Painting with Light: A Technical Overview of Implementing HDR Support in Krita

Executive Summary

Krita, a leading open source application for artists, has become one of the first digital painting software to achieve true high dynamic range (HDR) capability. With more than two million downloads per year, the program enables users to create and manipulate illustrations, concept art, comics, matte paintings, game art, and more. With HDR, Krita now opens the door for professionals and amateurs to a vastly extended range of colors and luminance.

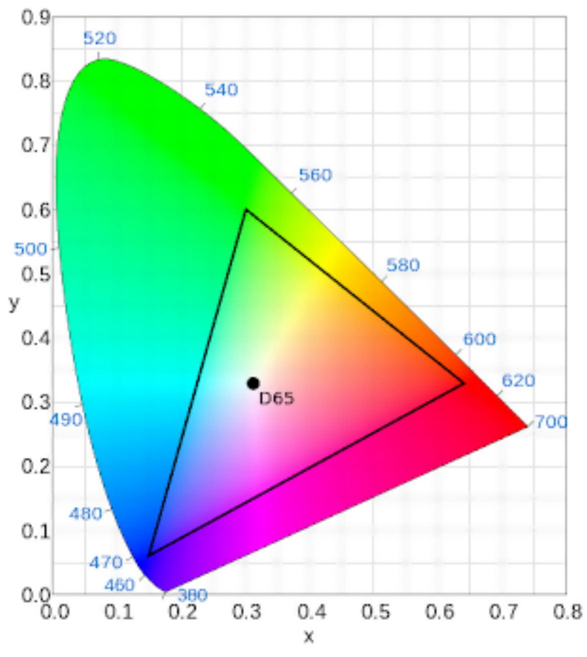
The pioneering approach of the Krita team can be leveraged by developers to add HDR to other applications. Krita and Intel engineers have also worked together to enhance Krita performance through Intel® multithreading technology, while Intel's built-in support of HDR provides further acceleration. This white paper gives a high-level description of the development process to equip Krita with HDR.

A Revolution in the Display of Color and Brightness

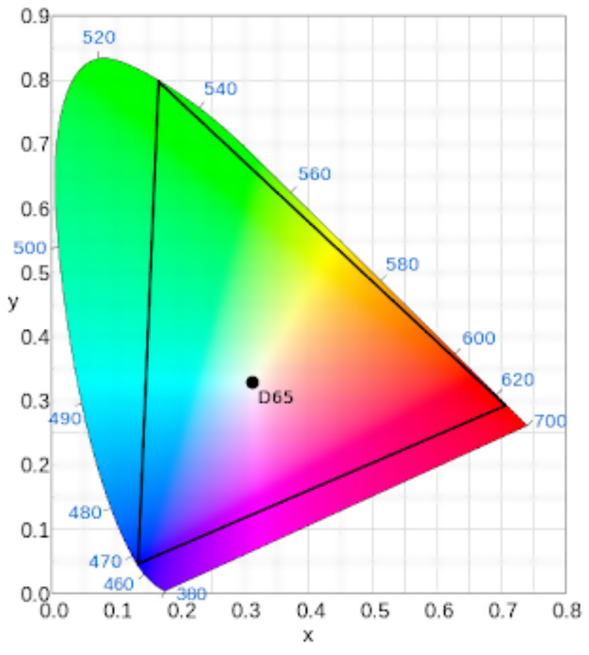
Before the arrival of HDR, displays were calibrated to comply with the D65 standard of whiteness, corresponding to average midday light in Western and Northern Europe. Possibilities of storing color, contrast, or other information about each pixel were limited. Encoding was limited to a depth of 8 bits; some laptop screens only handled 6 bits per pixel.

By comparison, HDR standards typically define 10 bits per pixel, with some rising to 16. This extra capacity for storing information allows HDR displays to go beyond the basic sRGB color space used previously and display the colors of Recommendation ITU-R BT.2020 (Rec. 2020, also known as BT.2020). This color space offers a significantly richer gamut of colors, notably with many more variations of greens and reds.

With HDR, screens can realistically depict sources of light. For example, they can show the lifelike effect of a sunbeam piercing through a window instead of a subdued, paper-style imitation. Whereas conventional screens are limited to a few hundred *nits* of brilliance, HDR screens can achieve 1,000 nits or more.



sRGB color space



Rec. 2020 color space
(spans much wider range of red and
green colors)

When displaying colors, HDR displays do not use traditional exponential gamma correction. Instead, they use a function called Perceptual Quantizer (PQ), which extends the dynamic range to both sun-bright and very dark values. HDR workflows generating images for HDR displays support a minimum of 10-bit color depth from start to finish, including the initial capture or creation of the images.

An Overview of the Krita HDR Implementation

Krita has been in continuous development since 2003 by volunteers and sponsored full-time developers. Krita development is based on the popular Qt* software development kit (Qt SDK). The Krita team has also built up a relationship with Intel, using Intel's multithreading technology and software development tools to enhance software application performance[1]. In addition, 7th generation and later Intel® Xeon® processors and Intel® Core™ processors support the Rec. 2020 Standard for 4K Ultra-High Definition (UHD) content creation and consumption, wide color gamut, and HDR with 10-bit enablement.

While several vendors offer HDR screens, and Intel has built HDR support into the latest generations of its products, the only operating system that currently supports HDR is Windows® 10 using DirectX*.

However, Krita uses OpenGL*, not DirectX. To support the extensions needed to work in HDR, the Krita team had to first modify the OpenGL to DirectX compatibility layer, called the Almost Native Graphics Layer Engine, or ANGLE.

The team also wanted to convert Krita's user interface from sRGB operation to p2020-pq operation, but without users being obliged to convert the main canvas immediately. The reason? HDR images are standardized to use the Rec. 2020 gamut and the PQ tone response curve (TRC). However, a linear TRC makes it easier to edit images, so the team decided to convert to PQ only when users were satisfied with their work.

This meant that additional modifications were needed for Qt. Finally, an HDR-capable color selector was also required for Krita.

Initial Challenges

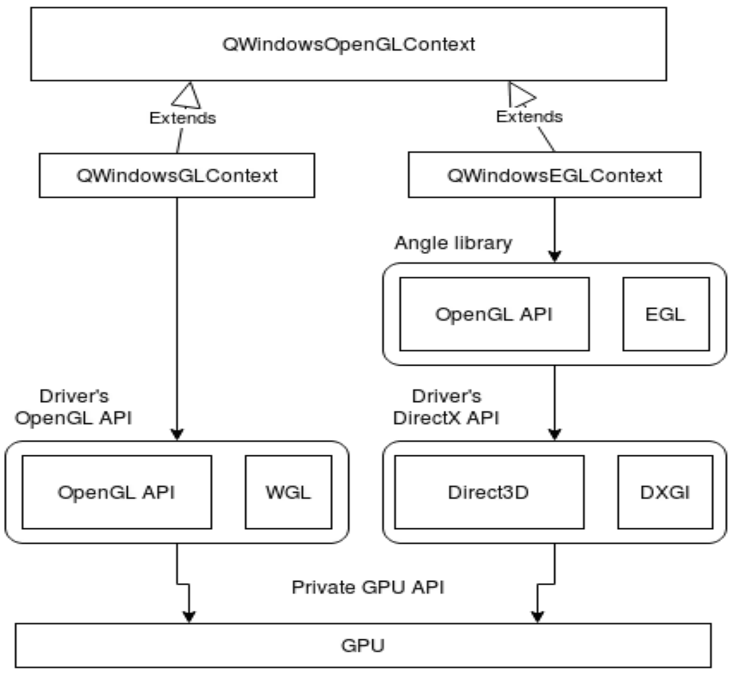
When the user switches a display into HDR mode, the Windows* operating system automatically starts to communicate with it in p2020-pq mode. All the colors that would otherwise be rendered in sRGB are automatically converted. If an application wants to render an image directly in p2020-pq, it should create a special *framebuffer* object (swap chain), set its color space to p2020-pq, and ensure that all the internal and intermediate textures have the correct color space and bit depth.

Because the graphical user interface (GUI) is usually rendered on the same swap chain, the GUI must be converted from sRGB into the destination color space; for example, 10-bit p2020-pq as above, or 16-bit scRGB that Krita also uses.

In Krita, Qt is used to render everything, including Krita's OpenGL canvas widget. Qt systematically uses an 8-bit color space for rendering window panels. Therefore, even though Krita passed 16-bit textures to the system, the data was still converted into 8 bits in Qt/OpenGL. Significant changes in Qt were needed.

HDR Support in ANGLE

ANGLE provides two interfaces: EGL* and OpenGL. EGL is used for creating windows, initializing surfaces, and configuring displays. EGL is often a substitute for platform-specific APIs like WGL (Windows) and GLX (X11). OpenGL API works with primitives, textures, and framebuffers, but not with surfaces or windows. The following diagram shows the relation between the libraries and APIs.



HDR Surfaces in ANGLE

EGL has well-defined extensions for receiving information about the color space of pixels being passed to the display. These extensions are EGL_KHR_gl_colorspace, EGL_EXT_gl_colorspace_scrgb_linear, and EGL_EXT_gl_colorspace_bt2020_pq. They were not originally supported by ANGLE. To implement these extensions inside ANGLE, the team made the EGL API function createWindowSurface() support the EGL_GL_COLORSPACE attribute, and pass the corresponding value to DXGI's swap chain (see diagram above).

HDR Surfaces in Qt*

After enabling the ANGLE EGL API to receive the color space information, work on Qt is the next stage for the team. Qt has a system for switching rendering engines at runtime. When a QApplication object is created, Qt creates one of the two *static* classes: QWindowsGLStaticContext or QWindowsEGLStaticContext (see diagram above). These classes define how Qt communicates with the graphics processing unit (GPU). They create surfaces and the OpenGL context classes, as defined by the standard.

- QWindowsGLStaticContext is a *legacy* way of communicating to the video driver. It locates opengl32.dll and tries to use its interface (typically known as WGL).
- QWindowsEGLStaticContext, on the other hand, uses the ANGLE library and its EGL interface. This was where the team made changes.

Basically, the only task was to modify the createWindowSurface() method to support color space selection. In the user code, the color space now can be passed via the default surface format:

```
// set main frame buffer color space to scRGB
QSurfaceFormat fmt;
// ... skipped ...
fmt.setColorSpace(QSurfaceFormat::scRGBColorSpace);
QSurfaceFormat::setDefaultFormat(fmt);

// create the app (also initializes OpenGL implementation
// if compiled dynamically)
QApplication app(argc, argv);
return app.exec();
```

Rendering to an HDR Surface

However, it is not enough just to create an HDR surface. HDR-canvas is not the only widget in Krita's GUI. There are also many conventional widgets like buttons, sliders, and checkboxes. All these widgets are rendered with standard QPainter, which is not aware of color spaces or HDR.

To overcome this issue, the team modified the way Qt composes windows internally. Each Qt widget has an associated OpenGL texture. When handling a paint event, the widget renders its contents onto this texture. At the end of the rendering cycle, Qt renders all these textures on screen, one by one (in QOpenGLCompositorBackingStore).

The team attached a color space tag to each texture (adding a special method QOpenGLWidget::setTextureColorSpace()) and modified QOpenGLTextureBlitter to handle this tag.

Thus, all the *legacy* QPainter-painted textures were in sRGB (SDR) color space, and all the tagged ones were handled accordingly.

Probing Supported Display Formats

A shortcoming of all OpenGL APIs is that there is no way to find out if the required surface pixel format (bit depth, RGBA versus BGRA) and color space is supported. The only solution is to request the creation of a real surface. If it succeeds, the format is supported. If it fails, it is not supported.

In Qt, this process becomes even more complicated, since the static OpenGL context is created during construction of the QApplication. The choice of the rendered type (WGL or EGL) should therefore be done in main() before the creation of QApplication.

An example of the probing code can be found in the testing application that the team added to Qt's sources.

Fetching Display Information

Yet another challenge was to fetch display information. The Windows API provides information about the connected display, its color primaries, white point, and HDR capabilities. But to fetch this info means knowing which HMONITOR handle is connected to which QScreen object. To do this, the team implemented an extension to QWindowsNativeHandleInterface. This patch to Qt can be found here, while an example of the code for fetching display information is available here.

Using Krita's HDR Capability

As an example, suppose that a user has an HDR-capable monitor, a DisplayPort 1.4 or HDMI 2.0a or higher cable, the latest version of Windows 10 with Windows Display Driver Model (WDDM) 2.4 drivers, and a PC with Intel graphics and a 7th generation or later Intel Core processor. The main steps to begin using Krita's HDR capabilities are as follows.

- Switch the display to HDR mode manually using the Windows settings utility.
- Select a default SDR brightness level to ensure that the screen display still looks normal.
- Configure Krita to support HDR. In the display settings panel, select the preferred surface (for example, **Rec. 2020 PQ** (10bit)). In the Docker* menu, select the HDR-capable small color selector.
- To create an HDR image, make a canvas using a profile with the Rec. 2020 gamut and a linear tone response curve. The appropriate profile for this is **Rec2020-elle-V4-g10.icc**.
- Create!
- Save the work. Krita's native .kra file format is a good way to save HDR images and can be used as the working format. For sharing with other image editors, use the OpenEXR format (:ref:`file_exr`). For sharing on the web, use the expanded PNG format (:ref:`HDR png export <file_png>`).
- For exporting HDR animations, Krita supports saving HDR to the new codec for MP4 and mkv: H.265.

Conclusion

This implementation of HDR support for the Krita painting application allows it to render HDR on its canvas. With the HDR color selector, users can *paint with light* by selecting the desired luminance directly on screen. The components that support this solution—Qt, ANGLE, Windows 10, Intel graphics, and Intel® 7th generation and later processors—are widely used and easily available. The approach taken by the Krita team can therefore be leveraged wholly or partly to implement HDR support for other software destined for digital artists.

Download the latest version of Krita with HDR support here

Access the KDE code repository for Krita here

[1] Performance results are based on testing as of November 23rd 2018 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

Configuration : Intel® Core™ i9-7980XE 2.6GHz, 18 Core(s), 36 Logical Processor(s), 24.75 MB CPU cache, Operating System Win10 Pro 64-bit Build 17134, 32GB RAM, Bios version / date: Alienware 1.1.10, 3/6/2018, Microcode Version: 200003A. Tested by Intel on November 23rd 2018.

Tweet: Krita is digital painting software with true HDR capability. Performance is enhanced through Intel® multithreading technology.

Summary: Krita is digital painting software with true HDR capability. Performance is enhanced through Intel® multithreading technology.