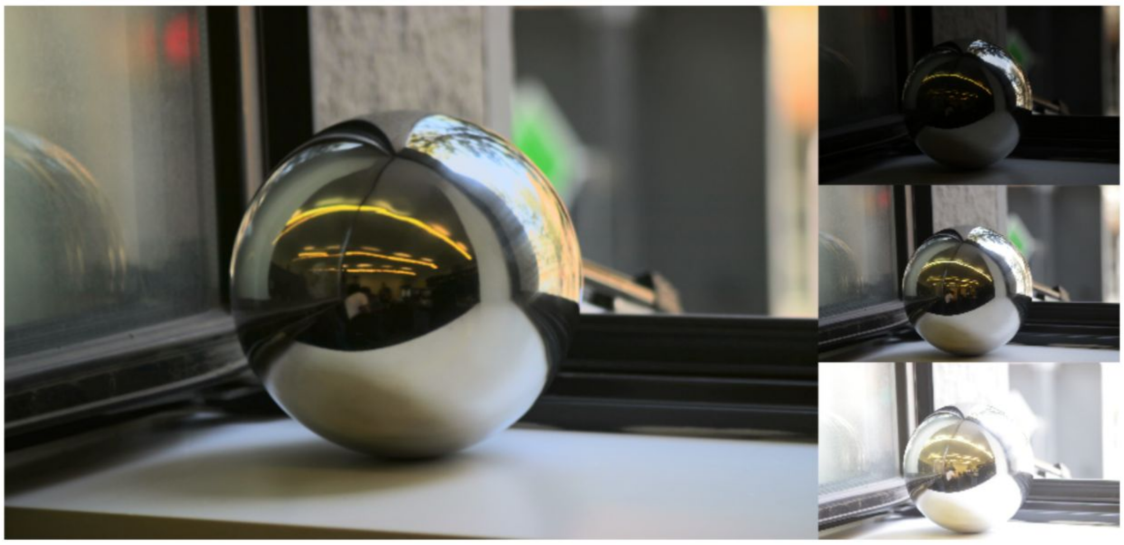

Figure 1: Result of the implemented assembly process using an LDR sequence of 3 images.

It is important to remember that this type of algorithms can be computationally expensive, depending on the final quality intended to be achieved. Since we work on the web, we face performance limitations; by now, as a first version of the tool, the results seem satisfactory, and the current goal is almost reached.

3.4 Storing HDR images: Shared repository

In order to support the collaboration over the web, the images created should be stored somewhere, and implement the concept of a collaborative repository of HDR images. One should recall that the most important use of these images is to become environment maps, to support PBR based on IBL. In this repository everyone should be able upload and download any image from an online platform. As the uploaded files should become environment maps, they will be saved as HDRE, the HDR format we have proposed, where some elements are pre-computed, for fast processing when using photorealistic rendering. The repository also supports searching by name and tag.

Although the webpage for this service is not enabled yet, it is possible to access the remote server (repository) through the current demonstrator, which is an evolution to prototype of the proof of concept demonstrator presented in **D4.4**. Indeed, the demonstrator includes an environment selection based on the environments stored in the online platform.

The functional repository, which contains the current default and tagged files stored is available at:

https://webglstudio.org/users/hermann/files/sauce_dev/files/8efb30d54aee665af72c445acf5328
4b/hdre/

4 Progress over the previous Web Demonstrator and Integration of the New Features

4.1 Progress over the previous demonstrator

The demonstrator presented in **D4.4** essentially represented an advanced proof of concept of HDR on the web. It needed quality improvements in the different functionalities to fully support advanced HDR with enough quality to be used by content creators.

The first, key aspect, relates to the pre-processing of the environment maps. Photorealistic rendering associated to Image Based Lighting based on a HDRi as an environment map, can be implemented through computing the illumination integral by splitting it into two parts which can be precomputed, namely the prefiltered environment map and the environment BRDF, this one stored in a 2D lookup texture (LUT). This is discussed in **D4.4**. As a very high number of samples in the environment map needs to be used to avoid artifacts in the rendered scene, the decision was made to carry out the pre-filtering of the HDRis when storing tem, and these computations stored in the file with the HDRE format, as these computations need to be carried out only once. This way the current demonstrator has had a significant performance boost, while, at the same time, producing rendered images of excellent quality.

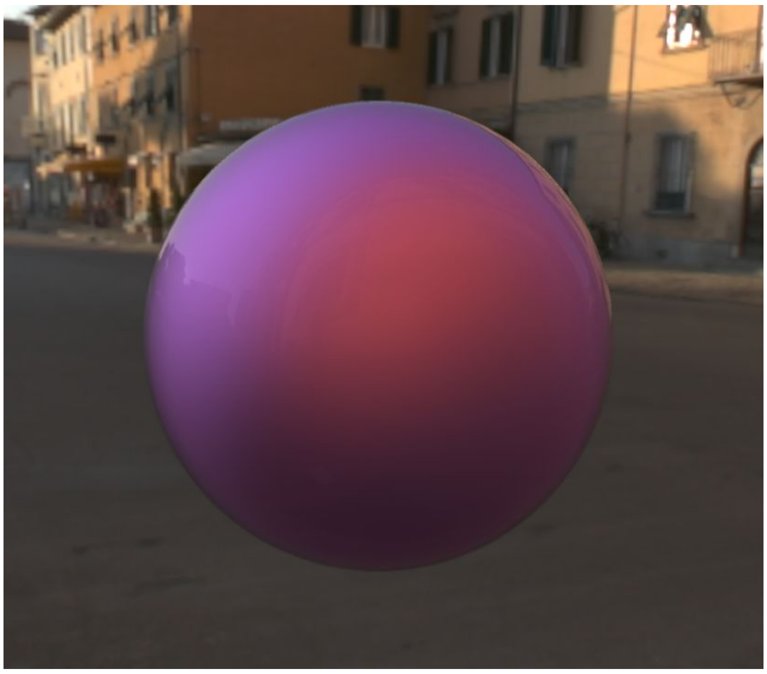
Figure 2: Mixing real and synthetic images

The PBR implementation has been improved as well in both quality and performance, with a range of realistic materials. The physically based camera simulation implementation has been improved as well, enhancing the post-processing effects presented in **D4.4**. See more detail about these aspects in [5]. Some of the tools which were not integrated in the first demonstrator have been integrated.

Another improvement over the previous demonstrator has been carried out in the tone-mapping. We already mentioned one of the tone-mappers, but several other ones have been implemented as well, to provide different alternatives for screen visualisation.

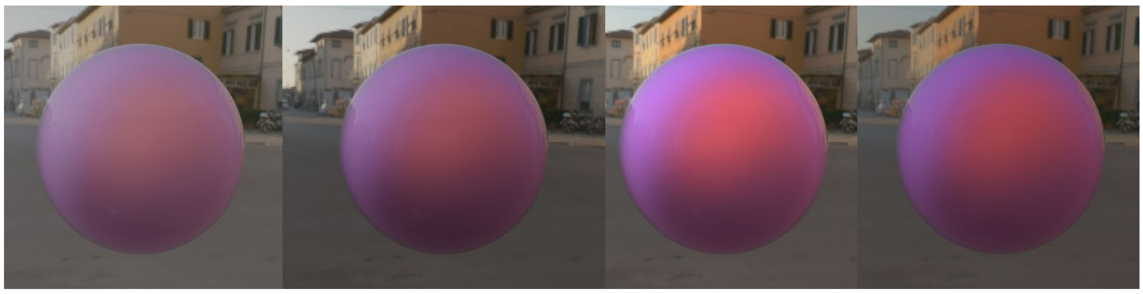
Figure 3: From Left to Right: Reinhard, Atmos, Exponential and PTR Tonemappers

The current demonstrator also integrates within the Graphical User Interface (GUI) the new tools presented in this deliverable. This is discussed in the following two subsections.

4.2 HDRE Repository

An independent webpage can be created for serving the HDR content created by ourselves, or by other creators (this is discussed later in the subsection *Future Work*) without the need of forcing users to go through the demonstrator. However, it seems easier for users to get advantage of the 3D environments stored in the HDRE repository directly from the demonstrator.

This has been implemented in the GUI through a folder button (📁), which now appears at the scene configuration panel, the right area of the UI, next to the environment selection option. When clicking on the button, a dialog appears and it is possible to choose from all the uploaded files, which are draggable, and filter per tag (e.g. outdoor, indoor, nature). Figure 4 shows the selection dialog.

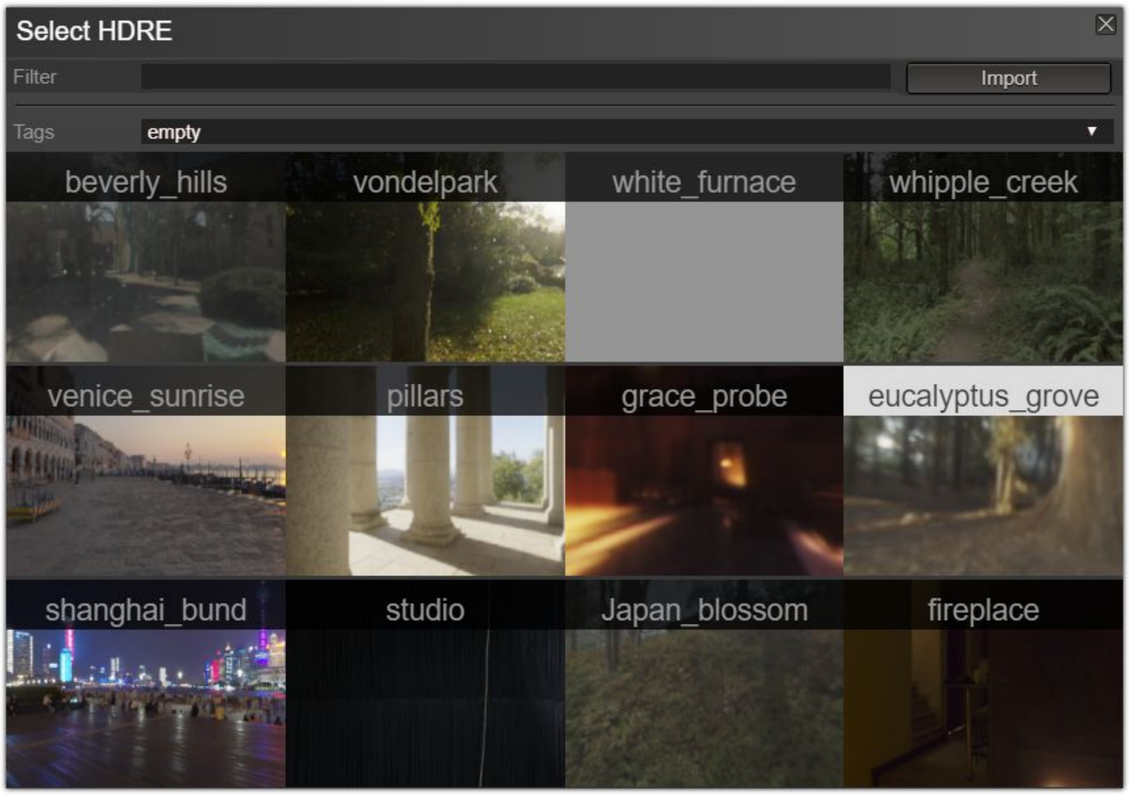

Figure 4: UI dialog for selecting 3D environments stored in the HDRE repository.

4.3 Assembly Tool

The Assembly Tool to create HDR images that can be used as environment maps appears with its own tab in the main canvas, since the functionality is quite different from the 3D scene manipulation, which has been included within a tab, to avoid potential confusion to the usera. The solution implemented appears as shown in Figure 5:

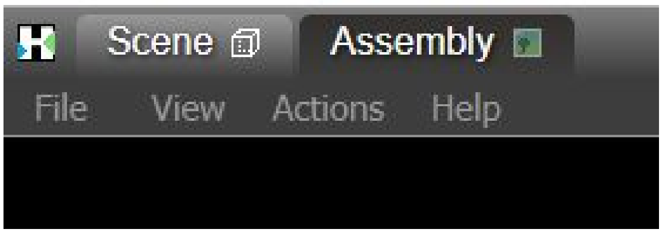

Figure 5: Tab organization of second web demonstrator.

Inside the new tab, the editor uses an empty panel for adding different user-defined parameters, the action buttons for assembling or uploading the final HDR image and other relevant information for the assembling process. The space at the bottom is reserved for LDR sequence previews together with its basic information. In case of using the response curve recovery algorithm (also implemented, although the potential drawbacks have been discussed earlier), users are able to change the exposure time of each image in the stack using a small button next to each image thumbnail.

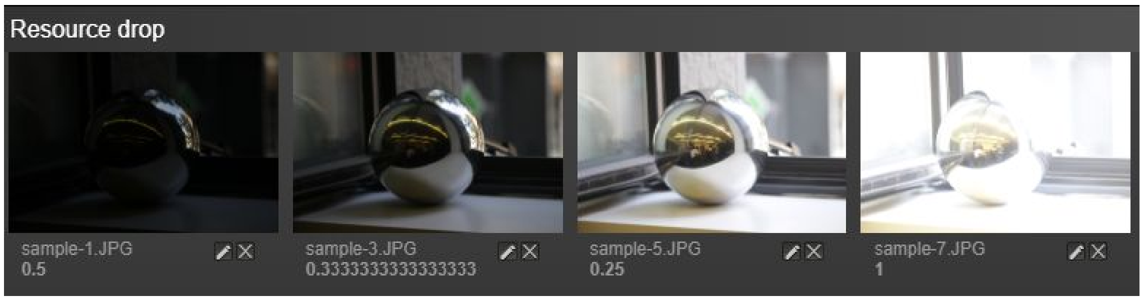

Figure 6: Resource area used to show LDR previews.

5 Conclusion and Future Work

5.1 Conclusions

This deliverable has presented the current demonstrator of web-based tools supporting HDR. The most novel features are the ability to assemble HDR images from LDR ones, and the creation of a repository to support the online collaboration around HDR content. On the other hand, it has presented the improvements over the previous proof of concept demonstrator, so that the current demonstrator shows prototype capability for a fully web-based pipeline for HDR content, which has been challenging because it has required the adaptation of the algorithms to the web since generally they are implemented for offline pipelines. A novel HDR format permits to improve performance without compromising rendering quality.

The assembly tool makes easier the job for HDR creators offering an online - and open-source - tool for creating environment maps, with the added possibility to upload them and contribute to a public repository, which should support online collaboration. The demonstrator showing collaborative possibilities has been made available.

5.2 Future Work

There are several lines of improvement in the demonstrator presented in this deliverable **D4.5**.

The first one relates to the general improvement of performance and quality of the web-based tools for HDR content, including the fusion of real and synthetic images.

The second one relates to enhancing the collaborative aspects; one of them could be the creation of an **autonomous platform** for serving our own HDR content and the one provided by creators. This platform would run on the web as an open-source repository for HDR environment maps. The different necessary elements to create the platform, such as SQL databases, remote server, tagging system, are already in place, and thus it would not be difficult to come up with a useful and user-friendly application in the short term to have this platform.

The third one relates to **improving the assembling** quality and performance, based on the first implementation which already works, exploring alternative algorithms and implementations.

6 References

[1] Akyüz, A. O. High dynamic range imaging pipeline on the GPU. *Journal of Real-Time Image Processing* **10**, 273–287 (2015).

[2] EXT_sRGB - Web APIs | MDN. URL https://developer.mozilla.org/en-US/docs/Web/API/EXT_sRGB (Accessed on 06/18/2019)

[3] Paul E. Debevec and Jitendra Malik. 1997. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 369-378. DOI: https://doi.org/10.1145/258734.258884 (also *ACM SIGGRAPH 2008 classes*)

[4] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. 2002. Photographic tone reproduction for digital images. *ACM Trans. Graph*. **21**, 3 (July 2002), 267-276. DOI: https://doi.org/10.1145/566654.566575

[5] Alejandro Rodriguez-Corrales: *HDR on the Web: A Pipeline and Tools Supporting Photorealistic 3D Rendering*, Bachelor Thesis in Computer Engineering, Universitat Pompeu Fabra, July 2019 (to appear)