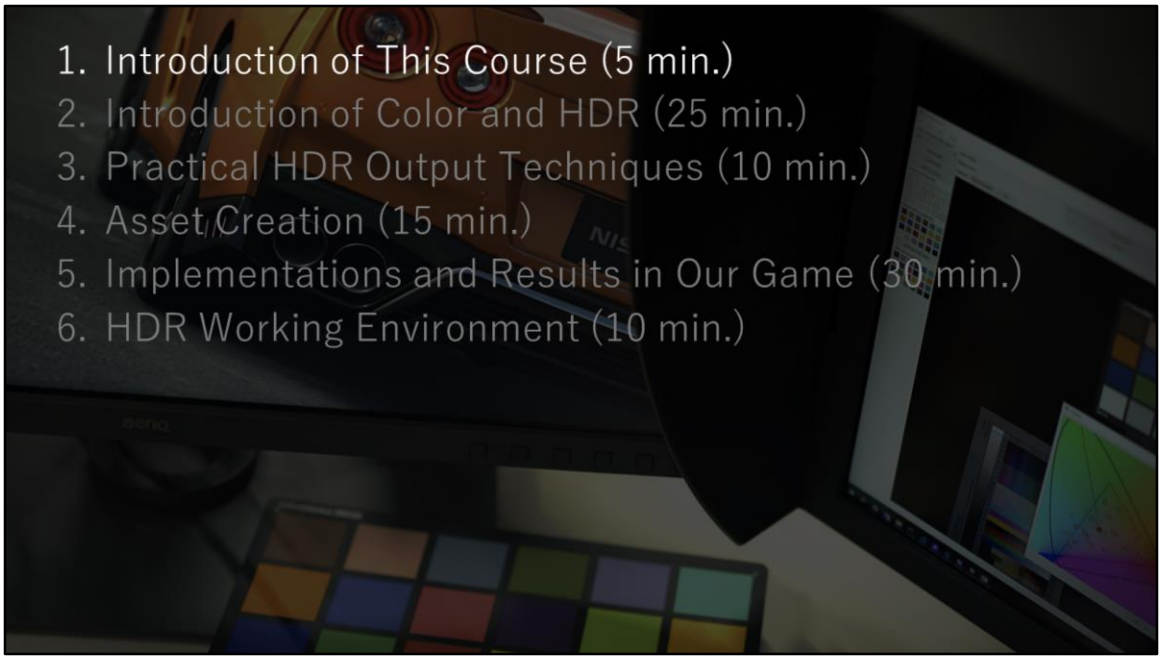
Practical HDR and Wide Color Techniques in Gran Turismo SPORT

SIGGRAPH ASIA 2018

Hajime UCHIMURA, Polyphony Digital Inc. / uchimura@polyphony.co.jp
Kentaro SUZUKI, Polyphony Digital Inc. / ksuzuki@polyphony.co.jp


1. Introduction of This Course (5 min.)
2. Introduction of Color and HDR (25 min.)
3. Practical HDR Output Techniques (10 min.)
4. Asset Creation (15 min.)
5. Implementations and Results in Our Game (30 min.)
6. HDR Working Environment (10 min.)

Presenters

Hajime UCHIMURA

- Graphics programmer at Polyphony Digital Inc.
- Image processing and color science.

Kentaro SUZUKI

- Graphics programmer at Polyphony Digital Inc.
- Post effects and so on.

Gran Turismo SPORT

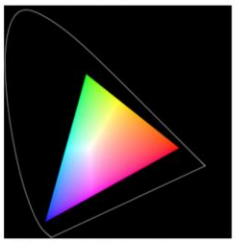
- Gran Turismo (GT)
  - “AAA” racing game series on PlayStation® platform
- Gran Turismo SPORT
  - Released in Oct. 2017
  - Highly reputation due to utilization of HDR and Wide Color technology.

Gran Turismo is “AAA” racing game series on PlayStation platform, and Gran Turismo SPORT is a latest title.
Gran Turismo SPORT was released in October 2017.
This title has a high reputation for its utilization of HDR and wide color technology.

Watch our trailer

Past of Real-time Rendering

- The quality of CG images has improved significantly over the past few years.
- However, final output quality was limited by conventional output devices.



Limited Color Space

Limited Dynamic Range

We'll talk about past days of real-time rendering.

The quality of computer graphics imagery has improved significantly over the past few years.

However, final output quality is restricted by the limitations of conventional output devices, mainly their limited color space and limited dynamic range.

New Technology: HDR and Wide Color

- HDR and Wide color technology has expanded these limitations.

Recently, HDR and wide color technology has expanded these limitations.

Using wide color space and high dynamic range standard, we can achieve more attractive images effectively.

Problems

- Finding a consistent interpretation between hardware behavior and software specification is difficult.
- As a result, implementing HDR and Wide Color technologies is difficult and non-trivial.

However, using HDR and Wide Color technologies are not trivial and are still difficult problems.
It's because consistent interpretations of both hardware behavior and software specification is difficult.

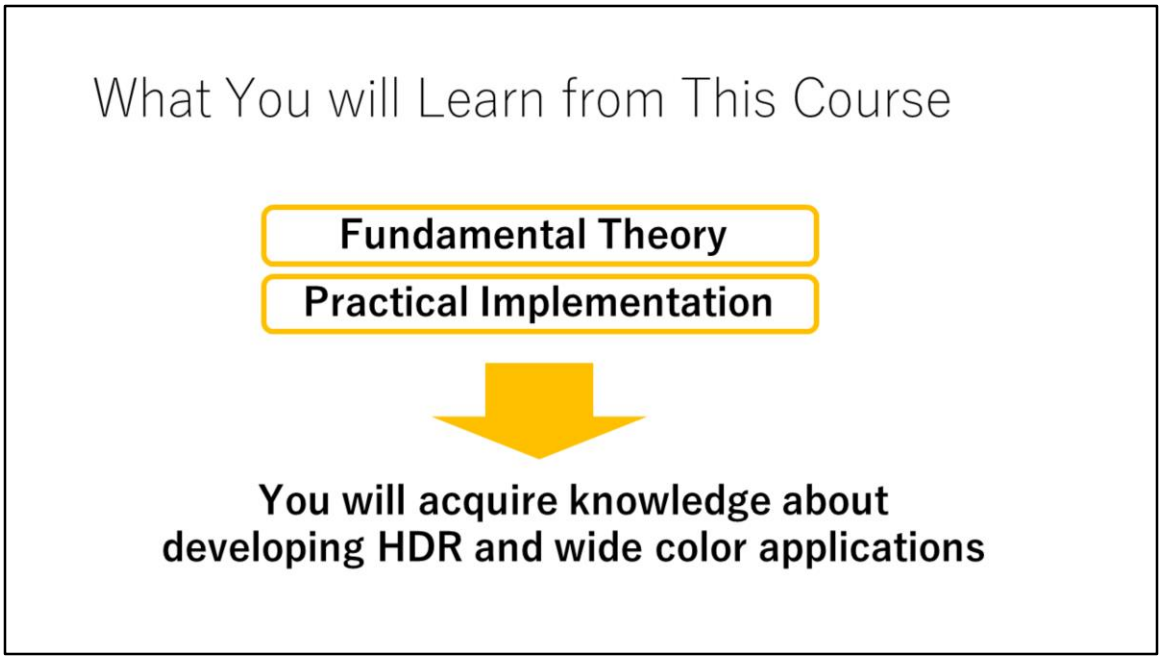
Our Solution from Game Industry

- Consistent theory-based approach for each aspect of the workflow
  - Asset collecting and editing
  - Interchangeable formats
  - Encoding
  - Working environment
  - Verification
  - Rendering pipeline
  - etc
- As a result, high quality output is achieved.

We are game developers.
So, our solution is from the game industry and it's a consistent, theory-based approach for each aspect of the workflow.
Such as

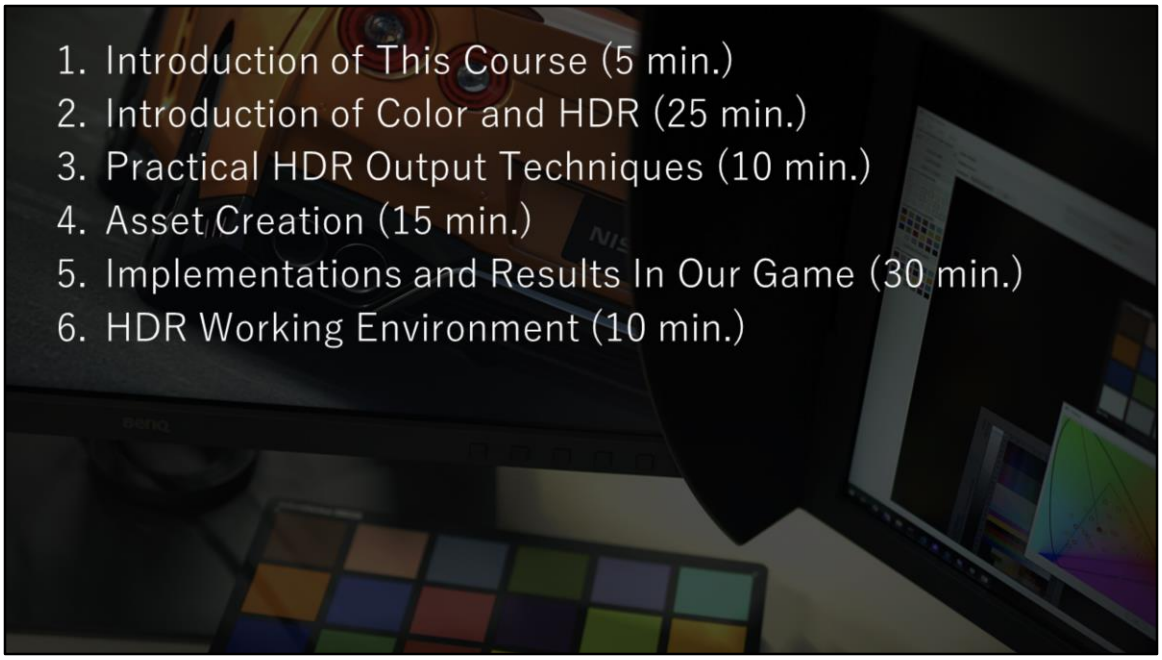
asset collecting and editing,
Interchangeable formats,
Encoding,
Working environment,
Verification,
Rendering pipeline,
And so on.

As a result, high quality output is achieved.



From our course, what you will learn are “fundamental theory” and “practical implementation” about a HDR and wide color technology. These knowledge are based on our case studies in game developing and “real” knowledge.

We believe your will get knowledge about developing HDR and wide color applications effectively and efficiently.

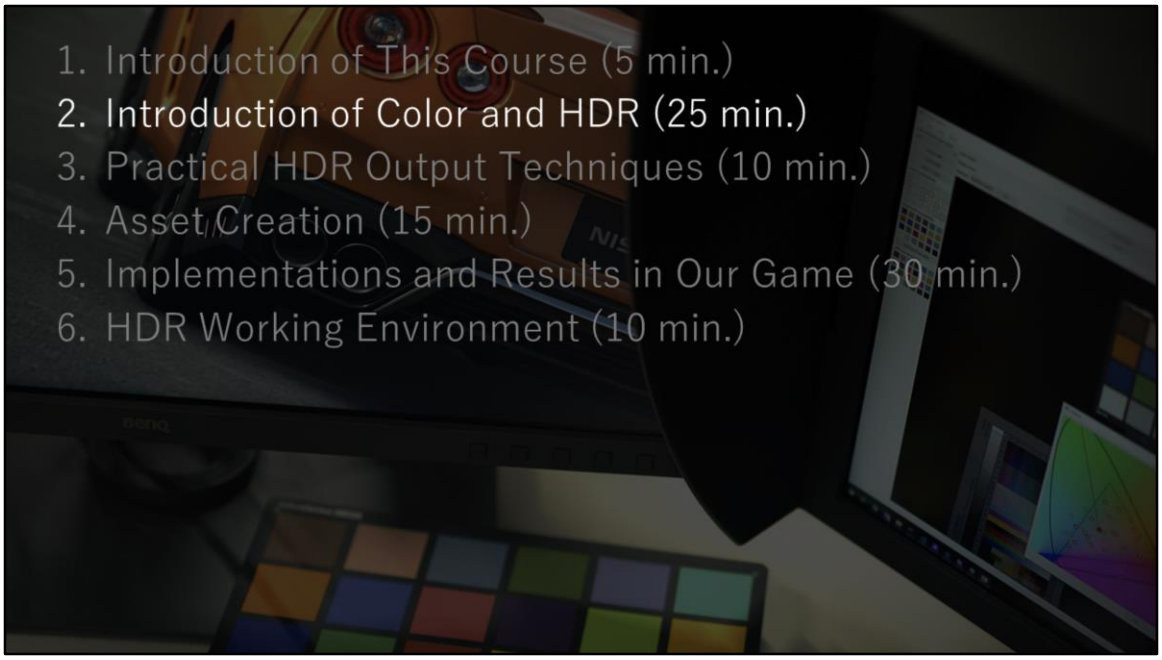


This is a overview of this course.
Chapter one is “Introduction of This Course” and we are talking now.
Chapter two is “Introduction of Color and HDR “.
In this chapter, we’ll talk about the fundamental theory about color system and HDR.
Chapter three is “Practical HDR Output Techniques”.
In this chapter, we’ll talk about the practical techniques such as tone mapping and gamut mapping.
Chapter four is “Asset Creation”.
In this chapter, we’ll talk about our HDR asset creation workflow.
Chapter five is “Implementations and Results”.
In this chapter, we’ll explain about case studies in our real game.
Implementations and results about problems we encountered.
Chapter six is “HDR Working Environment”.
In this chapter, we’ll discuss about exploring HDR working environment based on our experience.

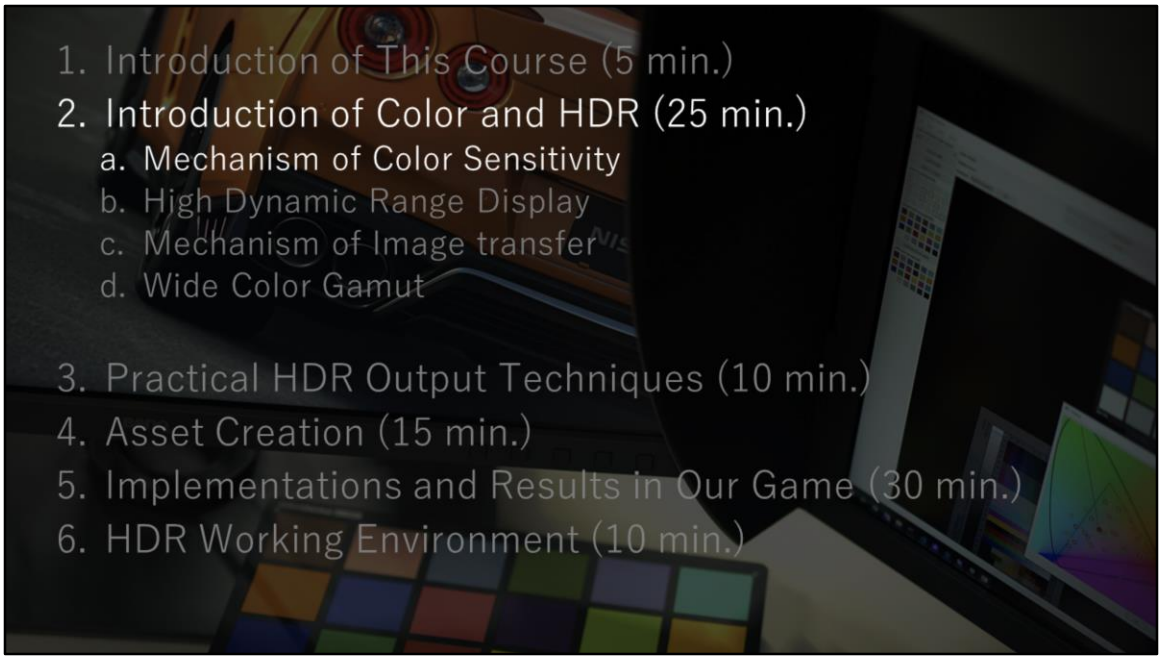
Course Materials

- http://www.polyphony.co.jp/publications/sa2018/

Our course materials are here.

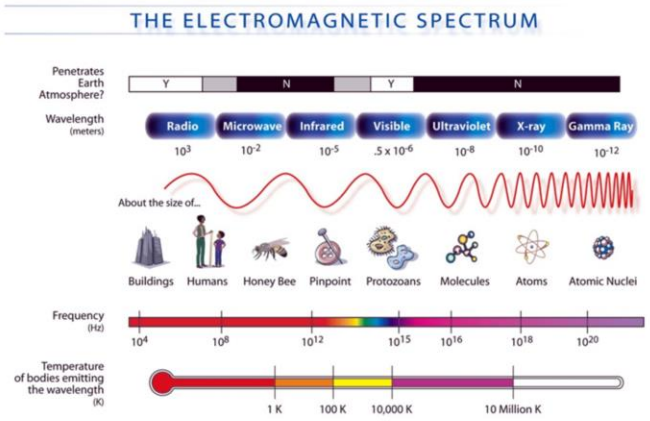


To discuss about color and dynamic ranges we have to start from some basics of color science.
We will talk about human eye mechanism and the basics of modern display technologies.



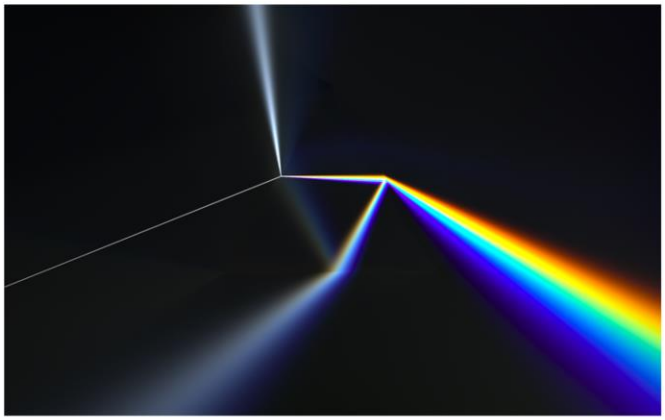
First, we'd like to talk about human eye color sensitivity.

Eye sees Electro Magnetic Wave

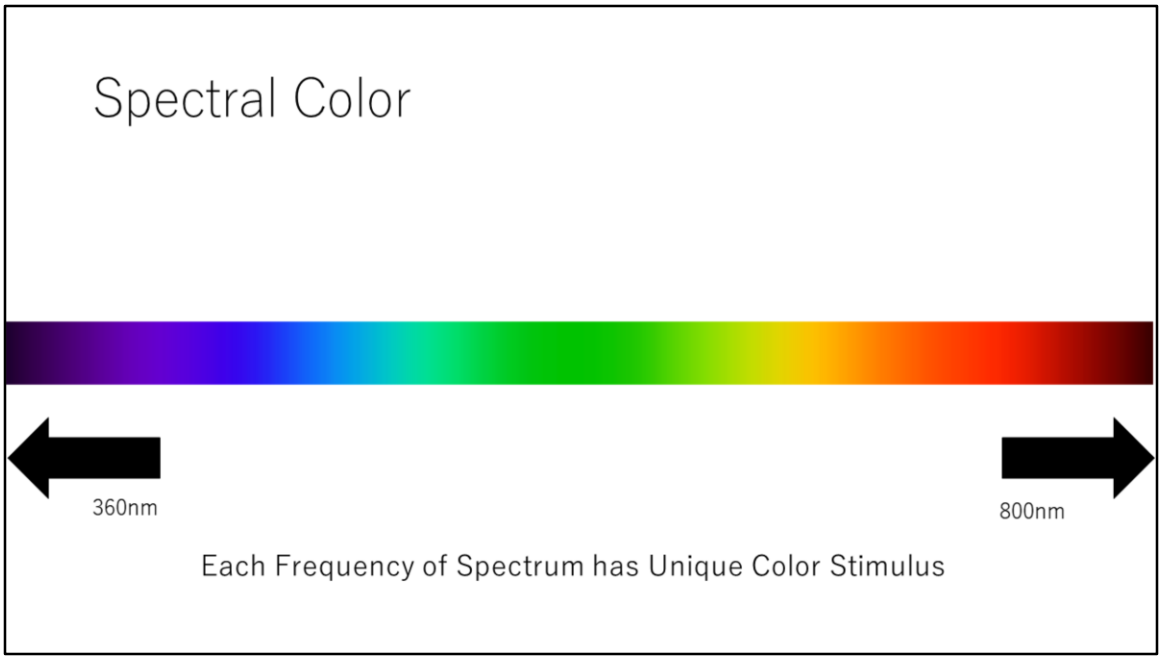


The human eye sees Electro Magnetic Wave as a light.

Light Consists of Spectrum

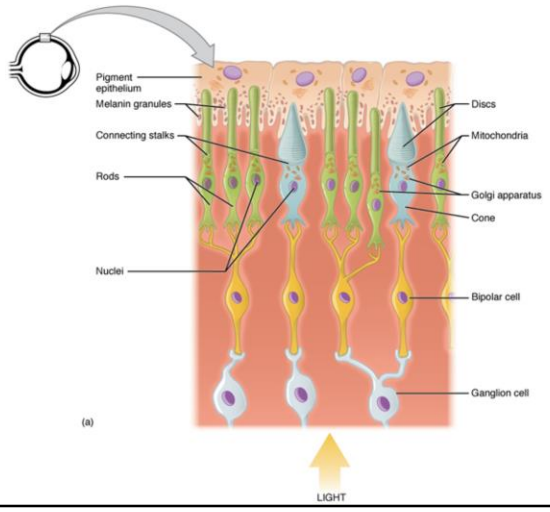


Light contains multiple different frequency electro magnetic waves.
This mixture is called spectrum.



The spectrum from 360 to 800 nm is visible to the human eye.
Each frequency of spectrum has unique color stimulation.

Cones (and Rod)



[College 2013]

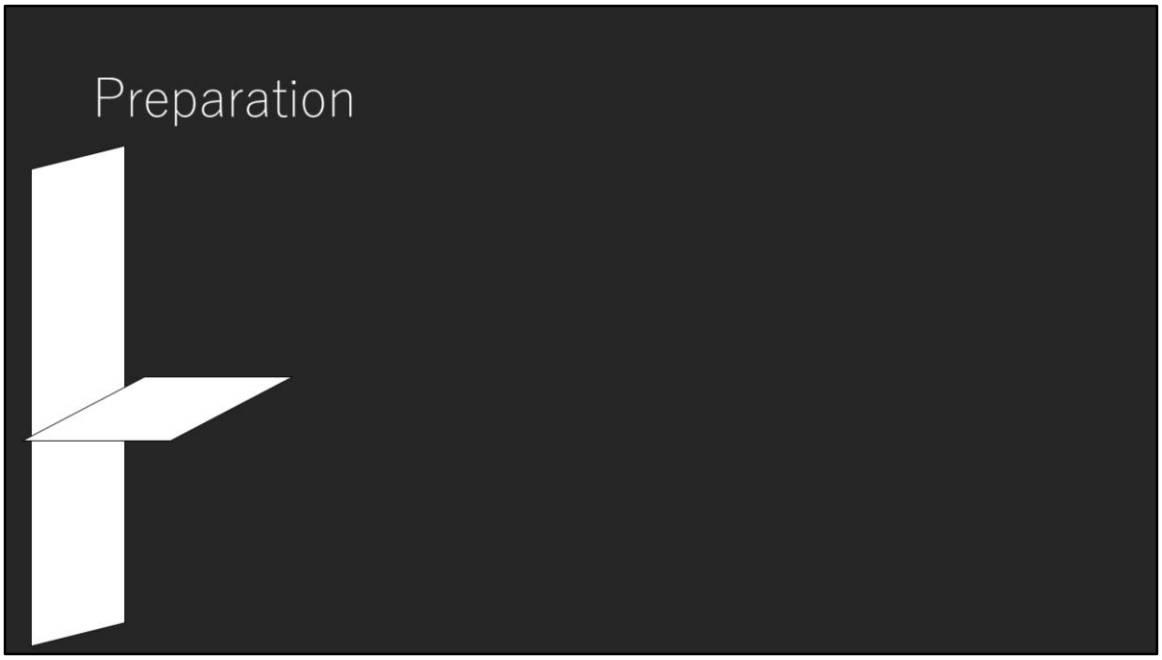
Because human eye has three cones that are sensitive to colors.

Measuring Human Eye

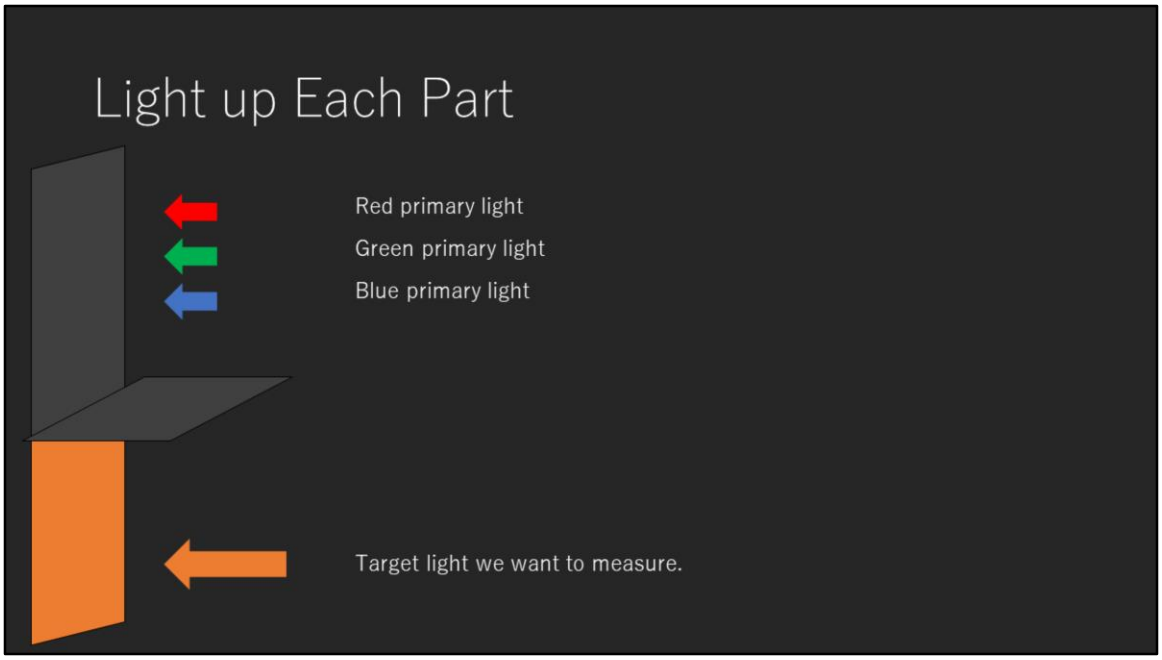
- CIE(Commission Internationale de l'Éclairage) did it in 1931. [Smith and Guild 1931]

http://www.cie.co.at/about-cie

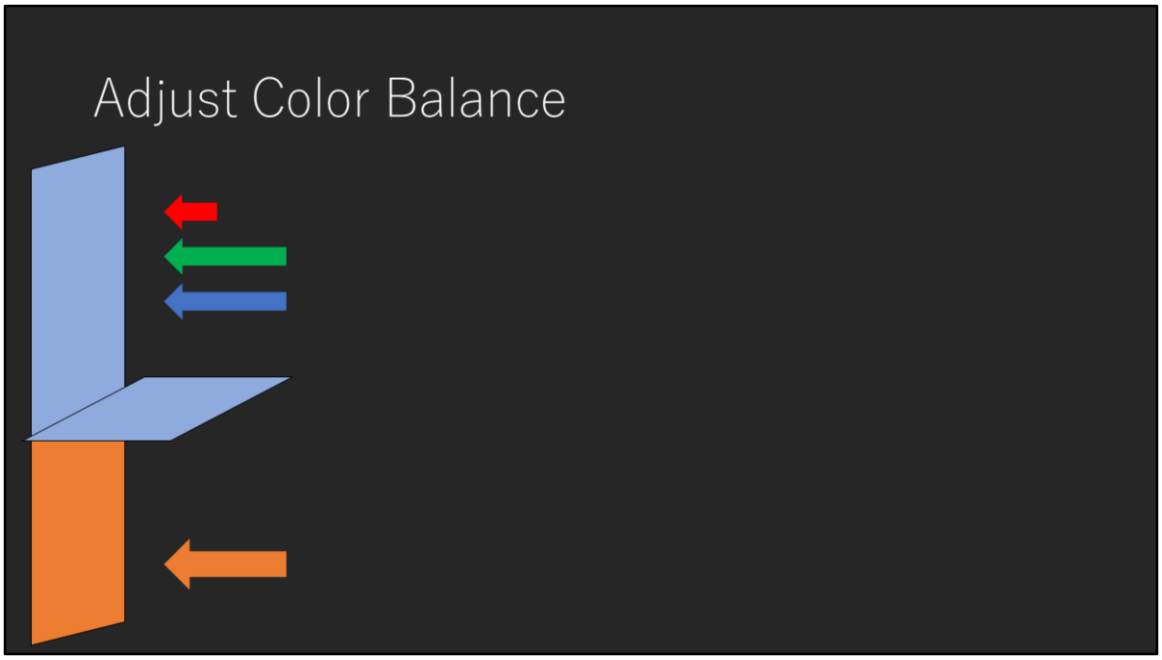
Scientists wanted to measure human eye sensitivity.
In 1931 CIE, a International Commission on Illumination did a research on it.



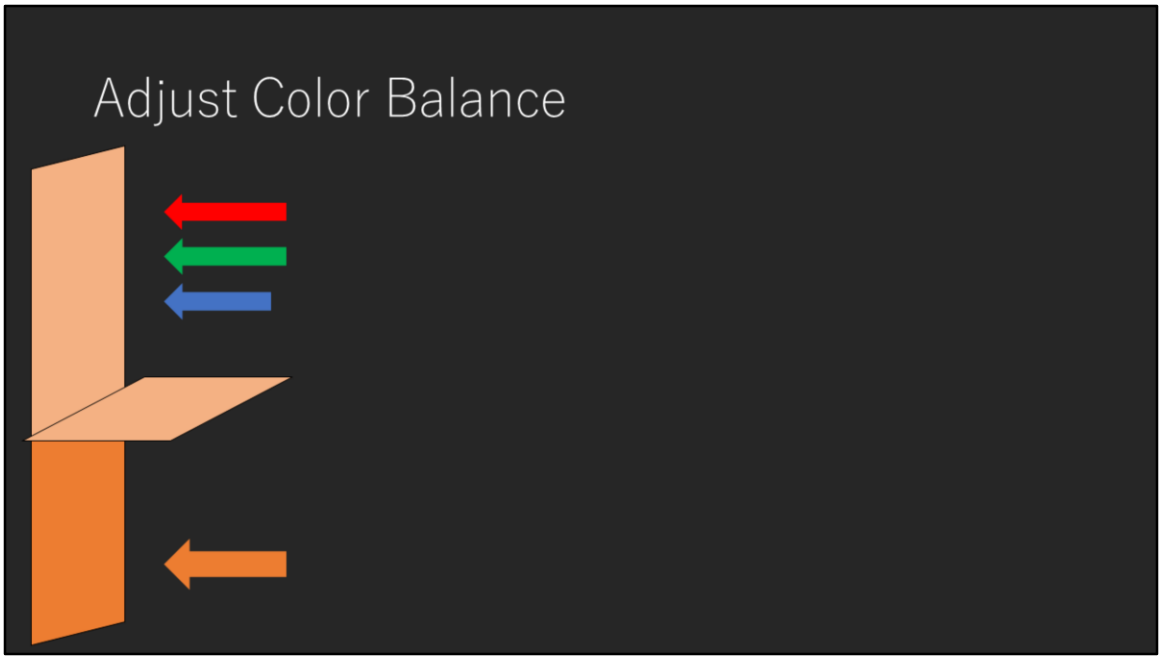
To measure a human eye sensitivity, first we prepare a white diffuse board and a separator.



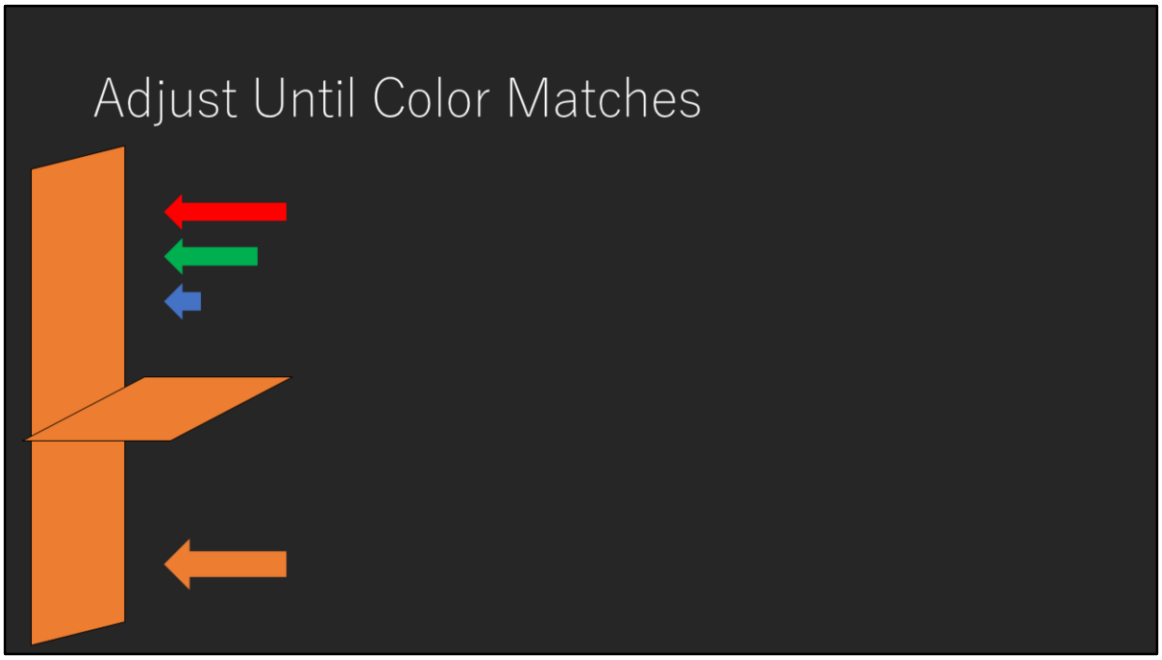
Then light up each part.
Upper part is a mixture of red, green, blue lights.
Bottom part is illuminated by target light which we want to measure.



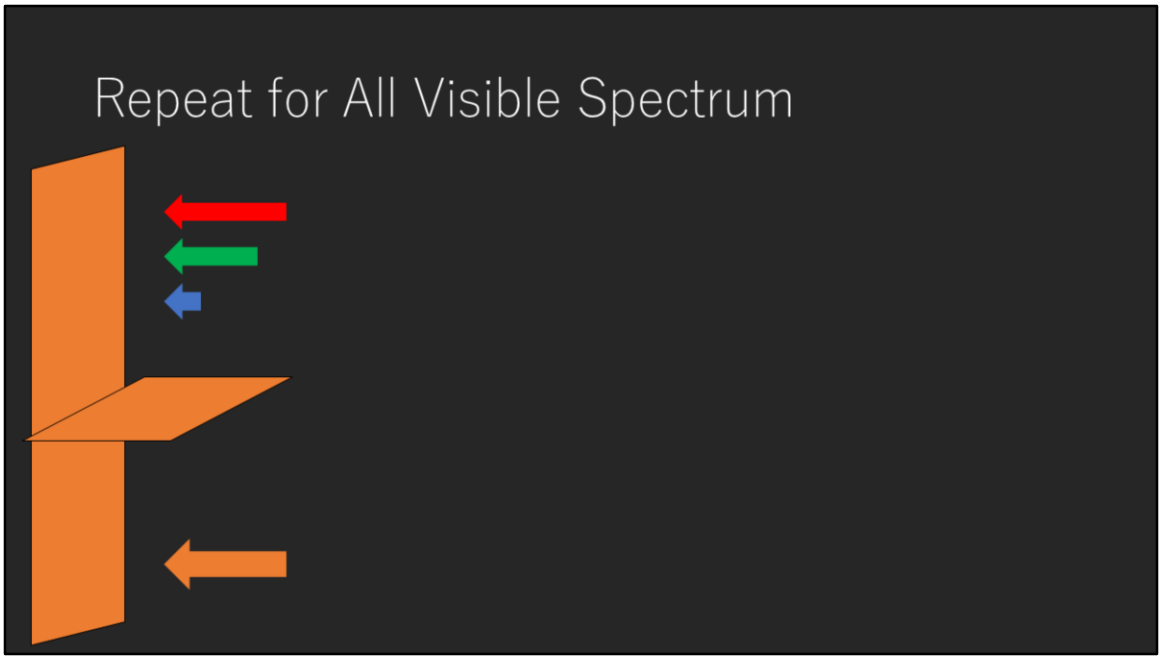
Then, adjust illumination balance to match two colors.



Keep Adjusting…

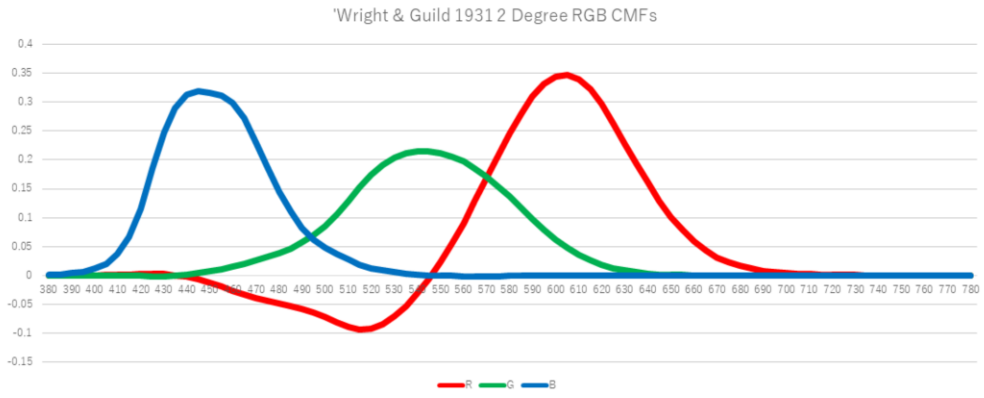


Adjust until two colors matches.



If these colors matched, the balance of the primary light represents human eye color sensitivity of that target spectrum.
CIE, a international commission on illumination repeated this process from 380 to 780 nm.

Color Matching Functions (CMFs)



[Colour 2018]

This is a result.
These graph is called CMFs, Color Matching Functions.

Variation of CMFs

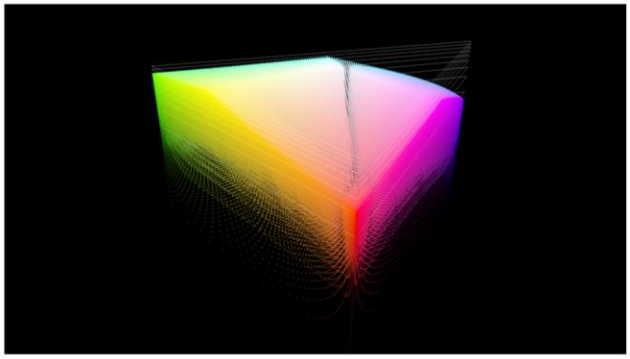
- CIE 1931 XYZ
  - The standard.
- CIE 1931 XYZ Judd-Vos-modified (Judd Vos)
  - Better at blue color representation.
- CIE 2012 XYZ(CIE2006)
  - New CMFs

[Brindley 1955][CIE 1931][Judd 1951] [Vos 1978][Wyszecki and Stiles 1982][CIE 2006]
[Stockman et al. 1999][Stockman and Sharpe 2000][Sharpe et al. 2005][Sharpe et al. 2011]

Many CMFs were proposed in history.
CIE 1931 XYZ is the most popular one.
CIE 1931 XYZ Judd Vos modified is closer to original XYZ functions and it is better at blue color representation.
CIE 2012 XYZ or CIE 2006 is relatively new and used in some TV manufacturer.

Color Representation

- Three degrees of freedom is sufficient to describe color.
- This three dimensional vector is called “tristimulus value”.



As mentioned before, human eye has three cones.
Due to that, three degrees of freedom is sufficient to describe color.
Technically all the colors can be described within three dimensions vector, which is called tristimulus value.

Tristimulus Value Calculation

- Given a spectrum $C(\lambda)$ and CMFs($\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$), you get tristimulus (X,Y,Z) by integral below:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \int C(\lambda)\bar{x}(\lambda)d\lambda \\ \int C(\lambda)\bar{y}(\lambda)d\lambda \\ \int C(\lambda)\bar{z}(\lambda)d\lambda \end{pmatrix}$$

For a given spectrum C lambda and Color Matching Function x, y, z,
We can calculate tristimulus values by just simple integral like this.

xy Coordinates

- We calculate xy coordinate (x, y) from tristimulus value $(\mathrm{X}, \mathrm{Y}, \mathrm{Z})$ as below:

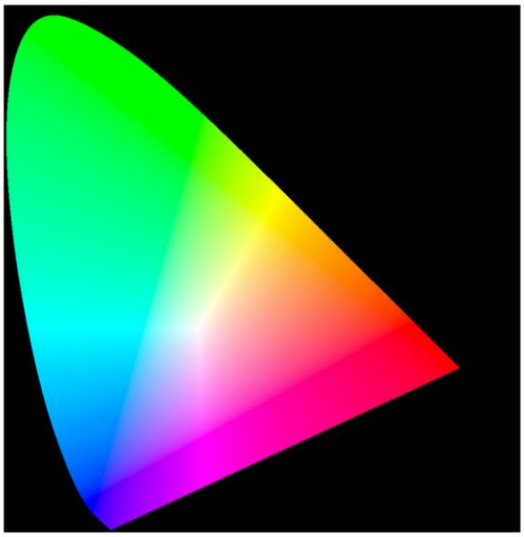
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{X}{X+Y+Z} \\ \frac{Y}{X+Y+Z} \end{pmatrix}$$

- This xy coordinate is often used when talking about any light's absolute color.

In some case three dimensions are too much complicated to draw on a paper or 2D surface.
And thus a smaller, two-dimensional xy coordinate is introduced.
This coordinate is used to discuss about any light's absolute color.

Color Locus

- “Chromaticity diagram”.
- All visible colors.
- “Color locus”
- No one can see colors outside. “Imaginary color”.



Using xy coordinate we can plot any color onto 2d diagram.
This diagram is called chromaticity diagram.

Once we plot all visible colors, we get this horse shoe shape, which is called “Color Locus”.
No one can see colors outside this locus. Those colors are called imaginary color.

Luminance Unit

- $\bar{y}(\lambda)$ of CIE 1931 XYZ CMFs is the exactly same function of Luminosity function of human eye.
- Luminous intensity in candela of any light wave $I_e(\lambda)$ can be calculated as this formula:

$$I_v(\lambda)_{cd} = 683.002 \frac{\mathrm{lm}}{W} \cdot \bar{y}(\lambda) \cdot I_e(\lambda)_{\frac{W}{Sr}}$$

- $1\ cd/m^2 = 1\ nit.$

y function of CMF CIE 1931 XYZ is exactly a same function of Luminosity function of human eye.

Any luminous intensity of any light wave can calculated with this formula. In TV specification, 1 nit is commonly used unit and 1 nit equals to 1 candela per meter square.

Color Matching

- Given tristimulus values $C_1 = (x_1, y_1, z_1)$ and $C_2 = (x_2, y_2, z_2)$

$C_1 = C_2$ means those colors are the same.

- Color matches even if spectrum is different.
- “metamerism”

With two colors c1 and c2 we can write color matching simply using equal operator.

Due to CMF shape different lights spectrum can be matched.
This phenomenon is called metamerism.

Tristimulus Color Basic Rules

- Rule 1:

$$\boldsymbol{C}_1 = \boldsymbol{C}_2 \text{ then } K\boldsymbol{C}_1 = K\boldsymbol{C}_2$$

- Rule 2:

$$\text{When } \boldsymbol{C}_1 = \boldsymbol{C}_2 \; and \; \boldsymbol{C}_3 = \boldsymbol{C}_4$$
$$then \; \boldsymbol{C}_1 + \boldsymbol{C}_3 = \boldsymbol{C}_2 + \boldsymbol{C}_4$$

- These rules are established from the experience. [Grassman 1953]

For tristimulus value pair c1 and c2 these rules are established from experience.
These rules are called Grassman ’s law.

Tristimulus Color Interpolation

- Color can be interpolated in form of tristimulus value.
- Given color C_1 and C_2 , a color between those two can calculated as

$$aC_1 + (1 - a)C_2$$

where a is a mixture ratio.

Color can be interpolated. For two colors c1 and c2 we can calculate blended color just by alpha interpolation like this.

Three Primary Colors

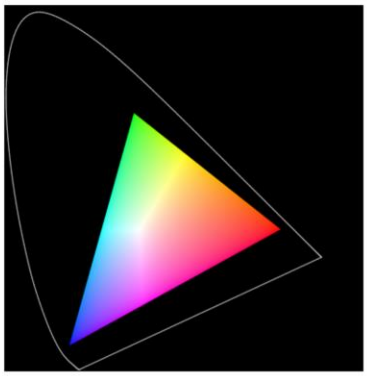
- Select arbitrarily primary colors P_r, P_g, P_b
- These primary colors can describe another color

$$C(r, g, b) = rP_r + gP_g + bP_b$$

When we select three primary color pr, pg and pb arbitrarily, these primary colors can act as a basis.
Three primary colors are enough to describe other color by linear combination.

The Color Inside Primary Colors

$$C(r, g, b) = rP_r + gP_g + bP_b$$



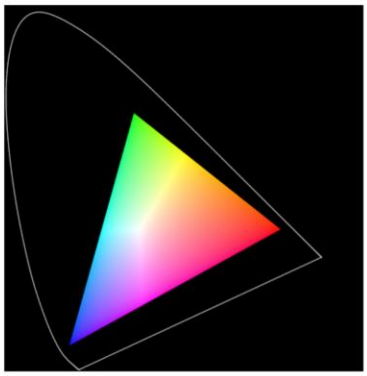
When the linear combination weight is limited to positive, the range becomes a triangle on the chromaticity diagram.

If we allow negative weight, any three primary colors can describe all visible color, but usually we don't do that.

Gamut

- Colors inside three primary colors.

$$C(r,g,b) = rP_r + gP_g + bP_b$$
while $r > 0 \cap g > 0 \cap b > 0$



Therefore, three primary colors define one color region.
This region is called Gamut.

Describing color using gamut is called the “colorimetric system”.

Gamut and Tristimulus

- Tristimulus value is colorimetric system dependent.
- “XYZ tristimulus”: CIE1931XYZ?

Basically, we have to know which gamut is used when any tristimulus value is present.
Otherwise, the color is not determined exactly.
For example, the XYZ tristimulus normally used with CIE 1931 XYZ gamut, but no one guarantees that.

Gamut Conversion Matrix

- With different color primaries (P_r, P_g, P_b) and (Q_r, Q_g, Q_b), the same color results in different tristimulus values.

$$rP_r + gP_g + bP_b = xQ_r + yQ_g + zQ_b$$

- So the conversion between two gamuts can be treated as a simple matrix transformation.

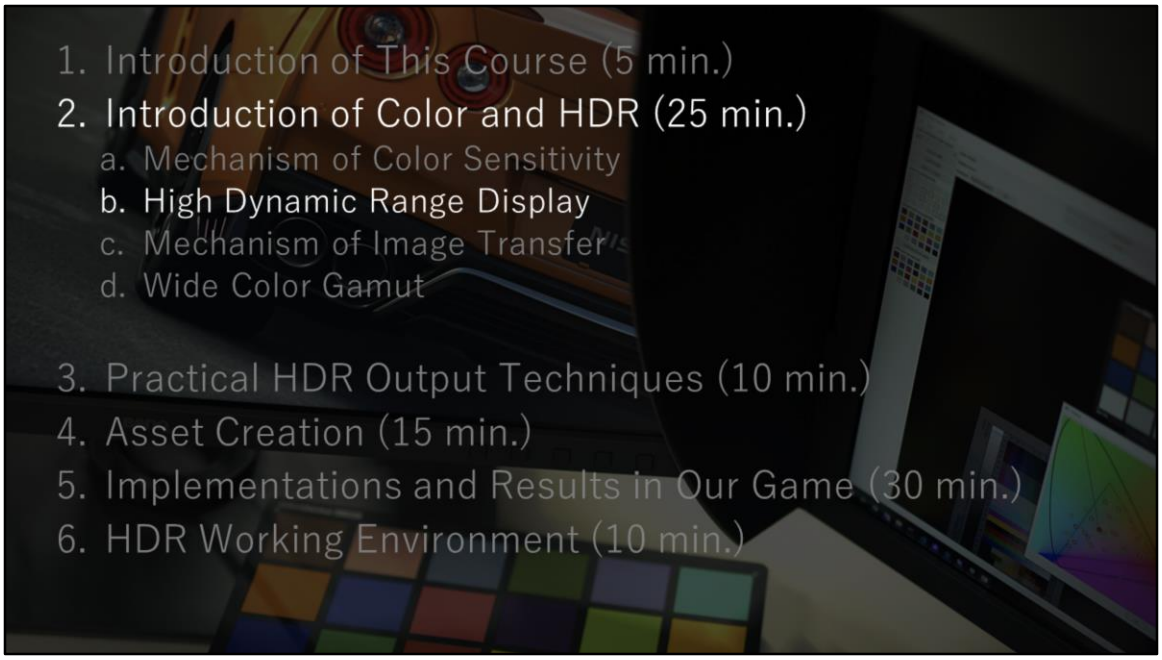
$$(r, g, b) = \begin{bmatrix} M_{11} & \cdots & M_{13} \\ \vdots & \ddots & \vdots \\ M_{31} & \cdots & M_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Different gamut describes same color in different tristimulus.
The conversion between any gamuts can be described in a simple matrix transformation.
This matrix is 9 degree of freedom and a gamut has three tristimulus, so the answer exists.

Section conclusion

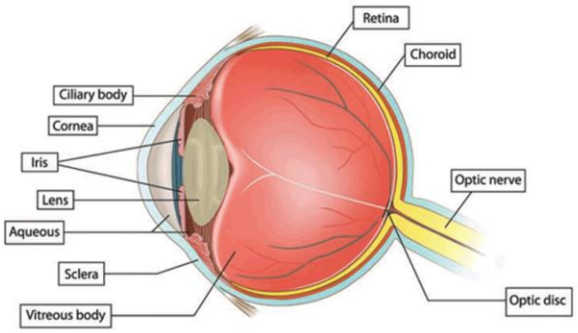
- Light is electromagnetic wave spectrum.
- Human eye perceives spectrum as a color.
- Color can be represented by a three dimensional vector.

To wrap up this section,
We can say, light is electro magnetic wave spectrum.
Human eye perceives spectrum as a color.
Color can be treated as a three dimensional vector.



From here we will talk about High Dynamic Range display.

Human Eye is Good Sensor



The motivation for HDR TV comes from a fact that human eye is really good sensor.

Dynamic Range of Human Eye

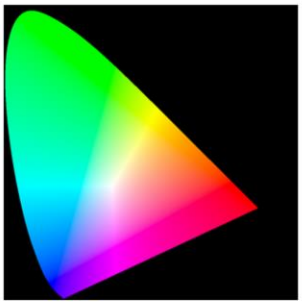
- Human eye can see from $10^{-6}cd/m^2$ to $10^{8}cd/m^2$ with pupil adaptation. [Pirenne et al. 1957][Hallett 1969]
- Virtually human eye has 10^{14} dynamic range of luminance.
- The Dolby Imaging research team found 1:10000 of dynamic range in picture satisfied 90% of subjects. [Dolby 2014]

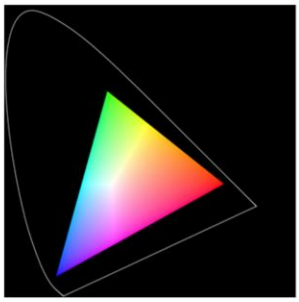
Human eye can see a light from very dark to very bright with pupil adaptation.
It has 10 powered by 14th of dynamic range, which is equivalent to 46 bits of precision.

This dynamic range is too wide to reproduce but recently
Dolby Imaging research team found 90 % of non professional people are satisfied with
1:10,000 of dynamic range in a picture.

Color Range of Human Eye

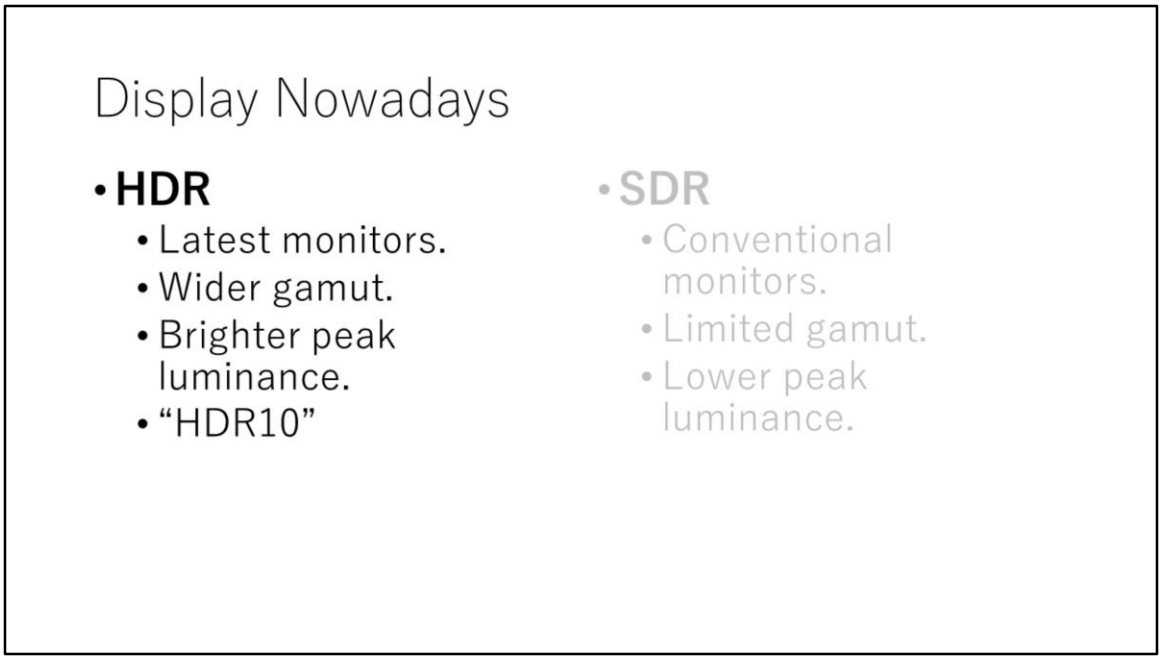
- Human eye can see 1nm difference[Hatada 1985].
- The conventional TV covers less than 80%.





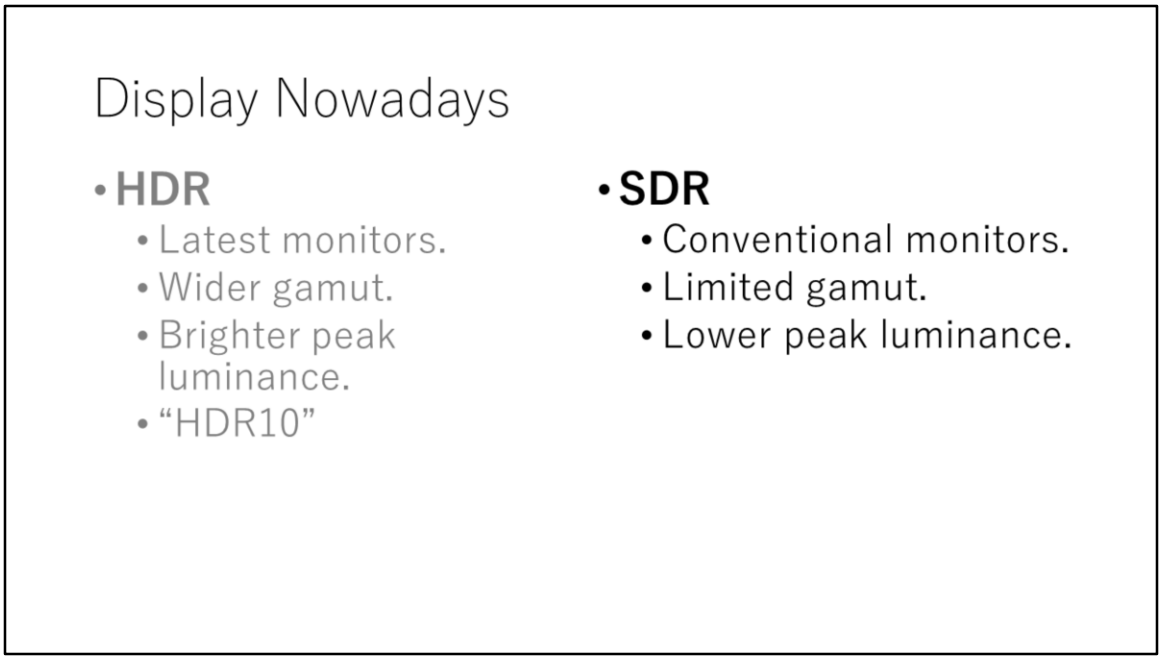
Human eye is also good at seeing colors.
For example, human eye can discover 1nm difference of spectrum.

But the conventional TV system can cover less than 80% of all visible colors.



Thanks to the technology evolution, devices nowadays are getting better.
To use advantages, HDR TV specification is established.
“HDR10” is the most popular HDR standard used in games.

HDR TVs means TVs with High Dynamic Range capability.
They still new and expensive, but it’s getting more common.
They can output more vivid pictures with wider gamut and brighter peak luminance.



On the other hand, conventional monitors are called SDR TV.

SDR means Standard Dynamic Range.

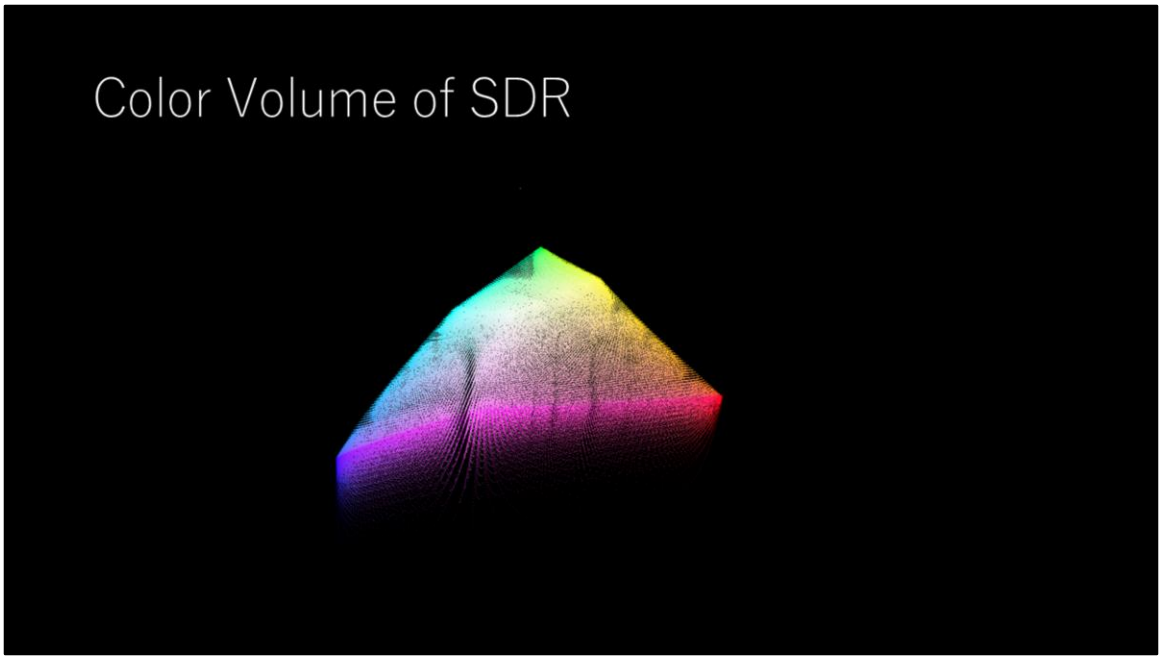
They supports limited gamut and limited peak luminance so the color reproductivity is limited compared to HDR TVs.

LDR: Low Dynamic Range

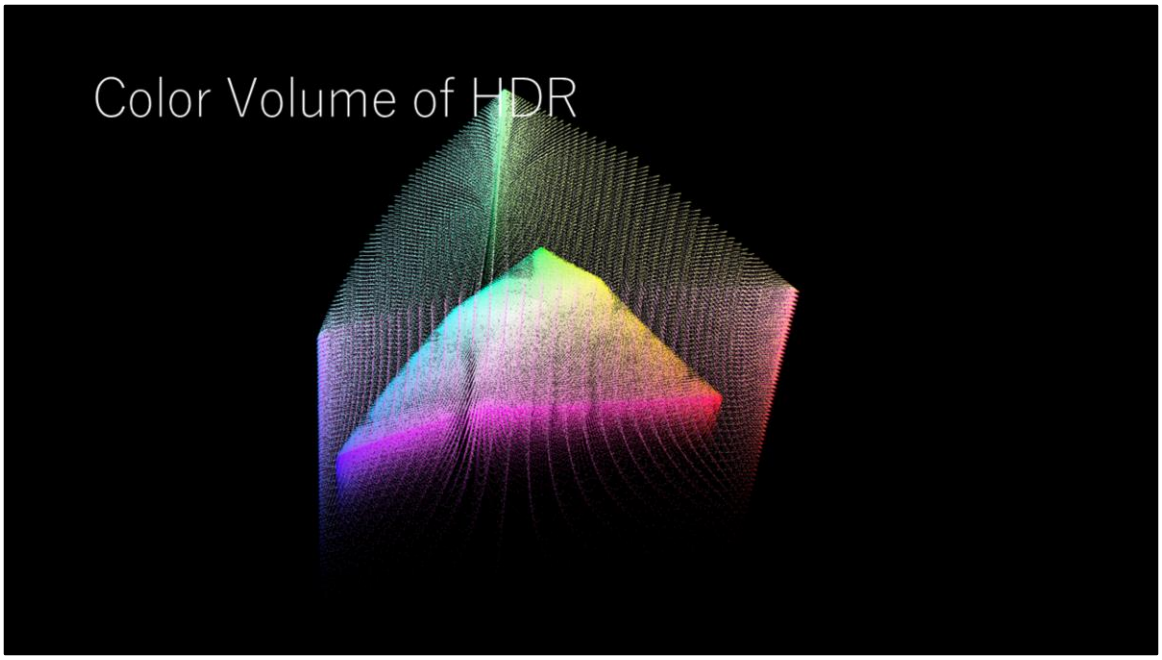
- Popular in Gaming industry.
- Conventional 0-255, 0-1 value.

- HDR means value which can be 1+.
  - float32, fp16, etc…

In gaming industry, the term LDR is commonly used to describe a value which range is limited.
In TV Broadcasting industry, the term SDR is commonly used instead.



This is a visualization of color volume of SDR TV.



This is HDR color volume.
It’s big and covers more colors.
This is the benefit of HDR TV.

HDR TV: Complicated Market

- HDR TV varies from cheap to expensive.
  - Some TV that is just SDR TV with a bit modified gamma.
  - Some TV that is better than professional monitor.
- Device specification calibration is important.

There is a small problem in the HDR TV market these days.
There are too many variations of specs between products and manufacturers.
Some TV monitor are really close to SDR TV and picture is not comparatively better.
Some TV monitor can output more than the professional monitor.
The market of HDR TV is full of dynamic range.
The most problem is that, the pictures are different.
We need to measure or calibrate the actual performance of each display in user environment.

From game industry point of view, we need to provide equal experience for each user regardless of environment.
For example, a scene like this, the track line in the bright white will be very important.

As recent games gets more competitive, proper specification of device calibration becomes more important.

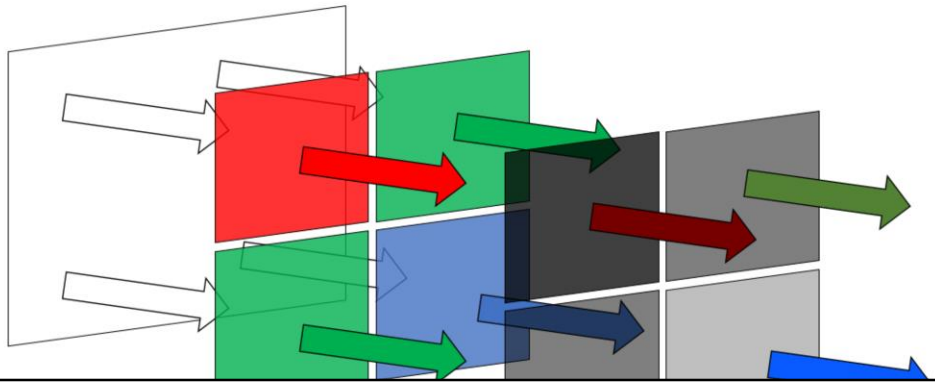
HDR TV Variations

- **Non emissive display**
  - LCD with backlight
  - Local dimming
- **Self emissive display**
  - OLED
  - Micro LED array

There are two variations in HDR TV mechanism.
One is non emissive and one is self emissive.

Non Emissive Display

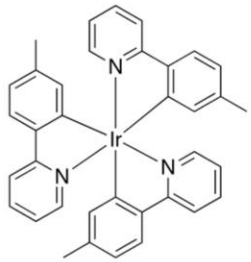
- Basically LCDs. (Liquid Crystal Display)
- Backlight + color filtered active shutter in front.



Non emissive display is the most common display these days.
The emission from backlight goes through color filter.
Then Liquid crystal polarized filter makes picture out of filtered light.

Self Emissive Displays

- OLED(Organic Light Emitting Device)
  - Good picture quality.
  - Perfect dark pixel.
  - Causes burn-in and damaged when picture is not moving.
  - Future games must think about it on OLED.

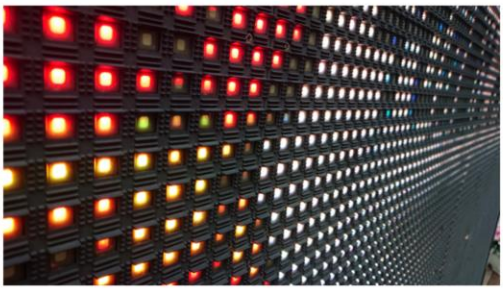


OLED is another popular display.
OLED has self emissive pixels made from organic chemicals.

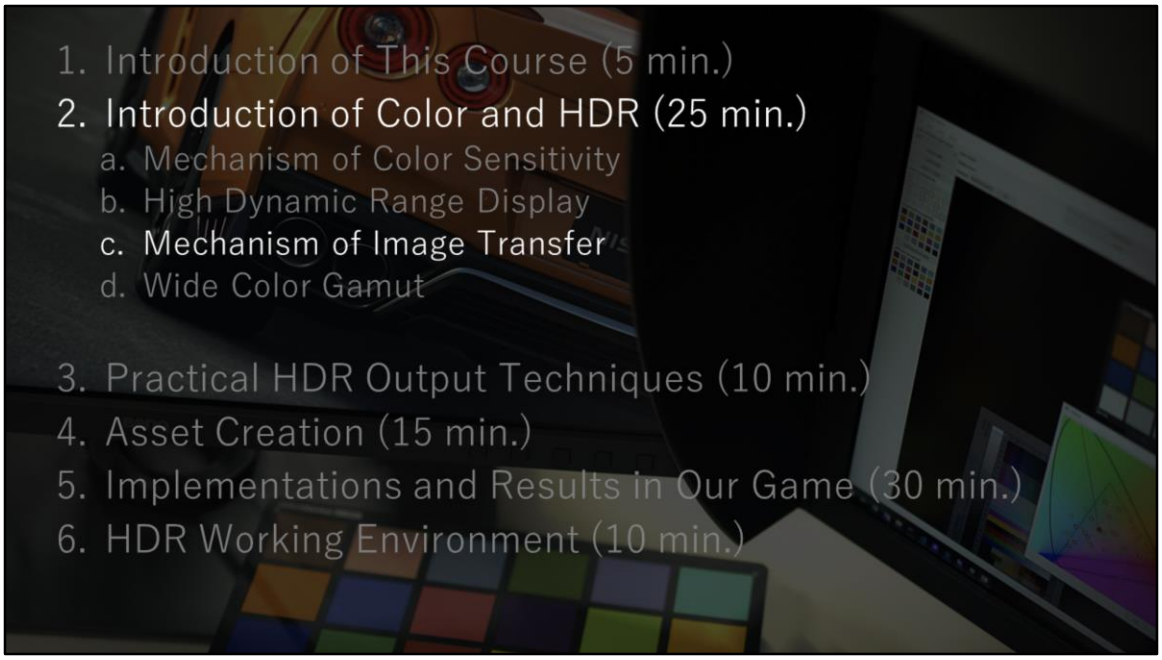
It has good picture quality thanks to the perfect dark pixel.
But OLED has a weak point.
It causes burn in when picture is not moving or pixel is too bright, heat damages the pixels.
Future games must think about it on OLED monitor.

Self Emissive Displays

- Micro LED
  - Great picture quality.
  - No burn-in.
  - Very expensive.



Micro LED array is the future of self emissive displays.
It has great picture quality and no burn in effect but very expensive right now.



Let’s talk about mechanism of image transfer.

Human Eye Nonlinearity

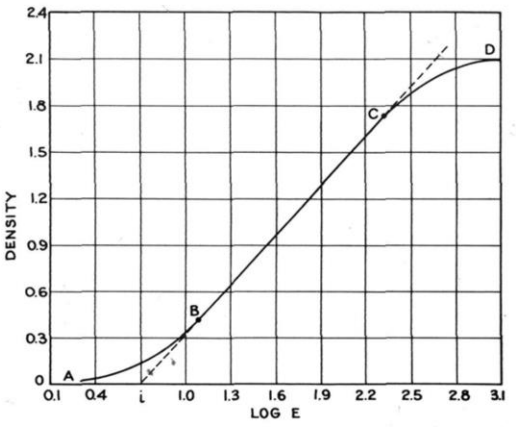
- Human eye has nonlinear sensitivity to images.
- Eyes are very sensitive at dark parts, but are not as sensitive at brighter parts.
- Using nonlinear function can reduce bandwidth.

Human eye has nonlinear sensitivity.

Eyes are more sensitive at the dark picture than the brighter picture.
This nonlinearity is very useful to reduce bandwidth of the data.

The Origin of Gamma Curve

- Photographic film. [Kenneth Mees 1954].
- Film tone reproduction characteristics was nonlinear log curve.
- This fits human eye sensitivity.



CHARACTERISTIC CURVE shows density versus exposure.

As you may already know of it, gamma curve is very popular in image transfer.

The original idea of gamma curve was an tonal reproduction characteristics of photographic film.

This nonlinear curve fits human eye sensitivity.

Gamma Curve in TV Monitor

- Reduces bandwidth of radio wave due to the same reason.
- Display tube nonlinearity.
- Keeps system simple and cheap.

The gamma curve was introduced to TV system to reduce radio bandwidth.
So the gamma curve was a compression technique at that time.
But not only that, TV display tube also had nonlinear characteristics.
Gamma curve solved everything at once.

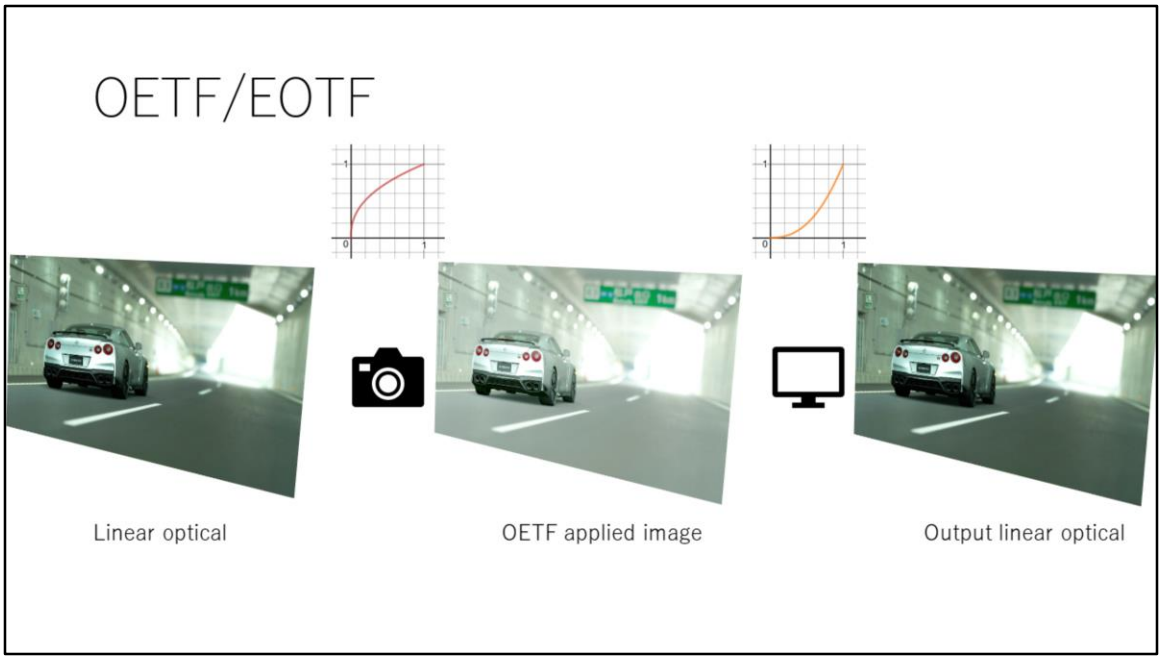
OETF / EOTF

- **OETF**
  - Optical to Electric
  - Used in Camera.
- **EOTF**
  - Electric to Optical
  - Used in TV monitor.

Recently, we call that nonlinear curve, OETF and EOTF.

OETF means Opto-Electric Transfer Function.
Camera uses OETF to convert optical signal to electric signal.

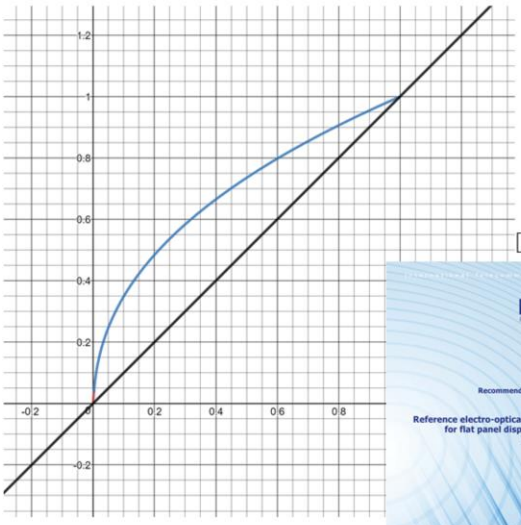
EOTF is opposite side of OETF and it means Electro-Optic Transfer Function.
TV monitor uses EOTF to calculate the power of output light.



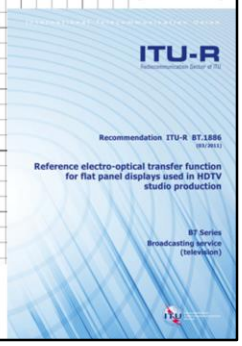
First, camera captures linear optical signal.
Then converts the picture using OETF.
Then the image is transferred.
Finally the TV monitor recovers output picture using EOTF.

sRGB Gamma (OETF)

- De facto standard.
- Almost all of SDR TV and PC monitors use this.
- Check ITU-R BT.1886 for more detail.



[BT1886]



This is sRGB Gamma curve.

A de facto standard OETF curve for non HDR monitors.
Please check ITU-R BT.1886 information for more detail.

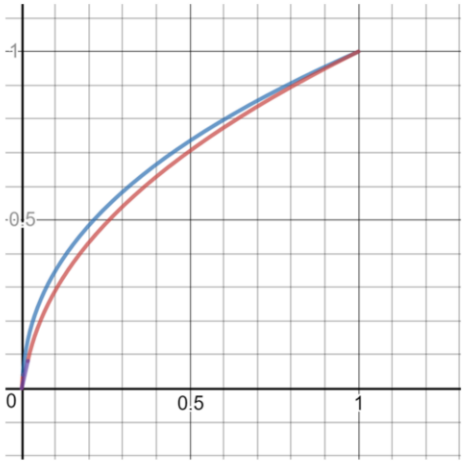
SDR Gamma

- Two variations are present.
- "ITU-R Recommendation BT.709" = Rec.709 gamma = BT.1886 gamma
  - $V = \begin{cases} 1.099L^{0.45} - 0.099 & (0.018 \leq x \leq 1) \\ 4.500L & (0 \leq x \leq 0.018) \end{cases}$
- sRGB gamma
  - $V = \begin{cases} 1.055L^{\frac{1}{2.2}} - 0.055 & (0.0031308 \leq x \leq 1) \\ 12.92L & (0 \leq x \leq 0.0031308) \end{cases}$

There are two variations for SDR Gamma. One is BT.1886, and one is sRGB gamma.
The difference between these two is just some parameter values.

Complicated SDR Gamma

- Those two are slightly differ.
- Actual gamma may vary.
- We will use “sRGB gamma”.



But the real world is more complicated.
For example, some TV monitors do not comply with the regulation and uses simple 2.2 gamma curve.

So we decided to use the “sRGB gamma” as a standard for SDR TV OETF.

sRGB Gamut

- “ITU-R Recommendation BT.709” = Rec.709 = BT.1886 = sRGB
  - Different names, same gamut.
  - sRGB 100 nit is our standard of SDR.

	x	y
White	0.3127	0.3290
Red	0.64	0.33
Green	0.30	0.60
Blue	0.15	0.06

And we will use “sRGB” as a standard name for SDR TV gamut.
Also defined 100 nits as a SDR TV maximum brightness.

OETF/EOTF in HDR TV

- HDR10, PQ, Dolby, ITU-T BT.2100
  - Same Curve, different name.
  - Peak brightness 10,000 nits.
  - Absolute luminance.
- Hybrid Log-Gamma, HLG
  - Has compatibility with SDR TV.
  - Relative luminance.

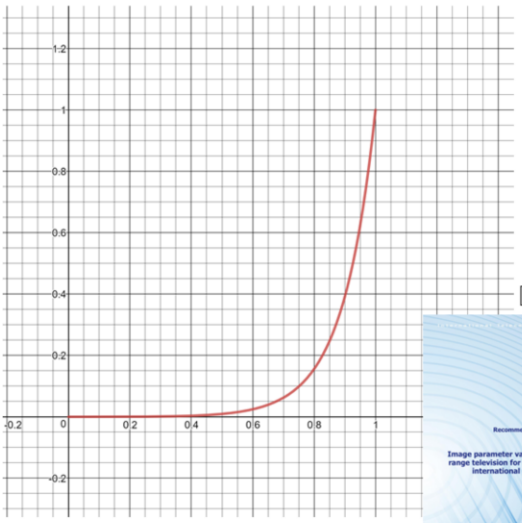
There are two variation of OETF for HDR TV.

One is HDR10. HDR10, PQ curve, Dolby Vision and BT.2100, all of this uses same curve.
In HDR10, pictures can contain absolute luminance of 10,000 nits maximum.

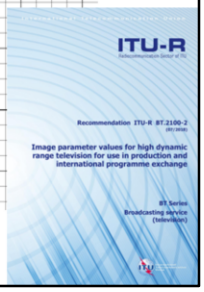
One another is Hybrid Log-Gamma, HLG.
It uses gamma-ish curve to have compatibility with SDR TV.

PQ Curve (EOTF)

- Standard of HDR TV.
- Based on Dolby Vision curve.
- Encodes absolute signal.
- 10,000 nits maximum.
- “ITU-R Recommendation BT.2100 HDR10 EOTF” = ST2084 = PQ Curve



[BT2100]



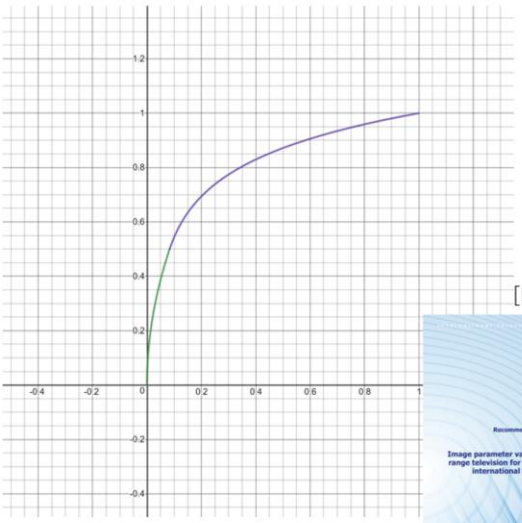
This is PQ Curve.
PQ Curve is a standard EOTF for HDR10 system and it is based on Dolby’s research.

PQ curve has another names.
ITU-R Recommendation BT.2100 HDR10 EOTF, ST2084.

We use “PQ curve” and “Inverse PQ curve” in this slide.

Hybrid Log Gamma (OETF)

- Some compatibility to SDR TVs.
- Encodes relative signal.
- Dynamic range x10 of SDR.



[BT2100]

This is Hybrid Log Gamma curve.

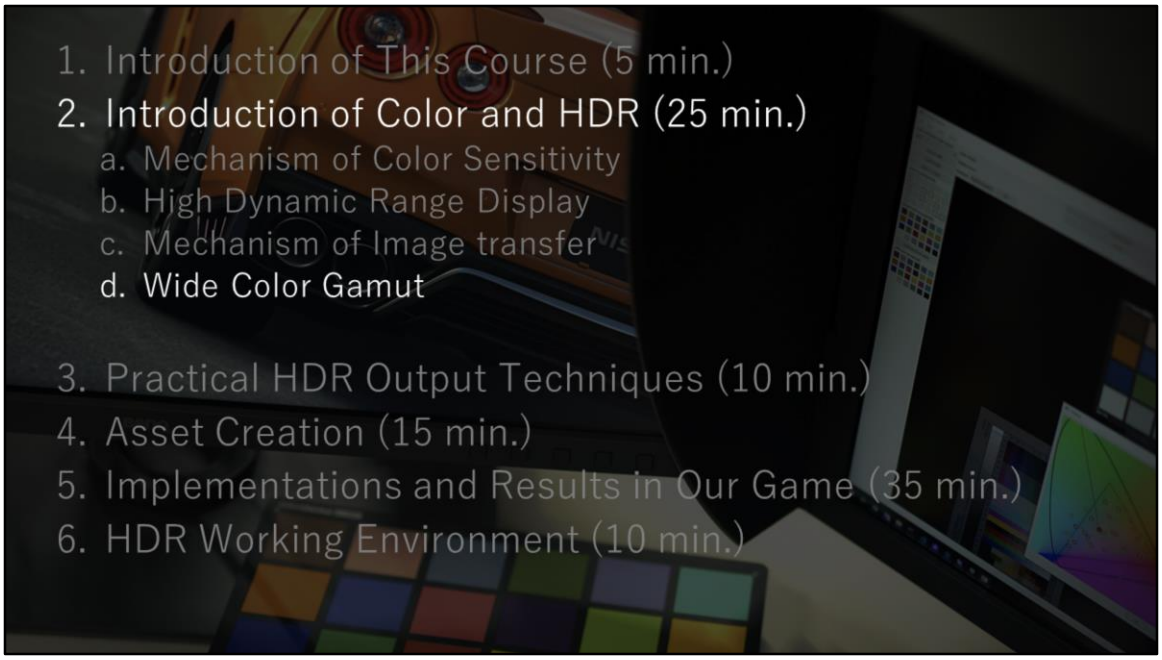
Hybrid log gamma curve is designed to compatible with SDR TVs.
So the dynamic range is not as wide as HDR10.

Inverse PQ Curve

- Games usually use OETF.
- PQ curve defines an EOTF.
- Inverse PQ Curve is more used in game graphics output.
- We will use “Inverse PQ Curve” as a OETF for HDR TV.

From game graphics point of view, we usually use OETF to encode scene linear image.
But PQ curve defines an EOTF.

So, Inverse PQ Curve will be more used in game graphics output.
In this slide, we use a word “Inverse PQ Curve” as a OETF for HDR TV.



Next to OETF and EOTF, we will talk about color.

What is Wide Color Gamut (WCG)?

- Wider color gamut than conventional sRGB.

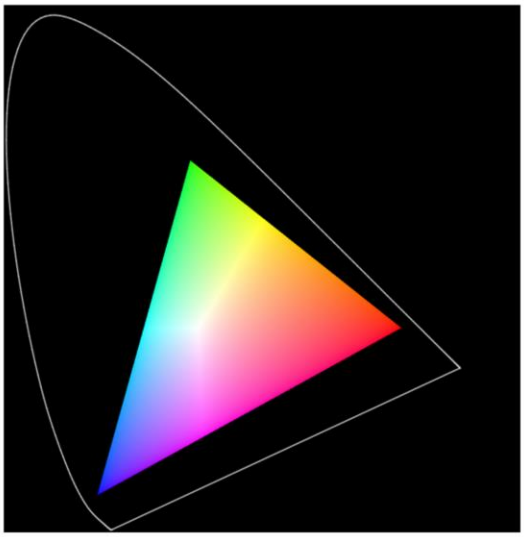
Wide Color Gamut is one of key technology of HDR TVs.

So what is wide color gamut?
There is no exact define of it but
We will use Wide Color Gamut, in meanings of a color gamut that is wider than conventional sRGB.

The main motivation to use wide color gamut can be found in these pictures.
The whole world is full of brilliant colors.
sRGB is not sufficient.

sRGB

	x	y
White	0.3127	0.3290
Red	0.64	0.33
Green	0.30	0.60
Blue	0.15	0.06



As mentioned before again and again, the gamut of sRGB is narrow.
We cannot represent actual color of tropical flowers or racing cars using sRGB.

Wide Color Gamut

- There are so many gamut proposed.
- Here is a list of popular wide color gamut.

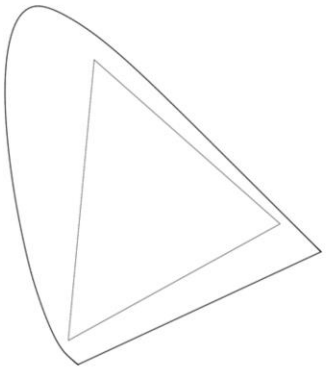
	Adobe	sRGB (reference)	**BT.2020**	**ACES2065-1**	**ACEScg**
xr	0.64	0.64	0.708	0.7347	0.713
yr	0.33	0.33	0.292	0.2653	0.293
xg	0.21	0.30	0.170	0	0.165
yg	0.71	0.60	0.797	1	0.83
xb	0.15	0.15	0.131	0.0001	0.128
yb	0.06	0.06	0.046	-0.077	0.044
White	D65	D65	D65	D60	D60

So wide color gamut is proposed and getting popular.
Of course, each of them has pros and cons.

Adobe RGB

- Most professional monitors and digital cameras support this gamut.
- Only green is wider than sRGB.
- Good for first step.

	x	y
White	0.3127	0.3290
Red	0.64	0.33
Green	0.21	0.71
Blue	0.15	0.06



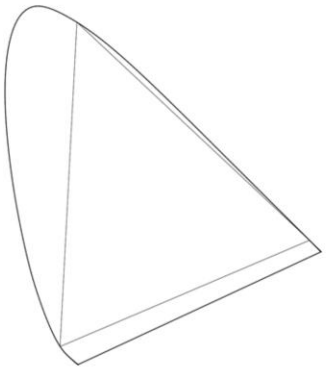
This is Adobe RGB.

The most popular wide color gamut.
This gamut has just green wider than sRGB.
But a lot of devices supports this gamut.
So this gamut will be very good first step.

BT.2020

- HDR TV standard.
- All primary colors are on locus.
- No TV output this gamut yet.

	x	y
White	0.3127	0.3290
Red	0.708	0.292
Green	0.170	0.797
Blue	0.131	0.046



This is BT.2020 gamut.

A standard gamut of HDR TVs.

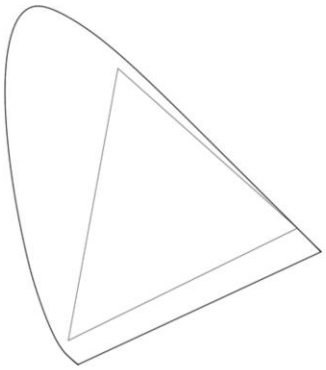
It has different names like ITU-R Recommendation BT.2020, or Rec.2020 but means same.

All primary colors are on color locus.
That means this gamut is very difficult to be implemented on real monitors.
So in 2018, no TV monitor can output this gamut 100% yet.

DCI-P3

- iPhone supports this gamut.
- Some good monitor also support.
- Mostly used in Cinema grading.

	x	y
White	0.3127 / 0.314	0.3290 / 0.351
Red	0.68	0.32
Green	0.265	0.69
Blue	0.15	0.06



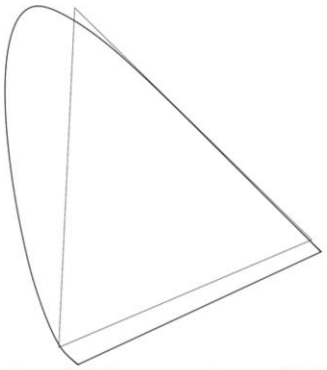
This gamut is DCI-P3.

Widely used in Cinema grading.
Recently iPhone and some apple product starts to supports this gamut.
Some good PC monitors and TV monitors also supports this gamut.

ACEScg

- Academy Color Encoding System
- Slightly larger than BT.2020.
- Designed for rendering and compositing.

	x	y
White	0.32168	0.33767
Red	0.713	0.293
Green	0.165	0.830
Blue	0.128	0.044



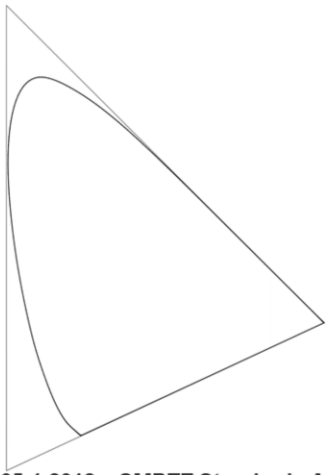
http://duikerresearch.com/2015/09/acescg-a-common-color-encoding-for-visual-effects-applications/

This is an ACEScg gamut.
ACES is Academy Color Encoding System, a group of colorimetry experts.
ACEScg is designed for rendering and compositing so some games in future will use this.

ACES2065-1

- Designed for storing image.
- Standardized by SMPTE as document ST2065-1.
- Covers full XYZ space.

	x	y
White	0.32168	0.33767
Red	0.7347	0.2653
Green	0.0000	1.0000
Blue	0.0001	-0.0770



ST 2065-1:2012 - SMPTE Standard - Academy Color Encoding Specification (ACES)

Lastly this is ACES2065-1 gamut.
Also known as ACES AP0.

This gamut is designed to store image and covers full locus.
Therefore loss in color precision can happen.

Benefits of Wide Color Gamut

- Better representation of most artificial lights
  - Traffic signal, Neon
- Better quality after heavy post process
  - For example : color temperature
  - 2500K-13000K with no saturation!
- Better lighting accuracy
  - Lighting calculation in WCG is better than sRGB.
- Benefits still work even output device is sRGB.

Wide color gamut has a lot of benefits.

WCG can represent most of artificial lights such as traffic signals and neon lights.
WC image keeps quality after heavy post process.
Some wide color gamut has better lighting accuracy than sRGB.

These benefits works fine even if the output device is sRGB.

Better Lighting Accuracy

- ACEScg and BT.2020 is more accurate than sRGB.
  - Fewer clipped colors.
  - Evenly distributed hue.

From here we will talk why wide color gamut is better in lighting accuracy.

In Wide color gamut, saturated colors such as 255,0,0 a vivid red are very rare.
Plus, wide color gamut is designed to have more evenly distributed hues.

That’s why ACEScg and BT.2020 has better accuracy in lighting calculation.

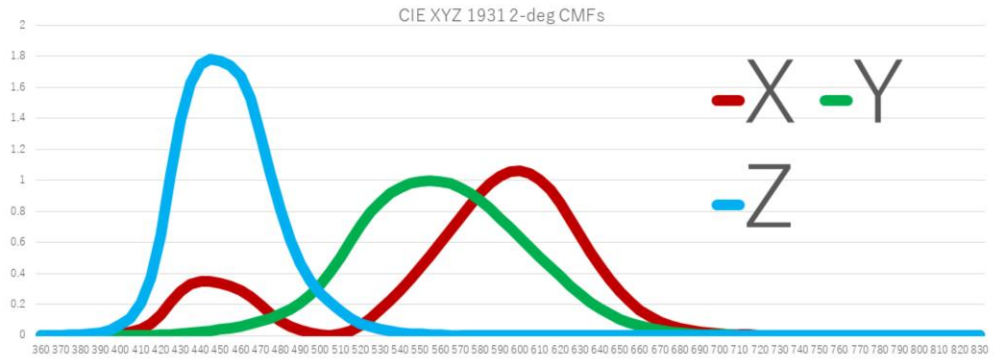
To test lighting accuracy, we will have side-by-side comparison.

Tristimulus lighting is a usual lighting calculation used in computer graphics.
Lighting is calculated by per channel multiplication.

Ground truth is calculated in spectrum domain.

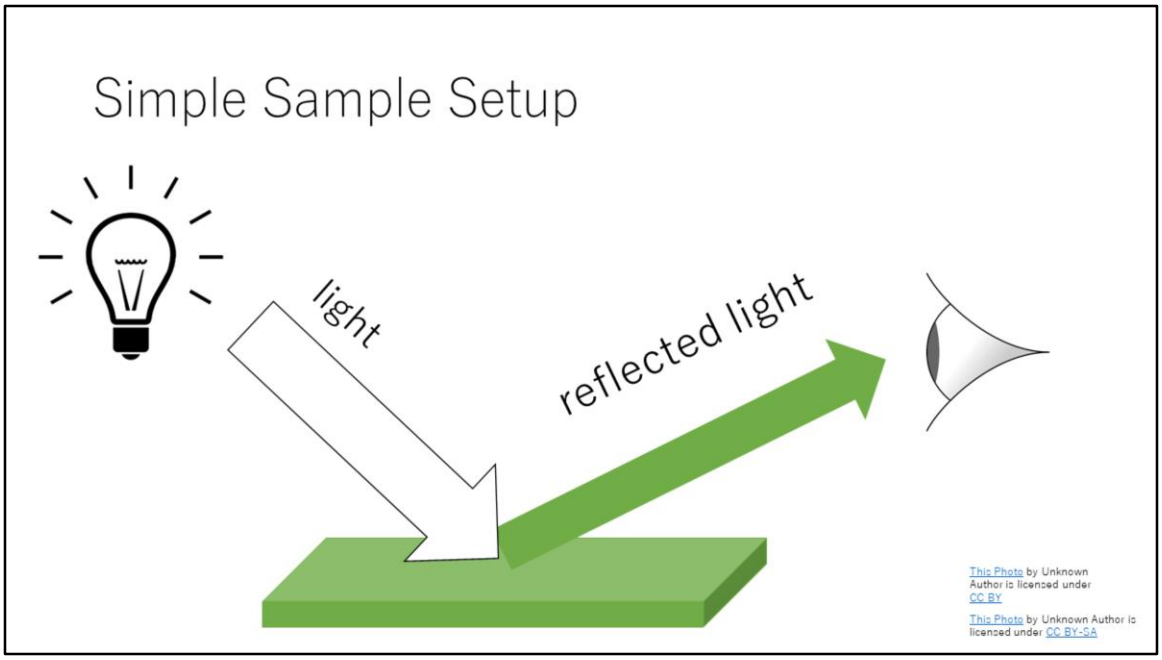
Finally we will calculate color difference using CIE Delta E 2000 formula.

Color from spectrum with CMFs.



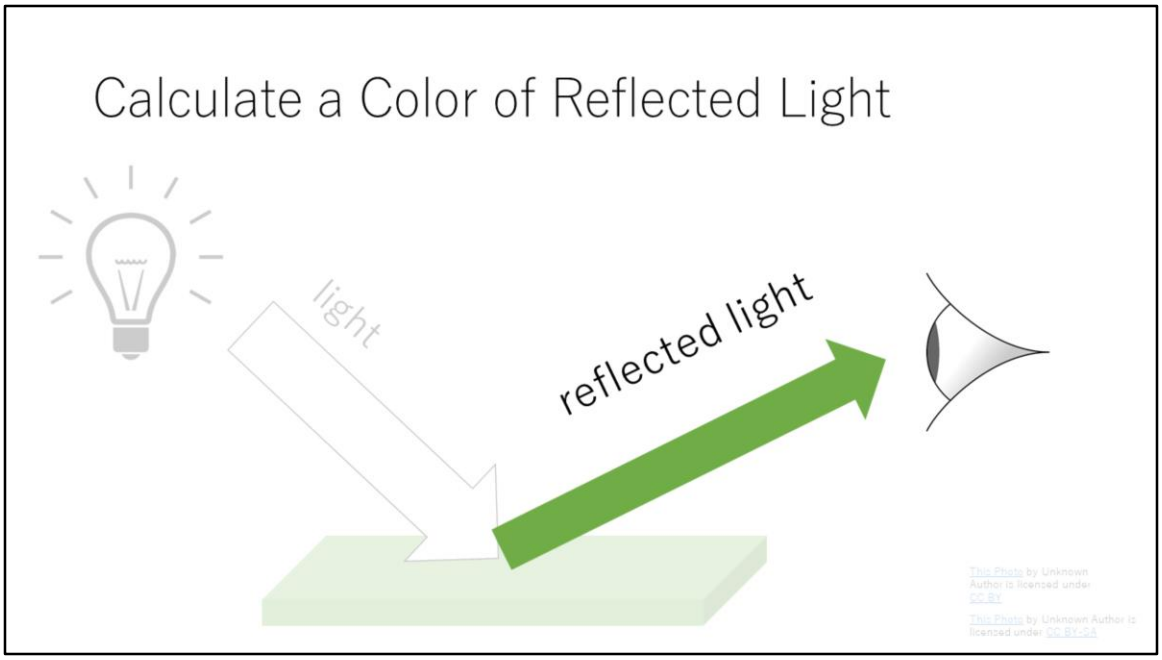
[CIE 1931]

This graph is a CIE 1931 XYZ Color Matching Functions.
X, Y and Z are the red, green, blue of the CIE 1931 XYZ color space.

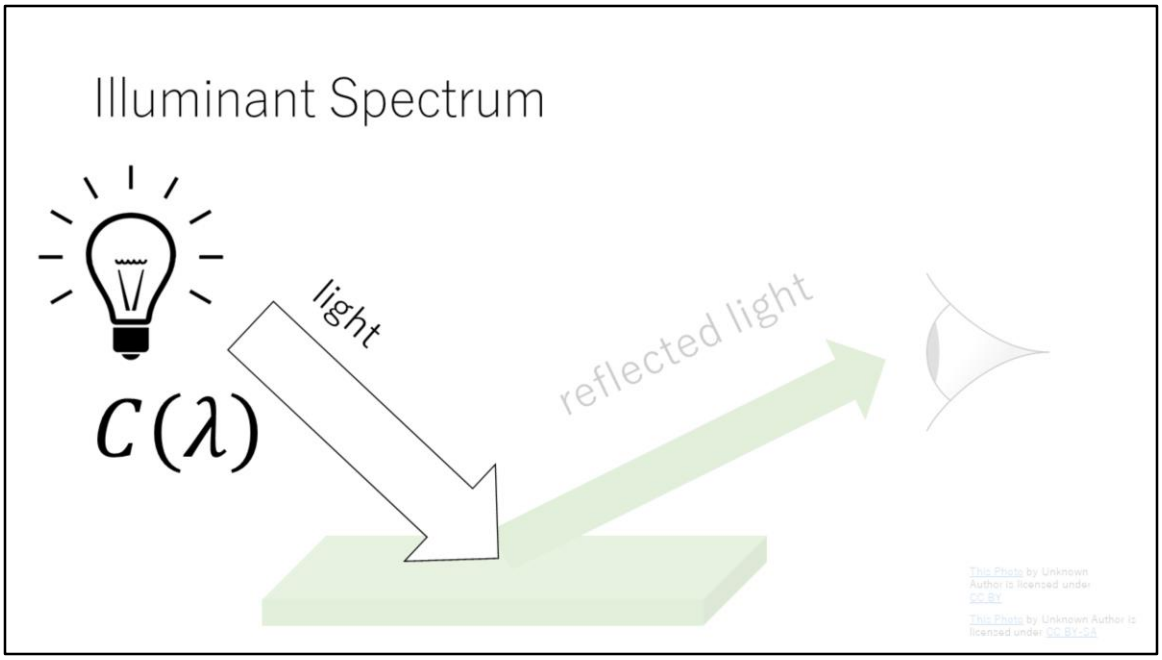


By this point you've become very acquainted with color science, but let me introduce some basic calculation of lighting in spectrum domain.

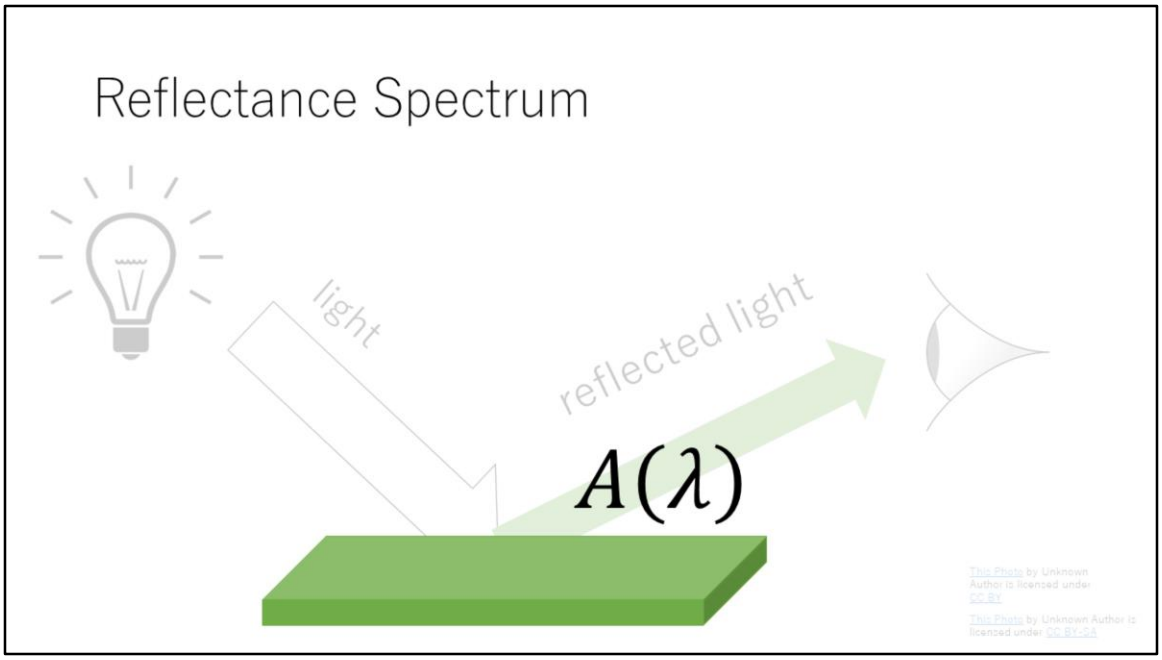
Let's think about a simple scenario.
There is a illuminant.
There is an surface.
And Human eye sees a color of reflected light.



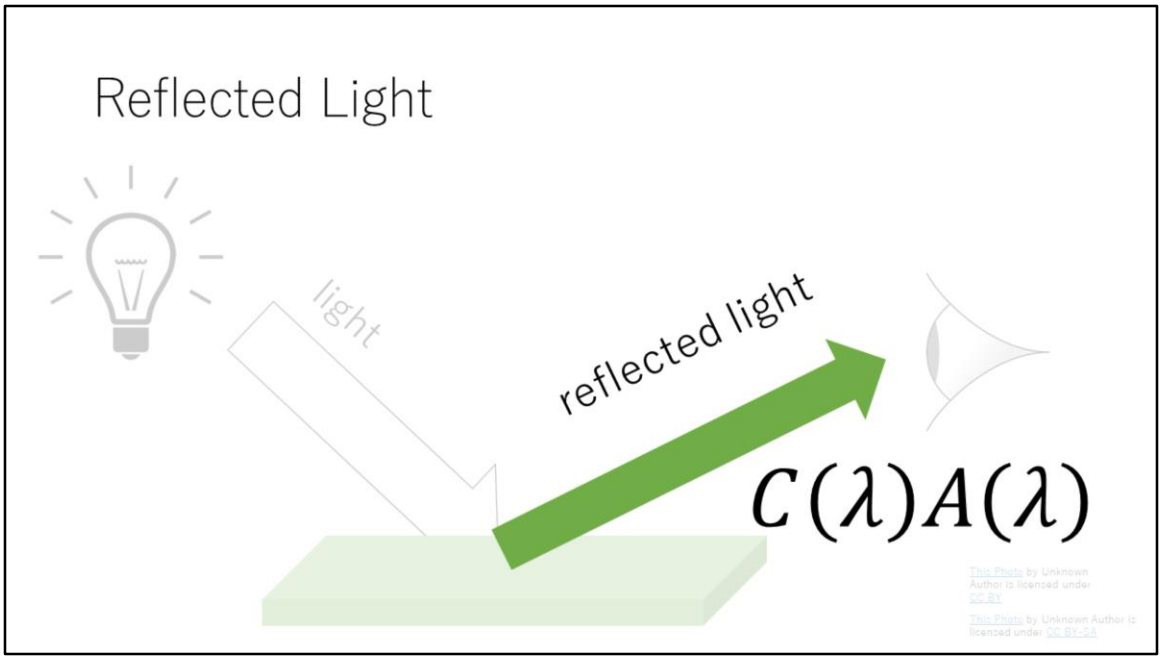
We'd like to calculate a color of this.



First we have to know a illuminant spectrum.
With no light, nothing is visible. Here I will call this C lambda.



Second we have to know reflectance spectrum of surface.
Let this be A lambda.



So, the reflected light is C lambda multiplied by A lambda

The Color We See in Spectrum Domain

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \int C(\lambda)A(\lambda)\bar{X}(\lambda)d\lambda \\ \int C(\lambda)A(\lambda)\bar{Y}(\lambda)d\lambda \\ \int C(\lambda)A(\lambda)\bar{Z}(\lambda)d\lambda \end{pmatrix}$$

We can calculate the color of the reflected light from convolution of the result spectrum and Color Matching Functions.

The Color We See in Tristimulus

$$\begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix} = \begin{pmatrix} \int C(\lambda)\bar{X}(\lambda)d\lambda \\ \int C(\lambda)\bar{Y}(\lambda)d\lambda \\ \int C(\lambda)\bar{Z}(\lambda)d\lambda \end{pmatrix}, \begin{pmatrix} X_A \\ Y_A \\ Z_A \end{pmatrix} = \begin{pmatrix} \int A(\lambda)\bar{X}(\lambda)d\lambda \\ \int A(\lambda)\bar{Y}(\lambda)d\lambda \\ \int A(\lambda)\bar{Z}(\lambda)d\lambda \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_C * X_A \\ Y_C * Y_A \\ Z_C * Z_A \end{pmatrix}$$

For real-time lighting, we usually use tristimulus values instead of full spectrum convolution.
Lighting calculation in Tristimulus values are very simple and fast but it will not match to ground truth result calculated in spectrum.

We repeated this process for popular 24 colors from xrite color checker.

This is a cropped sample from results.
In this case we used sRGB gamut for tristimulus calculation.

The left-bottom triangles are results of tristimulus calculation.
And the top right triangles are ground truth.

So the color reproduction error can be seen as a line between them.

Result of BT.2020

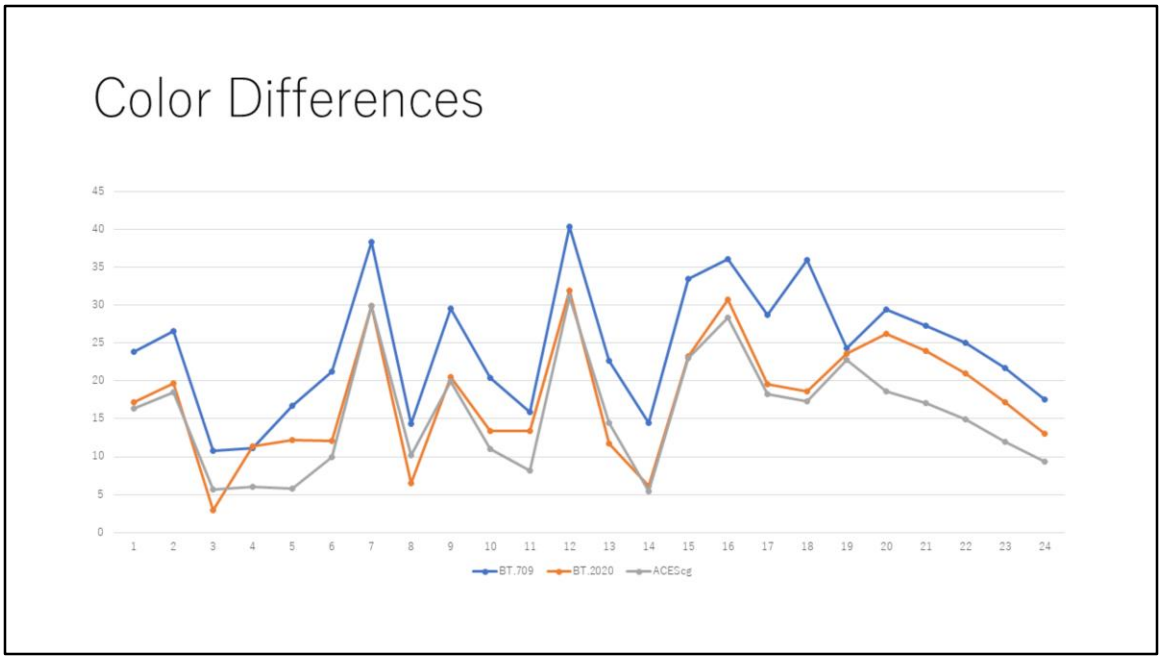
Right top : Full spectrum
Left bottom : BT.2020

This is a full visualization of results calculated in bt.2020.

Result of sRGB

Right top : Full spectrum
Left bottom : sRGB

And this is a result of sRGB.



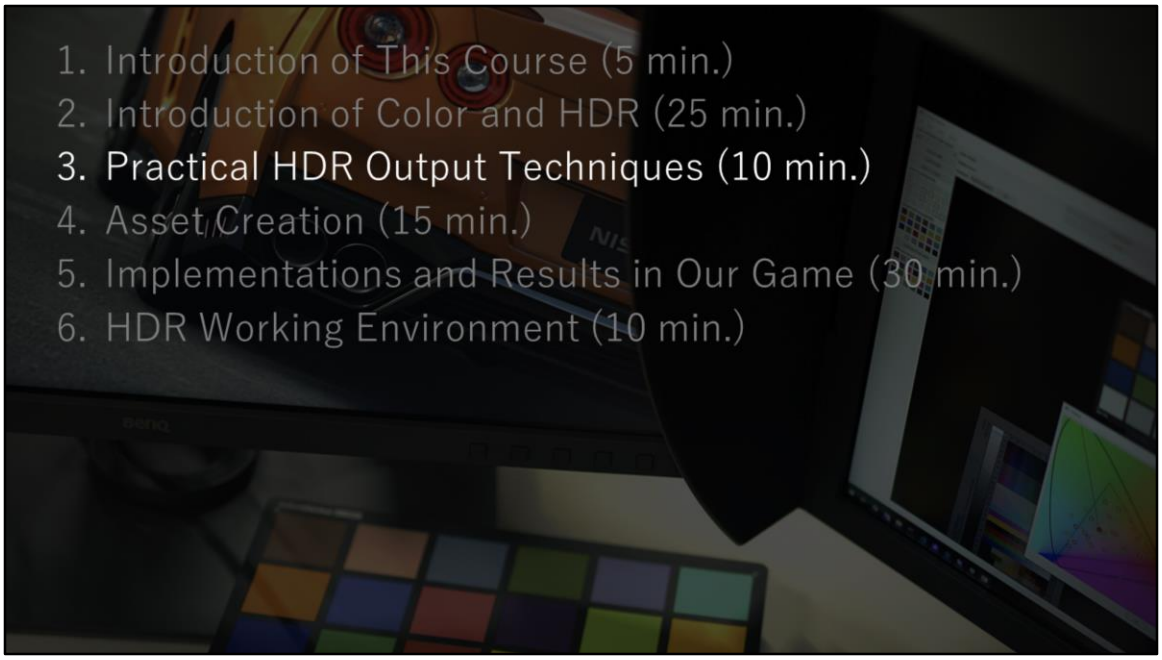
Here is a plot of color reproduction error in case of illuminant CIE D65.

Color differences are calculated using CIE LAB Delta E 2000.
Horizontal axis is patch index and vertical axis is color difference.
The line in Blue is sRGB, orange is BT.2020 and grey is ACEScg.

Average Error

sRGB	BT.2020	ACEScg
24.39502	17.75202	15.58134

This is the average error.
Each gamut has color reproduction error but BT.2020 and ACEScg are significantly better than sRGB.


1. Introduction of This Course (5 min.)
2. Introduction of Color and HDR (25 min.)
3. Practical HDR Output Techniques (10 min.)
4. Asset Creation (15 min.)
5. Implementations and Results in Our Game (30 min.)
6. HDR Working Environment (10 min.)

HDR10

- We selected HDR10 to used in the game.
- HDR10 has metadata.

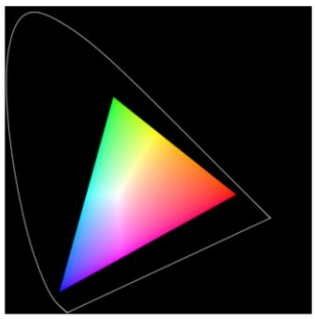
Before talking about practical HDR output techniques we have to talk about HDR10 itself a short.

We make games for PlayStation®4 platform and playstation4 supports HDR10,
So we decided to use HDR10 as a HDR output format.

In addition to that, HDR10 has a metadata system so here we introduce them very short.

Gamut: Side by Side

- **SDR: sRGB**



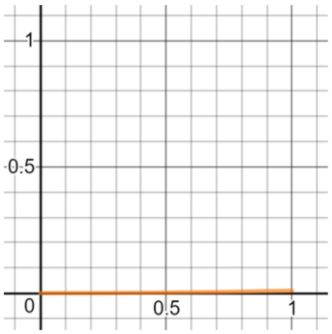
- **HDR: BT.2020**

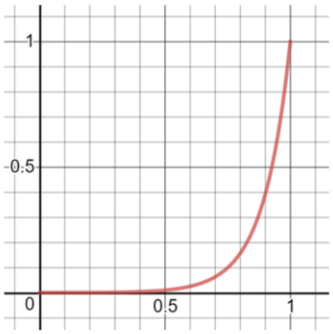
[BT2100][BT1886]

HDR10 uses BT.2020 compared to sRGB.

EOTF: Side by Side

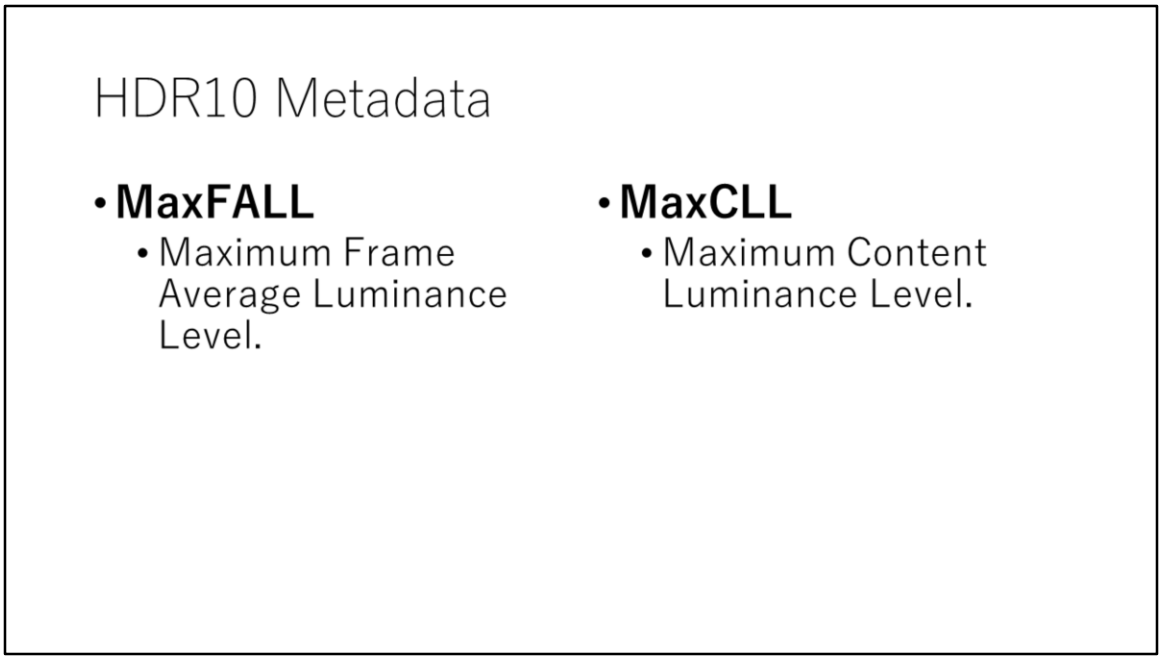
- **SDR: sRGB gamma**
- **HDR: PQ Curve**





[BT2100][BT1886]

HDR10 uses PQ curve for EOTF and it has 100times larger dynamic range.



HDR10 format can have some metadata.

These metadata is used by TV monitor to optimize their settings.

MaxFALL is a maximum frame average luminance, it describes content average luminance.
MaxCLL is a maximum content luminance, it describes content peak luminance.

Some TV just ignores these values, but some TV uses these value.
Some Game console can not send these metadata correctly.
Complicated.

MaxFALL

- MaxFALL of a picture frame can calculated like this:

$$\mathrm{m} = \frac{\sum_N \max\left(EOTF(R_i), EOTF(G_i), EOTF(B_i)\right)}{N}$$

- N: pixel count
- R_i, G_i, B_i: pixel value of each channel
- $EOTF(x)$: PQ Curve

This is a calculation of MaxFALL value.
Just an average of maximum luminance per pixel in each frame.

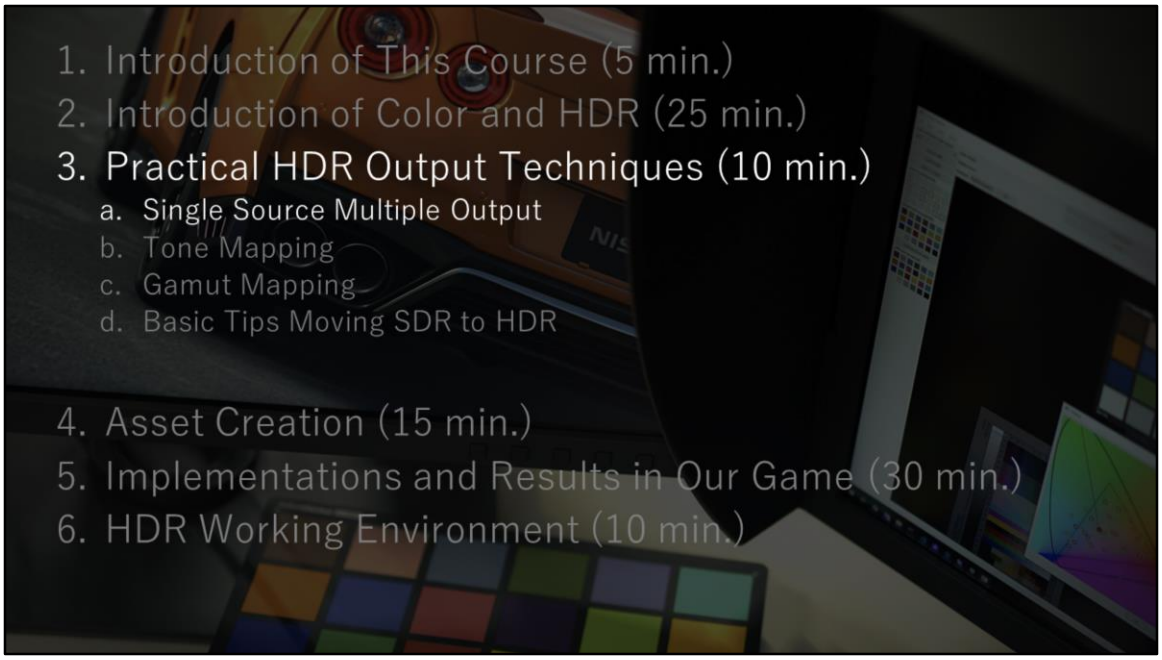
MaxCLL

- MaxCLL of a picture frame can calculated like this:

$$\mathrm{m} = \max\Big(\max\big(EOTF(R_i), EOTF(G_i), EOTF(B_i)\big)\Big)$$

- R_i, G_i, B_i: pixel value of each channel
- $EOTF(x)$: PQ Curve

And this is the formula of MaxCLL metadata.
Very simple.
It's just a maximum luminance for the content's whole frame.



From here we will talk about our HDR output strategy.

Different Gamut, Different OETF

- HDR OETF is not the same as the SDR OETF.
- HDR gamut is not the same as the SDR gamut.
- We have to apply different conversion functions.

As talked before, HDR and SDR uses different OETF and Gamuts.

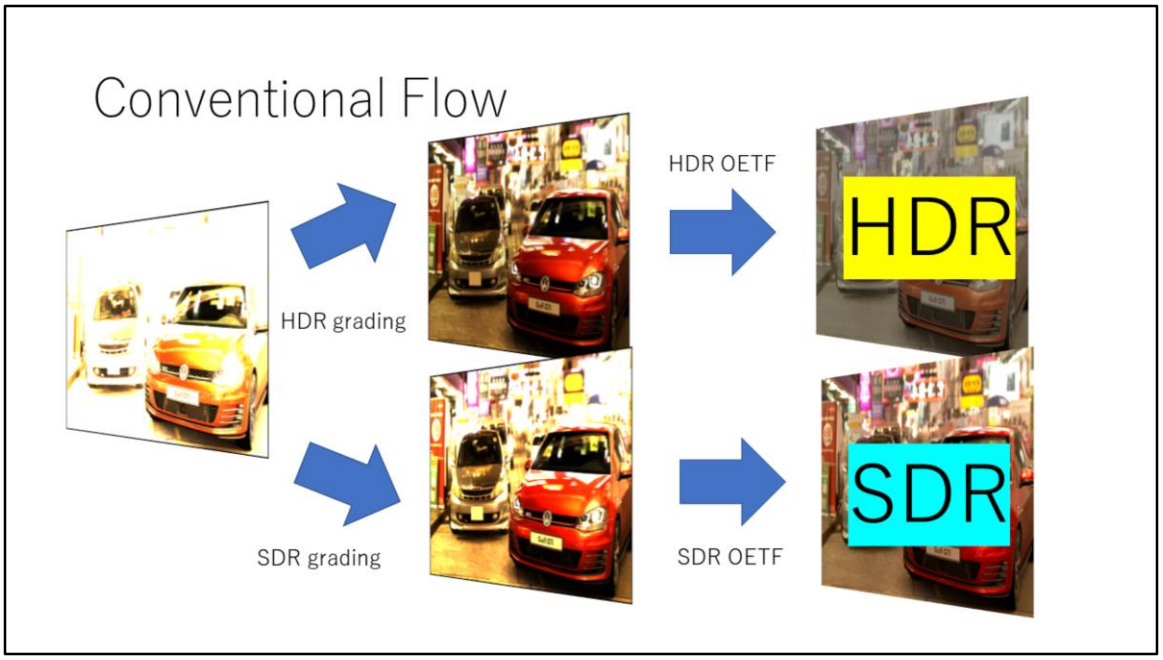
To output images for both TV, we have to apply different conversion functions.
Some of cinema industry uses different grading and post process to optimize image quality for each of them.

Single Source Multiple Output

- Game content is dynamic and unpredictable.
- Can not apply static method.
- Single source multiple target.

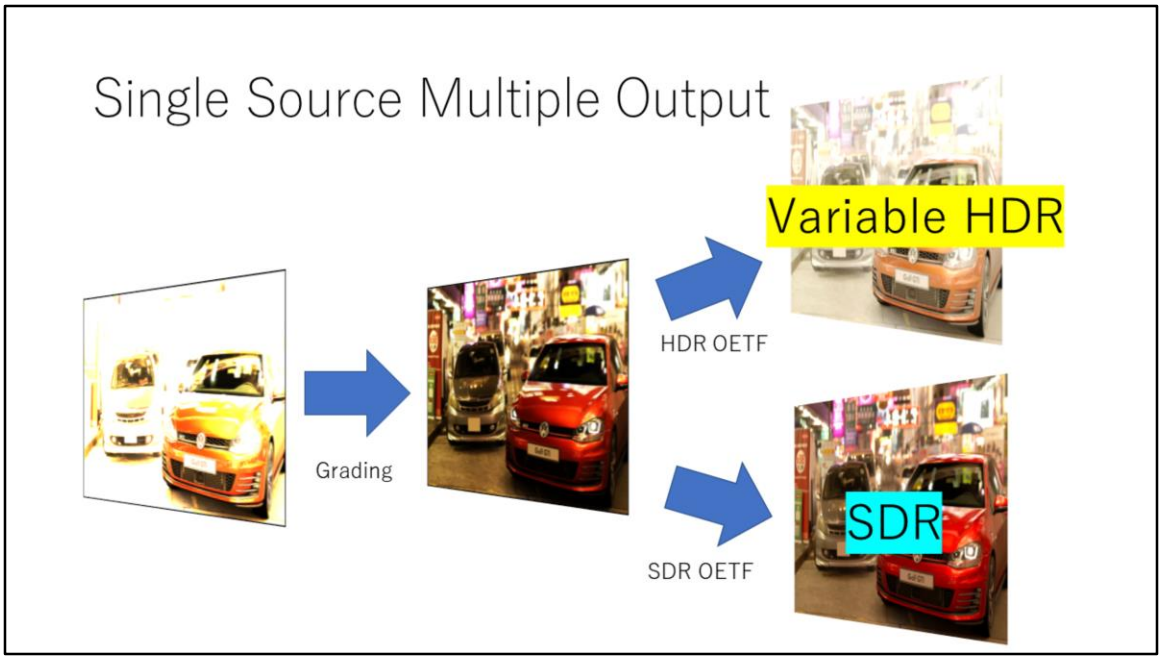
But we used single source multiple output style.

We prepare single beauty source and apply single grading on it.
Then convert it to the output format.
Movies uses multiple grading because they are static and predictable so multiple gradings are possible.
But Games are dynamic and unpredictable so static conversion or multiple grading is not able to applied.



This is a conventional flow of static content.

It applies multiple grading independently designed and optimized for each output devices.
This style needs a lot of time to prepare multiple source and difficult in real time.

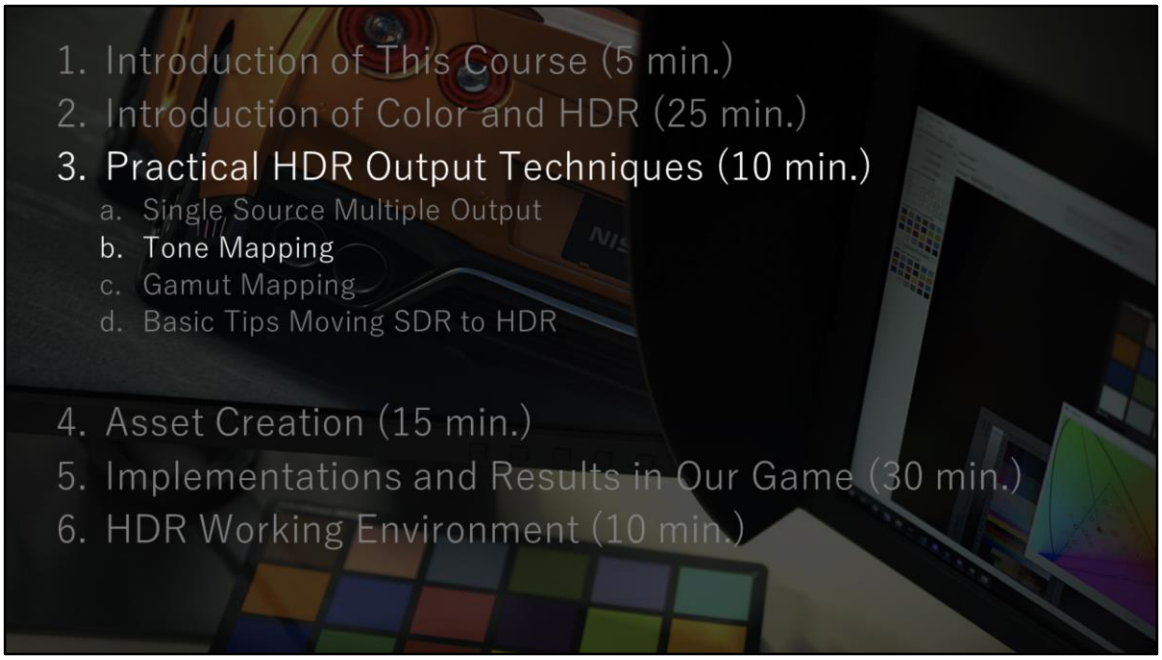


This is single source multiple output style.
First we render single source beauty image.
Using most wide dynamic range and most wide gamut.
And then apply proper output conversion for each device in real time.
HDR and SDR compatibility is achieved.

Two Technique to Make SDR/HDR Compatibility

- **Variable tone mapping**
  - Make image smoother on HDR TVs with different brightness spec.
- **Gamut mapping**
  - Convert wide color image to fit in SDR TV.

To achieve this we created two techniques.


1. Introduction of This Course (5 min.)
2. Introduction of Color and HDR (25 min.)
3. Practical HDR Output Techniques (10 min.)
a. Single Source Multiple Output
b. Tone Mapping
c. Gamut Mapping
d. Basic Tips Moving SDR to HDR
4. Asset Creation (15 min.)
5. Implementations and Results in Our Game (30 min.)
6. HDR Working Environment (10 min.)

Variable Tone Mapping

- All HDR TVs on the market have different brightness specifications.
  - Some TV can output 4,000 nits.
  - Some TV can output only 300 nits.
- So we need an adaptive mapping function for luminance.

Variable tone mapping is created to fill the gap of different peak brightness of different TV models.

Some TV can output 4000 nits, but some TV outputs only 300 nits.
So we need an adaptive mapping function for luminance.

Tone Mapping Variations

- Fixed tone mapping curve
  - ACES RRT+ODT
- Flexible, variable peak brightness tone mapping
  - Kawase's tone mapping[Kawase 2016]
  - Lottes's tone mapping[Lottes 2016]
- Both of them has limitations and not fit for us.
- So we developed our own tone mapping function.

There are three popular tone mapping functions introduced.

Both of them has pros and cons, but none of them fit our application.

So we developed our own tone mapping function.

ACES RRT+ODT

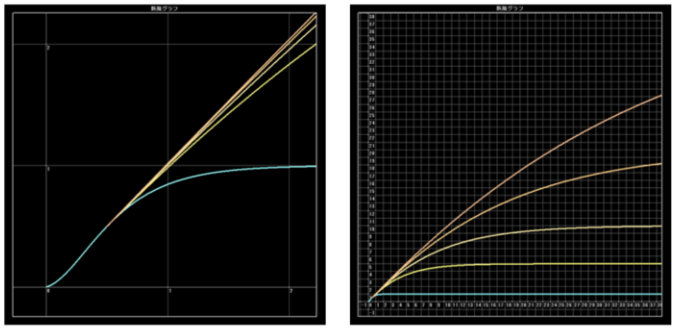
- Based on Fujifilm's film characteristic.
  - RRT : Reference Rendering Transform
    - Transform scene linear color to film look color.
  - ODT : Output Device Transform
    - Transform scene linear color to device signal.
- Pros:
  - Battle proofed. Professional designed.
- Cons:
  - Fixed variations. ODT only support 1000/2000/4000nits display.
  - Filmic look baked in RRT.

One popular tone mapping is presented from ACES.
It is called RRT and ODT.
This tone mapping is based on fuji film's physical film characteristic.

This tone mapping is designed by professionals and battle proofed but some color effects are baked in.
Our game is a car game so embedded color effects doesn't fit for us.

Kawase Variable Tone Mapping

- First presented at CEDEC2016
  - **HDR Output Theory and Practice by Silicon Studio** http://cedec.cesa.or.jp/2016/session/ENG/13532.html(Japanese) [Kawase 2016]
- From p74, he introduced a brief shape of tone mapping function with variable peak brightness.



This is Kawase's variable tone mapping function.
First presented at CEDEC2016, he introduced an idea of variable tone mapping.

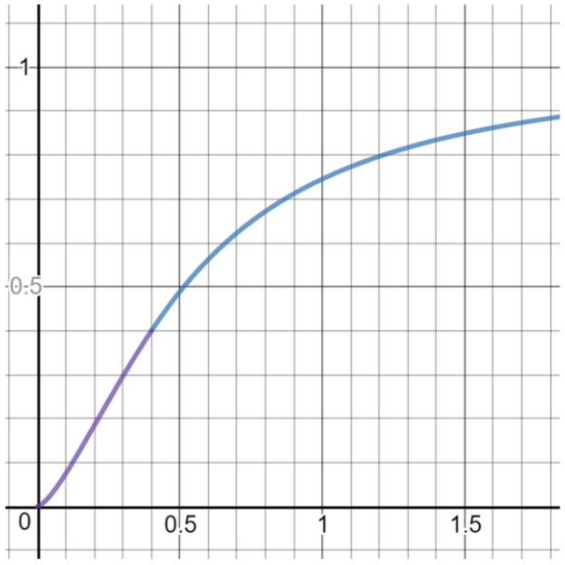
Kawase Tone Mapping

- Pros
  - Variable peak brightness!
  - Good idea to achieve HDR/SDR compatibility.
- Cons
  - No actual formulation.
  - No implementation information.

But there is no actual formulation and implementation information.

Lottes VDR Tone Mapping

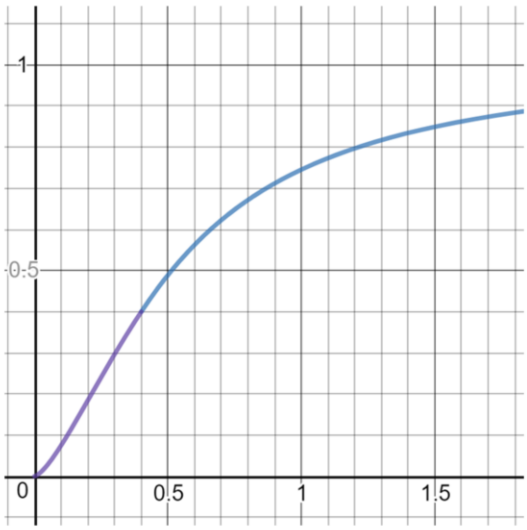
- First presented at GDC2016
  - Advanced Techniques and Optimization of HDR VDR Color Pipelines https://gpuopen.com/gdc16-wrapup-presentations/ [Lottes 2016]
- He presented actual tone mapping function formula



Lottes from AMD also presented variable tone mapping in GDC2016.

Lottes Tone Mapping

- Pros
  - Formulations are open.
  - Variable peak brightness!
- Cons
  - Little bit difficult to adjust.
  - No linear part.



His tone mapping is open and flexible.
But the function is little bit difficult to adjust.
Also this function has no linear part.

GT Tone Mapping

- We designed our own tone mapping function.

- Variable peak brightness!
- Linear mid makes picture photorealistic.
- Everything adjustable and very easy.

So we developed our own tone mapping function, GT Tone Mapping.

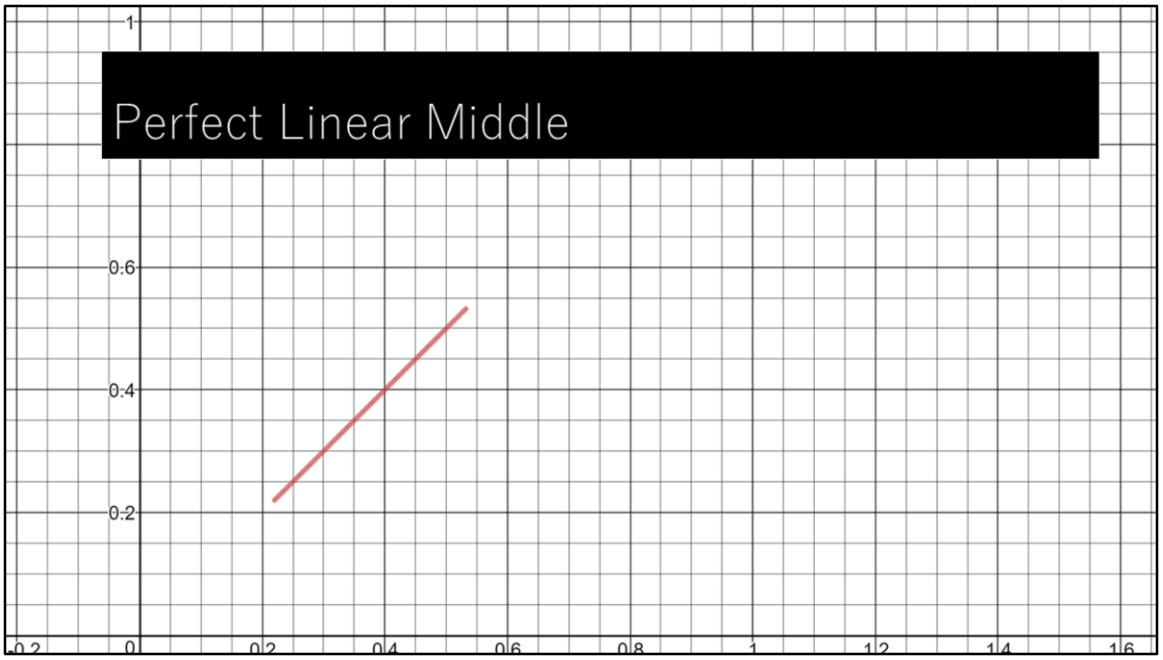
Design philosophy of our tone mapping are learnt from our artist and kept it simple in mind.

GT Tone Mapping Design Philosophy

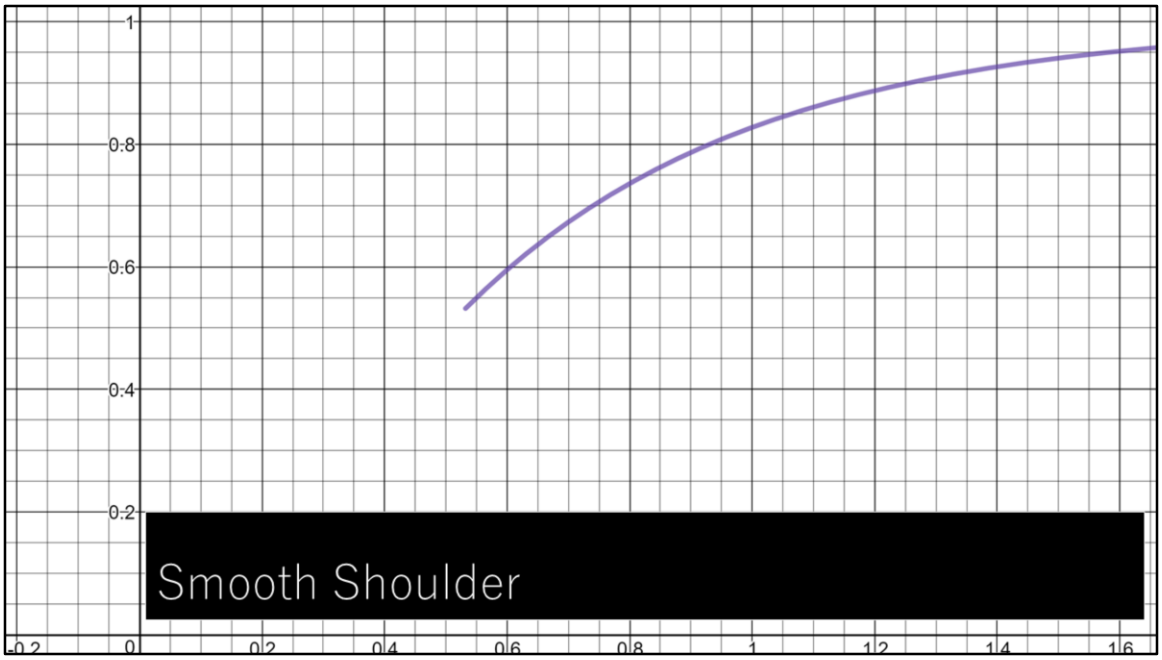
- Important three points we learnt from our artist:
  - Linear section in middle
  - Contrast adjustable toe
  - Smooth shoulder
- GT Tone mapping smoothly connects these three sections.

This is our tone mapping design philosophy.
There are three important points learnt from our artist.

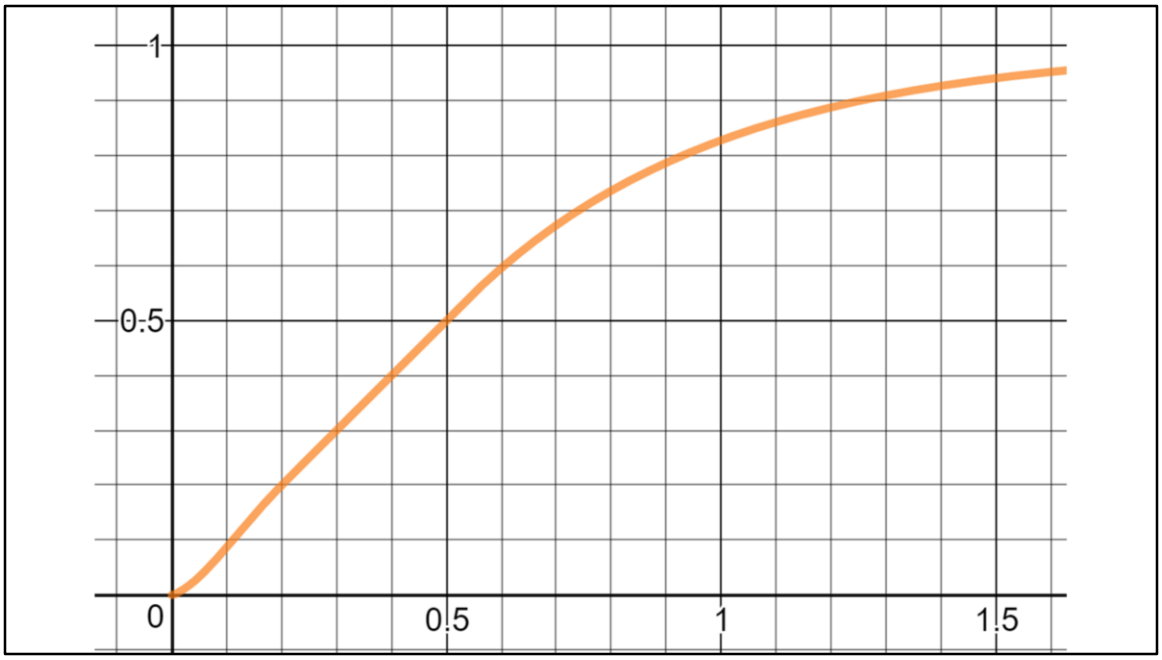
One is a contrast adjustable toe,



One is a middle linear section.
This makes whole picture photorealistic.
This section is the most important part.



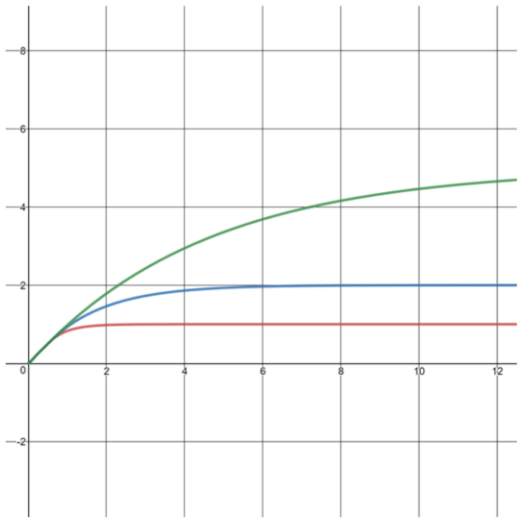
And a smooth shoulder.



GT Tone mapping smoothly connect these three sections.

GT Tone Mapping

- First presented at CEDEC2017 “HDR, Theory and Practice” [Uchimura 2017](Japanese)
  - https://www.desmos.com/calculator/mbkwnuihbd
  - https://www.slideshare.net/nikuque/hdr-theory-and-practicce-jp



We presented this function first at CEDEC 2017.
Formulation is on desmos.

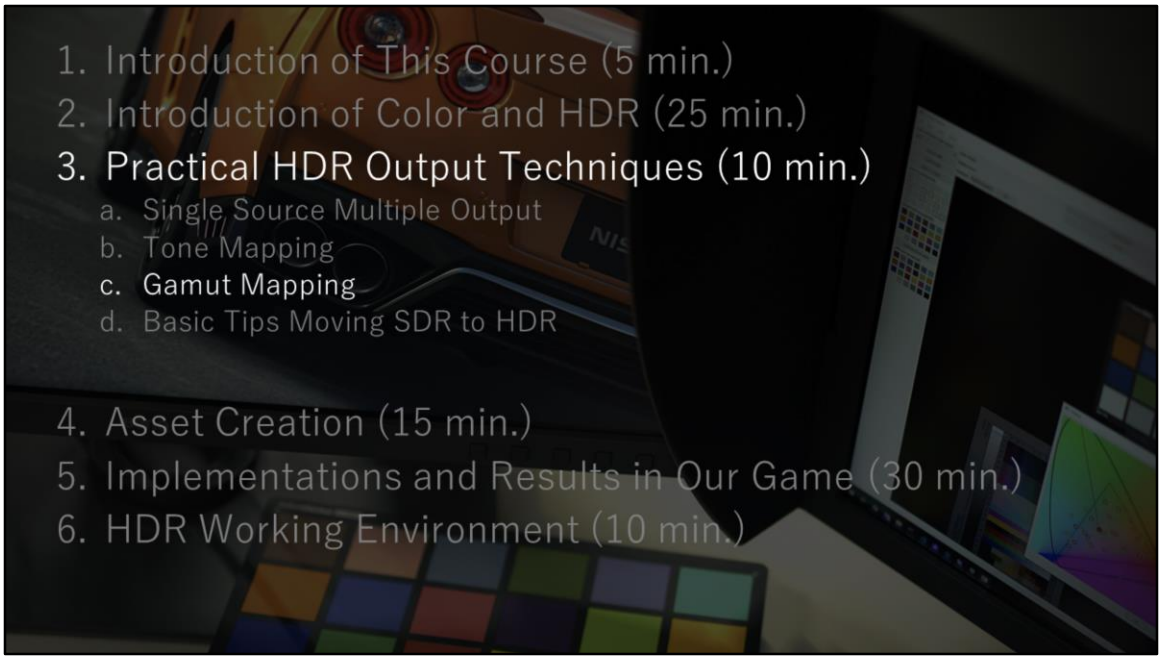
Inverse Tone Mapping

- We used smoothstep function to connect those parts.
- Unfortunately, it makes inversion harder.

- In fact, the necessity of inversion function was almost missed at that time.

By the way, we have made one mistake during GT Tone Mapping development.
It is inverse tone mapping function.

We used smoothstep function to connect sections.
But it made inverse tone mapping a little bit difficult.



Next we'd like to talk about gamut mapping.

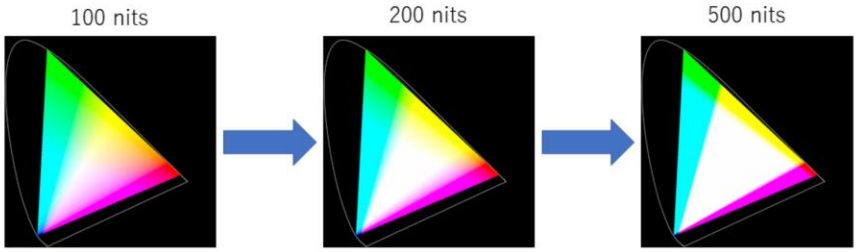
Gamut Mapping

- Simple gamut conversion makes noticeable artifact.
- Special care is required.

When converting vivid colors into narrower gamut,
simple gamut conversion is not enough to achieve visual quality.
In this pictures the yellow part actually is out of gamut,
so the simple conversion results in just saturated red.

Why This Happens

- Simple conversion is okay at 100 nits.
- Brighter color will be clipped and saturated.



Here is a illustration why it happens.
The color outside sRGB gamut saturates and clipped.

Gamut Mapping in Games

- Frostbite Method [Fry 2017]
- GTSport Method(Ours) [Uchimura 2017]

At the development time, there was only one method is proposed.
Alex Fry from EA proposed his method at GDC 2017.

But that method doesn't matched to our game,
so we decided to use our own method.

Frostbite Method

- HDR COLOR GRADING AND DISPLAY IN FROSTBITE[Fry 2017]
  - https://www.ea.com/frostbite/news/high-dynamic-range-color-grading-and-display-in-frostbite
- They used ICtCp [Dolby 2016] color space to split chromacity and lightness
- Lightness is tone mapped separately.
- Chroma desaturates with shoulder.

This is a frostbite method, presented at GDC 2017.

They used ICtCp color space to separate chromaticity and lightness of HDR framebuffer.
Then, they apply different curve to each channel.

For lightness, a simple tone mapping function is applied.
For chromaticity, a shoulder function is applied to have desaturation effect.

We also tested same idea, but the result doesn't fir for us.

GTSport Method

- We apply GT Tone Mapping function to each RGB channel, achieving gamut mapping effect and tone mapping simulataneously.

We used other method.

We apply GT Tone mapping function to each RGB channel independently. This gives us gamut mapping and tone mapping, two effects at once.

The figure below shows the gamut mapping effect of our method.
From left to right, the colors gets brighter, but gamut mapping gives us natural look.

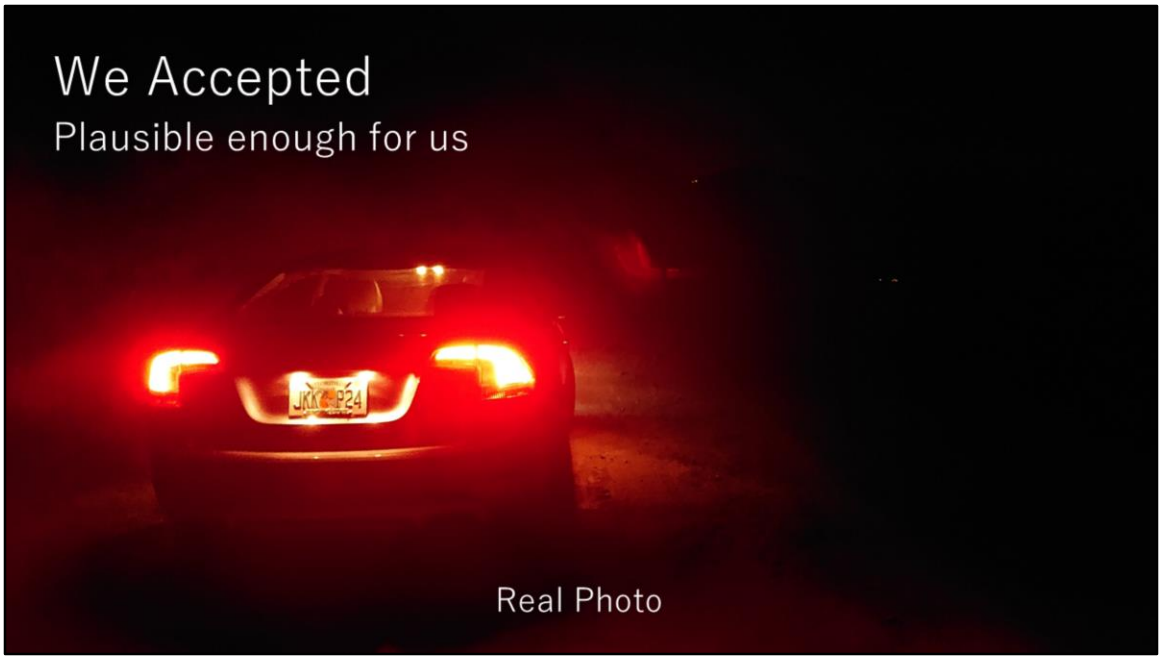
Hue Shifting

- When applying tone map function for RGB channel, hue shifts at brighter area.

By the way, our method has some minor issue. That is Hue shifting.

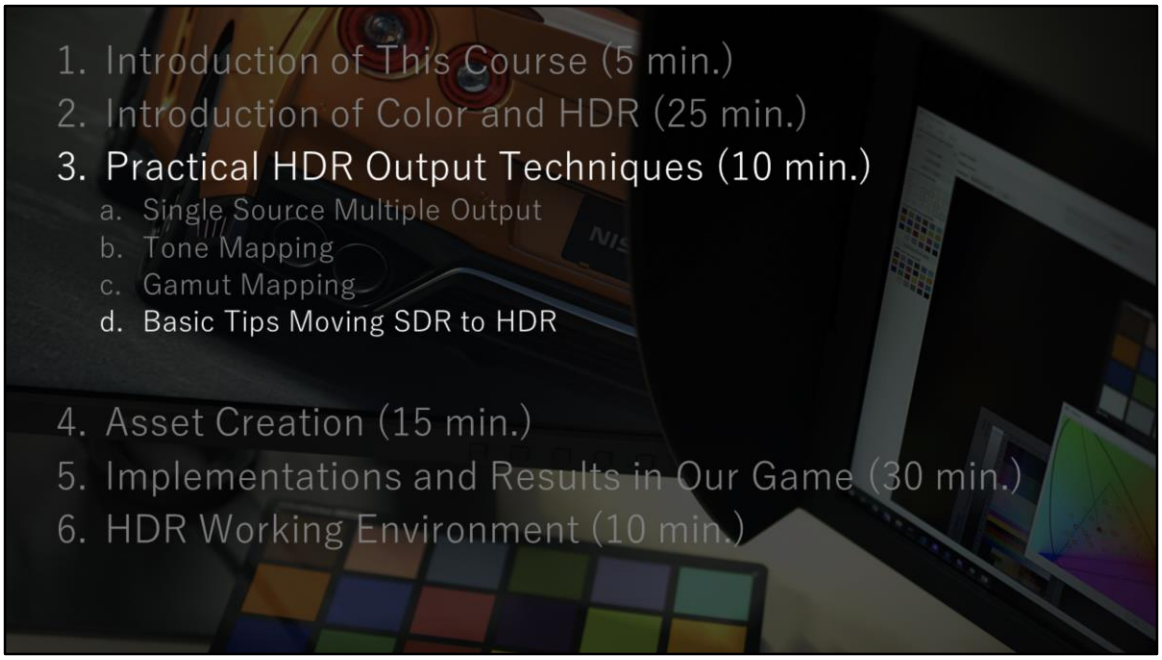
While applying tone mapping function for RGB channel independently, we don't care about hue of the pixels.

So the hue shifts when the color is very bright and vivid.

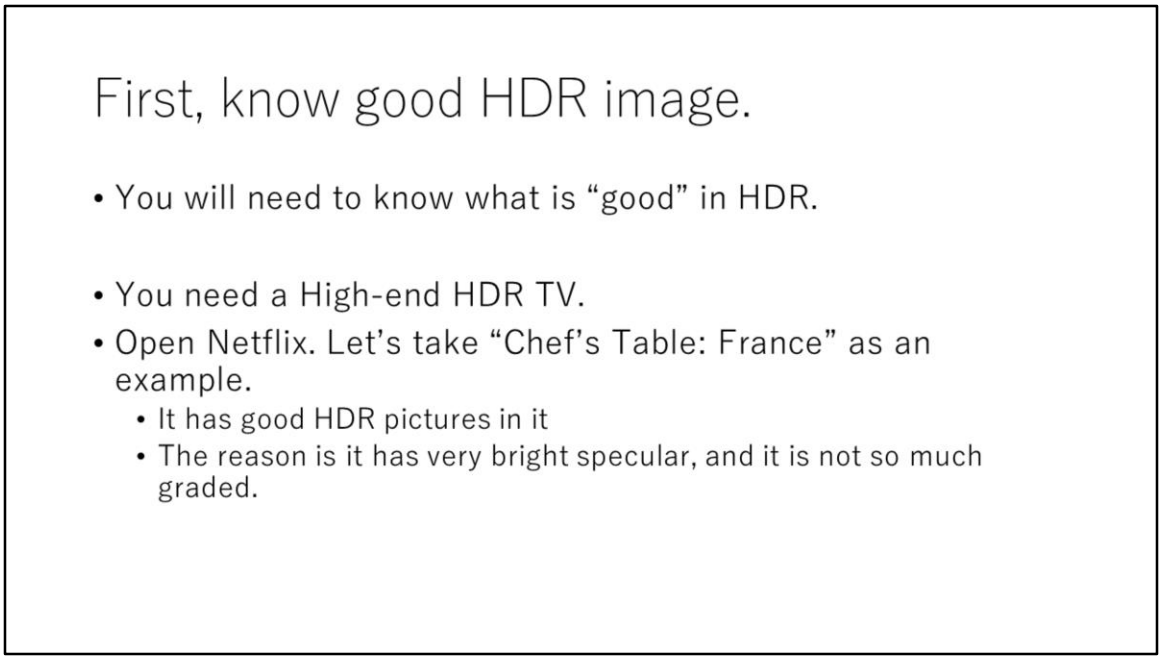


But we accepted it.

Take a close look at this real photography.
Hue shifting occurs even in the real world.



This section is just a small tips but we'd like to introduce some basic tips for moving SDR to HDR.



To make your game looks good at HDR TV,
you need to know what is “good” in HDR.

A high-end HDR TV will be required.
Netflix has a lot of great HDR contents.

I recommend to see “Chef’s table France” in HDR.
It has some really good HDR picture is in it.
The reason is the show is not so much graded and has very bright specular.

Second, just output it.

- Changing OETF and applying BT.709 to BT.2020 matrix is good point to start.
- Check them on HDR TV.

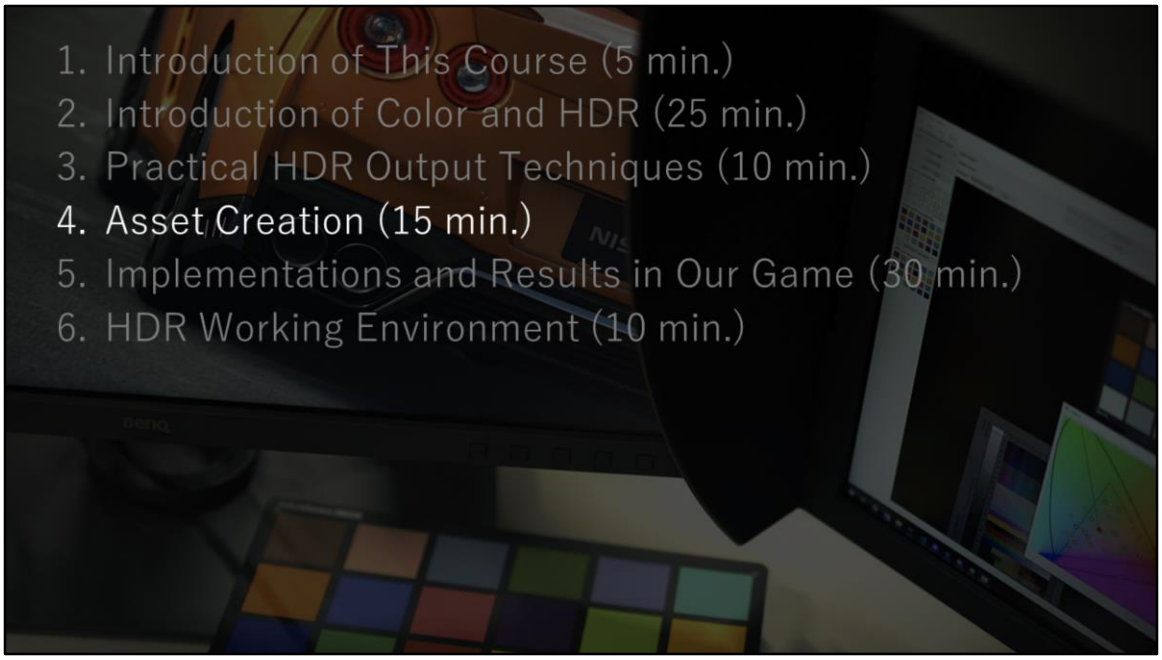
First step of move is converting your already owning assets.
Change OETF, apply color conversion.
Then check how your assets looks like on HDR TV.

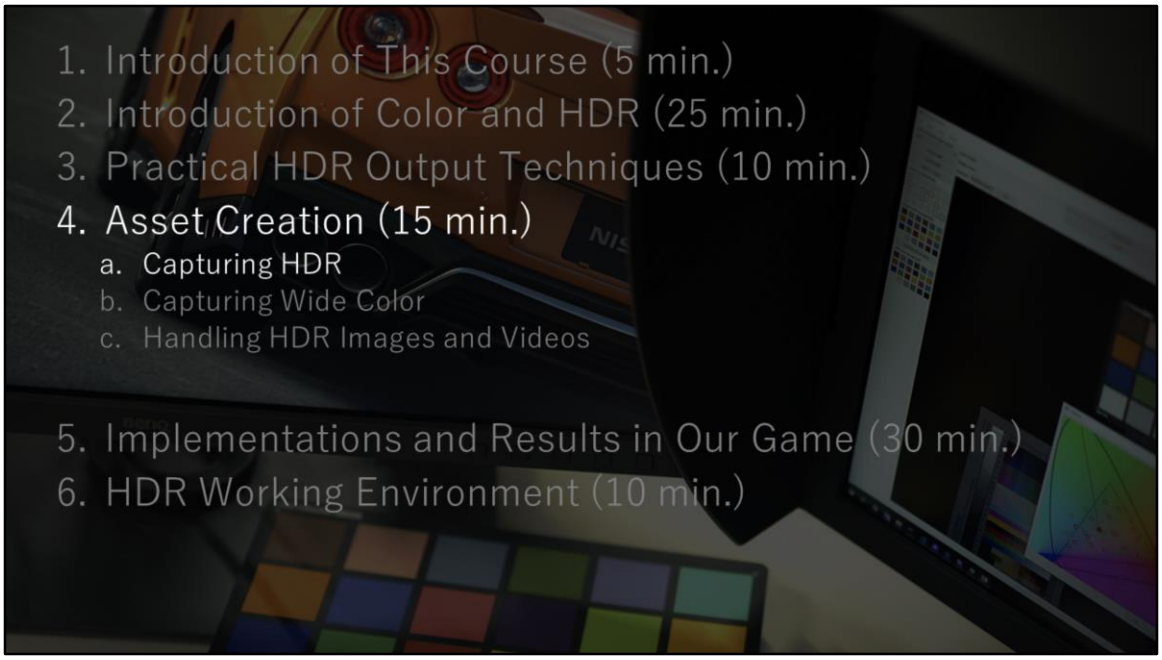
Expand your asset

- Inverse tone mapping for shininess.
- Chroma saturation for wide gamut.

- Not best answer but still a good point to start with.

Color expanding is also a good idea.
Some inverse tone mapping and chroma saturation can expand your assets.
These are not best answer but still is a good start line.

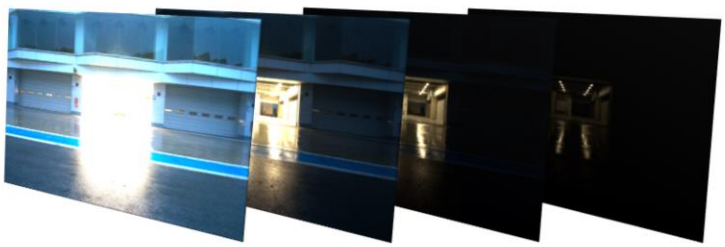

1. Introduction of This Course (5 min.)
2. Introduction of Color and HDR (25 min.)
3. Practical HDR Output Techniques (10 min.)
4. Asset Creation (15 min.)
5. Implementations and Results in Our Game (30 min.)
6. HDR Working Environment (10 min.)



From here we will talk about asset creation.
Especially for capturing HDR, wide color images.

HDR: Image Bracketing

- Image bracketing is a popular technique for capturing HDR images.
- Shoot multiple exposures at once.
- Estimate actual luminance from image brackets.



To capture the high dynamic range image we need to use image bracketing technique.
Image sensors in DSLR has 14 or more steps of dynamic range, it is equivalent to 10^4 of dynamic range.
So we need multiple exposure to cover up the dynamic range of human eye.

Debevec Method

- From each pixel on image $P_i(x)$ we estimate actual irradiance $I(x)$ using Debevec's weighted sum method [Debevec and Malik 1997]

$$I(x) = \sum_{i=1} \frac{P_i(x) W(P_i(x))}{W(P_i(x))}$$

$$W(z) = \begin{cases} z - Z_{min} & if\ z \leq \frac{1}{2}(Z_{min} + Z_{max}) \\ Z_{max} - z & if\ z > \frac{1}{2}(Z_{min} + Z_{max}) \end{cases}$$

From each pixel on multiple exposure pictures, we can estimate actual irradiance using Paul Debevec's weighted sum method.

Problem and Solution

- Naïve implementation of Debevec method is not good at dark picture.
- Dark picture contains a lot of noise.
- Weighted sum boosts the noise.
- To solve this, we created our own weighting function.
- Also implemented wavelet denoiser.

But naïve implementation will hit the problem.
Dark pictures contains a lot of noise by nature and Debevec method boosts the noise and, thus, the result will be very noisy.
To solve this we created our own weighting function that takes care of dark noisy image.
Also we implemented wavelet denoiser.

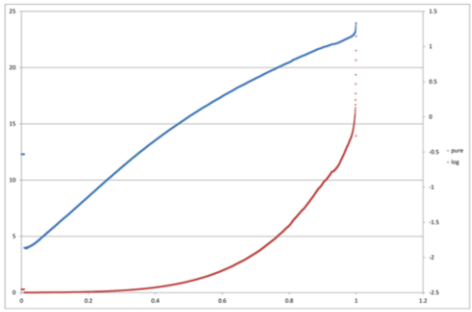
Importance of Capture Device Calibration

- To used in game, all image assets must be neutral as possible.
- No “look” or “develop” is required.
- A the calibration is very important.

To used in game, image assets must be neutral.
No look and develop is necessary.
So the image calibration process is very important.

Sensor Calibration

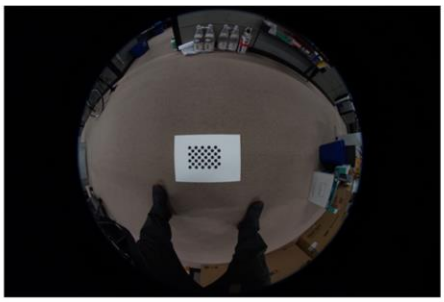
- We used Mitsunaga method to calibrate image sensor linearity. [Mitsunaga and Nayar 1999]



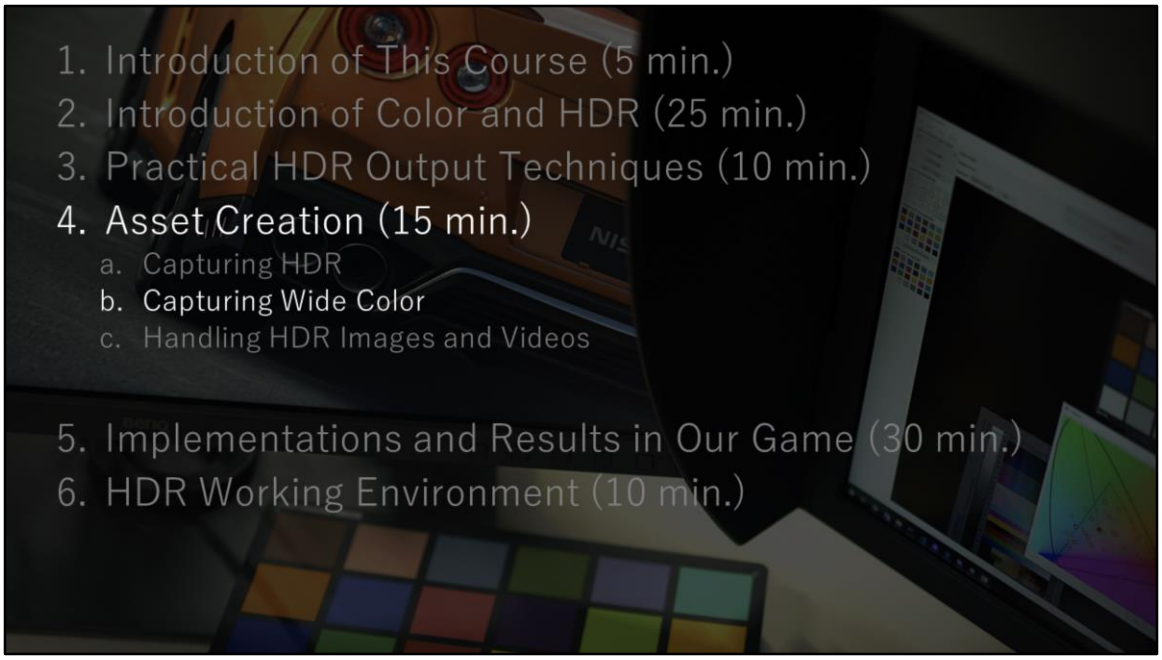
We used Mitsunaga's response curve calibration to measure image sensor linearity.

Lens Calibration

- We used OpenCV to calibrate lens distortions. [OpenCV 2018]



We used OpenCV to remove lens distortion.
Computer graphics basically has no distortion on it.
So lens distortion calibration is very important.


1. Introduction of This Course (5 min.)
2. Introduction of Color and HDR (25 min.)
3. Practical HDR Output Techniques (10 min.)
4. Asset Creation (15 min.)
a. Capturing HDR
b. Capturing Wide Color
c. Handling HDR Images and Videos
5. Implementations and Results in Our Game (30 min.)
6. HDR Working Environment (10 min.)

Color Calibration

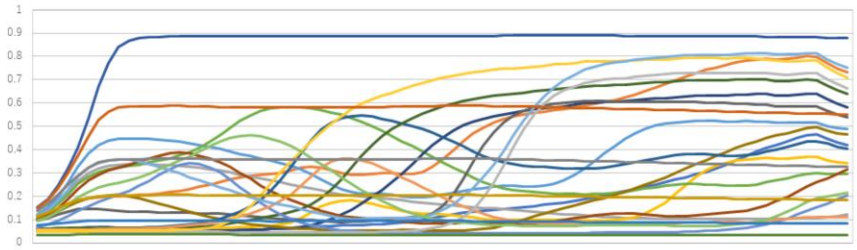
- X-Rite Color Checker. [Xrite 2018]
- SEKONIC C-700 Spectrometer.

We also calibrated image sensor's native color gamut.
We used X-Rite color checker and Sekonic C-700 spectrometer.

We take photo of color checker using high color rendition light to capture color sensor characteristics.

Color Checker Reflectance

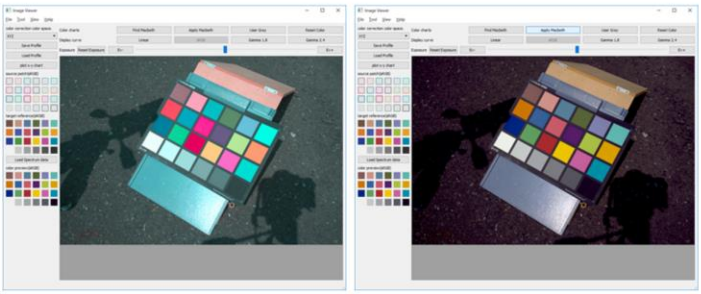
- Reflectance spectrum of Color Checker are precisely measured by Noboru Ohta. [Ohta 1997]
- We also measured reflectance spectrum by ourselves.



Xrite color checker has nicely controlled color patches.
Spectrum reflectance of each patches are well known.
We also measured spectrum reflectance of each patch by ourselves, but there is no significant difference.

Gamut Reverse Engineering

- We can calculate reverse gamut matrix from reflectance spectrum and illumination spectrum using Gauss-Seidel solver.
- See my talk at CEDEC2016(Japanese) for more detail. [Uchimura 2016]



From these reflectance and captured illuminant spectrum, we can calculate ideal absolute color of each patches.
Then we can solve gamut conversion matrix from color relation between patches and camera raw image using simple Gauss-Seidel solver.
For more detail please check my talk at CEDEC 2016.

Capturing Device Example

- ILCE-7 series DSLR from SONY.
- Calibrated all the lenses we used.
- Calibrated all the cameras we used.

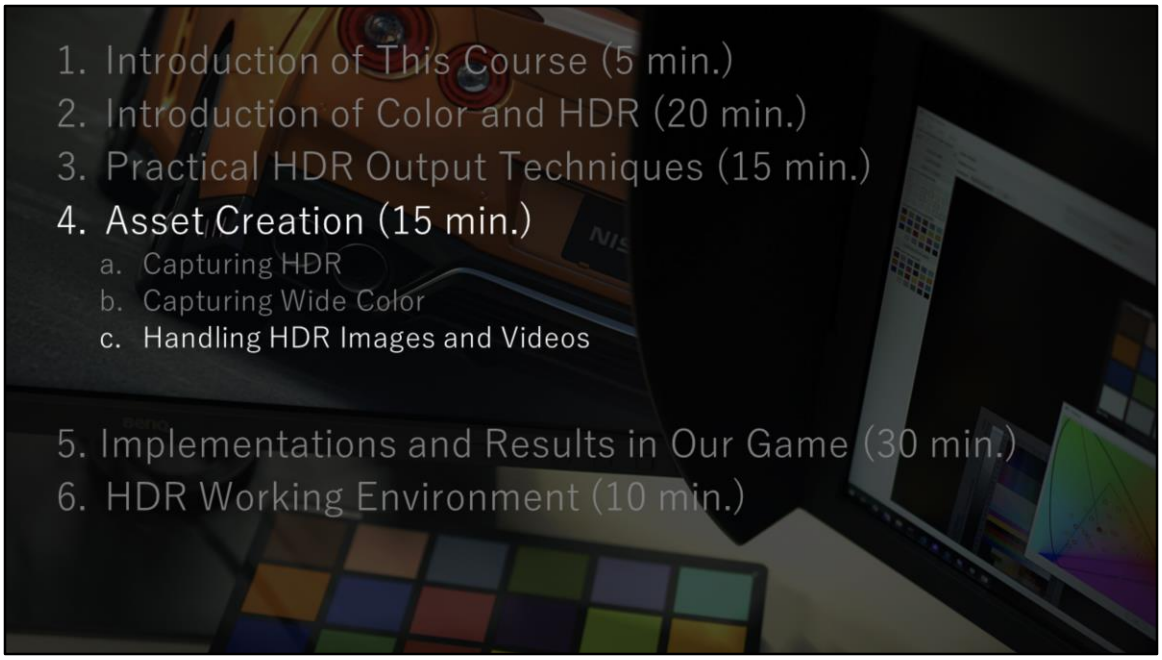
- That was really tough task.

We used SONY ILCE-7 series DSLR.
We have measured all distortions of lenses

Here are some results.
Actually this is a rendering from game itself, background images and light reflections are captured using the camera system we presented.

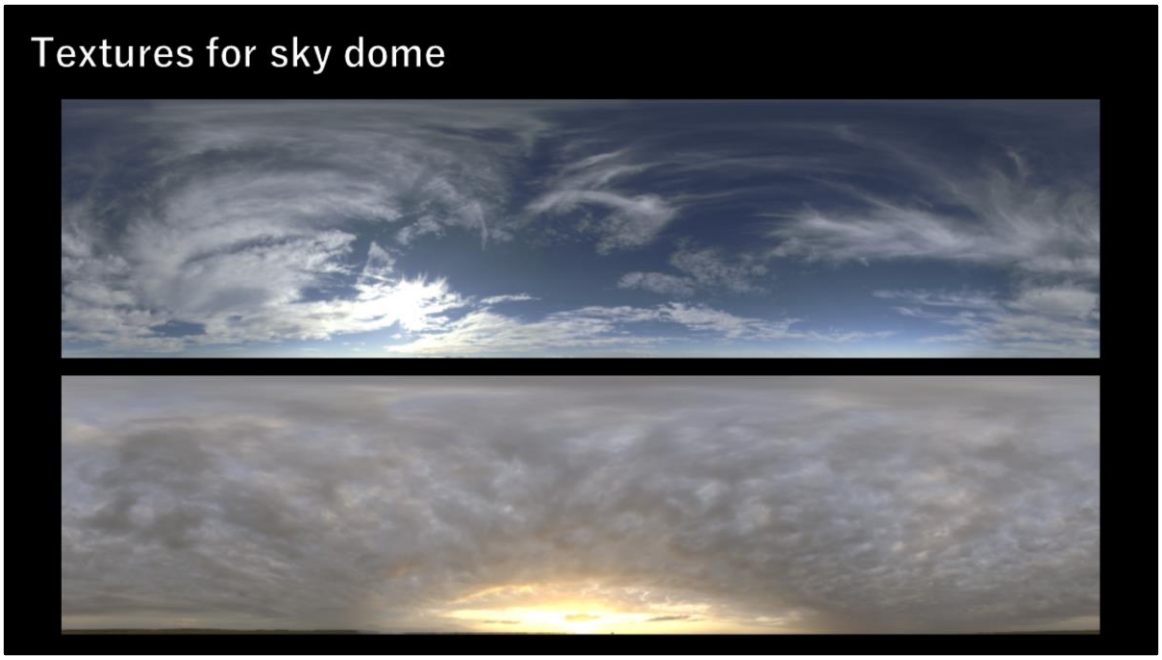
To achieve photorealistic blending between background images and car model, Lens distortion calibration is required.

Captured in High dynamic range and wide color gamut, these images also works well in HDR TV.

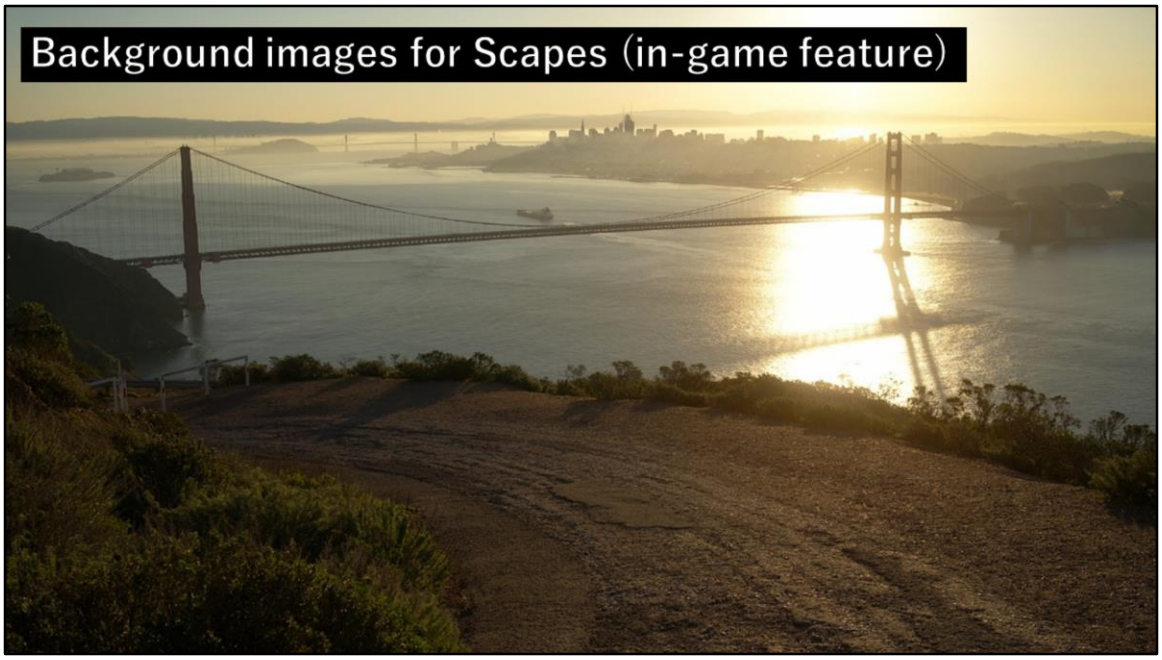


Next, we'd like to talk about handling HDR still images and videos in our development.

These are examples of our HDR still images.
This is an image of a texture used for light emission.



These are images used in our sky dome.



And this is an example background image used in Scapes, an in-game feature.
Scapes will be described in detail later.

HDR Image Formats

- We use a few HDR image formats

Asset Creation	Mainly used by artists during asset creation and editing
Runtime	Mainly used by the game itself

We use a few HDR image formats for each purpose. Mainly, we use different formats for asset creation and for runtime.

For asset creation, HDR images are mainly used by artists during asset creation and editing.
On the other hand, during runtime, HDR images are directly used by the game itself.

HDR Image Formats for Asset Creation

- There are many image formats that support HDR images
- We mainly use two formats:
  - **.hdr**: Radiance HDR / RGBE image format
  - **.exr**: OpenEXR

First, we would first like to talk about HDR image formats for asset creation.
There are many image formats that support high dynamic range images, and we mainly use two formats: Radiance HDR and OpenEXR.
We would like to explain some of the advantages and disadvantages of each format.

Radiance HDR / RGB image format (.hdr)

- http://radsite.lbl.gov/radiance/refer/filefmts.pdf
- **Pros**:
  - Easy to understand format specification
  - Many software exist that can handle it
- **Cons**:
  - Can't store complex data (e.g. multi channels)
  - Not so good precision
    - Shared exponent method
    - It is almost enough for our purposes…
  - Compression method is simple and inefficient

Radiance HDR is a widely known image format.
The format specification is easy to understand, and many public implementations exist for it.
In addition to that, many existing software can handle Radiance HDR, so it has good portability.

However, it has some weakness.
First, it cannot store complex data such as multi channel data, and due to its shared exponent method, the precision is not so good. (For our purposes, this precision is almost sufficient.)
Its compression method is also simple and inefficient, and high resolution images consume a large amount of storage.

OpenEXR (.exr)

- http://www.openexr.com/
- **Pros**:
  - Can store complex data (deep data)
  - Can store better precision data (16, 32-bit floating point)
  - Can be handled by major DCC software
  - Better compression methods than Radiance HDR
    - Lossy compression method also available
- **Cons**:
  - Lossy compression's quality is not as good as JPEG XR

Open EXR was developed by Industrial Light & Magic, and is the de facto standard HDR format in the computer graphics industry.
Open EXR can store complex data and better precision data such as 32 bit floating point values.
Major DCC software support and it has a better compression methods than Radiance HDR. Also, a lossy compression method is available.

Basically, Open EXR is a very good format, but lossy compression quality is not as good as JPEG XR.

In any case, artists can use these two formats for their work.

Why These Two Formats?

- There are other HDR image formats but we are using these formats because of portability and rich features

.hdr (Radiance HDR)	Good portability
.exr (OpenEXR)	Rich features

There are other HDR image formats of course, but we are using these two formats because of portability and rich features.

HDR Image Formats for Runtime

- We also need to use HDR images at runtime
- Runtime HDR image formats requirements
  - **Compact**
    - Needs to be stored in game disk, thus needs a compact size
  - **Fast Decoding**
    - Decoding performance directly impact loading time
  - **GPU friendly**
    - Runtime memory efficiency and cost of sampling textures

We also need to use HDR images during the game's runtime. There are some requirements for runtime HDR image formats.

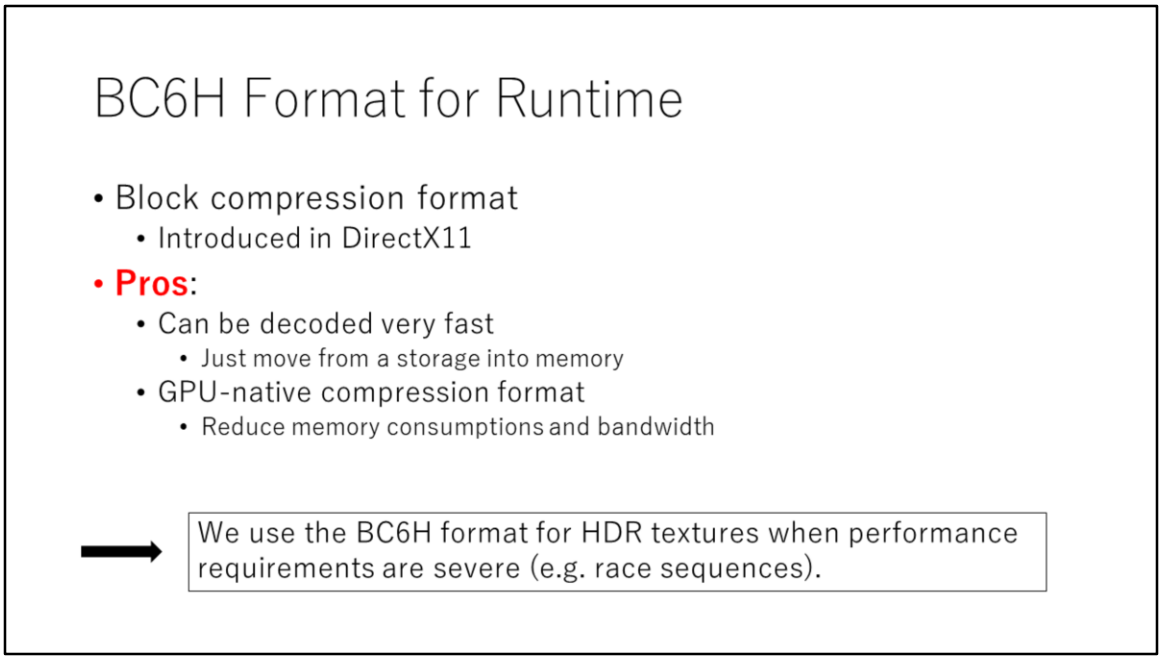
First, this format needs to be very compact, as we need to store it in the game's disk, which has a limited amount of space.
Second, it needs to have a fast decode time, as decoding performance directly impacts loading times.
Finally, it needs to be GPU friendly, as runtime memory efficiency and sampling cost is very important.

BC6H Format for Runtime

- Block compression format
  - Introduced in DirectX11
- **Pros**:
  - Can be decoded very fast
    - Just move from a storage into memory
  - GPU-native compression format
    - Reduce memory consumptions and bandwidth

So, we use BC6H format for runtime images.

This is a block compression format and introduced in DirectX11.
This format can be decoded very fast and it's designed for GPU so it's very GPU friendly.
Because of that, this format is appropriate for runtime.

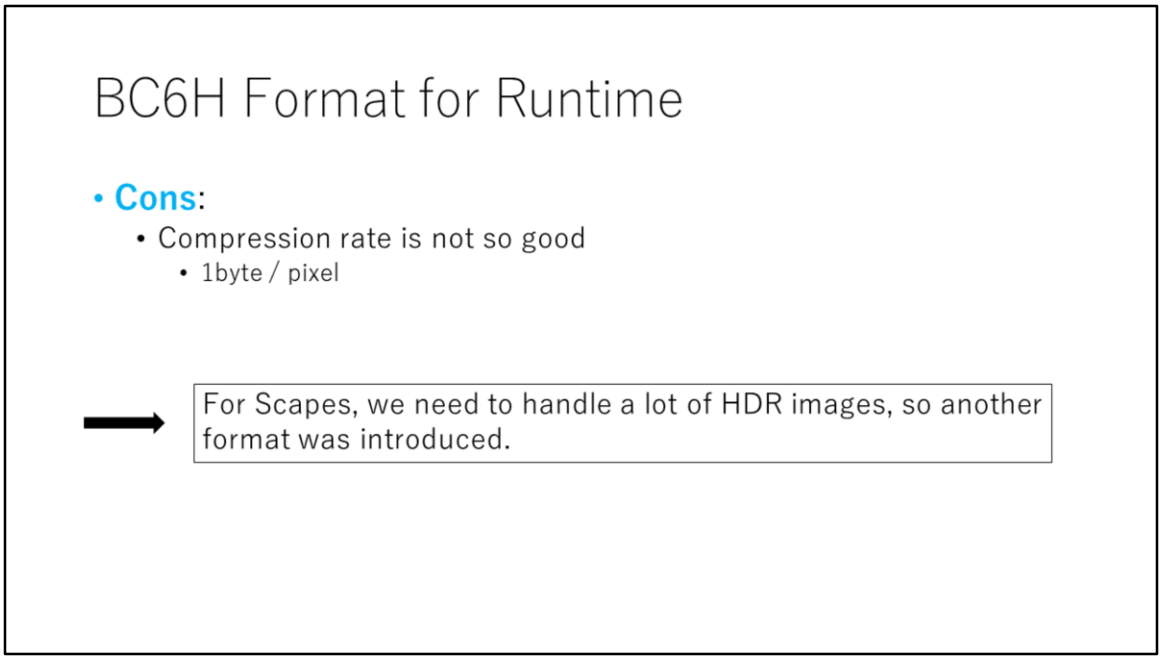


We use the BC6H format for HDR textures when performance requirements are severe, such as race sequences.

BC6H Format for Runtime

- **Cons**:
  - Compression rate is not so good
    - 1byte / pixel

BC6H is a good format, but its compression rate leaves much to be desired, as it can only achieve one byte per pixel.



Instead, for Scapes, we need to handle a lot of HDR images, and thus another format was introduced.

Scapes

"Scapes is an in-game feature that allows you to position a car in scenes from around the world and take photos."

By the way, let me explain a bit more about Scapes.

Scapes is an in-game feature that allows you to position a car in multiple scenes from around the world and take photos.

Scapes Background Images

Formats	HDR
Resolution	Very high (between 6000x4000 and 8000x5000)
Numbers	Over 1,000

In Scapes, all background photos are HDR and resolutions are very high, and each photo has a resolution between 6000x4000 and 8000x5000)
In addition to that, we had over 1000 background images at launch time.

Scapes Background Images

Formats	**HDR**
Resolution	**Very high (between 6000x4000 and 8000x5000)**
Numbers	**Over 1,000**

→ BC6H is inefficient! The estimated total size was over 30GB (Half the size of a game disk ☹) and the file size was a large issue.

And because of our requirements, BC6H is inefficient.
If we used BC6H for Scapes, the estimated total size for all images would be over 30GB, which is half the size of a game disk.

JPEG XR

- https://jpeg.org/jpegxr
- Developed by Microsoft
- **Pros**:
  - **High quality and efficient lossy algorithm**
  - Can store complex data
  - Supports not only HDR format (like fp32, fp16) but many other color formats
  - Reference implementation is open and under the public domain
- **Cons**:
  - There are few software that supports JPEG XR appropriately
    - But the image preview in Win10 supports JPEG XR

So we decided to use JPEG XR, which was developed my Microsoft.

This format can store complex data and support not only HDR format but many other color formats.
For our purposes, the most important aspect of JPEG XR is that JPEG XR has high quality, and an efficient lossy algorithm.

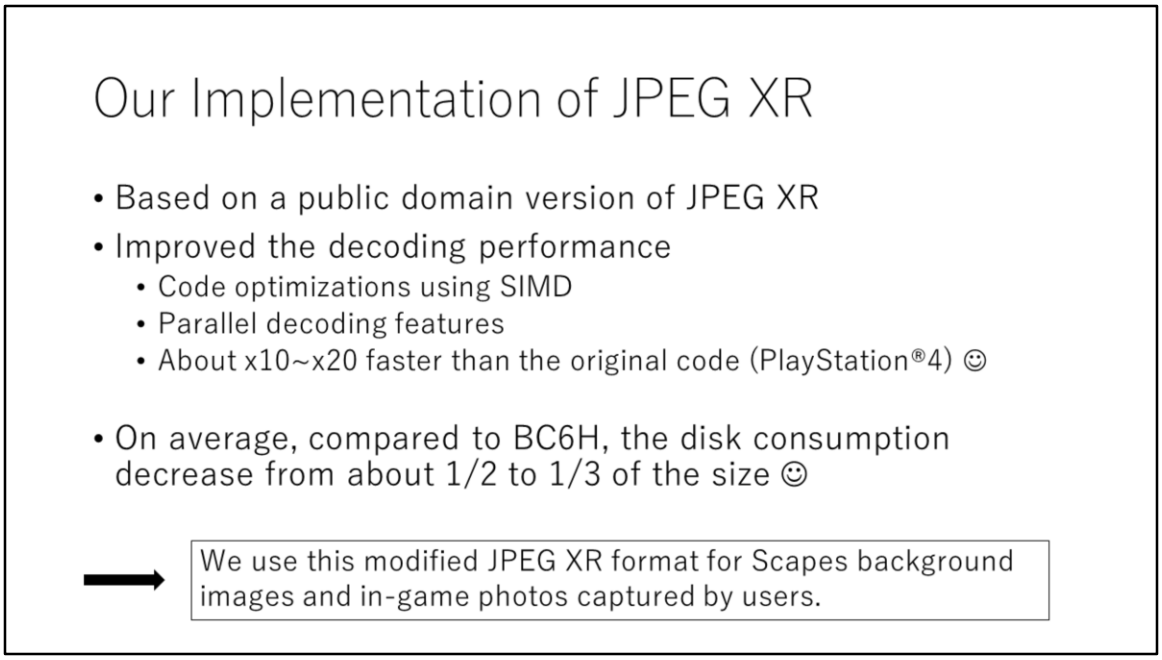
Unfortunately, there are few software that support JPEG XR appropriately, but the image preview in Windows 10 support it.

Our Implementation of JPEG XR

- Based on a public domain version of JPEG XR
- Improved the decoding performance
  - Code optimizations using SIMD
  - Parallel decoding features
  - About x10~x20 faster than the original code (PlayStation®4) ☺

Our implementation is based on a public domain version of JPEG XR.

We improved the decoding performance, and applied code optimizations using SIMD and added parallel decoding features.
Using these optimizations, we achieved about 10 times to 20 times faster decoding times than the original code.



On average, compared to BC6H, the disk consumption decreased from about one half to one third of the size.
Therefore, we use this modified JPEG XR format for Scapes background images and in-game photos captured by users.

HDR Video Formats

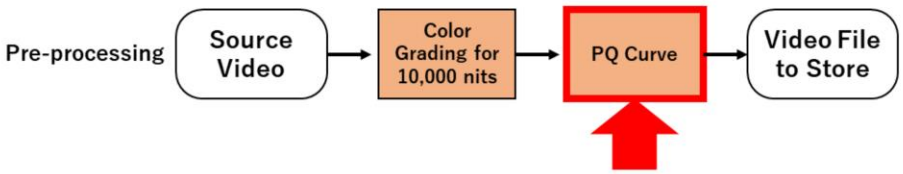
- **Pre-processing**
  - In-game videos are graded for 10,000 nits targets
    - e.g. opening and ending videos
  - Manually graded to expand the dynamic range

Next, we will explain our HDR video formats.

As a pre-processing step in-game videos, such as the opening and ending videos, are graded for 10,000 nit targets.
These videos are manually graded to expand the dynamic range.

HDR Video Formats

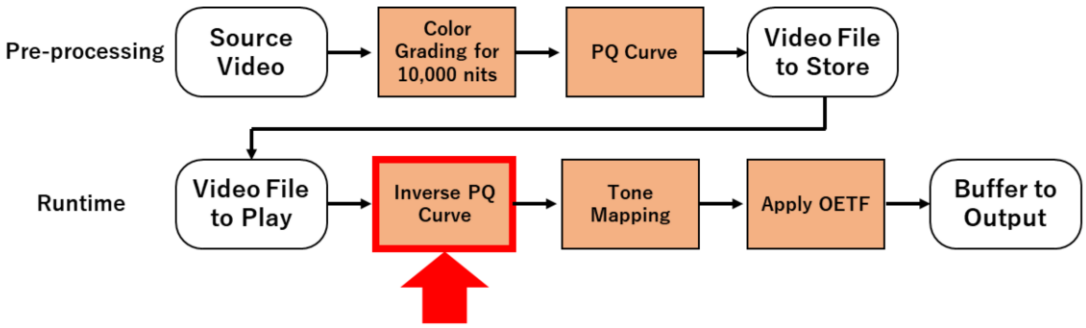
- **Pre-processing**
  - Stored as a file after PQ curve is applied



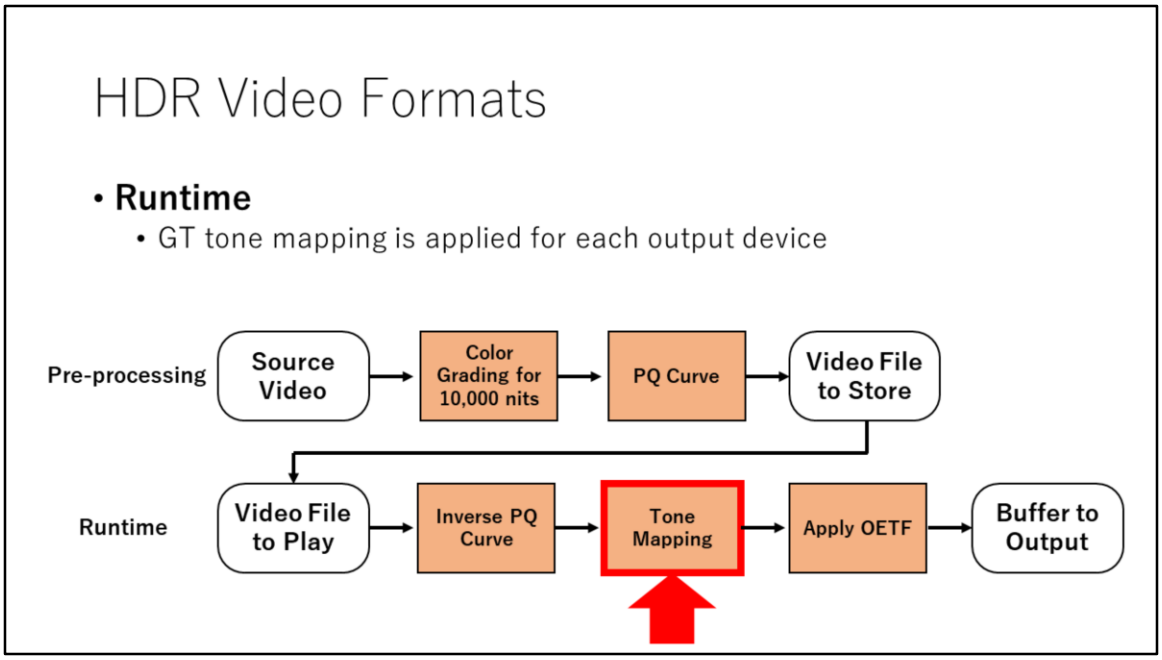
Then, graded videos are stored as files after the PQ curve is applied.

HDR Video Formats

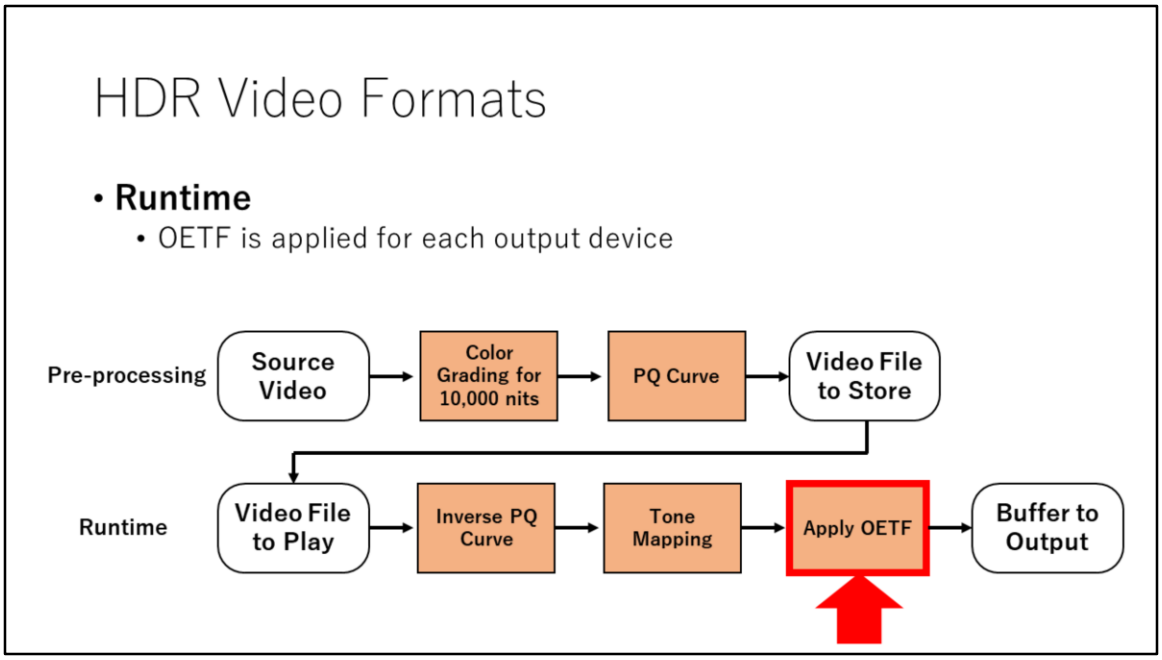
- **Runtime**
  - Inverse PQ curve is applied and then the buffer is linearized



At runtime, before playing the video, inverse PQ curve is applied to the video file to play, and then the buffer is linearized

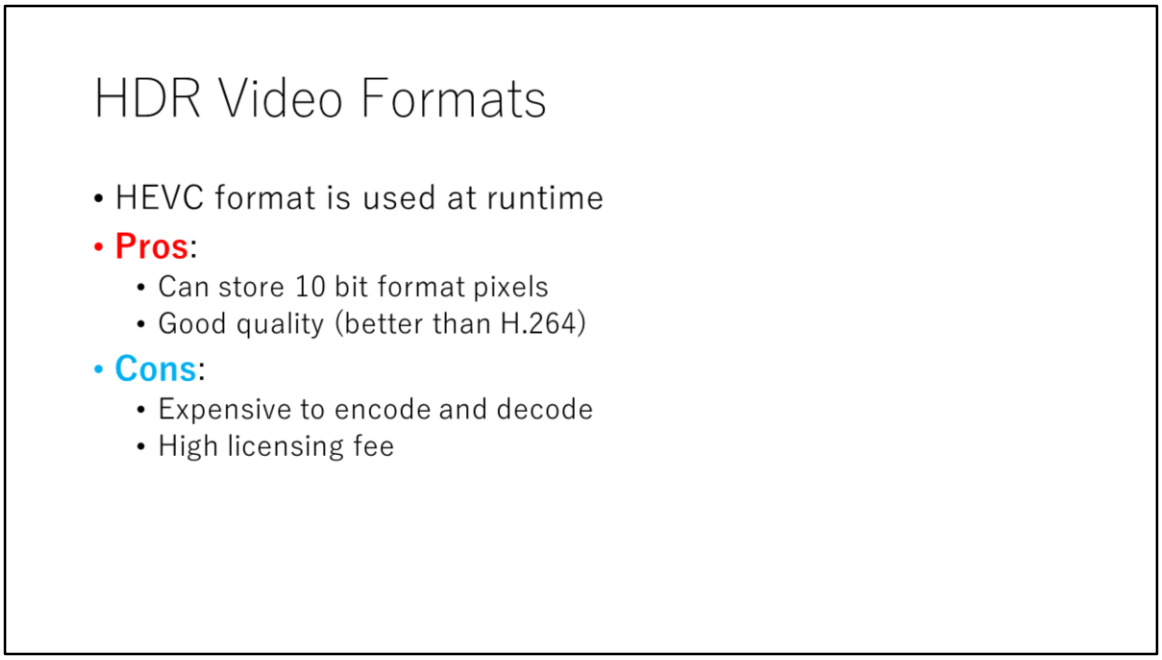


Next, GT tone mapping is applied for each output device.



Finally, OETF is applied for each output device.

In SDR, sRGB gamma is applied and in HDR, inverse PQ curve is applied. With this process, we can use the same source video for SDR and HDR output device.

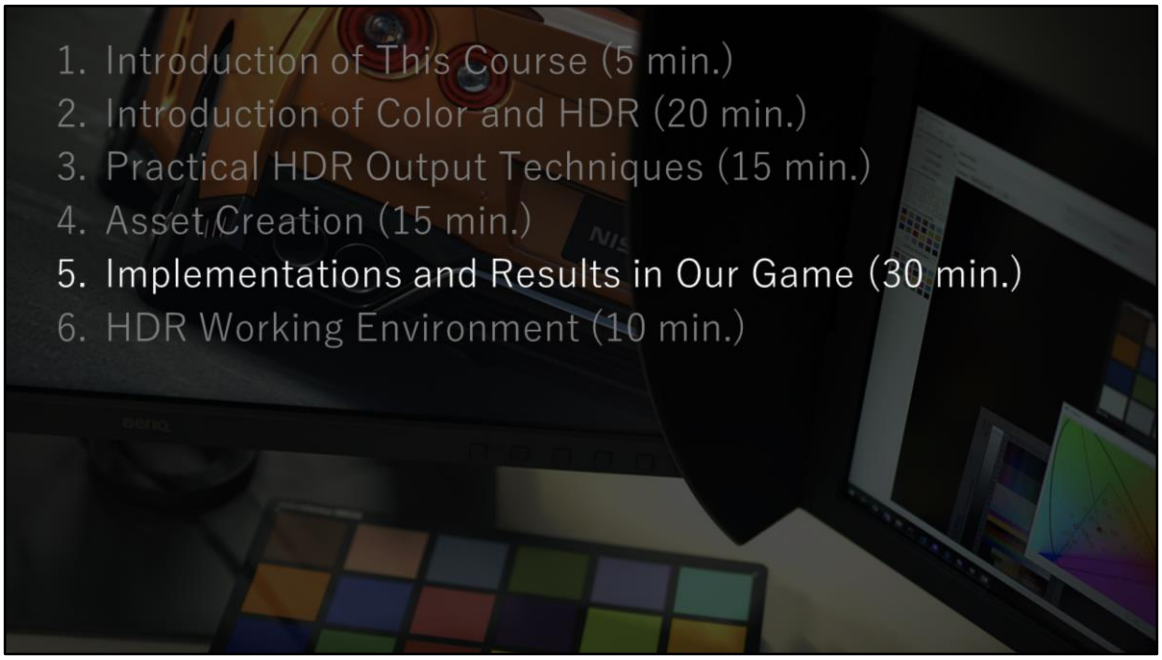


Our runtime video format is HEVC.

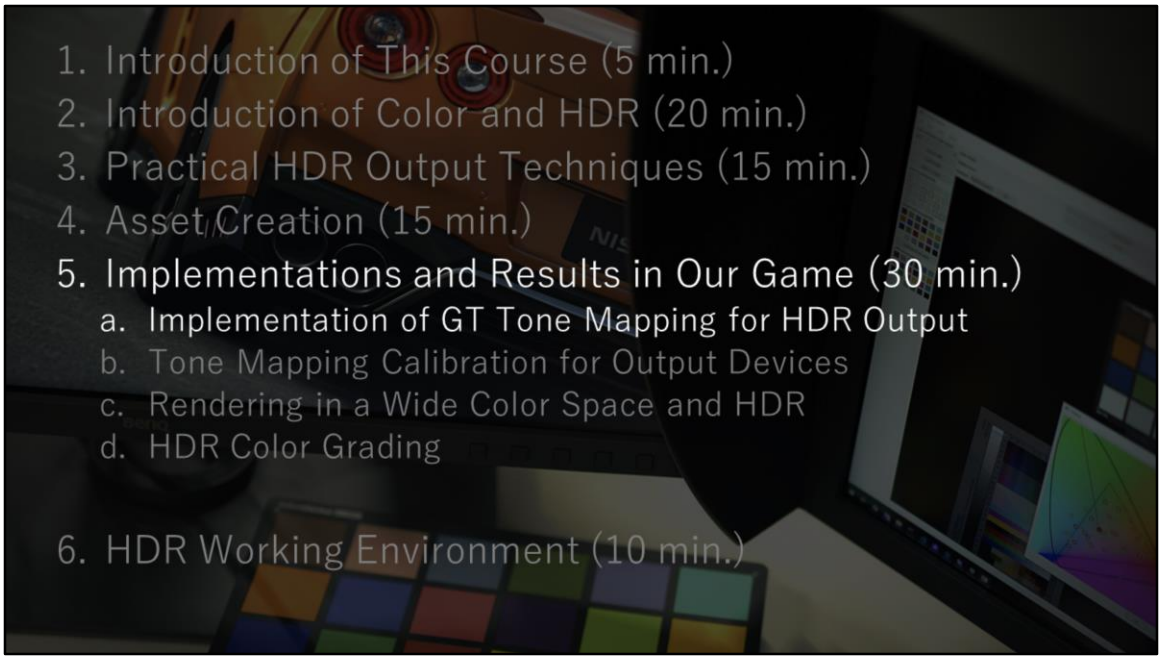
HEVC is a high quality video format.
This format can store 10 bit format pixels, which is useful for HDR outputs, and has better quality than H.264.

On the other hand, HEVC is expensive to encode and decode. In addition to that, it has a high licensing fee.

We needed a high quality runtime video, which is why we are using HEVC.



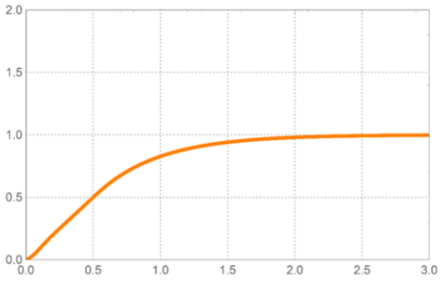
Next, we'd like to talk about



First we will talk about implementation of GT Tone Mapping for HDR Output.

Implementation of GT Tone Mapping for HDR Output

- Our runtime tone mapping method is “GT Tone Mapping”
  - We already explained the theory behind it in this course
  - How did we implement it in our game?



Our runtime tone mapping method is called “GT Tone Mapping”.

We already explained the theory behind it in this course, but how did we implement it in our game?

Performance Issue

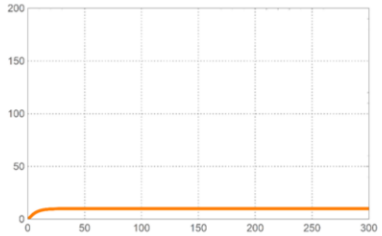
- The simple implementation has a high computational cost
  - Complex function (pow and exp functions are heavy on the GPU)

```
float GTTonemap(float x)
{
    float L = m + a * (x - m);
    float T = m * pow(x/m, c);
    float S = P - (P - S1) * exp(-C2*(x - S0)/P);
    float w0 = 1 - smoothstep(0.0f, m, x);
    float w2 = (x < m+l)?0:1;
    float w1 = 1 - w0 - w2;
    return (float)(T * w0 + L * w1 + S * w2);
}
```

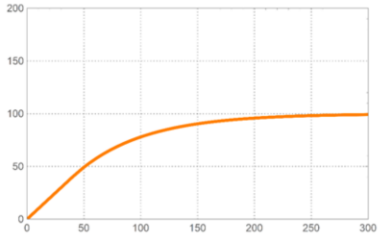
First we had an issue with the performance.
The simple implementation has a high computational cost because GT Tone Mapping is a complex function.
It requires the use of power and exponential functions, which are heavy to compute on the GPU.

Performance Issue

- Approximations (e.g. rational function fitting) need to be calculated for each tone mapping parameter
  - GT tone mapping is “variable” with respect to a device peak brightness



Tone Mapping for
Peak brightness: 1,000 nits



Tone Mapping for
Peak brightness: 10,000 nits

Approximations such as rational function fitting need to be calculated for each tone mapping parameter.
But because GT tone mapping varies with to a device’s peak brightness, we can’t simply apply rational function fitting.

Lookup table (LUT) approach

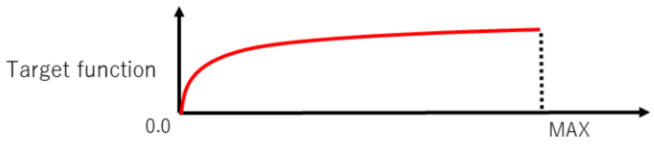
- We used a lookup table for computing the GT Tone Mapping at runtime
- **Pros**:
  - Approximation error can be reduced by just using a larger size table
  - Cost of sampling lookup table tend to be smaller than the analytic calculation
    - ALU pressure was the main bottleneck in our case
- **Cons**:
  - Memory consumption is larger than using an analytic function
  - Quantization error and discretization error

So we used a lookup table for computing GT Tone Mapping at runtime.
Using a lookup table means we applied a piecewise linear approximation to the function

By using a lookup table, approximation error can be reduced by just using a larger size table.
Sampling cost tend to be smaller than the analytic calculation, but in our case ALU pressure was the main bottleneck.

Of course, memory consumption is larger than using an analytic function, but it didn't matter for our case.
By using a lookup table, quantization and discretization error happen, so we need to decide appropriate parameters.

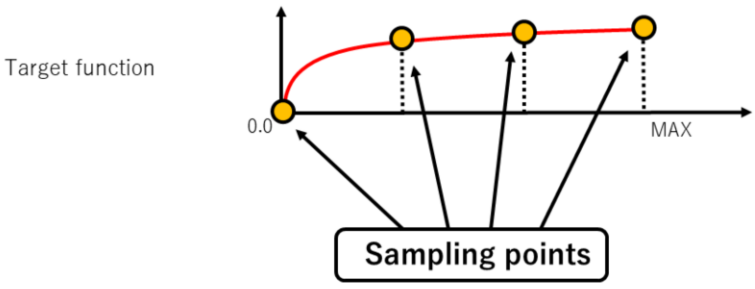
Usual Linear LUT Mapping



Let us begin by explaining about how usual linear lookup table mapping works.

Usual Linear LUT Mapping

- Uniformly sample a target function

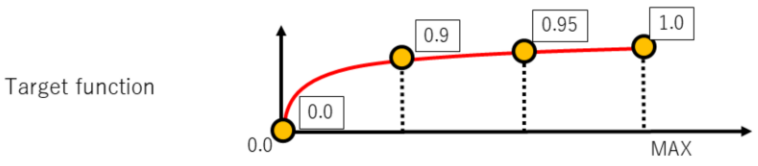



```
sampling_point = lut_index / (N-1) * MAX
```

In this mapping, we uniformly sample a target function.

Usual Linear LUT Mapping

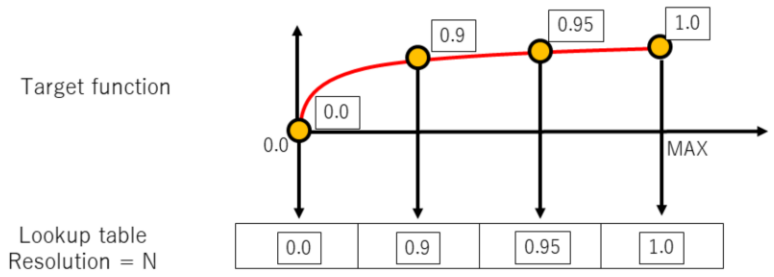
- Uniformly sample a target function



Example values are shown in the figure.

Usual Linear LUT Mapping

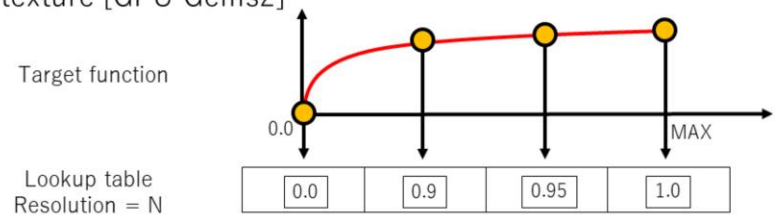
- Store sampled values onto a lookup table



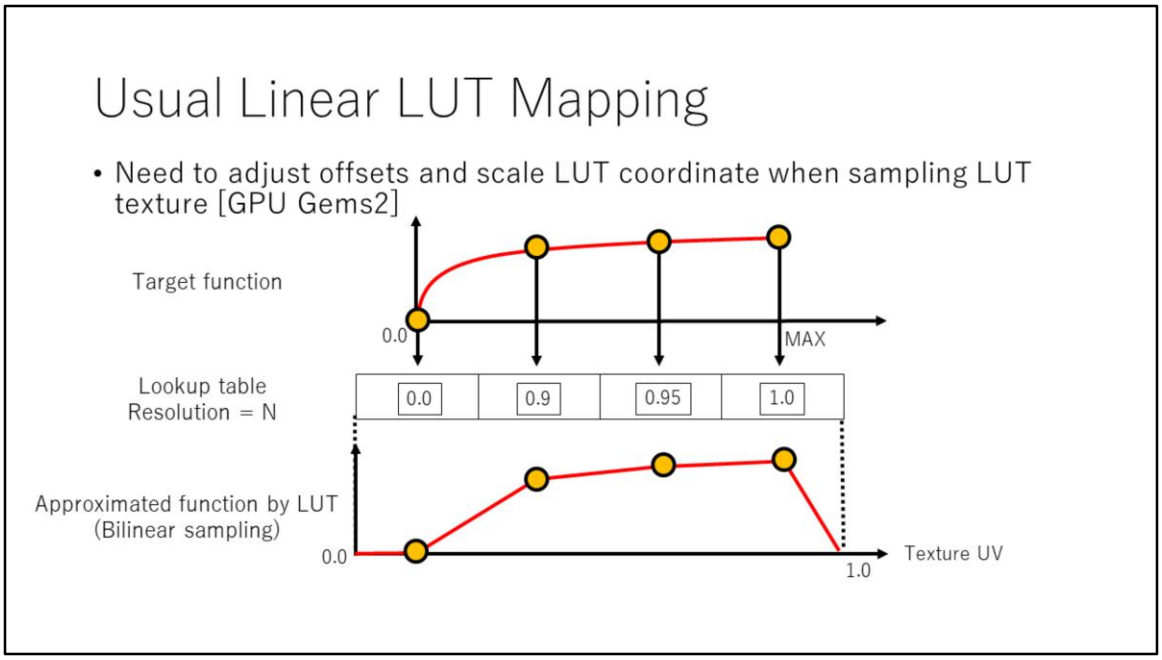
Then store sampled values onto a lookup table.

Usual Linear LUT Mapping

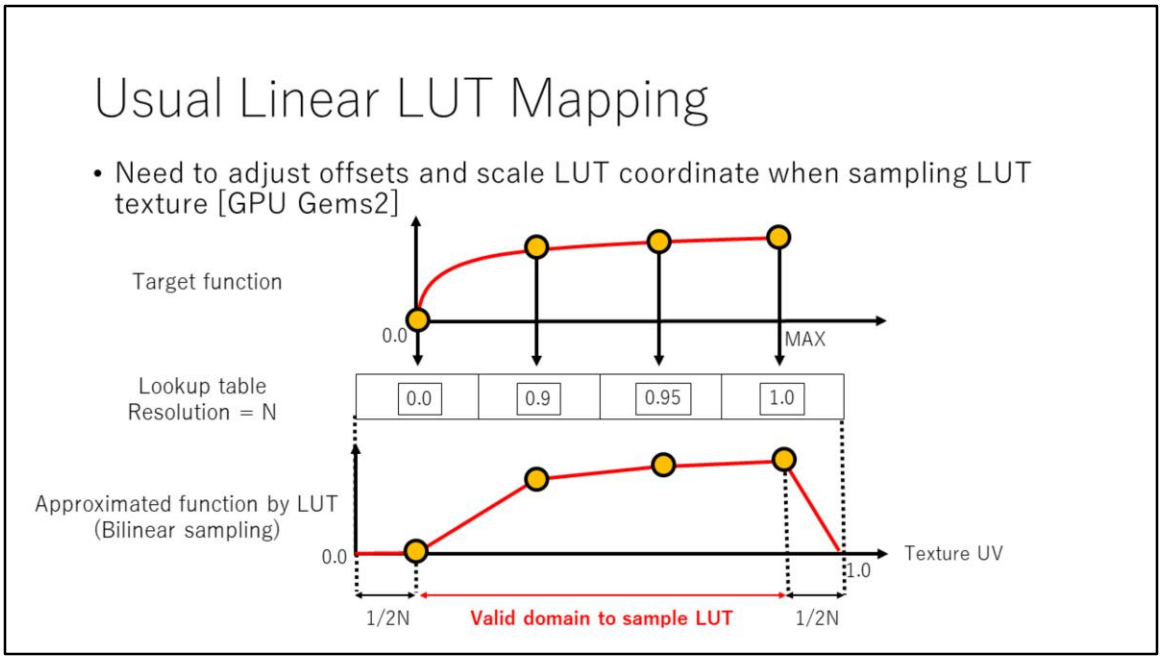
- Need to adjust offsets and scale LUT coordinate when sampling LUT texture [GPU Gems2]



At runtime, we need to adjust offsets and scale for lookup table coordinate.



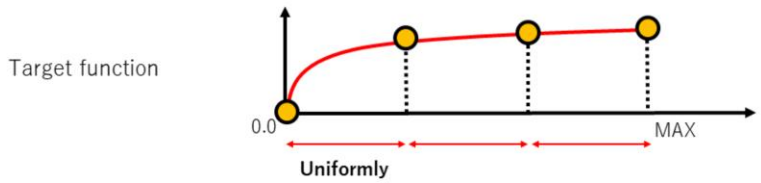
The figure shows a naïve bilinear approximation using lookup table.
As you can see, the UV range zero to one doesn't match the range of the target function.



So, we need to adjust offsets and scale of lookup table UV coordinates.
With this process, we can find valid positions to sample the lookup table.

Nonlinear LUT Mapping

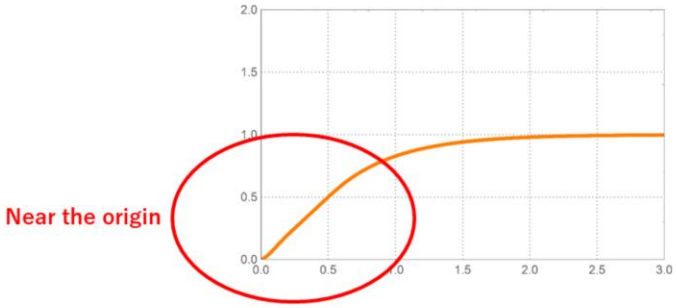
- **NOT** uniformly map the input value to the lookup table coordinates (X-axis)



Next, we will explain about our nonlinear lookup table mapping.
We do not uniformly map the input value to the lookup table coordinates.

Nonlinear LUT Mapping

- Accuracy is more important **near the origin**
  - The low part and middle part of the tone mapping function
- So, want to assign more texels near the origin (and less texels far from the origin)

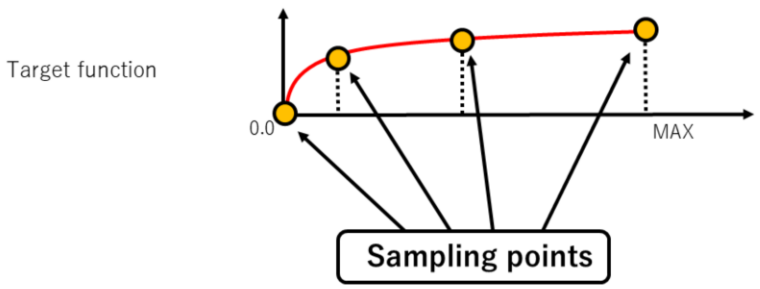


Accuracy is more important the near the origin, because there are low and middle parts of the function near the origin.
These parts decide the whole behavior of the function.

So we want to assign more texels near the origin to improve the accuracy and assign less texels far from the origin.

Nonlinear LUT Mapping

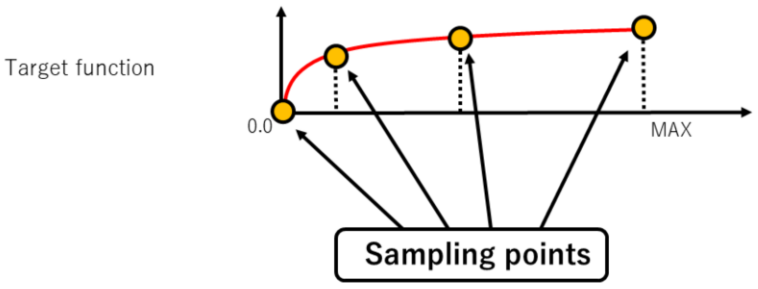
- Non-uniformly sample the target function to assign more texels near the origin



This is our nonlinear lookup table mapping.
We non-uniformly sample the target function to assign more texels near the origin.

Nonlinear LUT Mapping

- We use a x^2 mapping so that we can use square root at runtime
  - Other mapping can be used [Malin 2018]

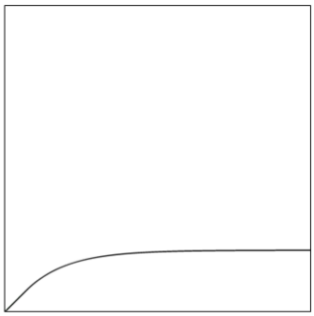



```
sampling_point = pow(lut_index / (N-1), 2.0f) * MAX
```

We use a square mapping because the inverse function, the square root, is easy to calculate at runtime.

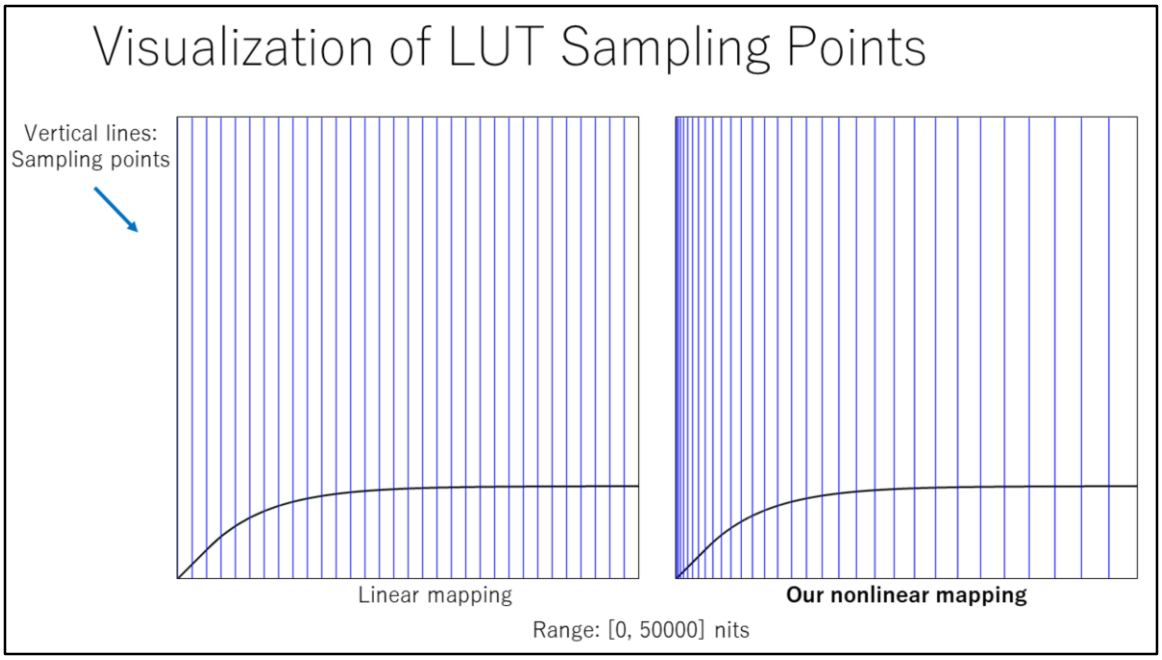
Visualization of LUT Sampling Points

- Whole input range: [0, 50000] nits
- LUT resolution: 32

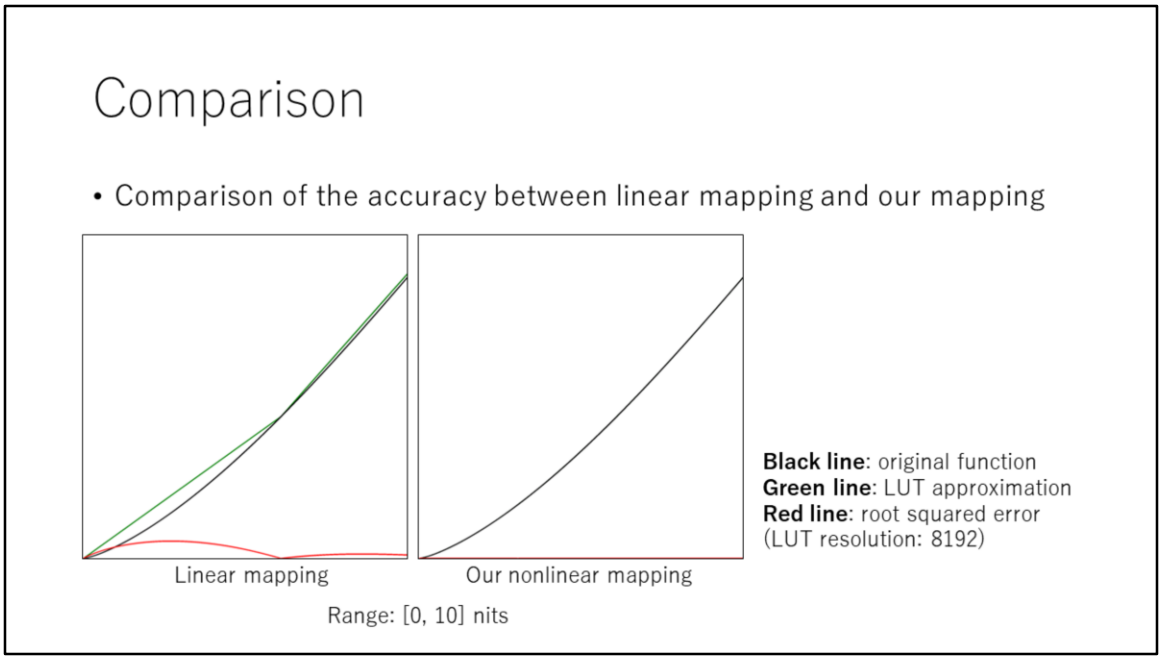


The whole tone mapping function
Range: [0, 50000] nits

We will show the visualization of mapping.
The whole input range is 0 to 50,000 nits and the lookup table resolution is 32.



And this is a comparison.
Vertical lines indicates sampling points.
As you can see, in our nonlinear mapping, there are more sampling points near the origin compared to the usual linear mapping.



This is a comparison of the accuracy between linear mapping and nonlinear mapping.

The black line is the original function, the green line is the lookup table approximation, and the red line is the root squared error.
The LUT resolution used here is 8,192.

The left figure shows the error of linear mapping is much larger than nonlinear mapping.

Results

- Specification of lookup table in our game

Texture	1D texture
Format	32 bit float
Width	8,192
Total Size	32 KB

- Performance results (PlayStation®4, 1920x1080)

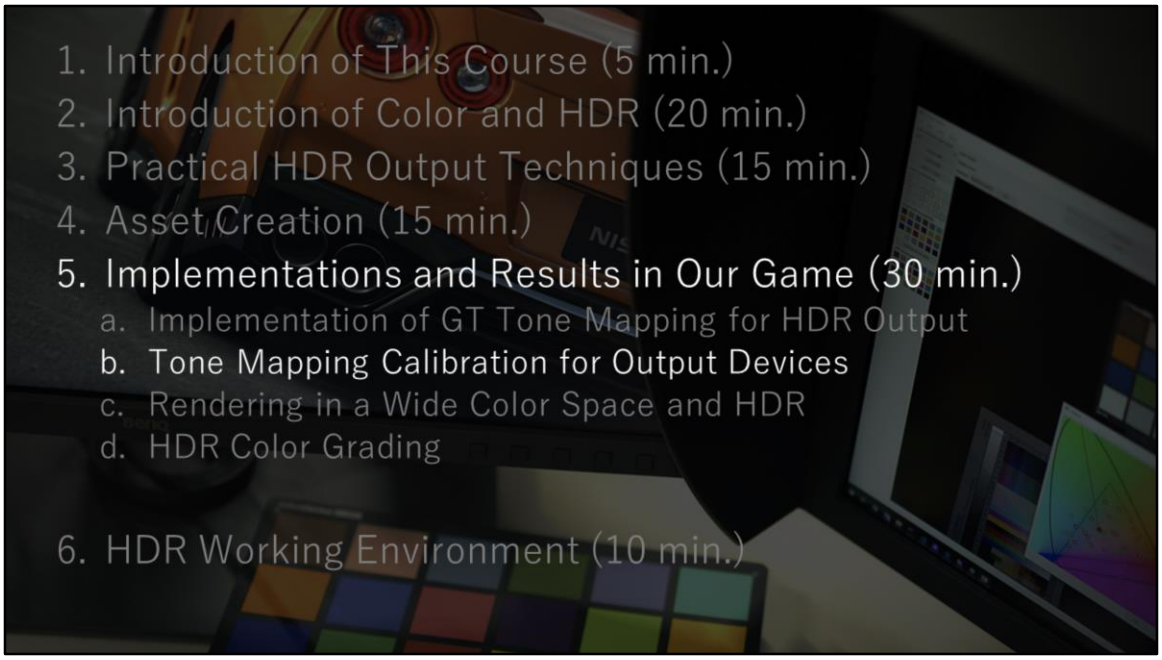
Analytic Calculation	0.40 ms
Our LUT Approach	**0.24 ms**

Our LUT approach includes creating LUT in the compute shader

Finally, our results.

On the top table you can see the specification for the lookup table in our game.
We used a 32-bit one dimensional texture, with 8,192 texels, using a total of 32kilobytes.

As for the performance, on the PlayStation®4, outputting a Full HD image, the analytic implementation takes 0.40 milliseconds per frame.
On the other hand, our lookup table approximation takes 0.24 milliseconds per frame.



Next, we will talk about tone mapping calibration for output devices.

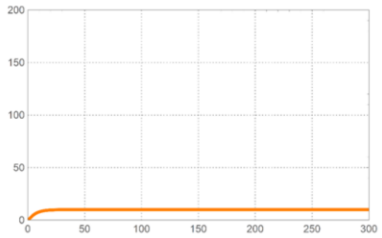
Tone Mapping Calibration for Output Devices

- Appropriate HDR output needs information about output HDR devices
  - Needs information about “peak brightness” of a device

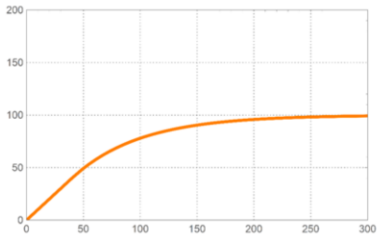
We want to output appropriate HDR images and for that purpose, we need information about the output HDR device.
Therefore, we need information about peak brightness of the device.

Tone Mapping Calibration for Output Devices

- Tone mapping uses this parameter to adapt rendered images to output devices
  - Want to output optimal images adapted to the dynamic range of the output device



Tone Mapping for
Peak brightness: 1,000 nits



Tone Mapping for
Peak brightness: 10,000 nits

The reason we need peak brightness is because tone mapping uses this parameter to adapt rendered images to output devices.

We want to output optimal images adapted to the dynamic range of the output device.
The detail has been already explained in this course.

Calibrations

- Want to know a peak brightness, so measure!
- Recent games often have their own calibration processes
- When starting games for the first time, users need to calibrate their screens

If you want to know something about devices, you should measure it. Recent games often have their own calibration processes. When starting games for the first time, users need to calibrate their screens.

Our Calibration Process (1/2)

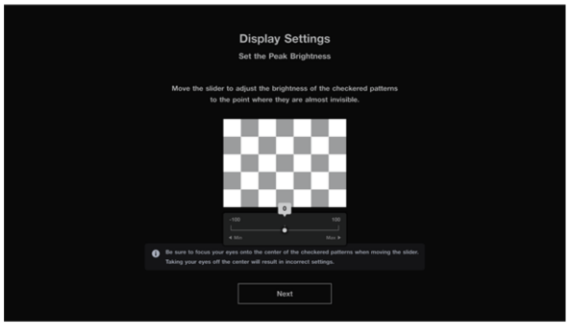
- First, make sure users are facing straight at the device
  - Particularly in HDR devices, viewing angle affects the visual appearance greatly

Next we will explain our own calibration process.

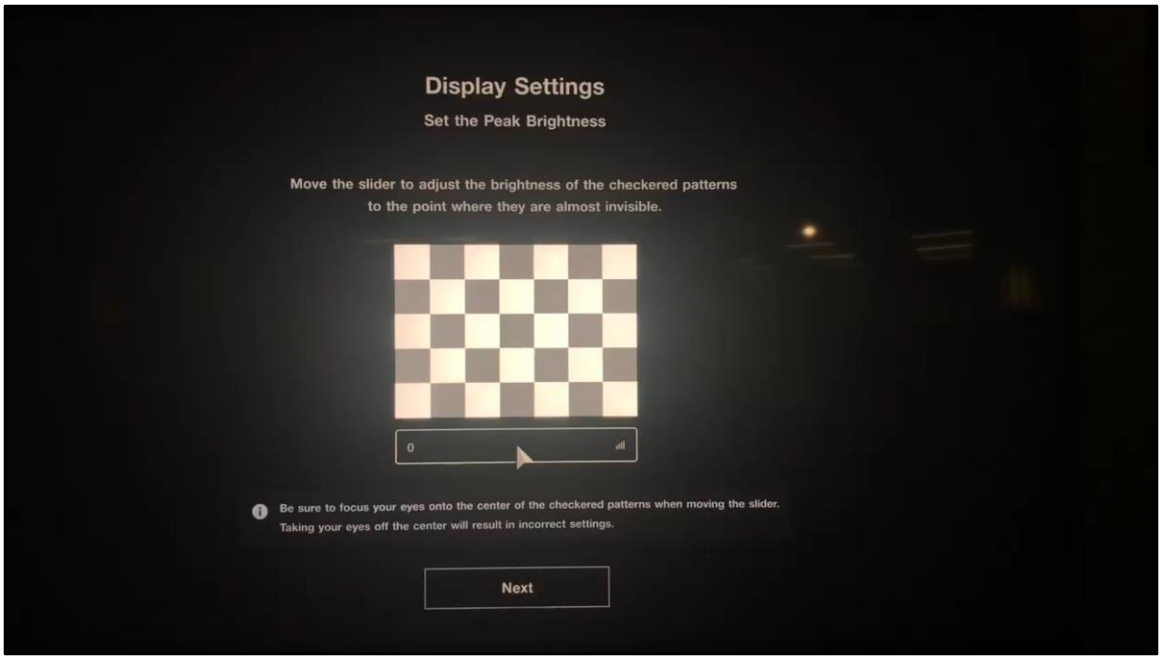
First, make sure users are facing straight at the device.
In HDR devices, viewing angle affects the visual appearance greatly, so this process is important.

Our Calibration Process (2/2)

- Second, move the slider to adjust the brightness of the checker board patterns to the point where they are almost invisible



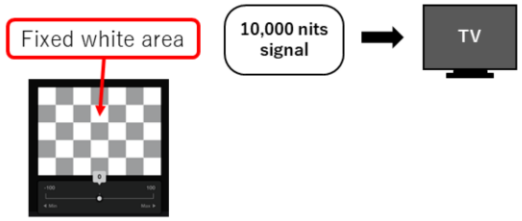
Second, move the slider to adjust the brightness of the checker board patterns to the point where they are almost invisible.



This is our calibration process movie.

How Can This Estimate the Peak Brightness?

- In the fixed white area
  - The input signal at the device is **10,000 nits** so the device tries to output **10,000 nits pixels**

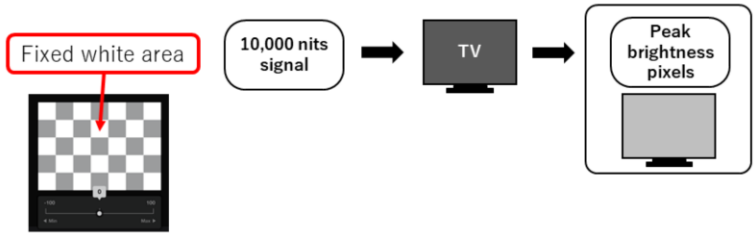


Then, how can this process estimate the peak brightness? Let us explain.

In the fixed white area, the input signal at the device is 10,000 nits so the device tries to output 10,000 nits pixels.

How Can This Estimate the Peak Brightness?

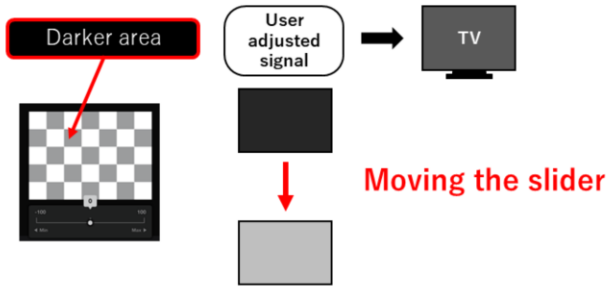
- In the fixed white area
  - However, the typical device can't output such bright pixels, so this is actually the output device's peak brightness



However, the typical device can't output such bright pixels, so the device will actually output its peak brightness.

How Can This Estimate the Peak Brightness?

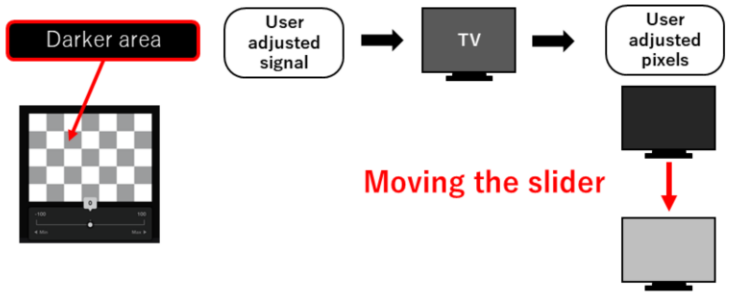
- In the darker area
  - When moving the slider, the input signal is adjusted and brightened



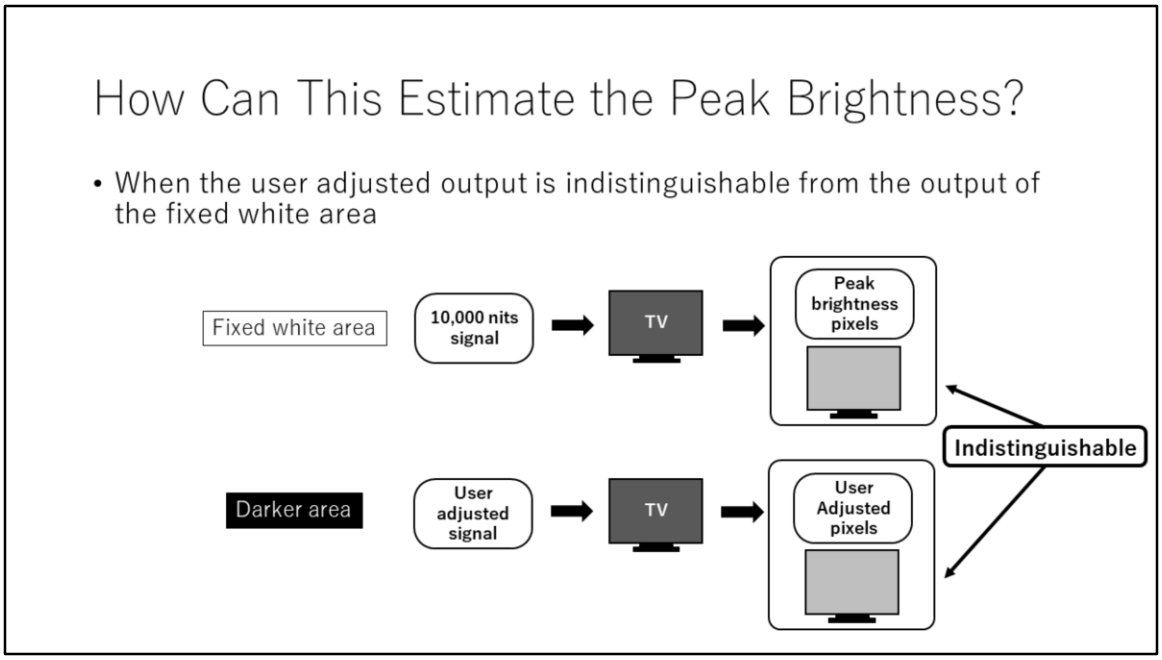
On the other hand, in the darker area, when moving the slider, the input signal is adjusted and brightened.

How Can This Estimate the Peak Brightness?

- In the darker area
  - The output pixels also gets brighter



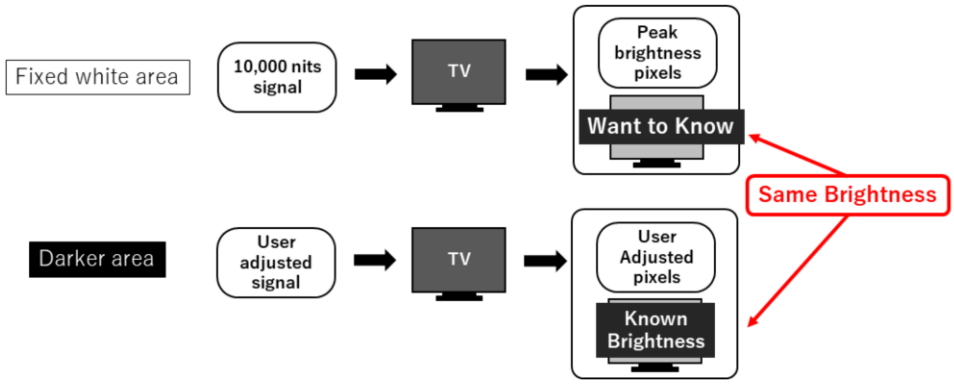
Then, the output pixels also become brighter.



When the user adjusted output is indistinguishable from the output of the fixed white area, the peak brightness can be estimated.

How Can This Estimate the Peak Brightness?

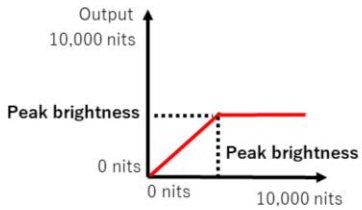
- When the user adjusted output is indistinguishable from the output of the fixed white area



If the output is indistinguishable, it can be thought that both area have the same brightness.
On the other hand, the brightness of the adjusted darker area is obviously a known value, so we can conclude that the device's peak brightness is the same as the adjusted brightness.

Important Points (1)

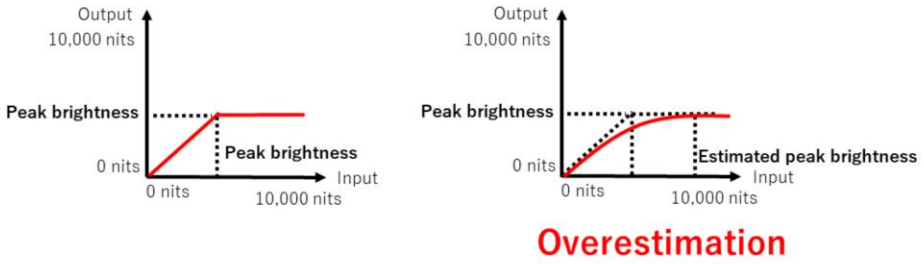
- This process assumes that devices directly map the input signal to the output brightness (the left figure)



There are however some important points to consider.
Our method assumes that devices map the input signal directly to the output brightness. The figure on the slide illustrates this assumption.

Important Points (1)

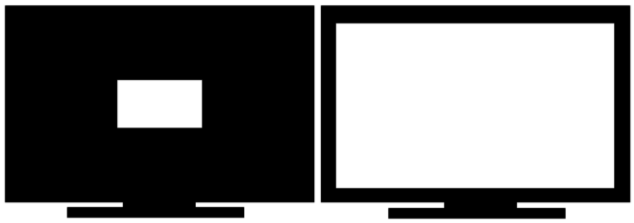
- This process assumes that devices directly map the input signal to the output brightness (the left figure)
- Actual behavior is often the right figure



However, an actual behavior is often the right figure.
In this case, so called display mapping is applied and an estimated peak brightness is overestimated.
But indeed the overestimation doesn't matter because it doesn't reduce the effective dynamic range, so our method works.

Important Points (2)

- This process also assumes that the peak brightness is always constant on the target device



The same input signal, the same output brightness

Also, this process assumes that the peak brightness is constant across all areas of the target device, and that the same input signal will output the same brightness.

Important Points (2)

- This process also assumes that the peak brightness is always constant on the target device
- Actual behavior depends on the entire output image
  - This happens because of each device's power management features[HGIG 2018]

The same input signal, **but the output can be different**

Animation

However, the actual behavior depends on the entire output image.

This behavior happens because of each device's power management features.
Whenever a bright area becomes wider, the power available will be distributed onto a wider area.
Because of that, the power per pixel will be reduced and pixel brightness will also be reduced.

Important Points (2)

- The peak brightness of the full screen is not necessary to estimate
  - It is not common to present a peak-brightness picture on the full screen
- This is why we use the checker board pattern on a part of the screen and not full screen

In our case, the peak brightness of the full screen is not necessary to estimate.
And this is why we use a checker board pattern on a part of the screen and not full screen.

Results

- Comparison between measured brightness and estimated value

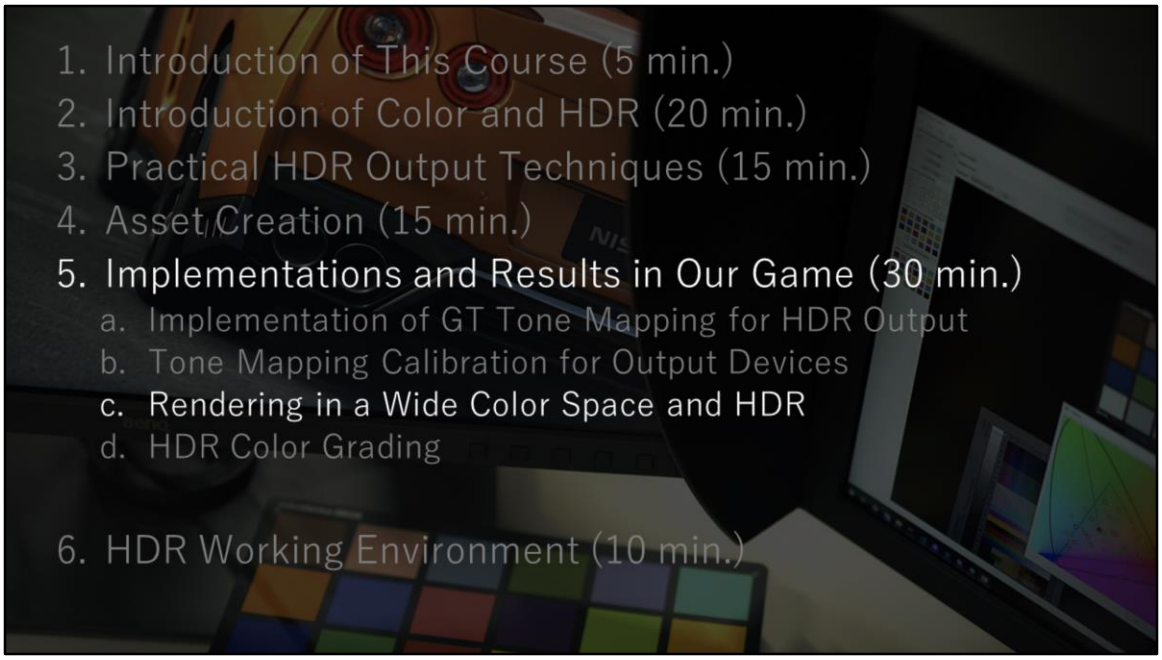
	Measured Brightness (nits)	Estimated Value (nits)
Mid-end TV	400	1100
High-end TV	1200	2000
Master monitor	410	470
Mid-end OLED TV	700	4300

This was measured using a luminance meter, but these values are not very accurate.
These values depends on devices and the measurement settings and are here as a reference

This slides shows some comparison results between measured brightness and estimated value after calibration.

Our method overestimate the peak brightness in ordinary TVs, but works very well in master monitors.
It's because the behavior of a master monitor is very simple.

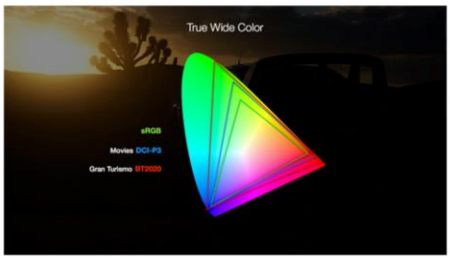
In a middle-end OLED TV, estimated value is much higher than the measured value.
Because their target can be thought of Dolby Vision, the estimated value is near 4,000 nits. (In Dolby Vision, the maximum brightness is 4,000 nits.)



Next, we will talk about rendering in a wide color space and HDR.

Rendering in a Wide Color Space and HDR

- For HDR rendering, we render frames in a wide color space and apply inverse PQ curve as OETF
  - Wide color rendering
  - Inverse PQ curve



For HDR rendering, we render frames in a wide color space and apply inverse PQ curve such as OETF.
So we will focus on wide color rendering and inverse PQ curve in this section.

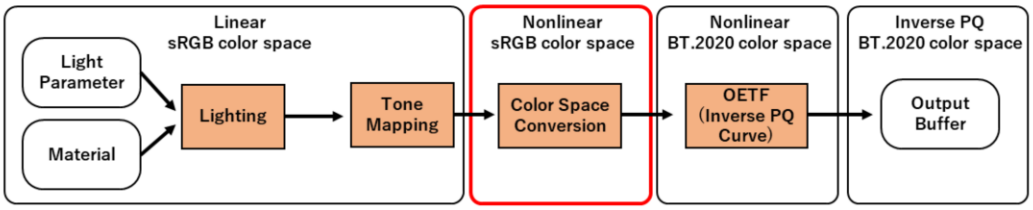
“HDR” Rendering

- Recently, “HDR” rendering has become the means to generating making output for HDR ready devices
  - In the past, we used to render on a linear space and a high dynamic range

Recently, HDR rendering has become the means to generating output for HDR ready devices.
In the past, HDR rendering we used to render on a high dynamic range linear space.

Conservative Approach for HDR Rendering

- Rendering in a narrow, conventional sRGB color space, then convert them for the wide, BT.2020 color space to output
  - Every sRGB color can be represented in the BT.2020
  - Conversion is very simple and easy



Let us begin with the conservative conservative approach for HDR rendering.
The bottom figure shows this approach.

In this approach, we will render in a narrow, sRGB-like color space, then convert them for wide, BT.2020 color space to output.

In the BT.2020, every sRGB color can be represented, so it's very simple and easy to implement the color space conversion, as we can just apply a conversion matrix.

Conservative Approach for HDR Rendering

- In this approach, the rendering process itself doesn't need to be changed a lot
  - No need to convert assets
  - Other assets may be easily expanded to new color space
    - E.g. particles, light sources

In this approach, the rendering process itself doesn't need to be changed a lot.

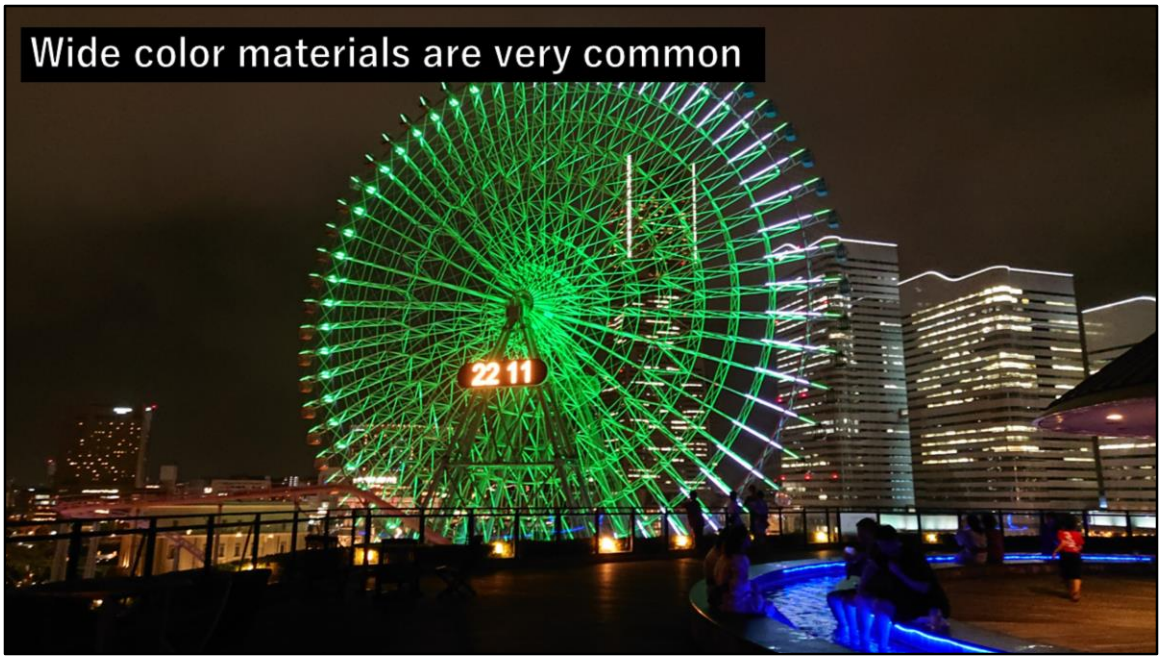
Every sRGB assets or texture can be used and doesn't need to be converted.
Some assets may be expanded to a wide color such as particles, lights.

Conservative Approach for HDR Rendering

- In this approach, the rendering process itself doesn't need to be changed a lot
  - No need to convert assets
  - Other assets may be easily expanded to new color space
    - E.g. particles, light sources
- **But, this approach can't make full use of a wide color and HDR output**
  - Limited color space assets
  - Limited rendering color space

However, this approach can't make full use of the wide color and HDR output because of limited assets and rendering color space.

Here are some examples of wide color materials in the real world.
Wide color materials are very common, so it is useful if we can render in a wider color range as well.

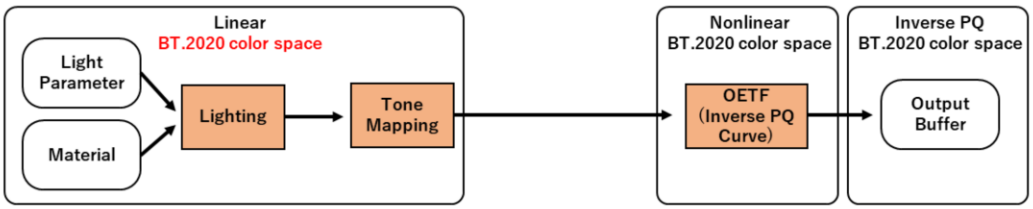


Like this.

And like this.

Our Approach for HDR Rendering

- Our approach for outputting to HDR devices is:
  - Rendering scenes in a high dynamic range, linear space
  - **Rendering scenes in a wide color space (BT.2020 in HDR10)**
    - Frame buffer is R11G11B10 (32bits/pixel) to reduce bandwidth



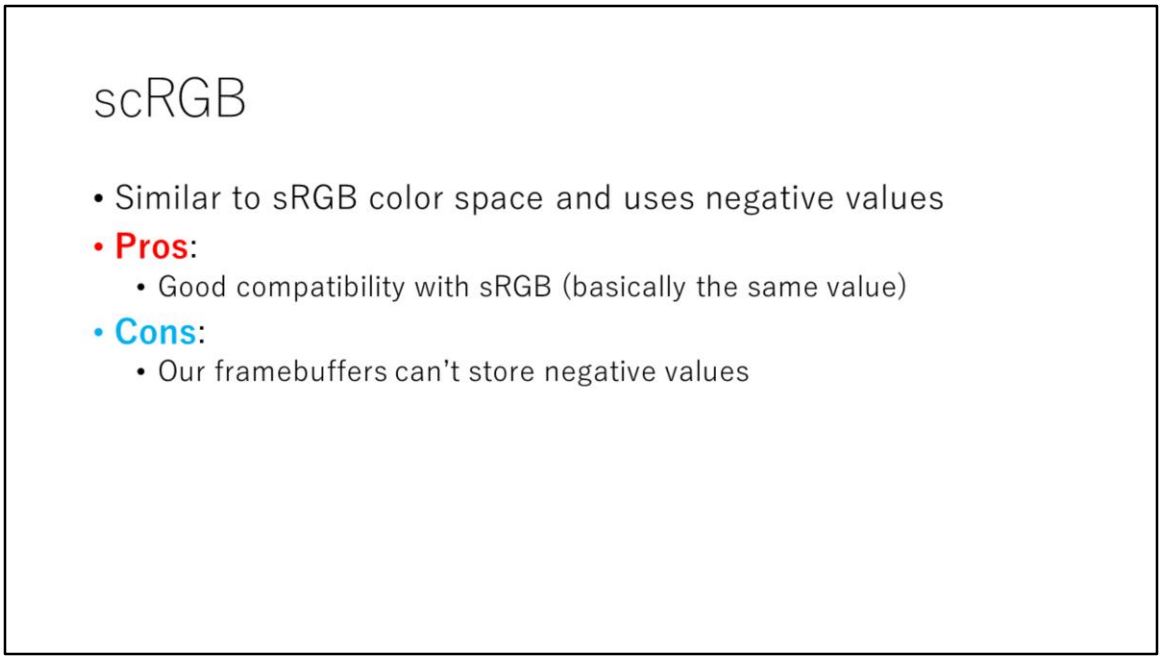
We will talk about our approach for HDR rendering.

In our approach, we render scenes in a high dynamic range, linear space.
And, we render scenes in a wide color space. Bt.2020.
Our frame buffer is RGB11 11 10 formats to reduce bandwidth.

Our Approach for HDR Rendering

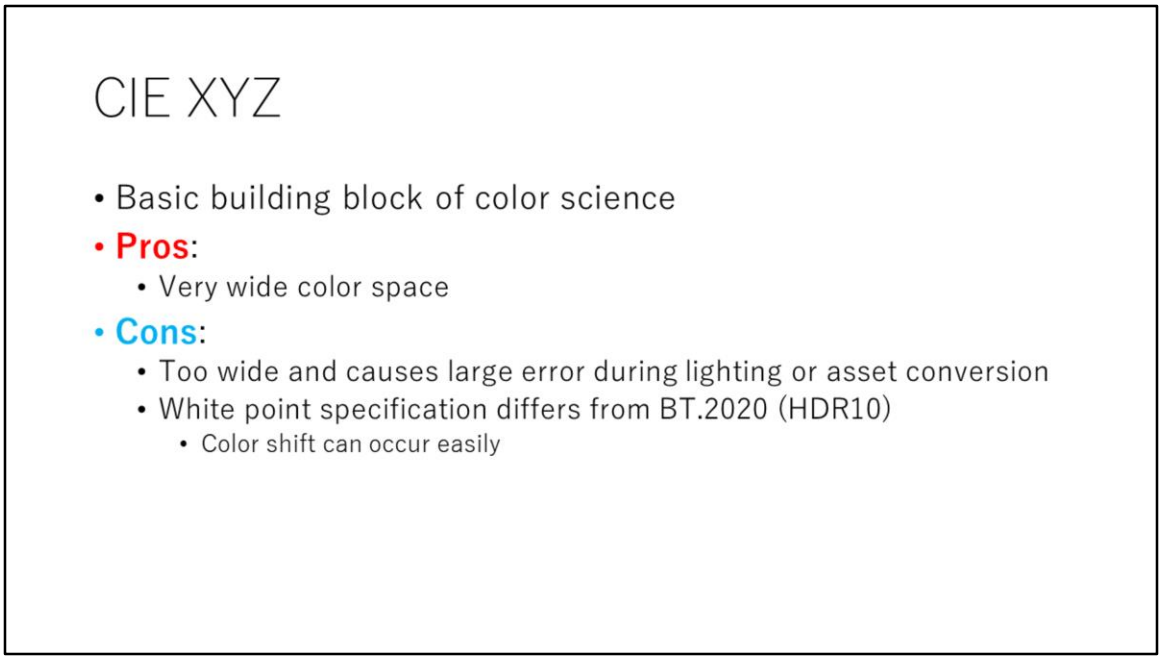
- Rendering color space candidates [Uchimura 2017]
  - scRGB
  - CIE XYZ
  - ACEScg (ACES computer graphics space)
  - BT.2020

We used BT.2020 but there were other candidates.
Next, we will talk about the reason we used BT.2020 instead of other color spaces.



scRGB is similar to sRGB color space and uses negative values.

This color space has good compatibility with sRGB, but our frame buffer can't store negative values, so we don't use scRGB.



CIE XYZ is a widely known color space.

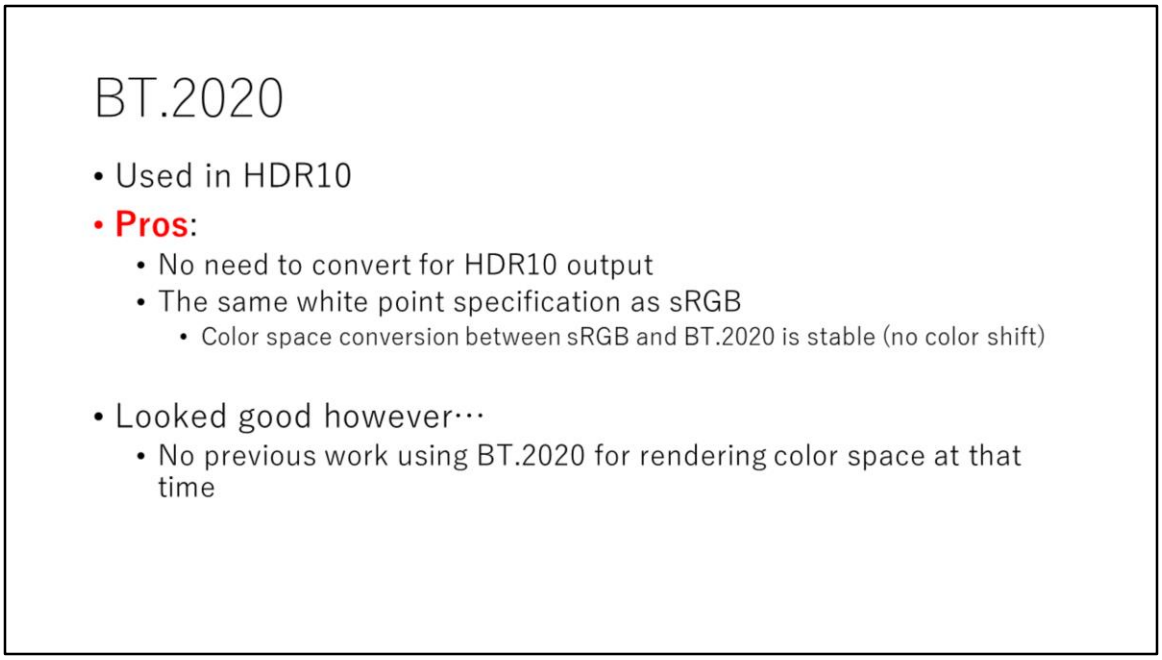
This color space is a basic building block for color science, and it's a very wide color space.
However CIE XYZ is too wide for us, and the error during lighting or asset conversion can be large.
In addition to that, white point specification differs from BT.2020, so color shifting can easily occur.
Because of that, we don't use CIE XYZ.

ACEScg

- Recently popular
- **Pros**:
  - Standard color transformation is well defined
    - RRT (Reference Rendering Transform)
      ODT (Output Device Transform)
  - Good portability
- **Cons**:
  - Many magic numbers exist in RRT and ODT
  - Some color grading processes are included in ODT
    - But we want a neutral one

ACES is has become popular recently.
In ACES color space, standard color transformation is well defined such as Reference Rendering Transform and Output Device Transform.
In addition to that, due to ACES popularity it has a good portability.

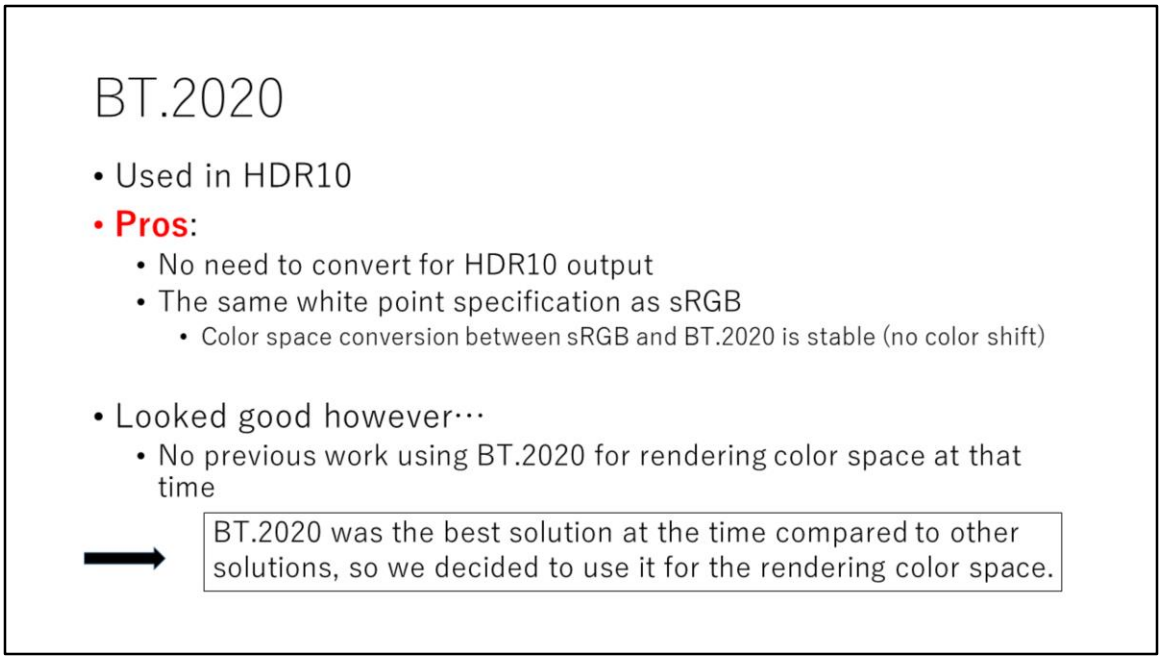
But in our opinion, many magic numbers exist in the RRT and ODT, and color grading processes are included in the ODT.
We want to use a neutral color space, so we don't use ACES.



BT.2020 is the color space used in HDR10.

So, there is no need to convert for HDR10 outputs.
And the white point in BT.2020 is the same as sRGB. Thanks to that, color space conversion between sRGB and BT.2020 is stable.

On the other hand, there was no previous work using BT.2020 for the rendering color space at the time.
We didn't have any knowledge.

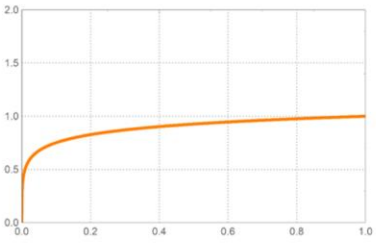


But, we thought BT.2020 was the best solution, so we decided to use BT.2020 for the rendering color space.

Inverse PQ Curve

- After rendering (in BT.2020), we need to apply the inverse PQ curve for HDR10 output
- Inverse PQ curve is described below (again)

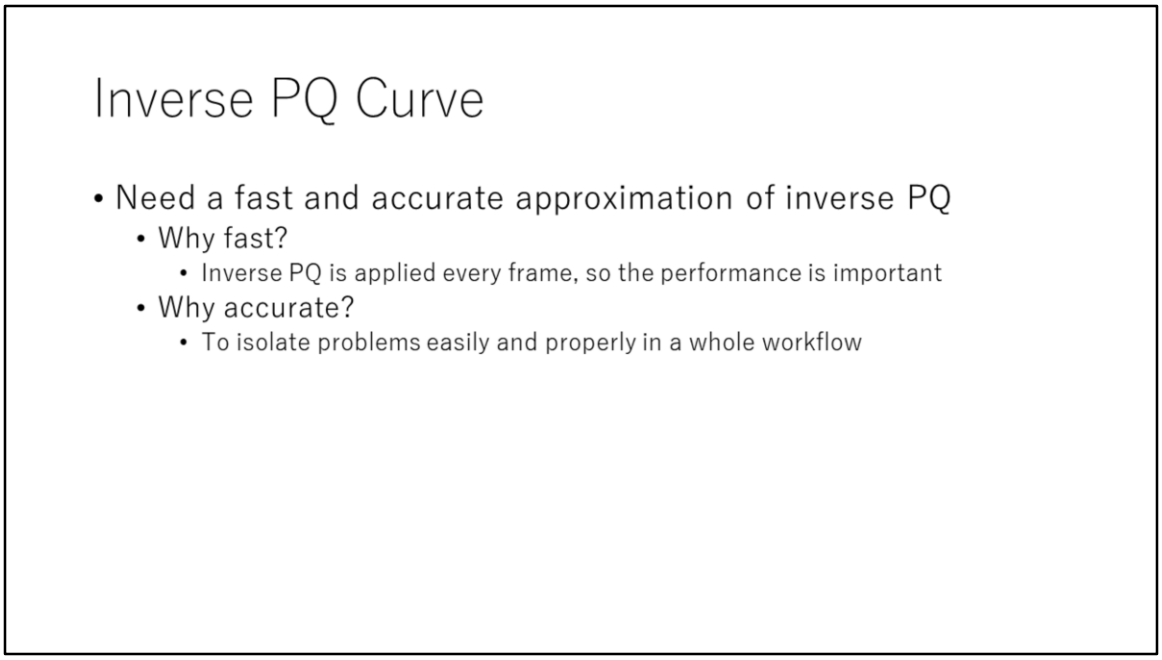
```
Float InversePQ(float3 input_RGB)
{
  const float m1 = 0.1593017578125;
  const float m2 = 78.84375;
  const float c1 = 0.8359375;
  const float c2 = 18.8515625;
  const float c3 = 18.6875;

  float3 ym = pow(input_RGB, m1);
  return pow((c1 + c2 * ym)/(1 + c3 * ym), m2);
}
```



[0, 10000] nits is mapped into [0, 1] for the function input

Next we will talk about the inverse PQ curve.

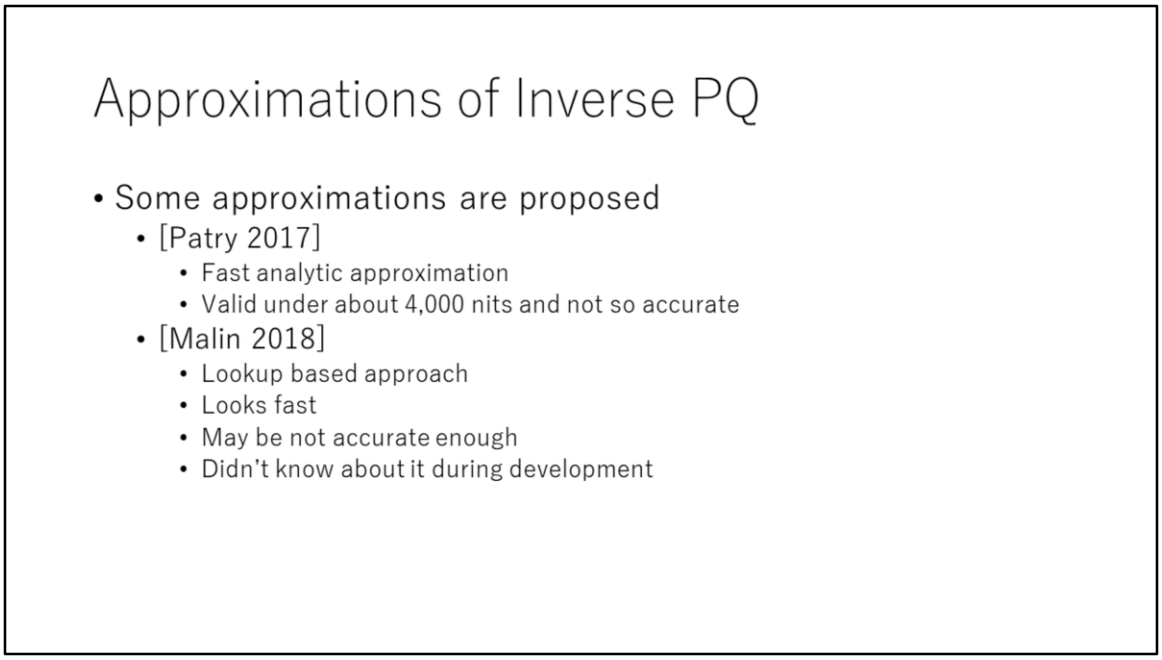
After rendering scenes and tone mapping, we need to apply the inverse PQ curve to the HDR10 output.
Inverse PQ curve is shown below.



For runtime, we need a fast and accurate approximation of inverse PQ curve.

Why fast?
It's because inverse PQ curve is applied every frame, so we want to reduce computational costs as much as possible.

Why accurate?
We wanted to use functions following the standard as much as possible.
We wanted to isolate problems easily and properly in whole workflow



Some useful approximations are already proposed.

[Patry 2017] is a fast analytic approximation but it is only valid under about 4,000 nits. And the accuracy is not so good.
[Malin 2018] is a lookup based approach and it looks fast, but we believe it is not accurate enough. We didn't know about it during development, so we couldn't use it.

Approximations of Inverse PQ

- Our approximation
  - Fitted by Mathematica

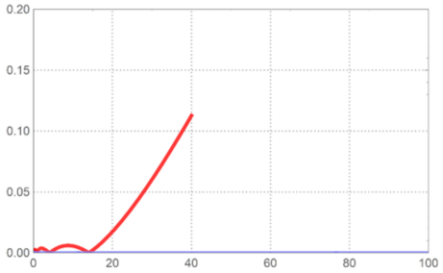
```
float InversePQ_approx(float x)
{
    float k = pow((x * 0.01f), 0.1593017578125);
    return
        (3.61972*(1e-8) + k * (0.00102859 + k * (-0.101284 + 2.05784 * k))) /
        (0.0495245 + k * (0.135214 + k * (0.772669 + k)));
}
```

In advance, [0, 10000] nits is mapped into [0, 100] for the function input

And this is our approximation.
First, we introduce a pow term, then we used Mahetmatica for fitting after that.

Approximations of Inverse PQ

- Comparison of the accuracy to [Patry 2017]
  - Desmos graph: https://www.desmos.com/calculator/0n402k2syc
    - Absolute error to the original inverse PQ
  - The bright input part of our approximation is much more accurate



Purple line: our approximation
Red line: [Patry 2017]

This is a absolute error comparison between our approximation and [Patry 2017].
The purple line is our approximation and the red line is [Patry 2017].
As you can see, in our approximation, the bright input part is much more accurate.

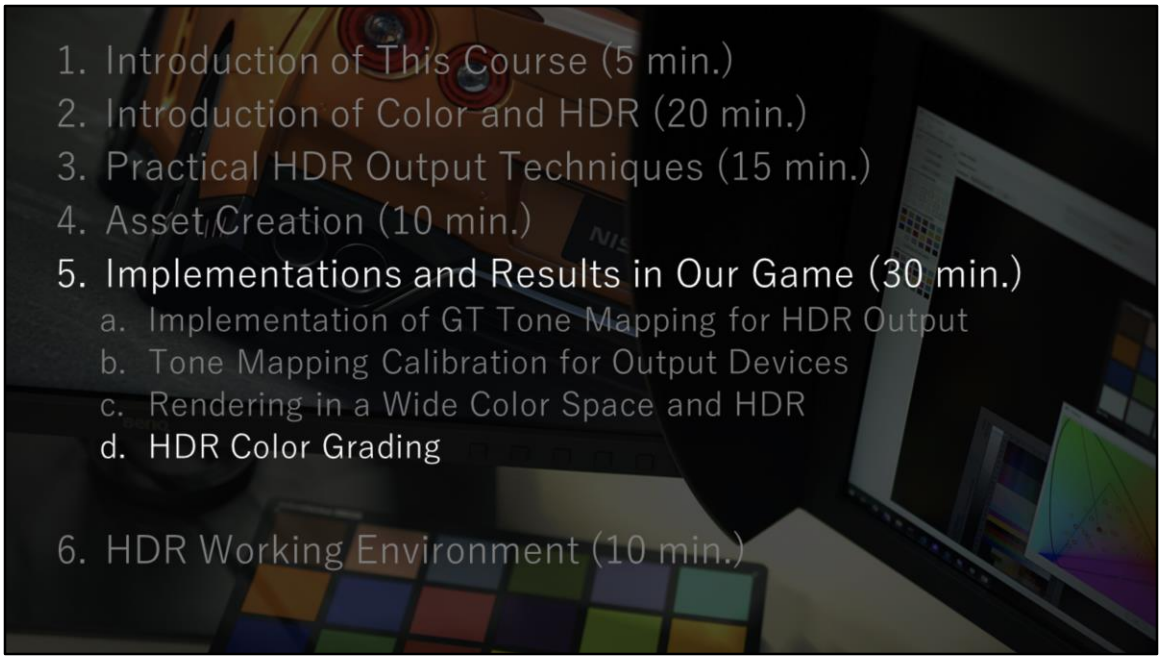
Performance Results

- PlayStation®4, 1920x1080 (includes tone mapping)

Reference	0.36 ms
Our Approximation	**0.33 ms (10% faster than reference)**
[Patry 2017]	0.29 ms

Finally, we will talk about performance.

The reference implementation takes 0.36 ms on PlayStation®4, full HD.
And our approximation takes 0.33ms which is 10% faster than the reference.
[Patry 2017] is faster but not so accurate, so we used our approximation.



Next, we will talk about HDR color grading in Gran Turismo SPORT.

Color Grading

- Color grading is the process of modifying input images in various ways to enhance final look
  - Brightness
  - Contrast
  - Hue
  - Etc
- Originally from the movie industry
  - In the past: directly applied to the film itself
    - E.g. bleach bypass
  - Today: digital processing workflow

Color grading is the process of modifying input images in various ways to enhance final look.
Some of the modifications include brightness, contrast, hue, and so on.

In the past, color grading was directly applied to the film itself such as bleach bypass.
Today, we can use a digital processing workflow, so can apply various effects much more easily.

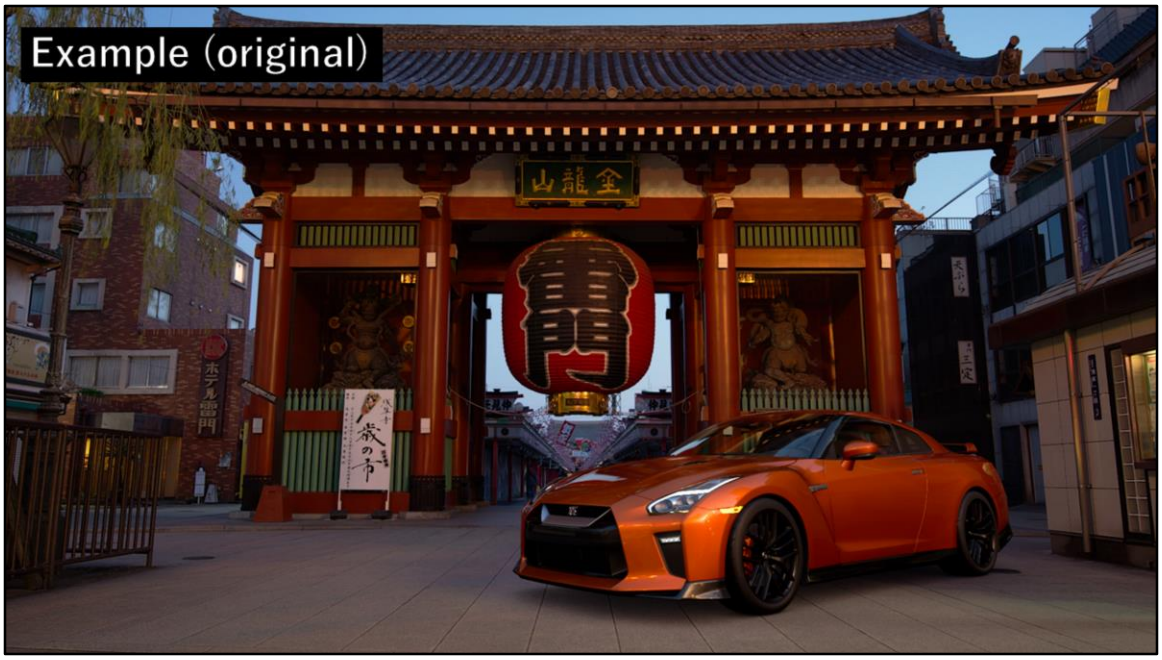
Color Grading in Gran Turismo SPORT

- Artists can enhance appearances in a cut scene
  - For attracting users with impressive images
- Users can edit in-game taken photos by in-game color grading tools
  - Race photo mode
  - Scapes photo mode

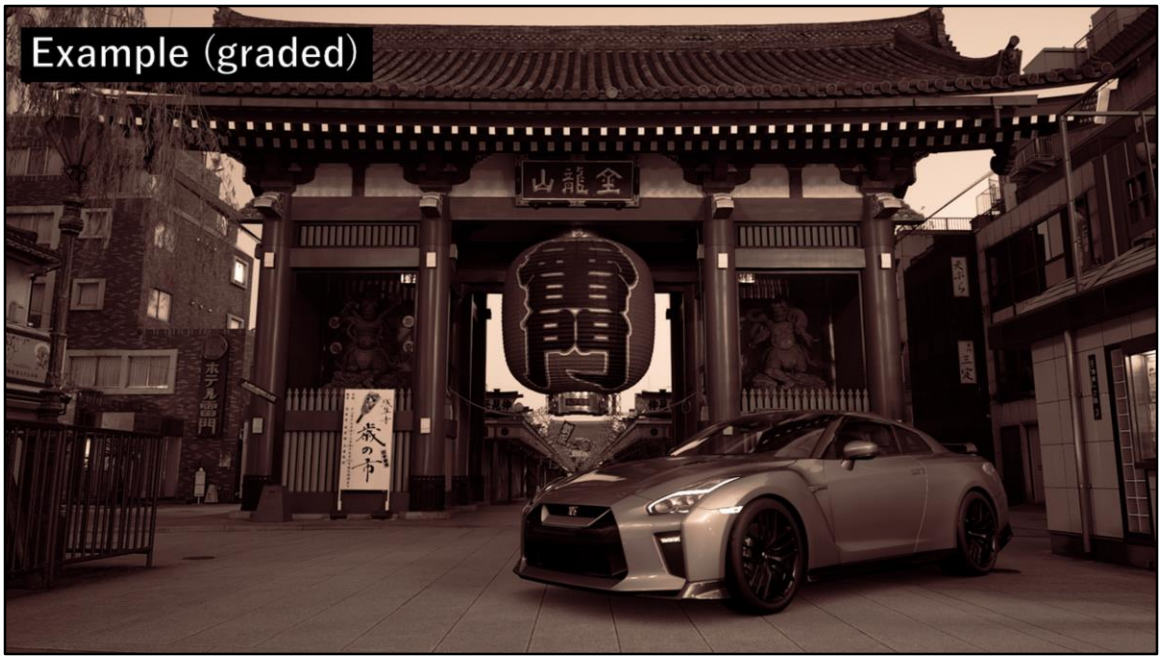
In Gran Turismo SPORT, there are two purposes for color grading.

The first is for artists.
Artists can enhance appearances in cut scene for attracting users with impressive images.

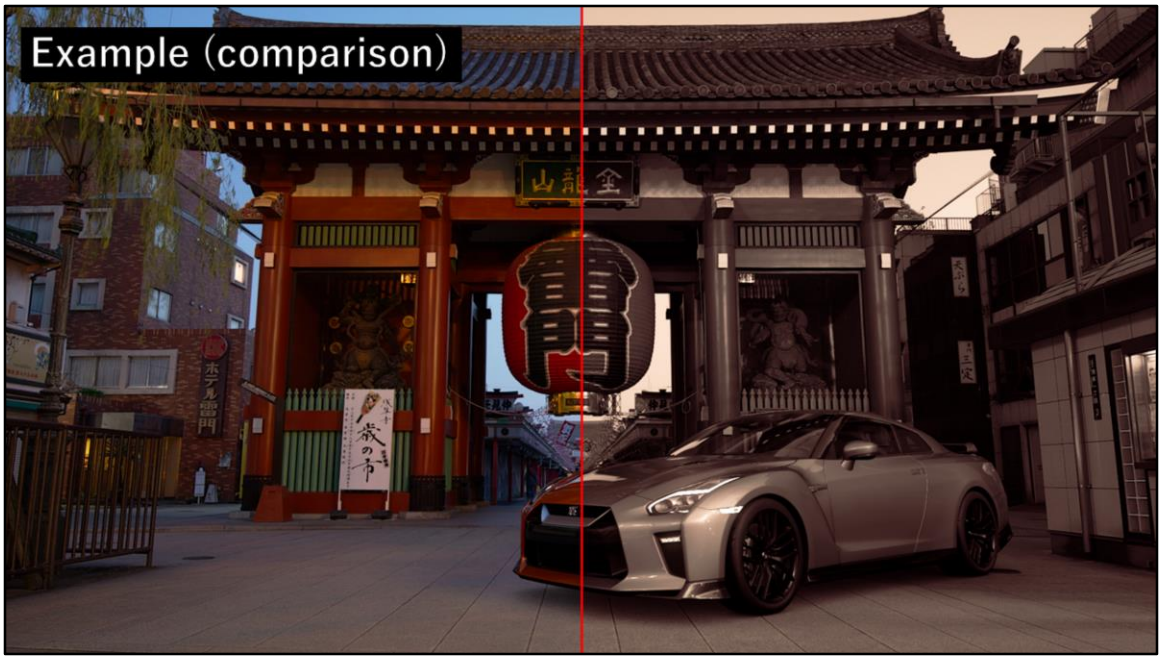
The second is for users.
Users can edit in-game taken photos by in-game color grading tools.



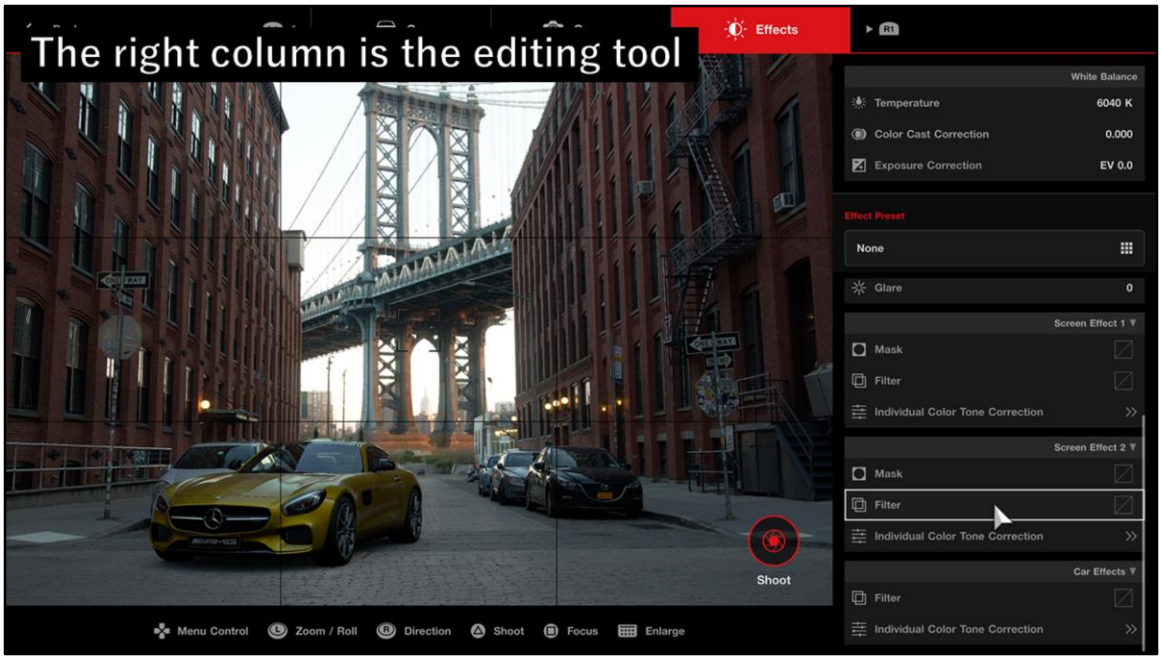
We will show an example of color grading.
This is the original image.



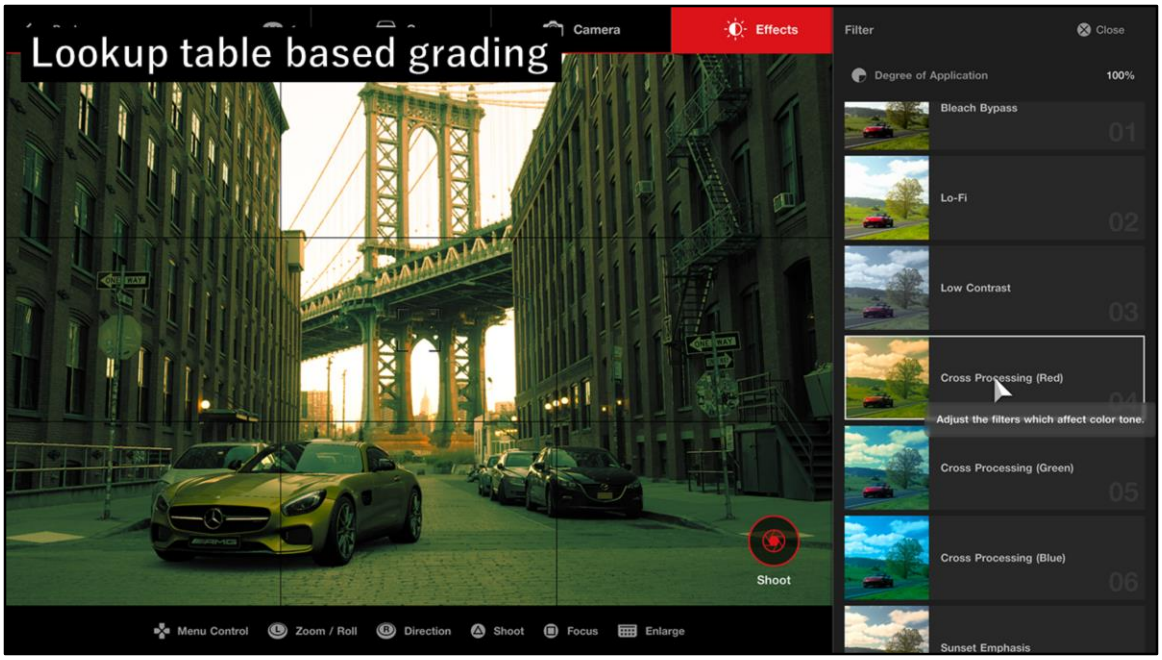
This is the color graded image.



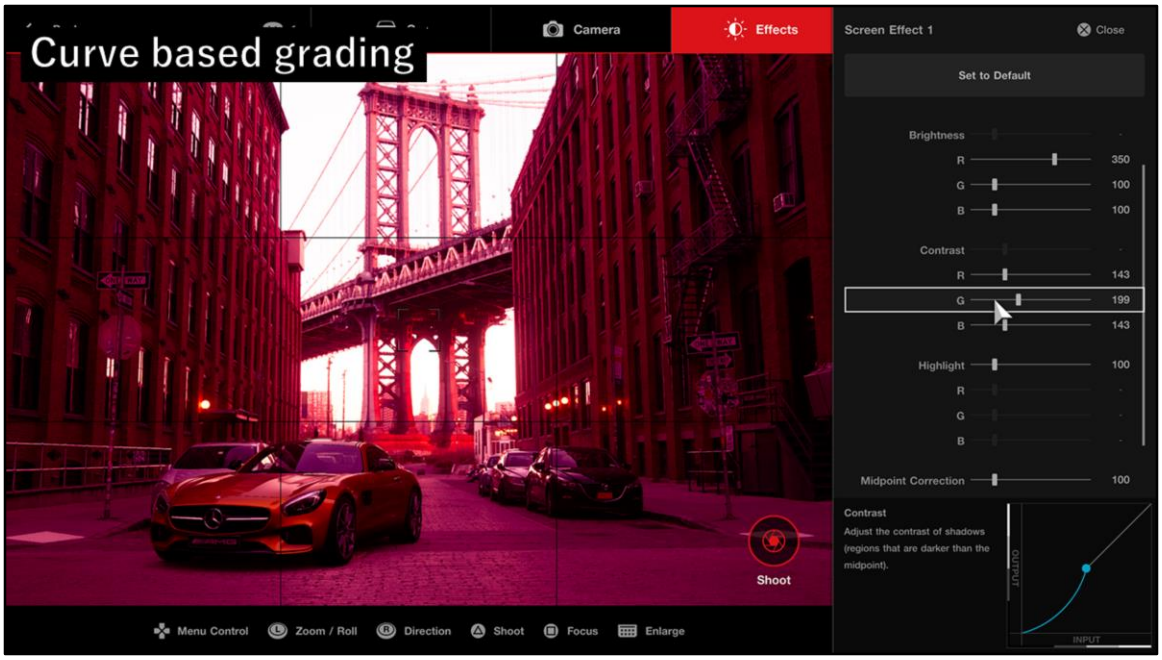
And this is a side-by-side comparison.



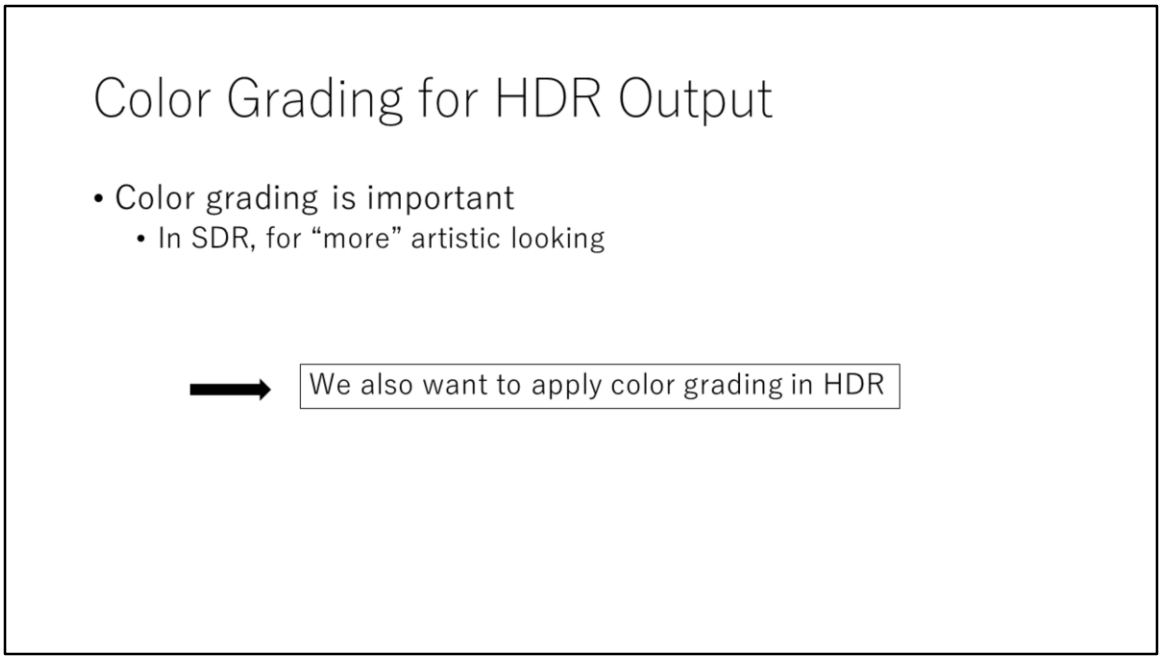
And this is our in-game color grading tool.
The right column is the editing tool.



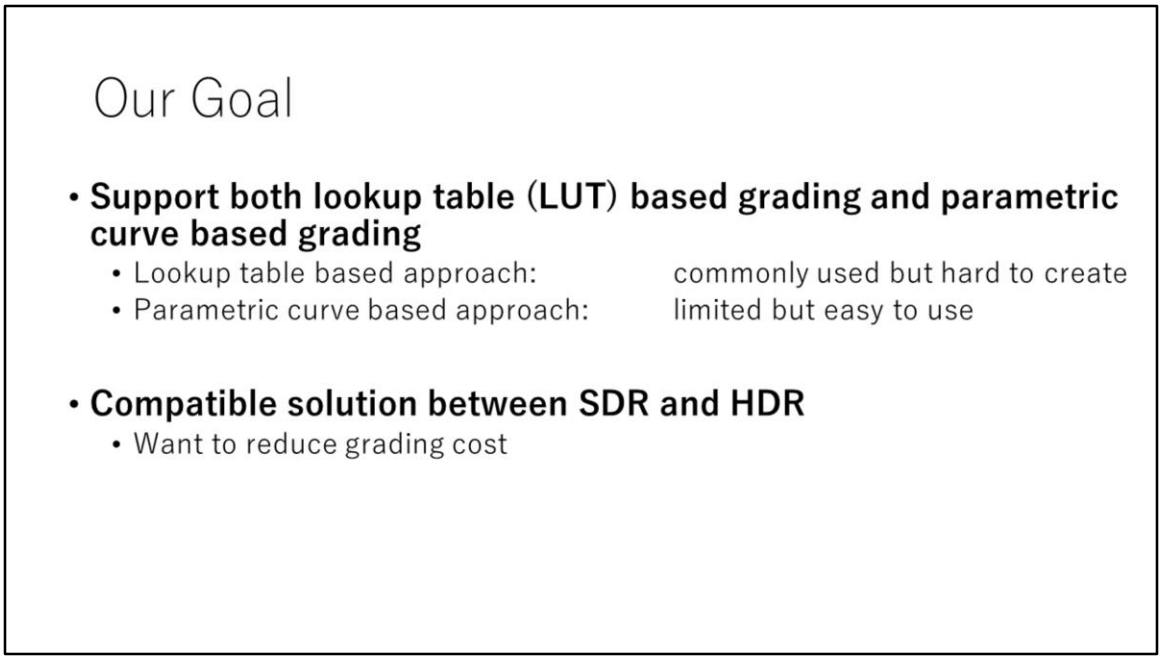
User can use lookup table color grading filters.



And curve based, parametric color grading.

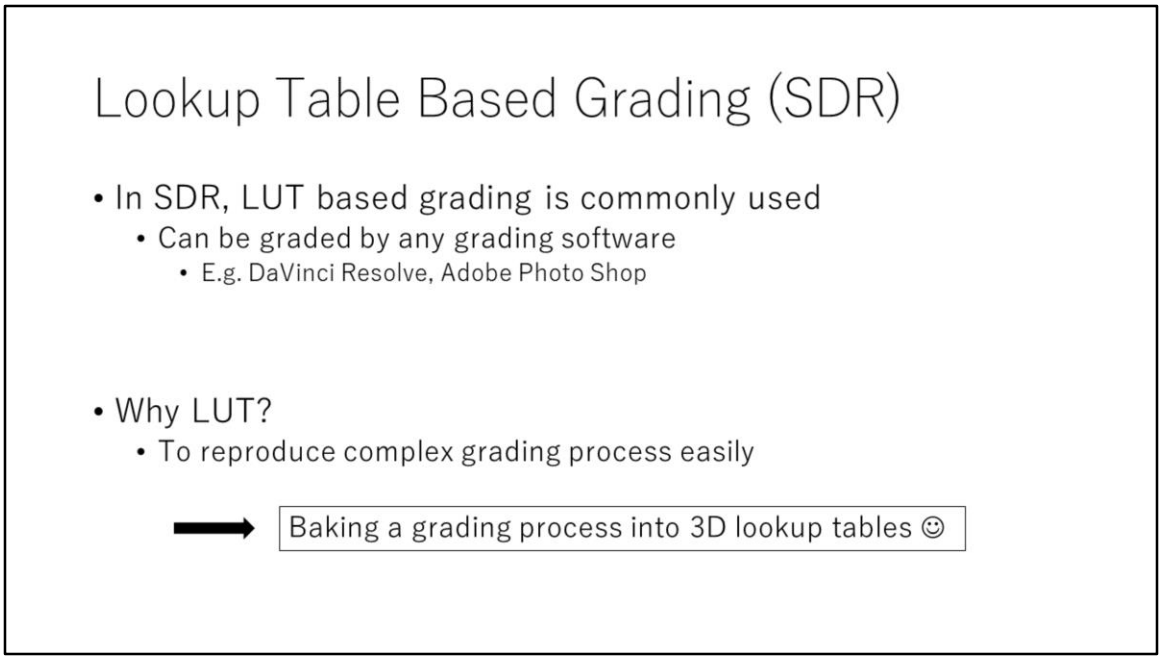


In SDR, color grading is important for a “more” artistic looking. And of course, we also want to apply color grading in HDR.



So, we had two goals.
The first was support both lookup table based color grading and parametric based grading.
And the second was to support a compatible solution between SDR and HDR.

Lookup table based grading is commonly used but hard to create and curve based grading is limited but easy to use.
And we want to reduce the grading cost.



First, we will talk about a general method of lookup based grading in SDR.

In SDR, lookup based grading is commonly used.
Grading itself can be authorized by any grading software such as DaVinci Resolve and Adobe Photoshop.

Then, why lookup table?
We wanted to reproduce certain grading processes easily.
Grading processes can be very complex, but by baking a grading process into lookup tables, we can easily apply the same color grading.

Lookup Table Based Grading (SDR)

- Using 3D LUT, arbitrary grading functions can be approximated in a piecewise-linear manner

$$(R_{graded}, G_{graded}, B_{graded}) = f_{grading}(R, G, B)$$

```
float3 graded_color = LUT3D.Sample(trilinearSampler, input_color).rgb;
```

Computing original function can be approximated by single 3D texture fetching

As you know, using three dimensional lookup table, arbitrary grading functions can be approximated in a piece-wise linear manner.
For that purpose, graded RGB values are stored into a LUT texture, and input RGB values are interpreted as positions on the LUT texture.

By this process, computing original function can be approximated by single 3D texture fetching.

Lookup Table Based Grading (SDR)

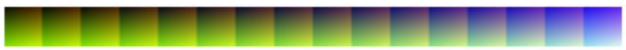
- SDR 3D LUT is simple and efficient

Resolutions	16x16x16 is enough
Texture bit-depth	8 bit and is compact ☺

- Examples

Neutral LUT (No-operation)

Monotone LUT



Something complex

In SDR, 3D lookup table is very simple and efficient.
The resolutions don't need to be so large. 16 by 16 by 16 is enough.
And the texture bit-depth is 8 bit and it's compact.

These figures are examples of SDR lookup table.

Lookup Table Based Grading (HDR)

- The main difference is an input and output format

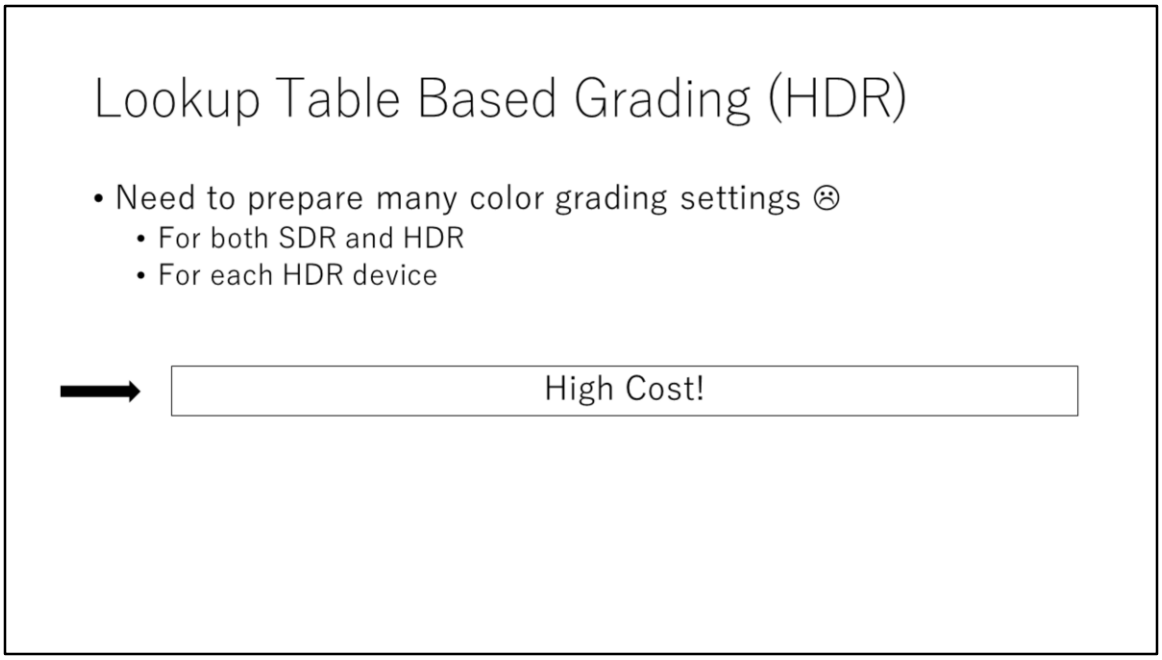
	Color Space	OETF
SDR	sRGB	sRGB gamma
HDR	BT.2020	Inverse PQ curve

- Applying process is the same, but grading and creation process is different
  - Need to do color grading and create 3D LUT independently for HDR
  - Independently for each HDR device (each peak brightness)

In HDR, what can be solution?

Indeed, the main difference between SDR color grading and HDR color grading is an input and output format.
In SDR, color space is sRGB and OETF is also sRGB.
But in HDR, color space is BT.2020 and OETF is inverse PQ.

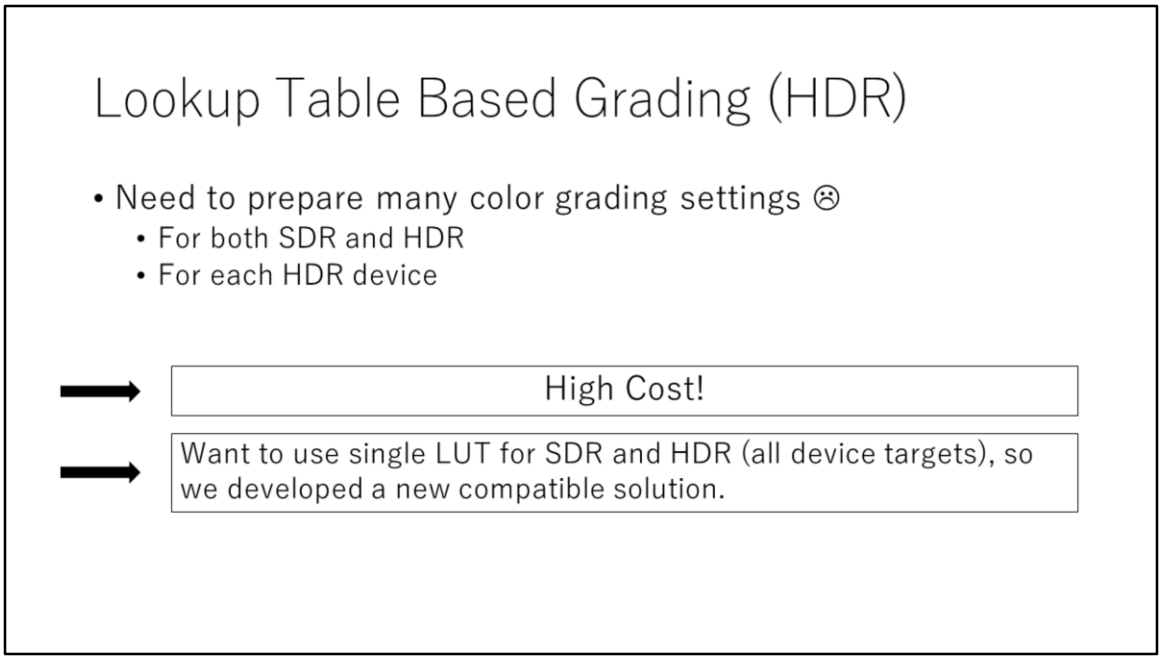
So applying process itself is the same but grading process is different.
We need to do color grading and create 3D lookup table independently for HDR.
In addition to that, for each HDR device and each device peak brightness.



That is why, we need to prepare many color grading settings, for both SDR and HDR and for each HDR device.

It is hard to maintain a similar appearance in both SDR and HDR manually

This has a high cost.



So we want to use a single lookup table for SDR and HDR and for all devices.
For that purpose we developed our new compatible solution.

SDR/HDR Compatible LUT Grading

- Once color grading is done (and make LUT), use it everywhere
  - The grading target is set for the most generic environment

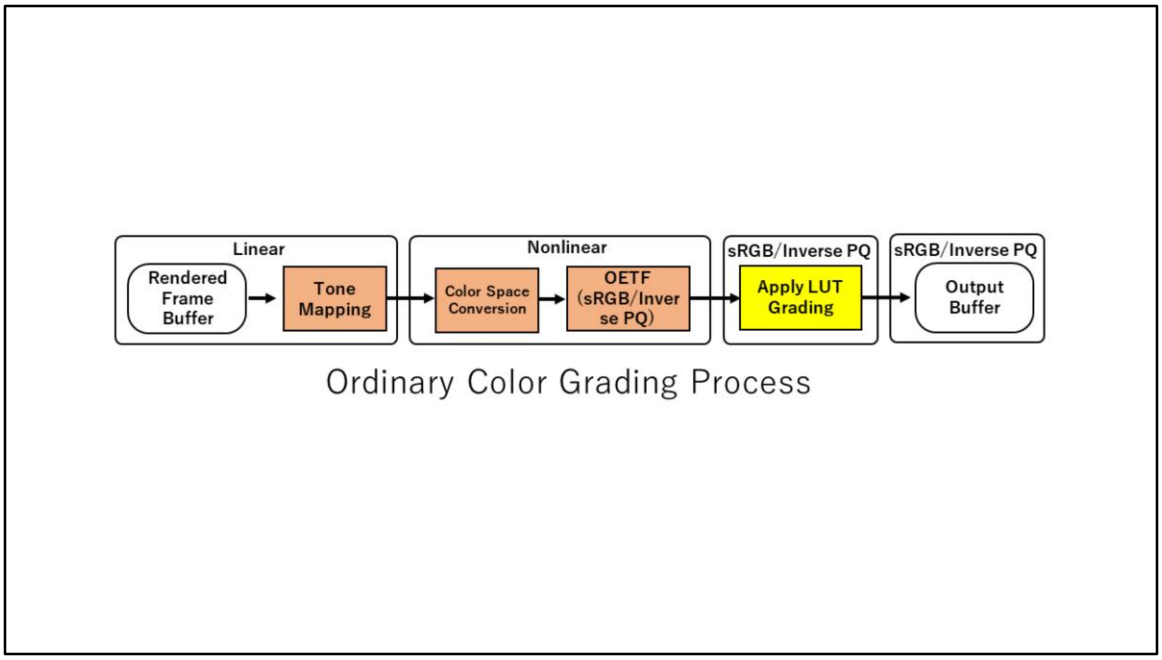
Color Space	BT.2020
OETF	Inverse PQ curve
Target nits	10,000 nits

- Software
  - DaVinci Resolve
    - de facto standard in the movie and game industry
  - For preview purposes, apply an appropriate display LUT

Next, we will talk about SDR and HDR compatible lookup table grading method.

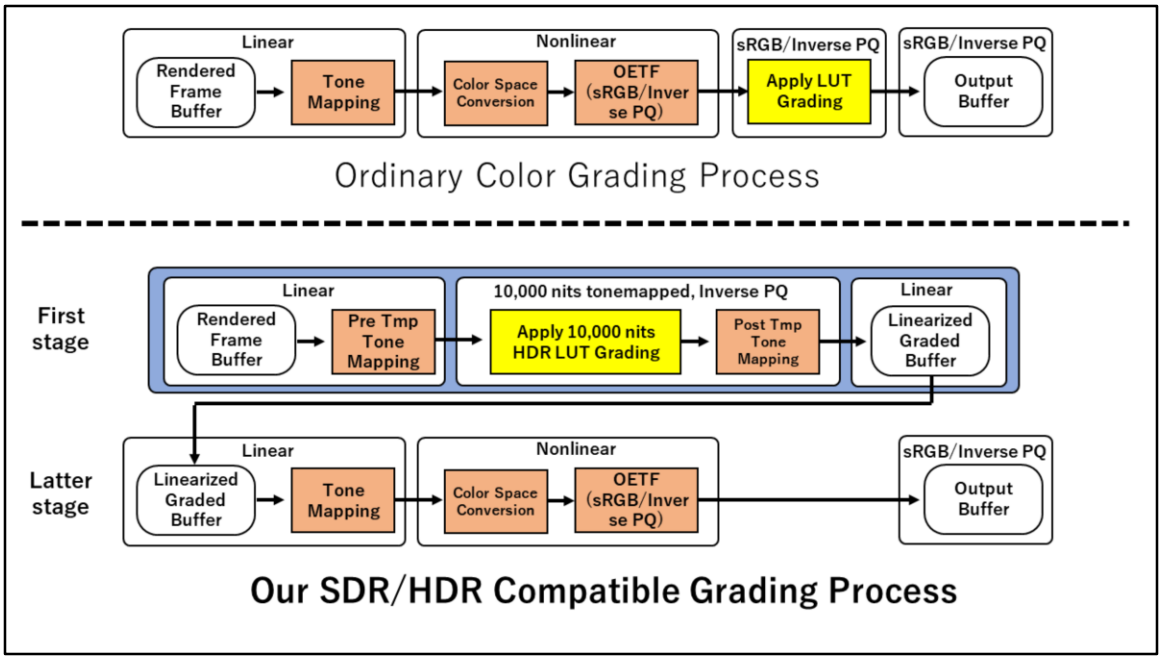
In our method, once color grading is done, we can use it everywhere.
For that purpose, the grading target is set for the most generic environment.
The color space is BT.2020, OETF is Inverse PQ (ST.2084) and the target nits is 10,000 nits.

And we are using DaVinci Resolve for grading.
This is a de facto standard in the movie and game industry.
For preview, we apply an appropriate display lookup table for DaVinci Resolve.



We will explain about our compatible solution.

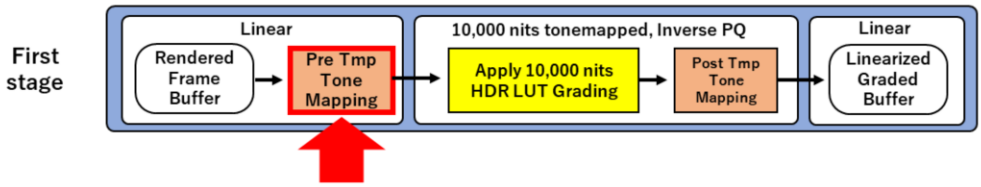
First, we will show an ordinary color grading process.
In this process, color grading is applied after OETF is applied.



And, this is an overview of our compatible grading process.
The latter stage is similar to the ordinary grading process, but the first stage is the key process.

Our SDR/HDR Compatible Grading Process

1. Pre temporal tone mapping is applied to a linear rendered buffer
   - Apply GT Tone Mapping (target 10,000nits) and then inverse PQ curve
   - The buffer is completely valid for 10,000 nits target HDR10 format



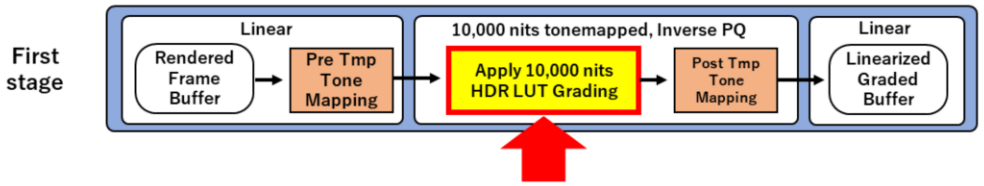
We will explain about the first stage more.

First, pre temporal tone mapping is applied to a linear, rendered buffer. We apply GT Tone Mapping and then inverse PQ curve in this process. The target of tone mapping is 10,000 nits.

At this point, the buffer is completely valid for 10,000 nits target HDR10 format.

Our SDR/HDR Compatible Grading Process

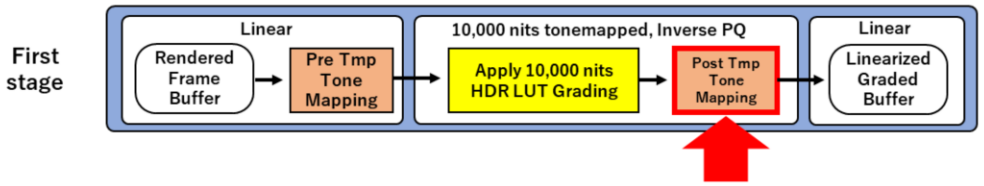
2. HDR 3D LUT (target 10,000 nits) is applied in an ordinary way
   - This process is valid because the input buffer's target and color space are the same as grading lookup table



Second, HDR 3D lookup table is applied in an ordinary way. This lookup table is for 10,000 nits.
This process is also valid because the input buffer's target and color space are the same as grading lookup table.

Our SDR/HDR Compatible Grading Process

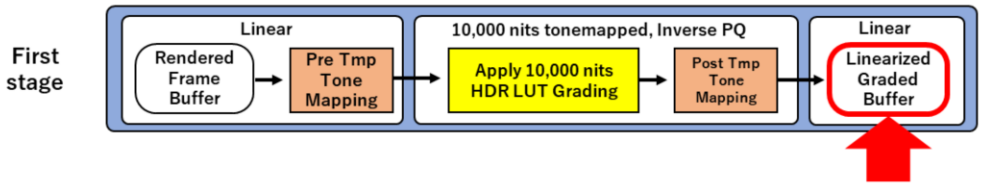
3. Post temporal tone mapping is applied
   - Apply PQ curve (not inverse PQ) and then inverse GT Tone Mapping to reconstruct a linear buffer



Third, post temporal tone mapping is applied.
We apply the PQ curve and then inverse GT tone mapping in this process to reconstruct a linear buffer.

Our SDR/HDR Compatible Grading Process

4. Finally, we can gain the linearized, color graded buffer

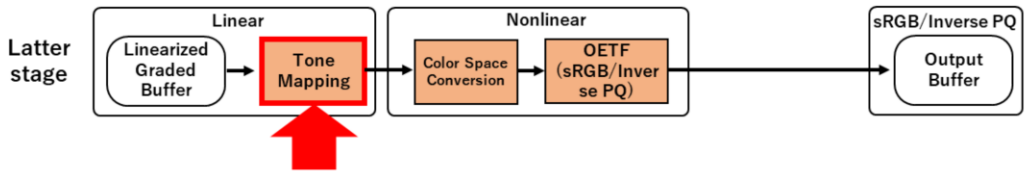


Finally, we can gain the linearized and color graded buffer.
Regardless of output devices, this process works so we achieve a compatible solution.

SDR/HDR Compatible LUT Grading

- After grading
  - Apply GT Tone Mapping to the linear, graded buffer

SDR	SDR tone mapping
HDR	HDR tone mapping for each output device



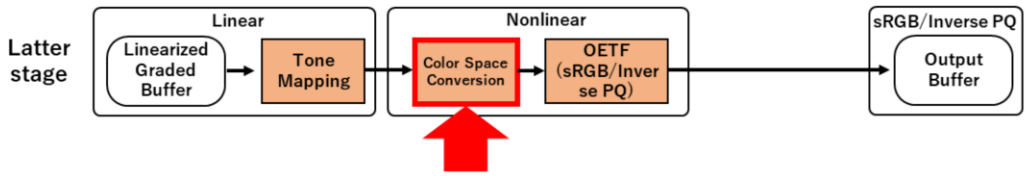
After grading, it's simple.

First, apply tone mapping to the linear graded buffer.
This tone mapping parameter varies for each output device as described before.

SDR/HDR Compatible LUT Grading

- After grading
  - Apply the color space conversion

SDR	BT.2020 color is converted to sRGB color
HDR	Nothing happens (the same color space in HDR10)

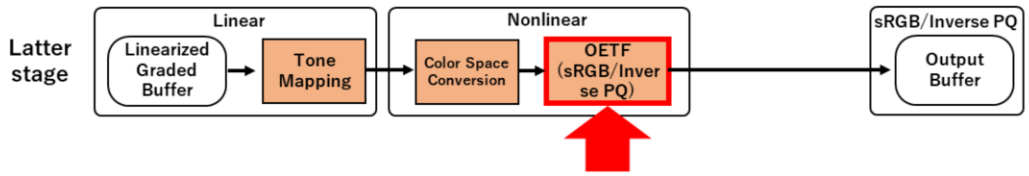


Second, apply color space conversion. In SDR, BT.2020 color is converted to sRGB color. In HDR, nothing happens.

SDR/HDR Compatible LUT Grading

- After grading
  - Apply OETF

SDR	sRGB gamma
HDR	Inverse PQ curve



Finally, apply OETF. In SDR, sRGB and in HDR, inverse PQ curve.

Here are some results of our compatible solution.
In this slide, the left is SDR and the right is the HDR output.
These are the same scene and images are directly captured from a TV.

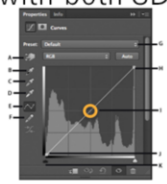
As you can see, in HDR, the dynamic range is preserved more than SDR.
In SDR, wider area is saturated because of its narrow range.

And these are images lookup table is applied.
As you can see, the whole impression looks similar, so our compatible solution works well.

off

Parametric Curve Based Grading

- LUT is great but often hard to create
  - Many methods to adjust
  - Not easy to learn grading process
- Simple, parametric curve based grading
  - Want a method like the “Curve” tool in Adobe Photoshop, but we need a tool compatible with both SDR and HDR.



https://helpx.adobe.com/jp/photoshop/using/curves-adjustment.html

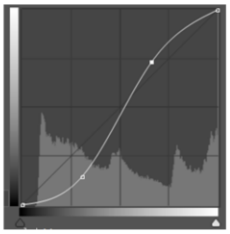
Next we will talk about parametric curve based grading.

Lookup table is great but is often hard to create, because there are many methods to adjust and it’s not easy to learn the grading process.

So, we need a simple, parametric curve based grading. We want a method like the “Curve” tool in Adobe Photoshop, but we also need it to be compatible with both SDR and HDR.

Parametric Curve Based Grading

- Requests from artists
  - Wanted to enhance contrasts easily!
- In SDR, “S” curve [Hable 2010] can be a solution but in HDR?
  - Apply directly in a linear space
  - Only dark part needs to be bended
  - Bright part can’t be bended
    - The range is not [0, 1]
    - Tone mapping will bend



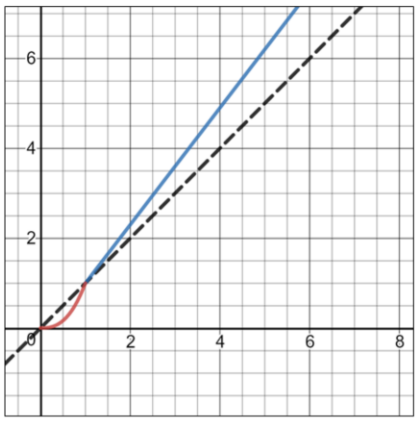
There was a request from artists. They want to enhance contrasts easily!
In SDR, “S” curve like filmic tone mapping can be a solution.

In HDR, we apply directly in a linear space and only the dark part needs to bended to enhance contrast.
The bright part can't be bended because the range is not between 0 and 1 and tone mapping will bend.

Parametric Curve Based Grading

- **Our solution**
  - The dark part is a simple gamma curve
  - The bright part is just a line!
  - 3 parameters
    - DarkPartGamma: p
    - MidPoint: m
    - BrightPartScaling: c

$$f(x) = \begin{cases} \dfrac{x^p}{m^{p-1}}, & 0 < x < m \\ c(x-m) + m, & m \le x \end{cases}$$

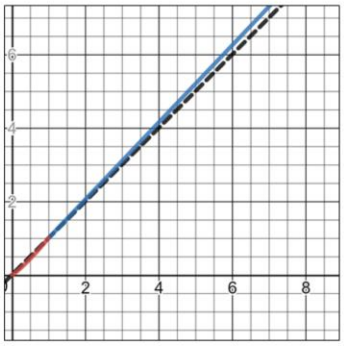


x is an individual component of RGB and f(x) is graded value

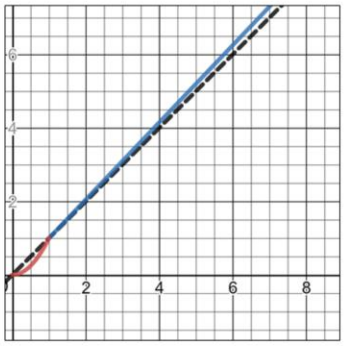
This is our solution.

The dark part is a simple gamma curve and the bright part is just a line and simple scaling function.
This curve has 3 parameters. DarkPartGamma, MidPoint controller, and BrightPartScaling.

Parametric Curve Based Grading

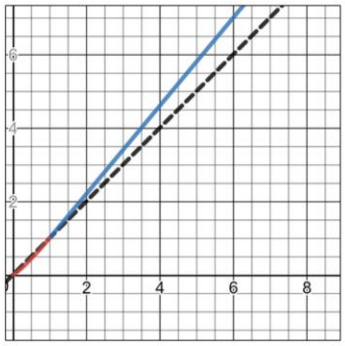


$p = 1.2, m = 1, c = 1.05$



$p = 2, m = 1, c = 1.05$

Strong emphasizing a dark part



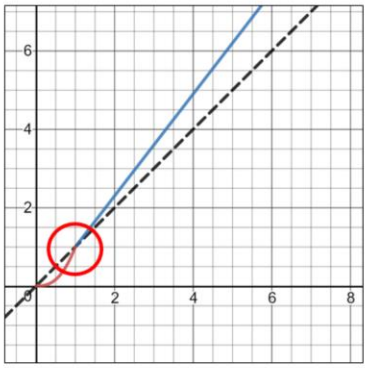
$p = 1.2, m = 1, c = 1.2$

Strong emphasizing a bright part

These are the behavior of the curve.
The center is strong emphasizing a dark part.
The right is strong emphasizing a bright part.

Parametric Curve Based Grading

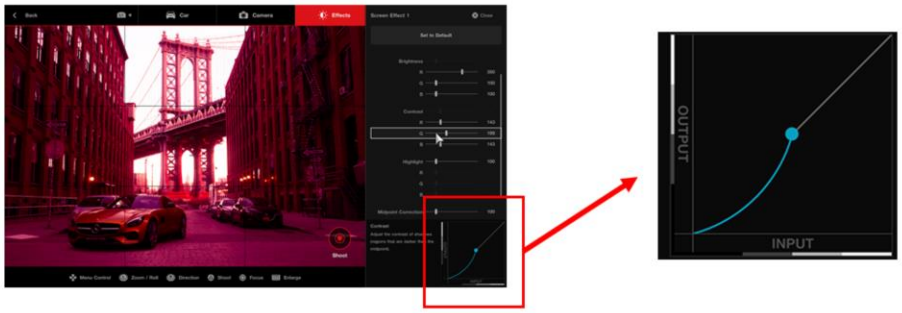
- The dark part and bright part aren't connected smoothly, but artists liked this curve, so that is OK!



The dark part and bright part isn't connected smoothly, but artists liked this curve, so it's OK!
The smoothness of the function is not so important in this situation.

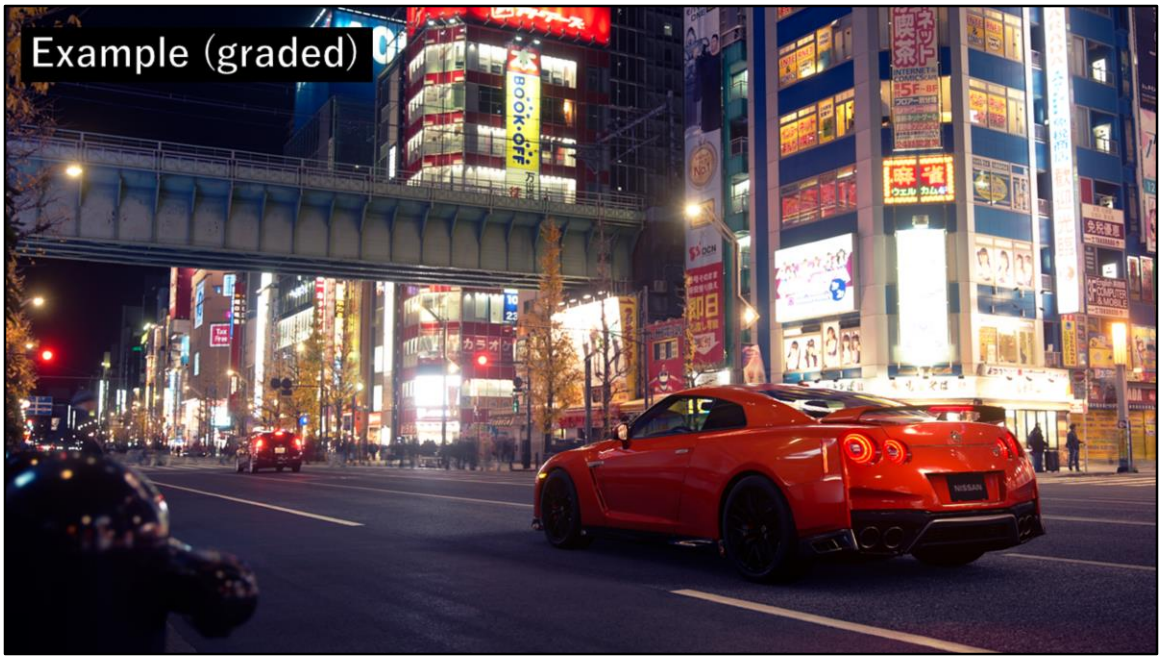
Parametric Curve Based Grading

- The same grading curve is implemented in a photo mode in Gran Turismo SPORT for photo editing tools
- World-wide players are grading their photos in a linear space using a SDR/HDR compatible parametric curve!



The same grading curve is implemented in a photo mode in Gran Turismo SPORT for photo editing tools.
World-wide players are grading their photos in a linear space using a SDR/HDR compatible parametric curve!

We will show an example of curve based grading.
This is the original image.



And this is the graded image.

And this is a side-by-side comparison.

Conclusions

- Achieved support of lookup table (LUT) based grading and simple parametric curve based grading
- Both methods are SDR/HDR compatible

To conclude.

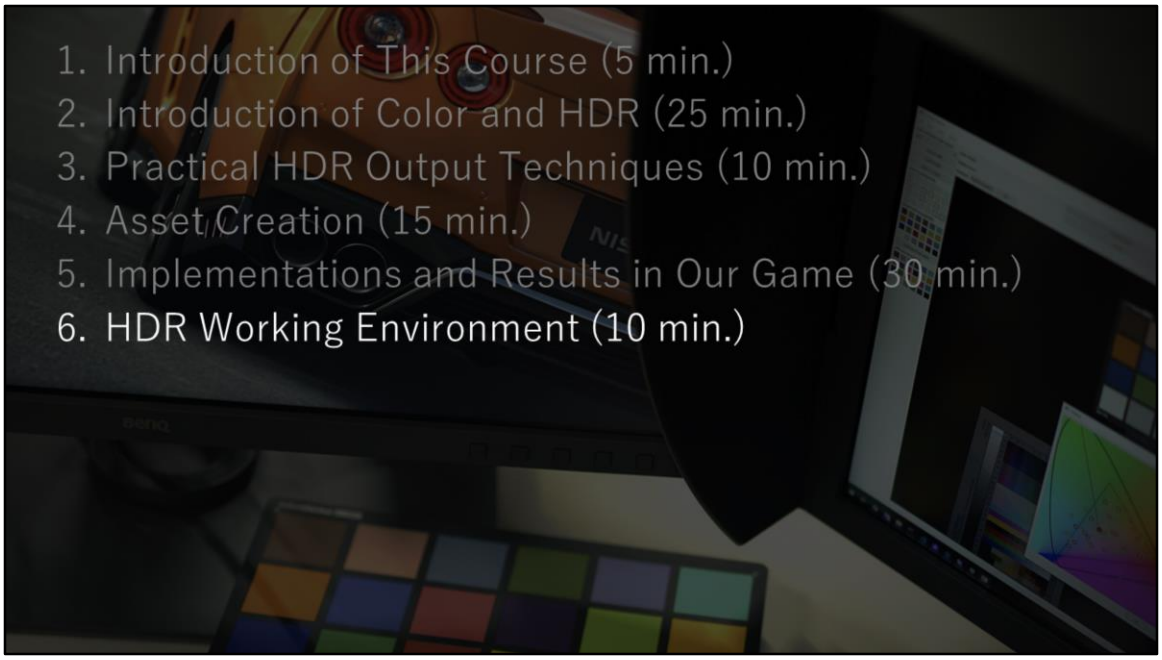
We achieved support of both lookup table based grading and simple parametric curve based grading.
And both methods are SDR/HDR compatible.

Conclusions

- Achieved support of lookup table (LUT) based grading and simple parametric curve based grading
- Both methods are SDR/HDR compatible

→ We achieved the goal: significantly enhance our HDR rendering output.

Thanks to these methods, we were able to significantly enhance our HDR rendering output.


1. Introduction of This Course (5 min.)
2. Introduction of Color and HDR (25 min.)
3. Practical HDR Output Techniques (10 min.)
4. Asset Creation (15 min.)
5. Implementations and Results in Our Game (30 min.)
6. HDR Working Environment (10 min.)

Build Neutral Image

- Verifying HDR output is very hard.
- Too many variations in HDR TVs.
- Neutral image output is very important.
- Needed for easy debugging.

For final section we'd like to cover how we built our HDR working environment.

I can say verifying collect HDR output is really hard problem.
The reason is there are so many variations of TV monitors.

To debug where the problem is,
Keeping output image as neutral as very important.

Watch It Physically

- Specification on internet is not enough to trust.
- Watching and testing actual monitor is important.
- We created our own testing suite.

Of course you can find specification numbers on internet,
But their number are sometimes not true.

So touching, watching and testing actual monitor is really important.
We have created our own testing suite.

Test All Monitors

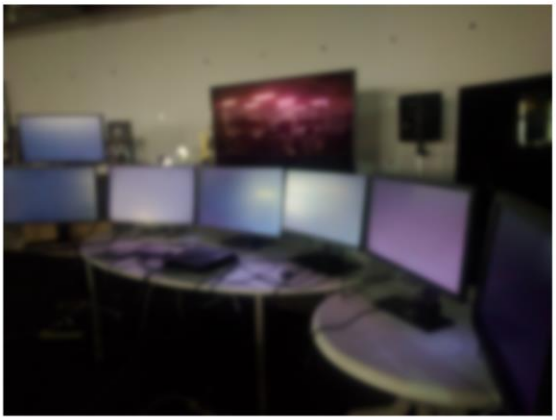
- To test the monitor we established standard testing patterns.
  1. Small window test
  2. Moving small window test
  3. Full locus image
  4. HSV scroller

- and so on.

Here is the list of the tests.

We will introduce what these tests do and what we check.

Some Day at Office



To test and verify, we collected a lot of monitors from cheaper to expensive. Price range varied.

Small Window Test

In small window test, we draw small white box in the tv monitor.
What we want to check here is the behavior of monitor tone mapping functions and backlight static specifications.

Small Window Test with Size

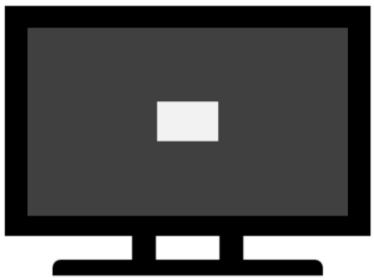
Changing size you can determine how peak brightness and tone mapping curve changes.

Small Window Test

We have found some TV monitor has interesting behavior.
When the size of the box is really small, the TV suddenly shuts down.

Maybe TV decided my precious white box is just noise.

Moving Small Window Test



Next up, moving small window test.

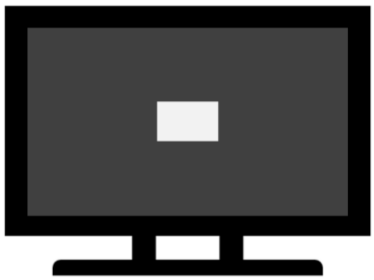
In this test, the white small box moves around.

This test is to check the behavior of local dimming backlight.
Local dimming is a technology used to enhance TV contrast.

Local dimming uses an array of small backlights.
But how this array is designed and located is hidden secret.
So we have to test the actual performance.

In local dimming monitor, halo artifact is well known phenomenon.
Halo is a blurry artifact around the bright pixels caused by light leaking.

Moving Small Window Test

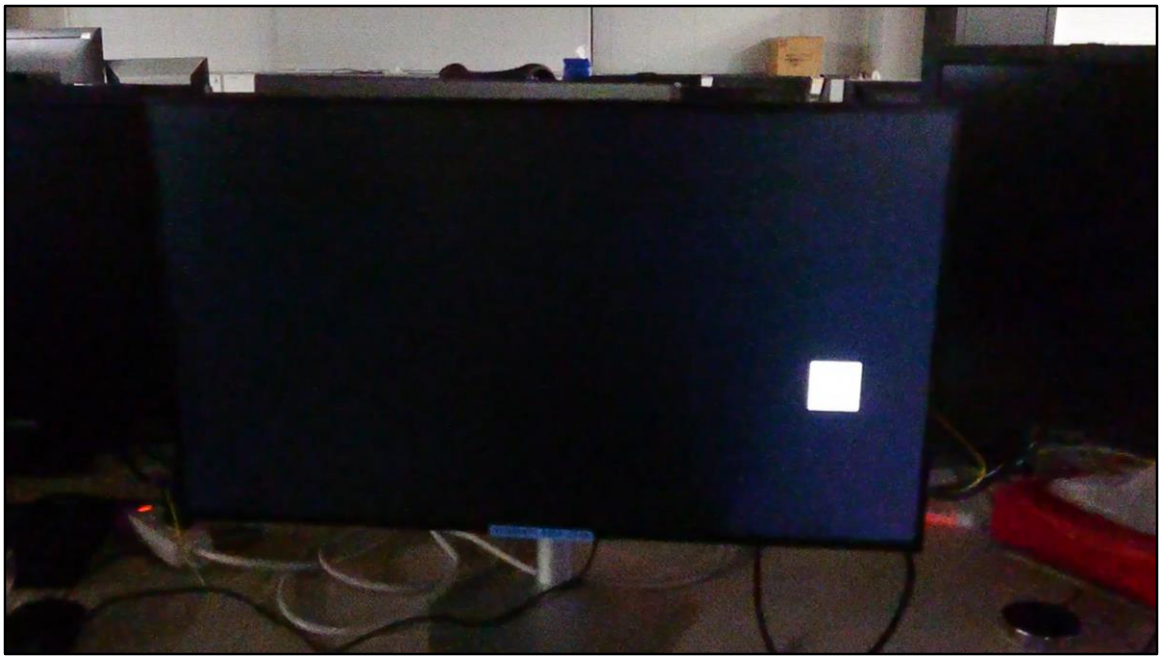


Next is moving small window test.
In this test, the white small box moves around.

Moving Small Window Test

With this test you can check the halo effect and behavior of the backlights.

This sample is to show how our actual test looks like on TV monitor.
This TV is a 100 inches SONY Z9D.
This TV monitor is actually the best in line up, no leaking can be noticed.

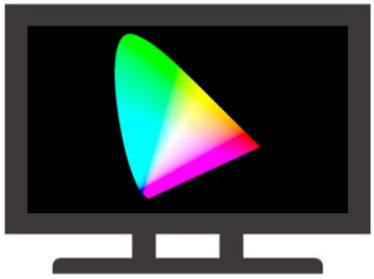


Compared to previous TV, this PC monitor has very interesting behavior.

From this test, we can say, the image quality may vary.
Imagine a situation that game player plays your game with this display, and he/she writes a review on amazon.

What can we do is, at least, is knowing.

Full Locus Test



Next test is full locus test.
We draw full color locus image on a display.

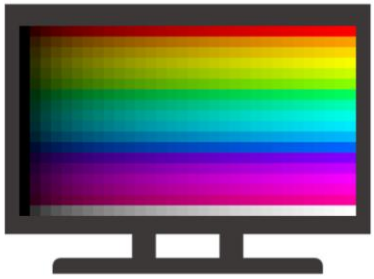
This test is to check how color signals are interpreted.

Full Locus Test: Result

This test is very simple but, a lot of thing can be determined.
These pictures are taken from actual TV monitors.

The left picture shows how colors are skewed and enhanced.
The right picture shows how colors are saturated and how narrow this monitor gamut is.

HSV Scroller



Next test is HSV scroller.

In this test we output HSV bars.
This test is to check how colors changed related to the luminance.
This is also very simple test, but works really well.

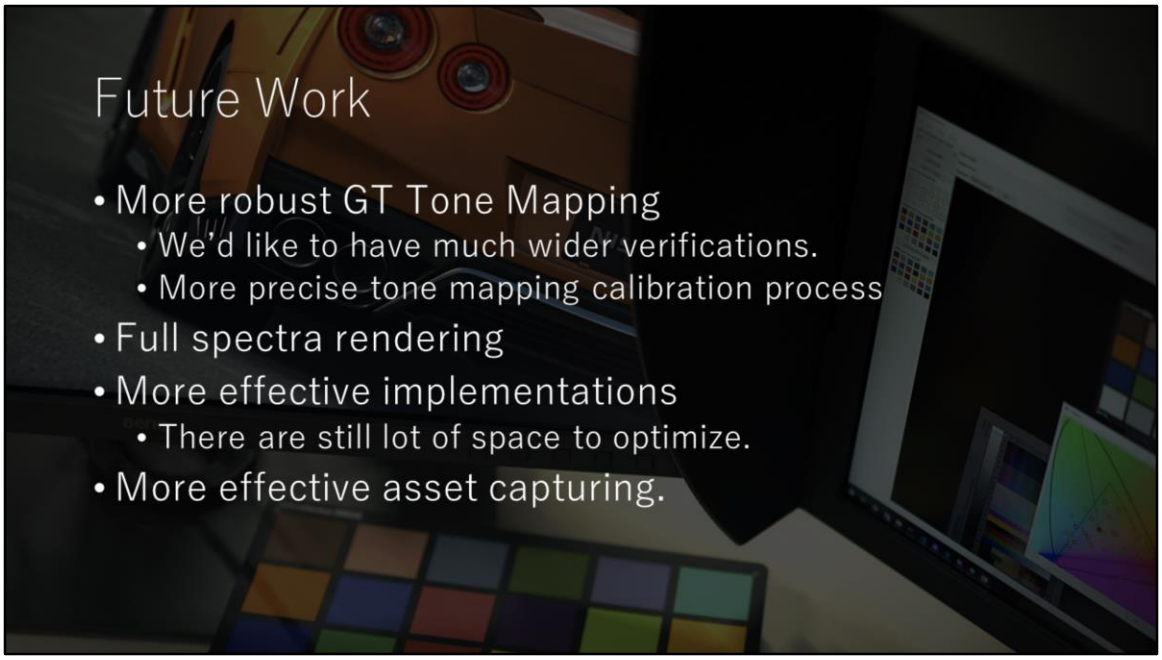
HSV Scroller Result Example

With this test we can find some TVs performs very well,
But some PC monitors have too much enhancement.
We can find which monitor is good for contents creation.

Conclusion

- HDR world is getting complicated by image enhancement, device limitations, etc..
- To build your believable environment, test everything.

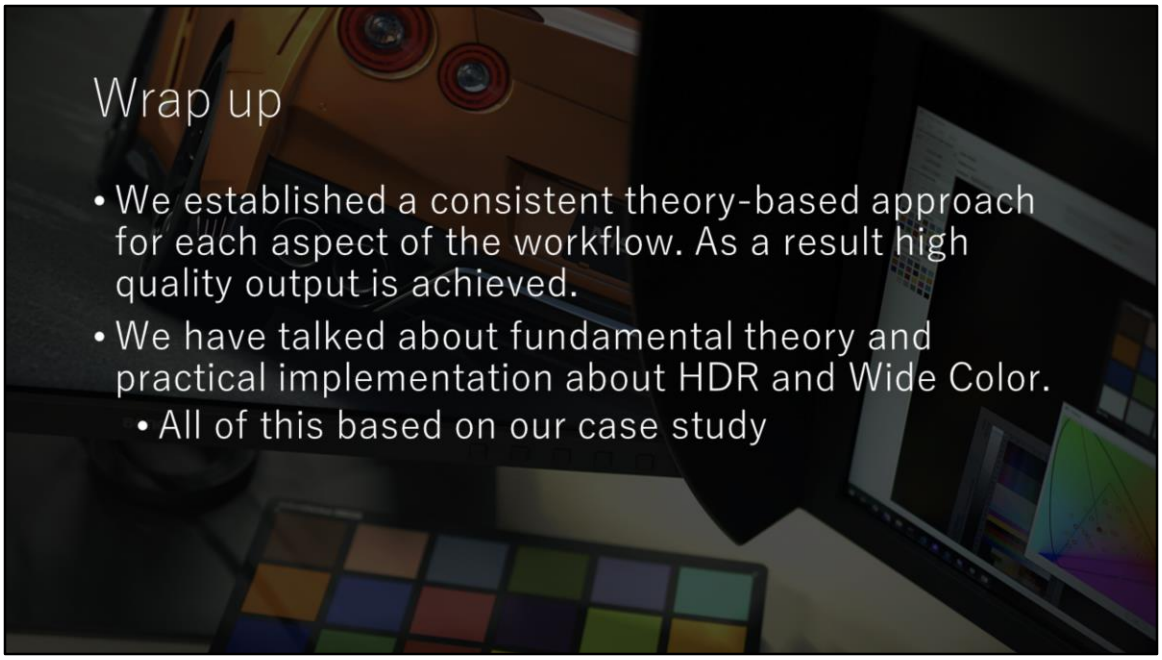
For conclusion, HDR world is getting complicated like SDR.
To build your believable environment, testing is important.



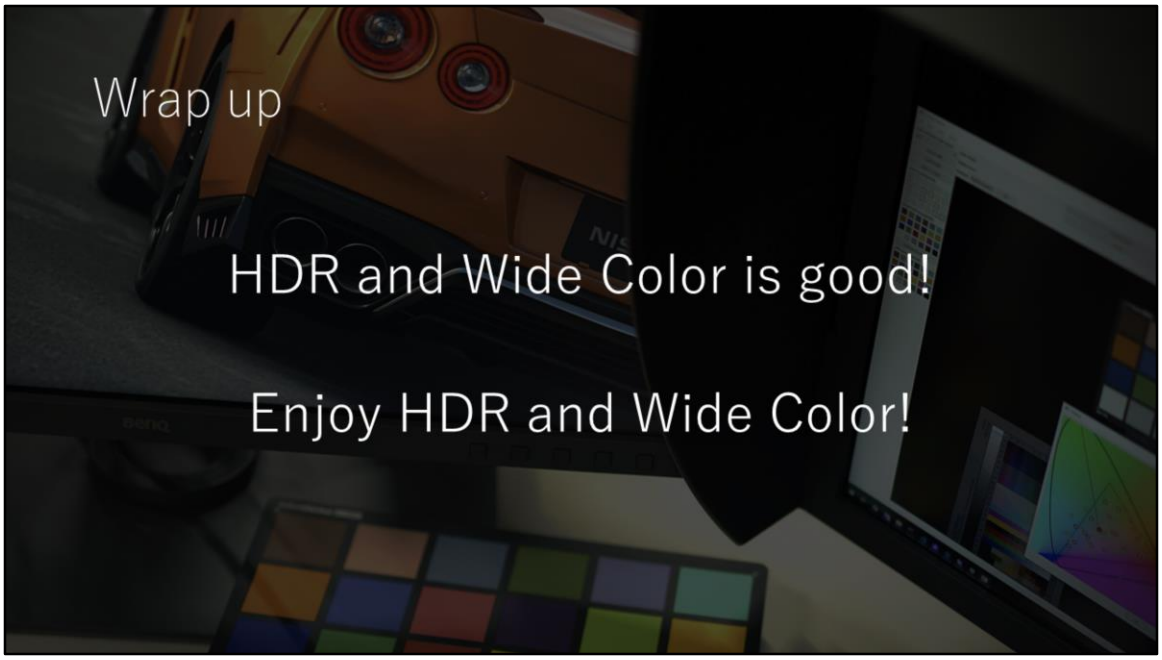
This is future work.
We'd like to build more robust tone mapping by much wider verify.
Also we'd like to build more precise tone maping calibration process.

We'd like to start full sptctra rendering.

Next, we'd like to build more effective implementations and asset capturing.
There are still lots of room to optimized.

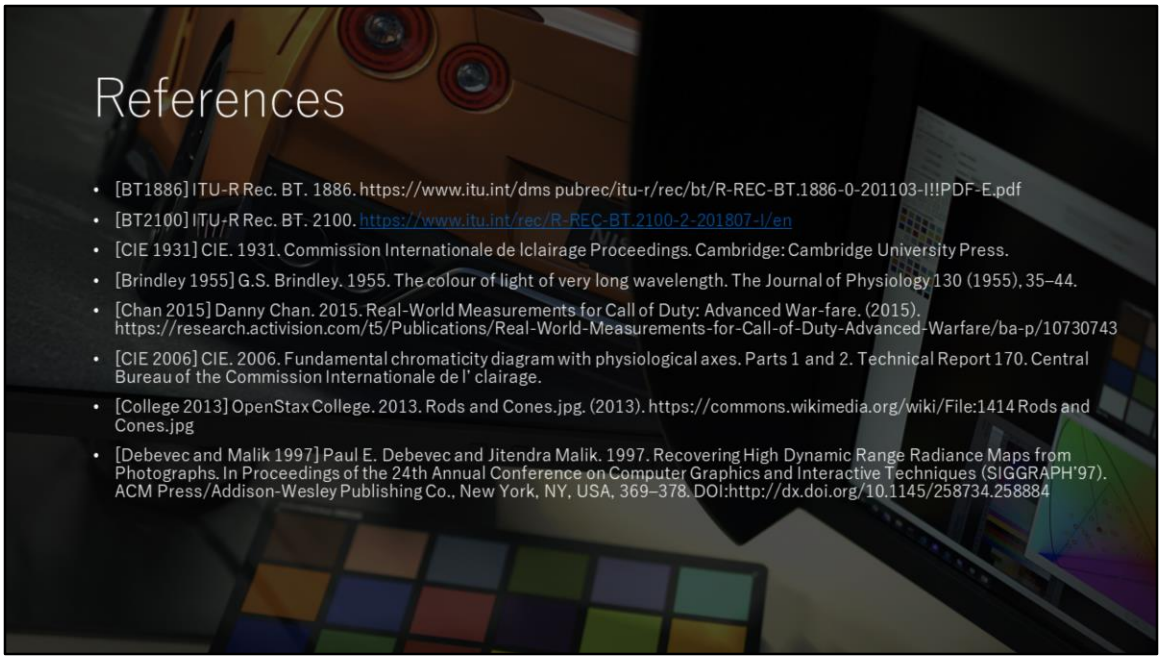


We established consistent theory-based approach for each aspect of the workflow. As a result high quality output is achieved.
We talked about fundamental theory and practical implementation about HDR and Wide Color. Based on our case study
We believe you have gotten knowledge about developing HDR and wide color applications effectively and efficiently.


Wrap up
HDR and Wide Color is good!
Enjoy HDR and Wide Color!

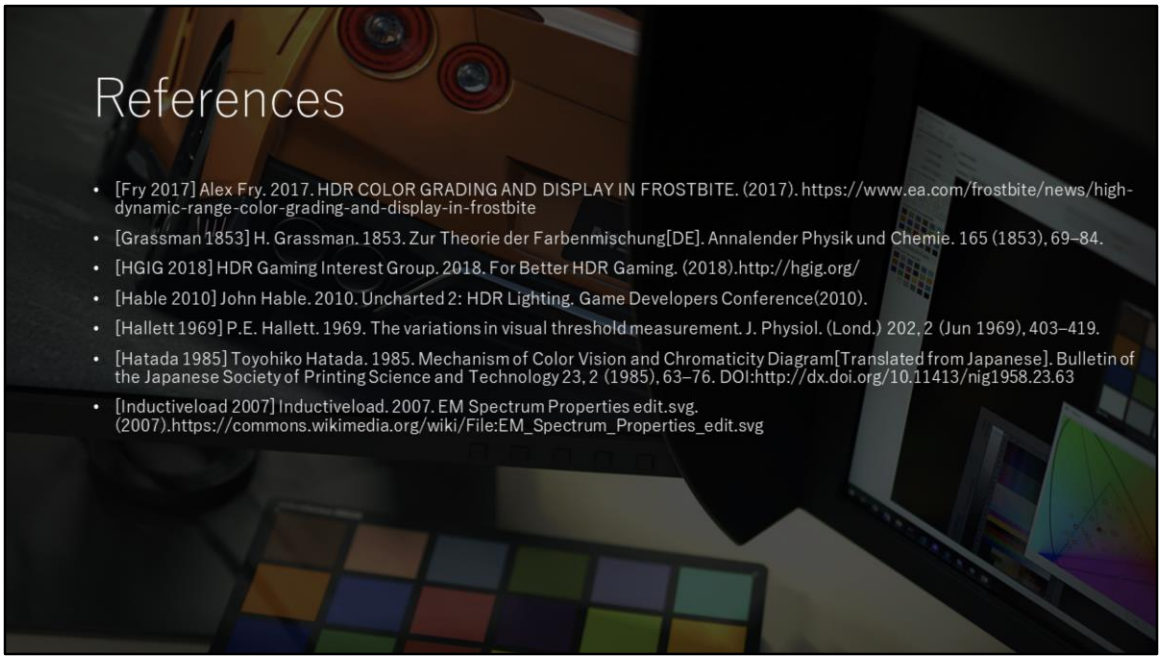

WE ARE HIRING
http://www.polyphony.co.jp/recruit/english/

Thank you for our acknowledgements.



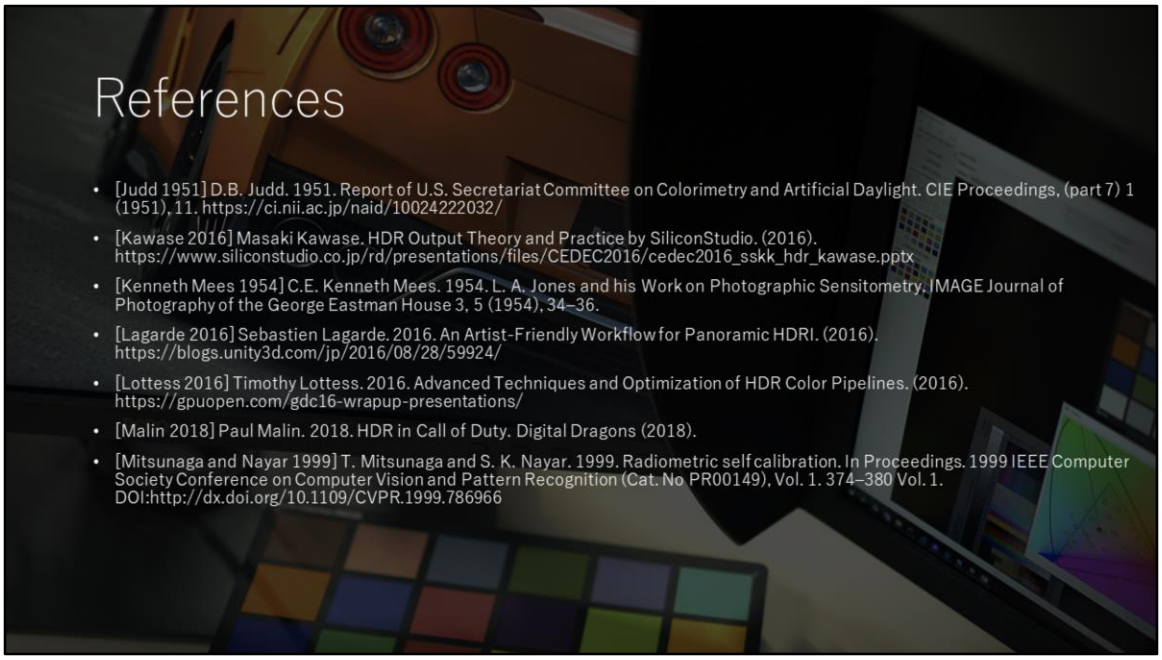
References

- [BT1886] ITU-R Rec. BT. 1886. https://www.itu.int/dms pubrec/itu-r/rec/bt/R-REC-BT.1886-0-201103-I!!PDF-E.pdf
- [BT2100] ITU-R Rec. BT. 2100. https://www.itu.int/rec/R-REC-BT.2100-2-201807-I/en
- [CIE 1931] CIE. 1931. Commission Internationale de lclairage Proceedings. Cambridge: Cambridge University Press.
- [Brindley 1955] G.S. Brindley. 1955. The colour of light of very long wavelength. The Journal of Physiology 130 (1955), 35–44.
- [Chan 2015] Danny Chan. 2015. Real-World Measurements for Call of Duty: Advanced War-fare. (2015). https://research.activision.com/t5/Publications/Real-World-Measurements-for-Call-of-Duty-Advanced-Warfare/ba-p/10730743
- [CIE 2006] CIE. 2006. Fundamental chromaticity diagram with physiological axes. Parts 1 and 2. Technical Report 170. Central Bureau of the Commission Internationale de l' clairage.
- [College 2013] OpenStax College. 2013. Rods and Cones.jpg. (2013). https://commons.wikimedia.org/wiki/File:1414 Rods and Cones.jpg
- [Debevec and Malik 1997] 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:http://dx.doi.org/10.1145/258734.258884



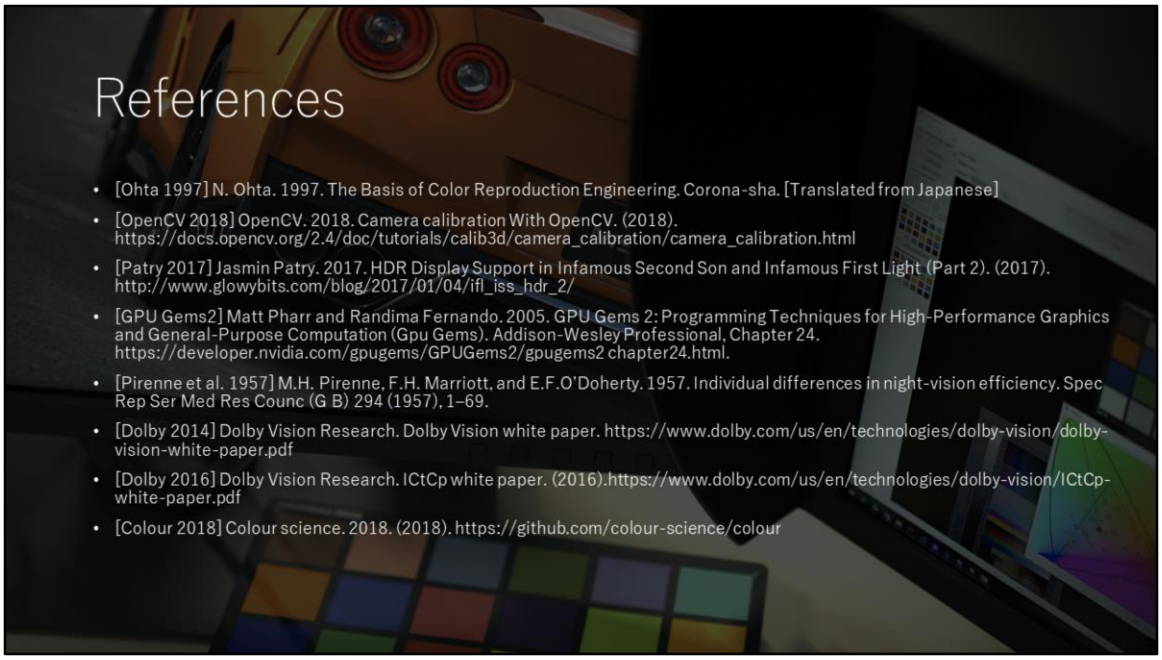
References

- [Fry 2017] Alex Fry. 2017. HDR COLOR GRADING AND DISPLAY IN FROSTBITE. (2017). https://www.ea.com/frostbite/news/high-dynamic-range-color-grading-and-display-in-frostbite
- [Grassman 1853] H. Grassman. 1853. Zur Theorie der Farbenmischung[DE]. Annalender Physik und Chemie. 165 (1853), 69–84.
- [HGIG 2018] HDR Gaming Interest Group. 2018. For Better HDR Gaming. (2018).http://hgig.org/
- [Hable 2010] John Hable. 2010. Uncharted 2: HDR Lighting. Game Developers Conference(2010).
- [Hallett 1969] P.E. Hallett. 1969. The variations in visual threshold measurement. J. Physiol. (Lond.) 202, 2 (Jun 1969), 403–419.
- [Hatada 1985] Toyohiko Hatada. 1985. Mechanism of Color Vision and Chromaticity Diagram[Translated from Japanese]. Bulletin of the Japanese Society of Printing Science and Technology 23, 2 (1985), 63–76. DOI:http://dx.doi.org/10.11413/nig1958.23.63
- [Inductiveload 2007] Inductiveload. 2007. EM Spectrum Properties edit.svg. (2007).https://commons.wikimedia.org/wiki/File:EM_Spectrum_Properties_edit.svg



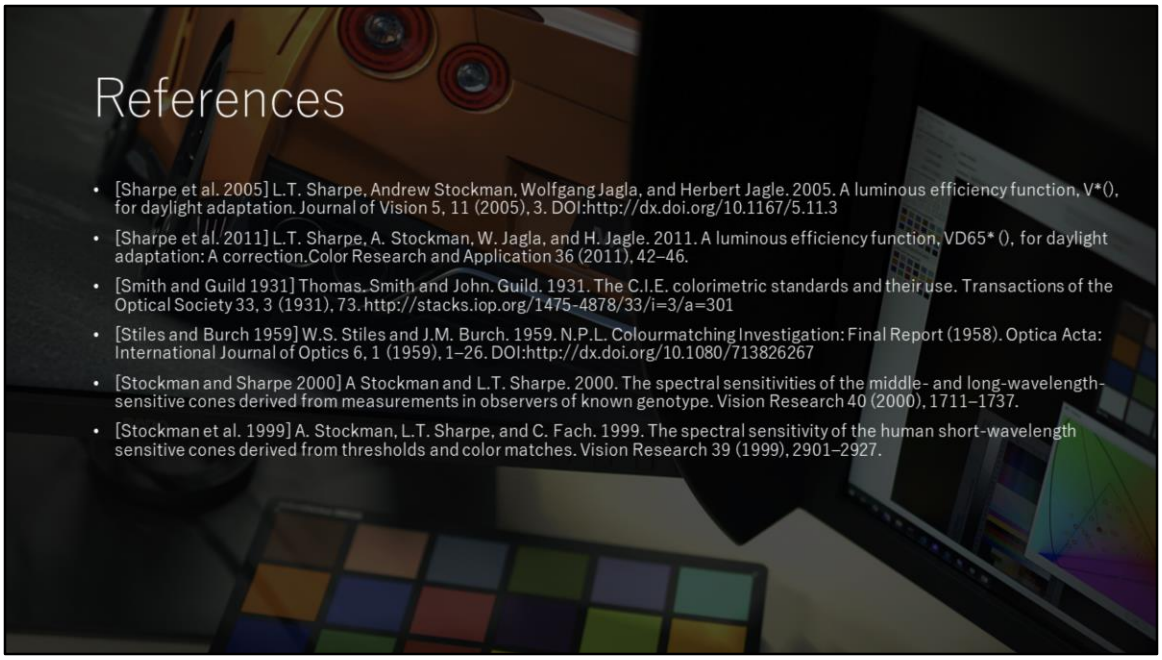
References

- [Judd 1951] D.B. Judd. 1951. Report of U.S. Secretariat Committee on Colorimetry and Artificial Daylight. CIE Proceedings, (part 7) 1 (1951), 11. https://ci.nii.ac.jp/naid/10024222032/
- [Kawase 2016] Masaki Kawase. HDR Output Theory and Practice by SiliconStudio. (2016). https://www.siliconstudio.co.jp/rd/presentations/files/CEDEC2016/cedec2016_sskk_hdr_kawase.pptx
- [Kenneth Mees 1954] C.E. Kenneth Mees. 1954. L. A. Jones and his Work on Photographic Sensitometry. IMAGE Journal of Photography of the George Eastman House 3, 5 (1954), 34–36.
- [Lagarde 2016] Sebastien Lagarde. 2016. An Artist-Friendly Workflow for Panoramic HDRI. (2016). https://blogs.unity3d.com/jp/2016/08/28/59924/
- [Lottess 2016] Timothy Lottess. 2016. Advanced Techniques and Optimization of HDR Color Pipelines. (2016). https://gpuopen.com/gdc16-wrapup-presentations/
- [Malin 2018] Paul Malin. 2018. HDR in Call of Duty. Digital Dragons (2018).
- [Mitsunaga and Nayar 1999] T. Mitsunaga and S. K. Nayar. 1999. Radiometric self calibration. In Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149), Vol. 1. 374–380 Vol. 1. DOI:http://dx.doi.org/10.1109/CVPR.1999.786966



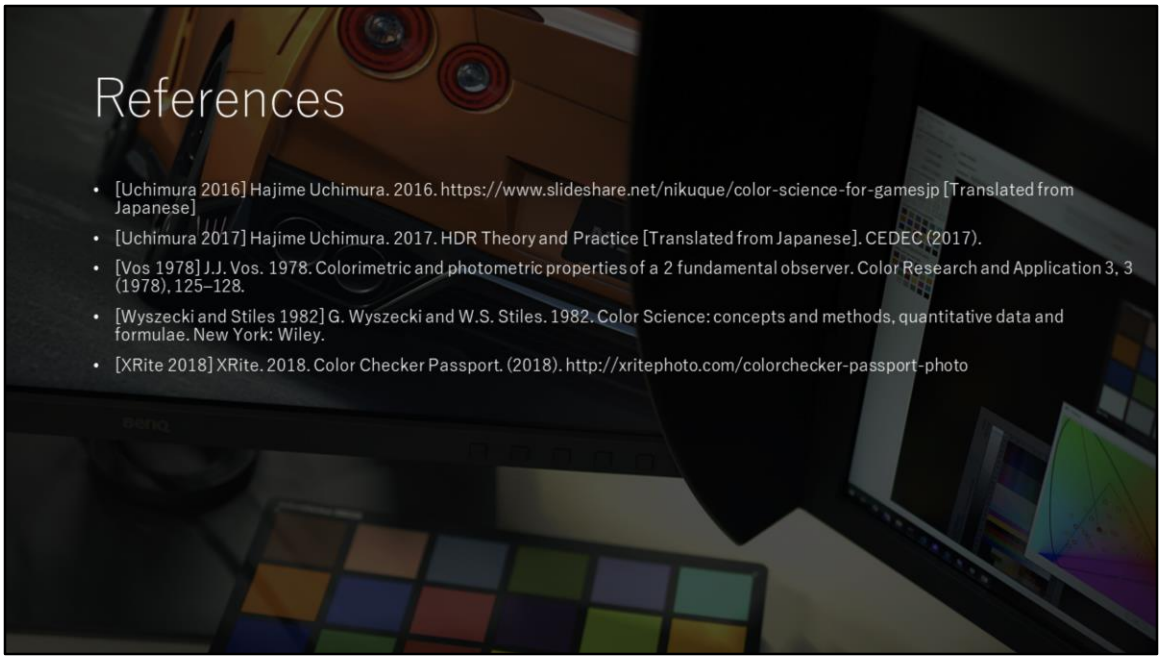
References

- [Ohta 1997] N. Ohta. 1997. The Basis of Color Reproduction Engineering. Corona-sha. [Translated from Japanese]
- [OpenCV 2018] OpenCV. 2018. Camera calibration With OpenCV. (2018). https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html
- [Patry 2017] Jasmin Patry. 2017. HDR Display Support in Infamous Second Son and Infamous First Light (Part 2). (2017). http://www.glowybits.com/blog/2017/01/04/ifl_iss_hdr_2/
- [GPU Gems2] Matt Pharr and Randima Fernando. 2005. GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems). Addison-Wesley Professional. Chapter 24. https://developer.nvidia.com/gpugems/GPUGems2/gpugems2 chapter24.html.
- [Pirenne et al. 1957] M.H. Pirenne, F.H. Marriott, and E.F.O'Doherty. 1957. Individual differences in night-vision efficiency. Spec Rep Ser Med Res Counc (G B) 294 (1957), 1–69.
- [Dolby 2014] Dolby Vision Research. Dolby Vision white paper. https://www.dolby.com/us/en/technologies/dolby-vision/dolby-vision-white-paper.pdf
- [Dolby 2016] Dolby Vision Research. ICtCp white paper. (2016).https://www.dolby.com/us/en/technologies/dolby-vision/ICtCp-white-paper.pdf
- [Colour 2018] Colour science. 2018. (2018). https://github.com/colour-science/colour



References

- [Sharpe et al. 2005] L.T. Sharpe, Andrew Stockman, Wolfgang Jagla, and Herbert Jagle. 2005. A luminous efficiency function, V*(), for daylight adaptation. Journal of Vision 5, 11 (2005), 3. DOI:http://dx.doi.org/10.1167/5.11.3
- [Sharpe et al. 2011] L.T. Sharpe, A. Stockman, W. Jagla, and H. Jagle. 2011. A luminous efficiency function, VD65* (), for daylight adaptation: A correction.Color Research and Application 36 (2011), 42–46.
- [Smith and Guild 1931] Thomas. Smith and John. Guild. 1931. The C.I.E. colorimetric standards and their use. Transactions of the Optical Society 33, 3 (1931), 73. http://stacks.iop.org/1475-4878/33/i=3/a=301
- [Stiles and Burch 1959] W.S. Stiles and J.M. Burch. 1959. N.P.L. Colourmatching Investigation: Final Report (1958). Optica Acta: International Journal of Optics 6, 1 (1959), 1–26. DOI:http://dx.doi.org/10.1080/713826267
- [Stockman and Sharpe 2000] A Stockman and L.T. Sharpe. 2000. The spectral sensitivities of the middle- and long-wavelength-sensitive cones derived from measurements in observers of known genotype. Vision Research 40 (2000), 1711–1737.
- [Stockman et al. 1999] A. Stockman, L.T. Sharpe, and C. Fach. 1999. The spectral sensitivity of the human short-wavelength sensitive cones derived from thresholds and color matches. Vision Research 39 (1999), 2901–2927.



References

- [Uchimura 2016] Hajime Uchimura. 2016. https://www.slideshare.net/nikuque/color-science-for-gamesjp [Translated from Japanese]
- [Uchimura 2017] Hajime Uchimura. 2017. HDR Theory and Practice [Translated from Japanese]. CEDEC (2017).
- [Vos 1978] J.J. Vos. 1978. Colorimetric and photometric properties of a 2 fundamental observer. Color Research and Application 3, 3 (1978), 125–128.
- [Wyszecki and Stiles 1982] G. Wyszecki and W.S. Stiles. 1982. Color Science: concepts and methods, quantitative data and formulae. New York: Wiley.
- [XRite 2018] XRite. 2018. Color Checker Passport. (2018). http://xritephoto.com/colorchecker-passport-photo