RGB values

Support
Materials and Textures

RickyBlender (RickyBlender) 2008-02-02 12:24:10 UTC #1

when i set the values of RGB in Blender how does it relate to other values of RGB that you can find in other software!

i'm having some difficulties relating the colors
Blender is 0 to 1 other are from 0 to 255

Is there a special wy to deal with these valuies

Tanks

Download a whole thread as PDF file

OBI_Ron (OBI_Ron) 2008-02-02 13:56:28 UTC #2

Basically, the number from other software/255

So, if the rgb color in GIMP is:
160
88
88

Then Blender would be
160/255 = .627
88/255 = .345
88/255 = .345

Apollos (Apollos) 2008-02-02 14:21:54 UTC #3

To add a little to what OBI_Ron pointed out, Blender uses normalized RGB values. These make it possible to do blending operations like multiply. Other programs will convert these internally at some point for color arithmetic.

The normalized values also make it intuitive to support higher color ranges, as a channel value of 0.627 could easily be representative of 8, 16, or 32 bit channels.

Blender does give byte representations in the color picker, but they're given in hex notation.

Mmph (Mmph!) 2008-02-02 14:42:52 UTC #4

I dont think I will ever understand this.
The way it is we have to type a "." (period) EVERY single time we enter a value.

All bow to the mighty "."!!!

I know it is only 1 wasted keystroke, you must enter almost every time, just that day after day it really starts getting to me. Plus I have arthritis, so sometimes it gets pretty painful when I am too lazy to go eat an aspirin.

Here is an idea! why not just add a static "." in the GUI? That way we wont have to type it every single time… if we need a 1 , then we just type a "." !!

rombout (rombout) 2018-11-10 04:35:01 UTC #5

This is super anoying for people who work color controlled. Why dont they add an option to show either normalized of regular method?

Who the hell is going to do the math on each value each time, that is time consuming. Luckily we got hex option but still that should not be needed

---

Secrop (Secrop) 2018-11-10 09:44:11 UTC #6

> rombout:
>
> Who the hell is going to do the math on each value each time, that is time consuming.

There's no need to do the math by yourself… you can just type '***val/255***' in each color component field, and blender will do the math for you.

You can even convert an hexadecimal value in the component field; just type: '***int('FF', 16)/255***' and you'll get the normalized value for the hexadecimal.

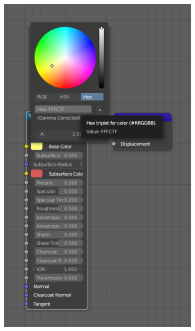
---

pixelgrip (pixelgrip) 2018-11-11 16:23:06 UTC #7

maybe have a look at this thread

> **sRGB Integer to linear calc**
>
> I often find sRGB integer values in the net ,without a picture from that color, that i could pick up with a colorpicker. Then i had the idea, to make a simple sRGB integer input nodegroup,there i could put in the values i have found. Since we use sRGB as Display mode,i have to gamma correct the values before,that way i can see the sRGB values in the Materialpreview and can pic it up. I have build a second gamma correction,for the case i want to display the values as linear with the sRGB Displ…

and the correct calc from the dev



Get Hex (Gamma Corrected) color

Hi! I would like to know how to get the Hex (Gama Corrected) value from a color in python. getting the default_value of the input i can get the float point color but i could't find a way to the hex gamma corrected one or at least the float point...

Reading time: 3 mins Likes: 3 ❤

---

rombout (rombout) 2018-11-13 04:25:38 UTC #8

> Secrop:
>
> There's no need to do the math by yourself… you can just type ' ***val/255*** ' in each color component field, and blender will do the math for you.

> You can even convert an hexadecimal value in the component field; just type: ‘ ***int(‘FF’, 16)/255*** ’ and you’ll get the normalized value for the hexadecimal.

In what compenent? Because when i add that i adjust the already blender value.

I mean i want to see RGB values so that mean val*255 than i have the RGB value. But thats is exactly what i mean, why do we need to calculations. There should be an option to just show RGB values as is, no calculation needed.

---

cgCody (cgCody) 2018-11-13 04:52:59 UTC #9

> rombout:
>
> But thats is exactly what i mean, why do we need to calculations. There should be an option to just show RGB values as is, no calculation needed.

Because it makes more sense in a linear workflow. In other situations like digital painting, 0-255 makes sense as it is a hard limit. In CG rendering, however, you’re often working with values beyond screen color space, in which case 0 - 1 (or 0% to 100%) becomes a soft limit, with values sometimes reaching in the 100’s or even 1000’s

A pixel from a light in your scene, for example, might have an RGB value of 250, 250, 250. You immediately know that this pixel is 250 times brighter than a white pixel. That same value with a 0-255 range would read as RGB 63750, 63750, 63750.

---

troy_s (troy_s) 2018-11-13 04:55:00 UTC #10

> rombout:
>
> There should be an option to just show RGB values as is, no calculation needed.

The problem is that integer based colour encodings are absolute garbage because they don’t represent anything.

So, what you are seeing with a float, is the actual colour ratio. Bear in mind that for things like emission, there is no such thing as “normalized”, hence the floats extend up to infinity.

TL;DR: The internal reference is float. The sooner everyone understands that colour as integer, or worse, hex, is absolute tripe, the sooner everyone has a more solid understanding of the core concepts.

> cgCody:
>
> In CG rendering, however, you’re often working with values beyond screen color space, in which case 0 - 1 (or 0% to 100%) becomes a soft limit, with values sometimes reaching in the 100’s or even 1000’s

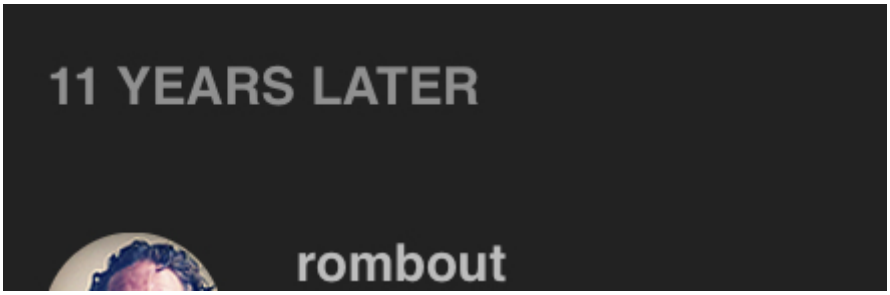
True!

> cgCody:
>
> A pixel from a light in your scene, for example, might have an RGB value of 250, 250, 250. You immediately know that this pixel is 250 times brighter than a white pixel. That same value with a 0-255 range would read as RGB 63750, 63750, 63750.

False! False! False!

The internal representation is float, hence converting to integer is actually completely backwards.

PS: Full marks for this!



---

cgCody (cgCody) 2018-11-13 05:02:34 UTC #11

> troy_s:
>
> False! False! False!
>
> The internal representation is float, hence converting to integer is actually completely backwards.

Well I know you're the man to talk to when it comes to color spaces, so I believe you!

However, what I mean in more of a general sense, is that if the input sliders had a range of 0-255, then values beyond that space would be human-non-readable without a calculator.

---

troy_s (troy_s) 2018-11-13 05:15:26 UTC #12

> cgCody:
>
> However, what I mean in more of a general sense, is that if the input sliders had a range of 0-255, then values beyond that space would be human-non-readable without a calculator.

Indeed!

And totally bunko! That is, there are a number of things going on here that are actually *greatly* improved when trying to communicate concepts simply by enforcing floats everywhere.

Folks go out a huntin' lookin' for dem hex codes, or some arbitrary value. The muddling commences.

As you rightly pointed out, sometimes the value is 0.0 to 1.0, and other times it could be 0.0 to infinity, or -5.0 to +5.0 or who knows what. The net sum is that *numbers are contextual*.

If we start muddying the waters with integer, now folks don't have a clue if 2000 is an integer normalized value, or 2000.0 units of emission colour intensity, or 2000.0 units of depth. By simply presenting and offering the pixel pusher the internal representation of the float, we level the playing field, and force people to think about the media itself. Is it a colour? Does it represent a percentage of reflection as with an albedo? Is it an emission? What the hell does an emission mean anyways?

Which nicely loops back into the other Devtalk thread that I was trying to highlight: *Hex codes are garbage*. They create the illusion of some sort of meaning, but all they are are ratios of something.

In the case of a slider in Blender, if you set an *albedo* value to 0.5, you are declaring that 50% of the incoming light will reflect back. What colour is the light? What the hell colour is this "red"? Are there other "reds"? Is it linearly or nonlinearly encoded? If this is an *emission*, how is 30,000.981 a legitimate value? How can I input 2172.721 if I need to? How do the pieces snap together?

Slowly, we can help each other learn and understand these things. Falling back on horrible and meaningless hex codes doesn't help anyone past these slippery questions.

cgCody (cgCody) 2018-11-13 05:34:50 UTC #13

> troy_s:
>
> What the hell does an emission mean anyways?

It's the measure of Ton-ness a given pixel holds influence over it's surrounding tangent space. 😄

This all reminds me of a decade ago or so when "linear workflow" was the hot buzzword in town, ("PBR" of it's day). I remember reading a Siggraph paper about it and going, "…Wait… there are colors beyond 255?!" ***HEAD EXPLODES***)

---

troy_s (troy_s) 2018-11-13 05:39:35 UTC #14

> cgCody:
>
> "…Wait… there are colors beyond 255?! ***HEAD EXPLODES***)

I can still tangibly recall the various steps of utter confusion regarding colour so well that I do my darndest to try and keep an empathetic grip on that feeling when trying to explain various bits to others.

Language and concept muddling is at the core of so much of the rot. The legacy "comfort" of broken mental models help none of us. It is remarkable how a carefully chosen unknown term such as "scene referred" can shake just enough discomfort for someone to rethink how their mental models are constructed and arrive at a much better comprehension.

As silly as it sounds, the very same thing applies when I say *hex codes are garbage*; it is a helpful push down the rabbit hole of beginning to unravel firmer understanding.

---

cgCody (cgCody) 2018-11-13 05:56:21 UTC #15

> troy_s:
>
> Language and concept muddling is at the core of so much of the rot.

Very true! And it's not just limited to our tiny universe of digital content creation. Look at all of the display manufacturers with HDR, for example. Not only is that industry divided on a standard, but each manufacturer has a different implementation (Not to mention marketing buzzwords that are incorrect/obscure).

Then we have the content meant to take advantage of this technology. Aside from a select few videos on Youtube and Amazon Prime (Top Gear: Grand Tour looked amazing!), most of the "HDR" stuff I've seen seems to show a fundamental lack of understanding about having an extended color space.

CarlG (CarlG) 2018-11-13 10:15:56 UTC #16

So if you find a random texture you want to use, and want to make sure it adheres to a **pbr albedo cheat sheet**, what would be the correct procedure? Assume the sheet cheat is listed in either sRGB or Linear, rather than both as here.

---

cgCody (cgCody) 2018-11-13 11:34:37 UTC #17

If what you're asking is along the lines of; You used a color picker on your sRGB texture and got value S, and you want to make sure that matches your linear cheat sheet's value L, then look up the formula for converting from sRGB to linear. Convert your S to it's corrisponding L and see if that matches your table.

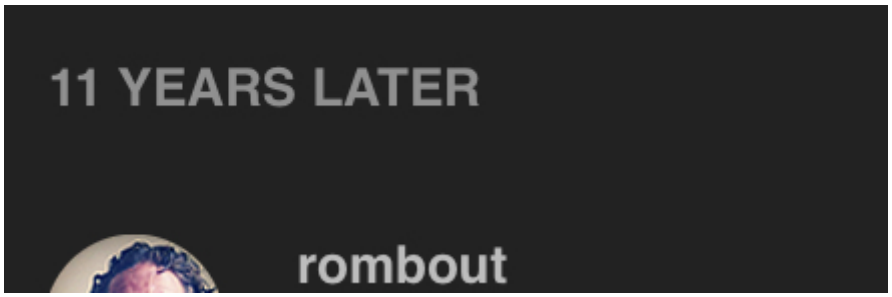
EDIT:
((S + 0.055) / 1.055)^2.4 = L

So given the first example in your link, let's assume the sRGB value was't listed, but you used a color picker on your texture and got the value 148.

148 / 255 = 0.5804
((0.5804 + 0.055) / 1.055)^2.4 = 0.3
Yep, looks like it adhares to the table.

---

rombout (rombout) 2018-11-14 04:13:57 UTC #18

> troy_s:
>
> PS: Full marks for this!
>
> 
>

Ooopssss! 😉

I do understand that for editing in certain ways this makes sense. But if im looking at textures say in UV editor or a render and i want to check color values and compare this with other software, im screwed. I need to do calculations all the time or copy/paste the HEX. I think it would be useful if we can read "regular" RGB values when picking them in the, say uv editor.

I dont think there are a lot of people which can read and understand color values in float numbers? Im a graphics designer i look at colors in CMYK, RGB or HEX, not in float 🙂

I come from a graphics world where the limits dont go passed 255. How far do they go with other bit images than? Because in 2d software they are still read as 255 limit, white wont get passed that 255-255-255 limit.

PS what is that for calculation you put in the end **@cgCody**

> ((S + 0.055) / 1.055)^2.4 = L

Edit
I should have done a google search first

http://entropymine.com/imageworsener/srgbformula

---

cgCody (cgCody) 2018-11-14 04:46:48 UTC #19

> rombout:

> How far do they go with other bit images than? Because in 2d software they are still read as 255 limit, white wont get passed that 255-255-255 limit

In CG rendering, there is really no limit, and values are only relative to the context in which they are being used. That is why having an integer range from 0 - 255 would make the already confusing subject of a scene referred system even MORE difficult to grasp. Troy put it best in post #12. Research the subject of “scene referred space” if you really want to dive into the deep end.

About that equation. That’s really only helpful in rare situations like CarlG’s where you want to make sure an sRGB value matches an expected linear value. In day to day usage, Blender is automatically converting your textures to linear color space. This is what that drop down on the image texture node with the options color and non-color have to do with. Non-color is used in cases where you don’t want the color space converted (eg normal maps, roughness maps)

---

troy_s (troy_s) 2018-11-14 05:46:51 UTC #20

> rombout:
>
> I dont think there are a lot of people which can read and understand color values in float numbers? Im a graphics designer i look at colors in CMYK, RGB or HEX, not in float

What colour space are the code values in?

See the problem?

Hex codes literally mean nothing, nor does “CMYK”, nor “RGB”, without coupling them to a colour space.

In the case of RGB for example, are the lights sRGB / REC.709 or are you reading the values on an Apple MacBook Pro as are very common in graphic design? The values in each case here are completely different lights, and as such, the ratios between them mix entirely different colours depending.

Even within Blender in the default state, are the RGB values sRGB nonlinear? Are they scene referred linear? Are they Filmic code values from the Base Log? Are they aesthetic values after the contrast? Do they represent a reflective albedo, an emission, non colour alpha, non colour depth, non colour normal?

CMYK? More meaningless. Is it Fogra36 code values, US Web Coated V2? Any one of the other many CMYK ink / paper combinations in the world?

As you can see, integer based encodings don’t tell you anything, despite folks thinking they do, and hex is worse. It really is high time to let them die, where they belong, so the fewer people get confused. Sadly there are already too many confused people out there who think hex codes mean something, and doubly so mean something in a compositing / rendering pipeline.

CarlG (CarlG) 2018-11-14 10:42:18 UTC #21

How are hex worse? They’re just base16 of base10 integers.

RickyBlender (RickyBlender) 2018-11-14 13:26:29 UTC #22

these number represent proportion !
does not matter if in HEX binary or base 16 ect…

happy bl

---

lolwel21 (lolwel21) 2018-11-14 14:35:28 UTC #23

They are mainly “worse” in 2 fashions:

1. the 0-255 is a hard-coded limit in hex notation.
2. Hex is generally not human-readable (without extensive practice).

RickyBlender (RickyBlender) 2018-11-14 14:51:41 UTC #24

there are converter on the net not a problem!

happy bl

---

troy_s (troy_s) 2018-11-14 15:24:51 UTC #25

> RickyBlender:
>
> does not matter if in HEX binary or base 16 ect…

> CarlG:
>
> How are hex worse? They’re just base16 of base10 integers.

Express float ratios such as an emission of 1022.9171 in hex, or the linear nit output from ST2084 in hex. In normalized integer domains, trivial values are impossible to represent.

Further, hexadecimal notation only took root due to people *fundamentally misunderstanding* pixel management. Some people actually believe that if you copy hex codes, you get identical colours. Or identical precision floats for that matter. *This simply is not the case*, and is extremely nonsensical in a rendering environment. *This has nothing to do with hexadecimal representation and is a broken conceptual model*.

Use floats everywhere, and learn why.

RickyBlender (RickyBlender) 2018-11-14 15:51:27 UTC #26

HEX
2 years ago I think there was like a bug in the color picker
the only valid value was from HEX representation
other were not the real value

I think this has been corrected by now
but have not check it in a long time

the use of HEX or Octal or Binary has to do more with how program are written I think and how you format your output
it could be anything but require some formula to show it

and there are also the fact that color space between Printed matters and Computer screen are not the same
Printed matter use Negative color as Screen use positive additive colors
and also color space is not the same between RGB or CYM ect…
so it does complicated things

it is not an easy and simple subject !

happy bl

CarlG (CarlG) 2018-11-14 18:52:17 UTC #27

Sorry, I misinterpreted “integer based encodings” as the classical 0-255 range, which in hex is the same as $00-$ff range and where encoding was just the icc tag. I’ve been using only floats, since ever (**1990’ish**, so a while). Although back then they didn’t go above 1 🙂

---

rombout (rombout) 2018-11-15 00:27:40 UTC #28

I dont get the harsh response every time. HEX does mean something for big part of designer, perhaps not in your world. I dont think it will be dropped.

I know the values are depended on which color profile is used in Blender. So i would guess it will be possible to output RGB value as there as well.

Not everybody is interested in post production values where you perhaps need more control. I get the feeling you dont understand that part.

---

troy_s (troy_s) 2018-11-15 01:59:07 UTC #29

> rombout:
>
> Not everybody is interested in post production values where you perhaps need more control. I get the feeling you dont understand that part.

This has nothing to do with post production.

If you consider a normalized float representing a typical colour expression, say `0.5, 0.5, 0.5`, does anyone here know what colour that is? How about `0.1, 0.0, 0.0`? Can anyone identify that colour?

If your answer isn’t a question, you missed everything that I tried to express above. I’d encourage you to re-read it.

Changing those two simple values to hex codes, doesn’t change them, it simply makes the basic problem more obfuscated, and *it certainly doesn’t add to meaningfulness in any way*.

Hence, again, hex codes and integer representation are hot garbage. 😉

PS: Respect for your answer on the Devtalk forum, you stinking cheater. LMAO.

---

sundialsvc4 (sundialsvc4) 2018-11-15 02:55:06 UTC #30

The key thing to remember, I think, is that “that hex stuff” is an *output encoding.* It’s intended to tell a [stupid …] display device exactly what to do. “Image files” and “movie files” are directed at devices. They might need to consider gamma, color profiles, compression, and other characteristics (and limittions) of devices. You think about these things when you’re ready to prepare your render to be seen on some specific (or, generic) *device,* or devices.

Up to that point, the render information is really quite abstract – it’s just “floating-point numbers.” They’re normalized so that you can meaningfully work with them using mathematical operations, but it’s perfectly okay (at *this* point …) to have a pixel that is blacker than black or whiter than white, because it’s all just digital data. The *OpenEXR* and *MultiLayer OpenEXR* file formats are specifically designed to represent this sort of information, one file per frame: loss-less, accurate, and of course, big.

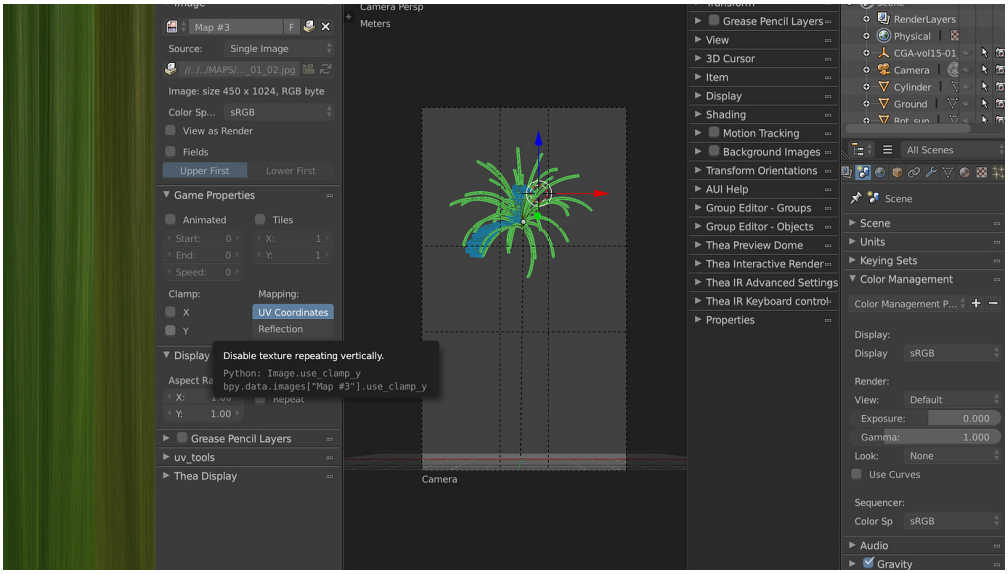
Execute your entire render sequence to produce a “final print” that consists of OpenEXR files. Then, and *only* then, concern yourself with producing the various presentation-files that you require. Each presentation file is generated, separately, from the same “final print” master source, never from one another. Encoding, color mapping, compression, and so forth is decided individually for each presentation.

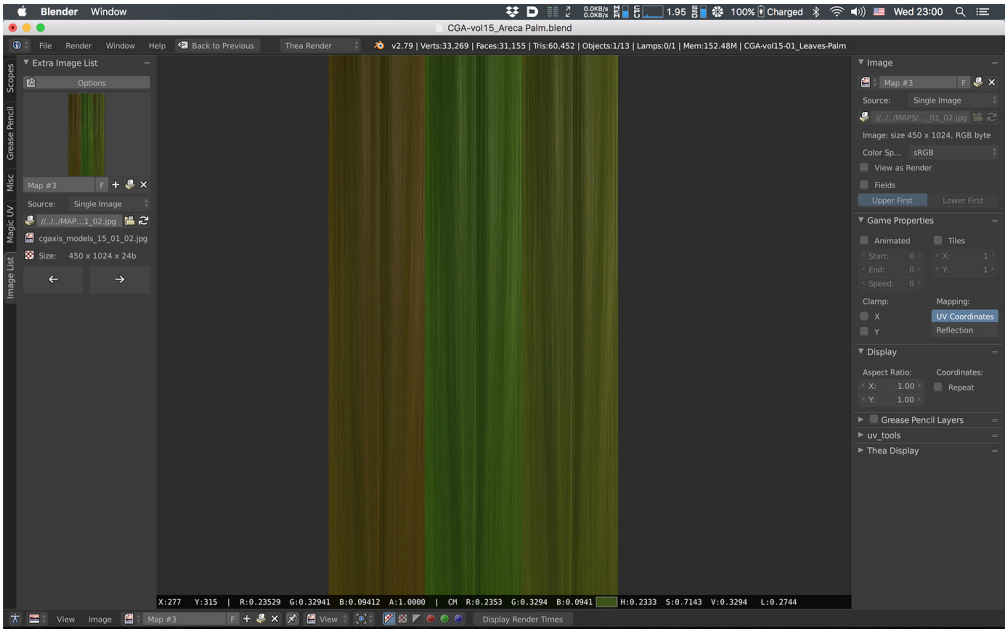
rombout (rombout) 2018-11-15 03:03:28 UTC #31

> troy_s:
>
> If you consider a normalized float representing a typical colour expression, say `0.5, 0.5, 0.5` , does anyone here know what colour that is? How about `0.1, 0.0, 0.0` ? Can anyone identify that colour?

I would think that when i use sRGB in scene > color settings and also sRGB in image > texture > color space that these numbers should be correct.

I'll think i do some test on this see if the numbers are the same as in other apps. See if i can find logic in this





sundialsvc4 (sundialsvc4) 2018-11-15 03:16:16 UTC #32

You could say that the first color is "mid-gray" and that the second is "dark pure red," and there are certain standard color-spaces so that we're not *completely* speaking-in-tongues. Displays including Blender's own display logic will use color-space information to determine what you see on the screen. But digital intermediate files during a multi-stage render might contain data that doesn't [yet …] meaningfully fit into a standard color space – and, it doesn't have to. (It's just data.)

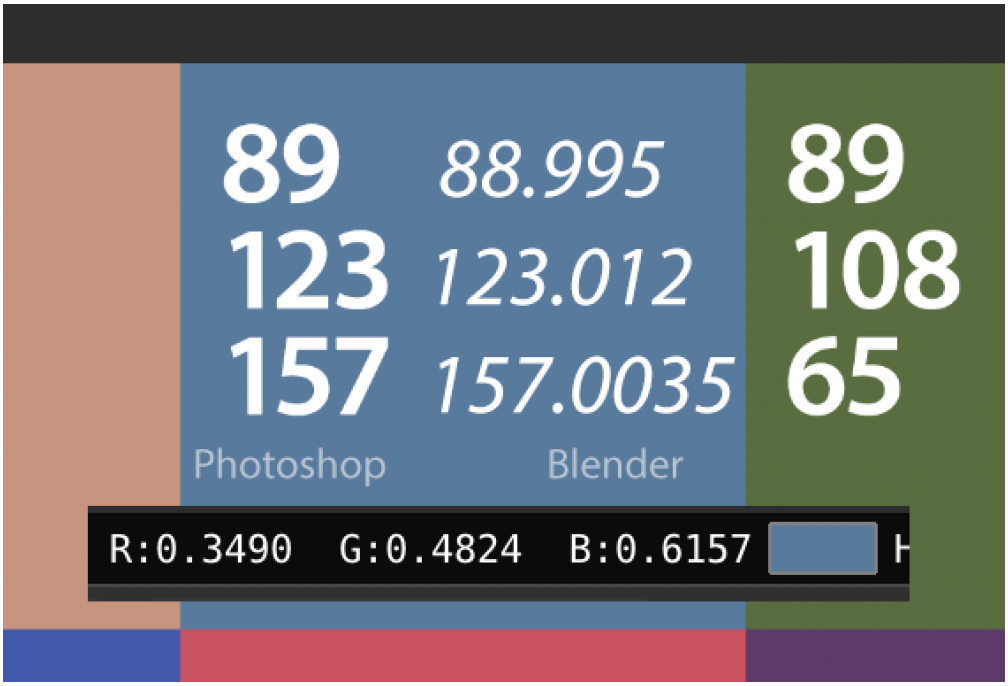
The doc files are pretty good on this:
https://docs.blender.org/manual/en/dev/render/post_process/color_management.html

---

rombout (rombout) 2018-11-15 03:32:15 UTC #33

Here's a file i tested and calculated. using that value*255 its near in most cases. Probably because i only see 4 digits of the float. If i could see more then it would be more precise?

Perpaps ill try to make an addon orso. But than i would need to figure out to combine it with the color spaces i guess.



It would be nice if Blender had more profiles. I use an external render engine which outputs images with ColorMath RGB or RGB color space. Its a bit vague which one it is. The devs them self so no space is applied, but i can only get the same image when i open it using either ColorMatch RGB or Apple RGB.

Problem is blender cant show these images properly 😟

---

pixelgrip (pixelgrip) 2018-11-15 04:11:29 UTC #34

i think that in the most cases,if you colorpick from a photo in the web ect,they are sRGB colors.because they are gamma corrected for the monitors.even most cameras have some rough gamma correction,iirc my old sony had a gamma of 2.5 as default.

if you colorpick from another software,you have to keep in mind,what you see is the result of a render ,and the color in the render, must not be the value, the material has in the nodes for example.
but in most cases you see sRGB values on screen.

and if you import a image tex, with the color option selected,then blender treat the colorvalues as sRGB too, and makes it linear for you.

so in most cases ,you can colorpick a value from a photo and asume, that the picked color is sRGB.

if you want to get sure,then you have to know, what the color is representing (sRGB or linear…)

---

troy_s (troy_s) 2018-11-15 04:38:02 UTC #35

> sundialsvc4:
>
> You could say that the first color is “mid-gray” and that the second is “dark pure red,” and there are certain standard color-spaces so that we’re not *completely* speaking-in-tongues.

It’s remarkably close to entirely speaking in tongues, more so in software similar to Blender.

The main issue is that folks forget that *RGB* is a model with no inherent meaning other than “it’s three lights”. XYZ, for example, is also three lights. Don’t expect the mixture to look anything like skin tones when dumped through an sRGB display without a transform though.

What colour is the mid-grey, for example? Is every RGB encoded image R=G=B for an achromatic value? Does `0.5` represent a mid grey in Filmic? A Filmic image? An albedo?

Key points, in order to be an additive RGB colour space according to the ISO, it must define three characteristics:

1. A clearly defined chromaticity for each of the three lights.
2. Transfer functions (EG: The sRGB OETF)
3. An clearly defined achromatic colour (aka White Point)

If we can’t identify all three facets, whatever it is, is not an RGB colour space.

> rombout:
>
> Here’s a file i tested and calculated. using that value*255 its near in most cases. Probably because i only see 4 digits of the float.

What happens in a render? As an albedo?

What was the reference space in Photoshop? What happens if you are in Adobe RGB? ACEScg? Do the values have absolute meaning?

> rombout:
>
> Problem is blender cant show these images properly

It can display them just fine, assuming you have the configuration set up accordingly.

> pixelgrip:
>
> and if you import a image tex, with the color option selected,then blender treat the colorvalues as sRGB too, and makes it linear for you.

Are you sure about that? What if it is a photo? 😉

---

pixelgrip (pixelgrip) 2018-11-15 04:52:17 UTC #36

> troy_s:
>
> Are you sure about that? What if it is a photo? 😉

not completly sure but i guess 😀

here someone has written,its based of encoding.hmmm

Color vs. Non-Color data

cycles, materials, color
asked by **icYou520** on **06:14PM - 01 Aug 17 UTC**

---

lolwel21 (lolwel21) 2018-11-15 15:17:24 UTC #37

Photos are (usually) encoded with an sRGB gamma, so yes, it always converts from sRGB to Linear if you have the ‘color’ option checked.
As a test, if you save your photo encoded with another color space (ex: Adobe RGB), then the photo will look weird in Blender.

---

troy_s (troy_s) 2018-11-15 23:14:29 UTC #38

> lolwel21:
>
> Photos are (usually) encoded with an sRGB gamma, so yes, it always converts from sRGB to Linear if you have the ‘color’ option checked.

Are you sure? 😉

The answer of course is “No this is entirey false” and segues down a deep rabbit hole. The TL;DR is that on a photo either a pure 2.2 power function or inversion of the sRGB OETF is not taking it to linear.

---

pixelgrip (pixelgrip) 2018-11-16 00:29:59 UTC #39

has a image some information data stored,like a header data or what encoding was used? some photos have this exif format.is something like that stored in the images,about encoding used ect?

i guess the answer is no,because the pixel dont know it was manipulated

---

troy_s (troy_s) 2018-11-16 01:04:11 UTC #40

> pixelgrip:
>
> i guess the answer is no,because the pixel dont know it was manipulated

Correct. There are many ways to pixel management, and sadly ICCs etc. can’t solve this.

The problem is that folks texture hunting are looking for *data*, not a “pretty picture”. Every photo online isn’t data, but rather some goddamnidiot’s idea of the aforementioned “pretty picture”.

It doesn’t take a rocket scientist to realize that an sRGB OETF or pure 2.2 power function encoded image is *not even remotely* a path to getting to linear; aesthetic images are mangled up beyond recognition. Further, the sRGB OETF / 2.2 power function approach is strictly for *display referred encoding*, so it can never deliver proper linear data.

This is why log transforms exist. And DPX. And float values. And…

FinalBarrage (FinalBarrage) 2018-11-16 01:47:04 UTC #42

That was actually the system flagging it, i cant see why, i will make sure to look into it.

---

cgCody (cgCody) 2018-11-16 02:03:51 UTC #43

Is idiot a flagged word? 😄

---

cgCody (cgCody) 2018-11-16 02:04:06 UTC #44

That answers that. Haha

---

troy_s (troy_s) 2018-11-16 02:04:26 UTC #45

You figured it out.

Also, I quote the message I received:

> Multiple community members flagged this post before it was hidden

---

troy_s (troy_s) 2018-11-16 02:05:06 UTC #46

This post was flagged by the community and is temporarily hidden.

---

cgCody (cgCody) 2018-11-16 02:06:46 UTC #47

I'll say. The website filtering is a bit too sensitive, and overly dramatic with the PM (which I also recieved.

---

troy_s (troy_s) 2018-11-16 02:07:19 UTC #48

Too late, *multiple* members of the forum all disagree with you.

FinalBarrage (FinalBarrage) 2018-11-16 02:07:34 UTC #49

I actually agree, ive removed it from the list of flagged words, but that is not a reason to use it. Careful with context, dont use it to harass other people

---

cgCody (cgCody) 2018-11-16 02:08:20 UTC #50

Thanks, master!

Haha sorry had to go there. 😄

Edit: now thats just sneaky! 😂

---

troy_s (troy_s) 2018-11-16 02:08:26 UTC #51

I'd probably slap a caveat into the default message that multiple members of the community haven't flagged it, and rather that the filternet caught the post.

---

rombout (rombout) 2018-11-17 20:09:13 UTC #52

> troy_s:
>
> What happens in a render? As an albedo?
> What was the reference space in Photoshop? What happens if you are in Adobe RGB? ACEScg? Do the values have absolute meaning?

I always use sRGB for my tex when rendering. Blender will show other spaces not correct if they are not in one of Blender spaces. Atleast as far is i have knowledge of this

> troy_s:
>
>  rombout:
>
> > Problem is blender cant show these images properly
>
> It can display them just fine, assuming you have the configuration set up accordingly.

All set to sRGB, i know how to properly do that, thats why the numbers are close. It will show my textures properly. But it wont show my render properly from the external render engine i use. As noted earlier, this one outputs images with RGB space. When i open it in Photoshop i either use ColorMatch RGB or Apple RGB.

---

troy_s (troy_s) 2018-11-17 21:57:41 UTC #53

> rombout:
>
> I always use sRGB for my tex when rendering. Blender will show other spaces not correct if they are not in one of Blender spaces. Atleast as far is i have knowledge of this

Good on you for spotting this. That is, the light ratios are expressed using the unique ratios of the particular space. You *can* of course transform most spaces into other spaces, if you know how. If you have something you need, feel free to link it and I'll try my best to get you a solution.

> rombout:
>
> But it wont show my render properly from the external render engine i use. As noted earlier, this one outputs images with RGB space. When i open it in Photoshop i either use ColorMatch RGB or Apple RGB.

This is fishy given that "RGB" isn't a space, but rather a model.

If you know the details, and your engine dumps a float framebuffer or whatever, it should be almost trivial to get it working properly.

I *can* say with a degree of speculative certainty, that ColorMatch or Apple RGB is completely wrong. We should sort that out.

---

burnin (burnin) 2018-11-18 01:23:30 UTC #54

Funny, how much more comprehensible it all was in the analog days.
While today most can't even grasp the fundamental difference between a photo, digital print and digital information for a machine to represent.

"Am i screaming, saying it loud enough?"

As mind starts twisting thoughts out of control it plays trick on own personality… why i love my sexy eyes, no matter what numbers say. I have intentionally couple of decalibrated displays (one never knows all the clients perks). If it looks good on all, then CC/tone mapping is fine.
Discretion and accepting, knowing that tolerances are just a part of getting the work done, are well advised.

Keep up the spirit, next generations will love reading this.
🤗

---

rombout (rombout) 2018-11-18 01:49:51 UTC #55

> troy_s:
>
> I *can* say with a degree of speculative certainty, that ColorMatch or Apple RGB is completely wrong. We should sort that out.

What exactly do you mean by “completely wrong”?

When can use relight which is a light buffer but this output is in a certain file type. Im not sure you could decode this.

> Good on you for spotting this. That is, the light ratios are expressed using the unique ratios of the particular space. You *can* of course transform most spaces into other spaces, if you know how. If you have something you need, feel free to link it and I’ll try my best to get you a solution.

The engine uses this lib to save its images, **http://freeimage.sourceforge.net/**
Ive did try and ask around there i believe on time. DId really got far there,
https://sourceforge.net/p/freeimage/discussion/36110/thread/a4ee3cff/?limit=25#0eb7

One of the devs in the su version stated nothing is added. But i need a profile otherwise it wont match the profile or preview as it looks in the engine. They say the engine use linear settings, but never really got an answer what display space it uses

Ive attached a psd file, in the info it doesnt contain a profile. So im really in the dark what the proper method is opening these files. For years im opening them as ColorMatch because they than look just like in the render engine darkroom.

☐ **Vacuumbottle_thea-v2_8bit.psd.zip** (4.9 MB)

---

troy_s (troy_s) 2018-11-18 04:50:51 UTC #56

> rombout:
>
> What exactly do you mean by “completely wrong”?

Exactly what I typed; completely and utterly wrong.

> rombout:
>
> The engine uses this lib to save its images,

Hot stinking garbage. Unsurprisingly, your output is broken.

> rombout:

> But i need a profile otherwise it wont match the profile or preview as it looks in the engine. They say the engine use linear settings, but never really got an answer what display space it uses

That sure sounds like clueless developers. I would run far away from whatever you are using that is using it, and any of the wisdom the developers are offering.

If it were “doing nothing” to your image, the linear encoding would be maintained in the file, which clearly isn’t the case if slapping a broken profile on top of the junk appears to do something correct.

My guess is that you are noticing a difference in the transfer function, which means the clueless library is completely mangling up the output.

Just speculating here, but I am now sadly too well versed in untangling absolute crap that some random developer of some random file library has done to colour encodings.

Can you do a clean render of a very high intensity scene with values that hit scene referred 16+ and dump a TIFF out from this godforsaken dumpster fire?

---

rombout (rombout) 2018-11-18 08:08:15 UTC #57

> troy_s:
>
> Exactly what I typed; completely and utterly wrong.

Still not sure what you mean by wrong… I open the file in PS using that profile otherwise it wont match the preview in renderer.

Im not running away from an engine ive been using for 10 years… Sorry to much time investment here.

> Can you do a clean render of a very high intensity scene with values that hit scene referred 16+ and dump a TIFF out from this godforsaken dumpster fire?

By 16+ you mean 16bit i guess or not. I can only save tiff in 8 bit. we can save EXR in 16 bit or HDR. But both of these look different as well but thats normal for the HDR case than in photoshop.

I can use a different app which can load the light buffer and save out a 16 bit psd. But that would not be any different that the earlier 8 bit i posted. Both dont have a profile attached

Here’s an EXR image. it opens as 32bit file. I never use EXR because it opens with different toning in photoshop. Ive also attached and PNG which is converted to sRGB which has the proper toning, this is open as ColorMatch RGB and than converted to sRGB. I resembles the same toning as the preview in the render engine.
☐ **Vacuumbottle.exr.zip** (3.4 MB)

troy_s (troy_s) 2018-11-18 16:00:04 UTC #58

> rombout:
>
> Still not sure what you mean by wrong… I open the file in PS using that profile otherwise it wont match the preview in renderer.

Did you render using Colormatch RGB primaries?

If not, then wrong.

Now on the upside, the Colormatch primaries are close-ish to REC.709 lights, but with a transfer function of 1.8, which means you are seeing the transfer function differences.

So applying an incorrect profile on your data to adjust the transfer function is the wrong approach. There is a more appropriate method, including getting directly to an sRGB encoded image.

> rombout:
>
> By 16+ you mean 16bit i guess or not.

No, I mean checking the scene referred values. In the UV editor you can sample the values via mouse click. When you do so, the left side of the information panel shows the reference space values that extend from zero to infinity. The right side show the after-colour transformed values.

I suspect your software is broken and only using some hard coded transfer function. Perhaps that can be tweaked and we can use a more appropriate transform for you.

First I would need to see what is going on with the file encoding. To do this, I would need:

1. An EXR with large exposures in the scene referred domain
2. An “untouched” display referred output version. That is, a PNG or a TIFF at 8 bit in addition to the EXR.

Just make sure that your test has high exposure values that extend upwards to 16.0+ or higher.

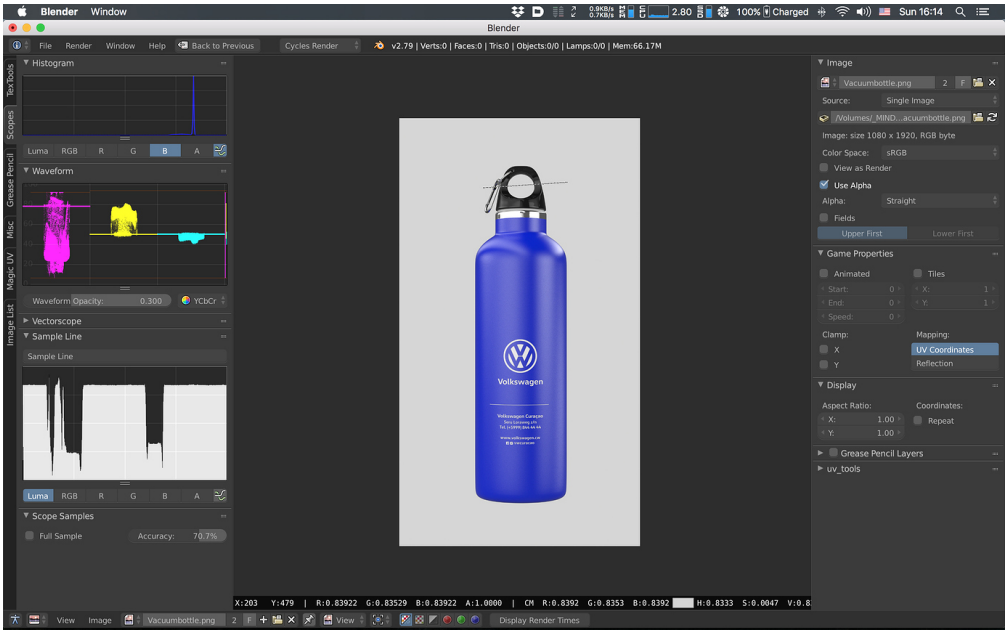
Second, **can you test this profile instead of Colormatch RGB or Adobe RGB**. In theory it would seem this would be the correct profile for the way the software is mangling up the work, but hard to be certain.

---

rombout (rombout) 2018-11-18 20:14:39 UTC #59

> troy_s:
>
> No, I mean checking the scene referred values. In the UV editor you can sample the values via mouse click. When you do so, the left side of the information panel shows the reference space values that extend from zero to infinity. The right side show the after-colour transformed values.

Not sure which panel you mean. I only see the color picker info at the bottom and the scopes. Dont know which info i should see on the left?



Ill try that profile and see what this does.

Peetie (Peetie) 2018-11-18 20:24:23 UTC #60

On the render itself. Then you see values in the black bottom bar. On the left: scene refered, on the right after transform.

---

rombout (rombout) 2018-11-18 20:26:10 UTC #61

That what my image is showing, i was referring to this text.

> No, I mean checking the scene referred values. In the UV editor you can sample the values via mouse click. When you do so, the left side of the information panel shows the reference space values that extend from zero to infinity. The right side show the after-colour transformed values.

Peetie (Peetie) 2018-11-18 20:32:23 UTC #62

And then you can click in the picture and you see the values changing. If not, I must understand it wrong.

---

troy_s (troy_s) 2018-11-18 20:35:07 UTC #63

The problem is you are using an already display referred image as a reference, which is already mapped to the output referred domain.

Load your model in Cycles, and increase a light intensity. You will see the left values as **@Peetie** has pointed out, extend from zero to infinity.

Those are the actual scene referred values. Only an EXR is capable of revealing the actual scene values easily, assuming the rendering engine isn't mangling those up.

---

rombout (rombout) 2018-11-18 20:50:15 UTC #64

> troy_s:
>
> Those are the actual scene referred values. Only an EXR is capable of revealing the actual scene values easily, assuming the rendering engine isn't mangling those up.

Im not sure i get this part though. Where should i be able to see those values, in those scopes or at the bottom. The bottom always shows so i dont think that is the one. But than the scopes always show info when i open an image.

PS that Elle profiles comes close, but it looks flatter in darker parts. The vibrance was almost the same but some of dark/materials are less dark. The Elle sRGB look a bit creamy, sort of dreamy (not sure how to explain it better)

EDIT
It seems they are about the same. I noticed some difference in images when i saved out a PSD. They look different as well. So i guess something is happening there as well.

Left opened as sRGB-elle-V2-g18, opened as ColorMatch RGB

Below is screengrab of the application with a loaded image buffer.

troy_s (troy_s) 2018-11-18 21:34:19 UTC #65

> rombout:
>
> that Elle profiles comes close, but it looks flatter in darker parts. The vibrance was almost the same but some of dark/materials are less dark. The Elle sRGB look a bit creamy, sort of dreamy (not sure how to explain it better)

Let me be very clear; this isn't an aesthetic option. *Incorrectly tagging your imagery to overcome broken output is not an option*.

Aesthetic choices are intended to be made during a grade, *not* via profiles which define the colourimetry of the encoded values.

What it seems like is the software is entirely broken, and the output is being encoded to legacy Apple standards that specify a 1.8 transfer function. I haven't evaluated the process yet however, so I can't state with certainty yet until I get the sample EXR. Hopefully we can solve this broken pipeline and avoid the incorrect tagging. The Elle 1.8 is a transfer function of 1.8, with appropriate primaries. It is essentially identical to the completely incorrect ColorMatch profile, without the incorrect primaries.

> rombout:
>
> Im not sure i get this part though. Where should i be able to see those values, in those scopes or at the bottom.

Make a default scene using Cycles and render into the UV Image Viewer. Left clicking will reveal the black bar along the bottom, and show the sampled RGB values from the scene as per previous descriptions.

rombout (rombout) 2018-11-18 21:37:41 UTC #66

Yes i got that part, i alread showed that in earlier images. I though you meant something else.

PS what is so bad about the ColorMatch RGB profile. You make it sound like its something really bad. I read some where its kind a outdated or so.

PS i did add an EXR earlier, isnt that one any good?

---

troy_s (troy_s) 2018-11-18 23:51:19 UTC #67

> rombout:
>
> PS what is so bad about the ColorMatch RGB profile.

It isn't how the values are encoded, so it describes an alternate colour encoding entirely. We need to root out exactly why your rendering engine is rendering whack values and find the appropriate method to get it to render proper sRGB values.

> rombout:
>
> PS i did add an EXR earlier, isnt that one any good?

The easiest way for me to diagnose what is going on is to have a reference render of a high dynamic range scene, which your bottle is sub optimal. I would need a render of both EXR encoding and the TIFF with no mangling through other software. Directly from the engine.

There is a good chance you can improve your renders by a not insignificant degree by sorting this out.

---

rombout (rombout) 2018-11-19 03:30:46 UTC #68

These images are straight from the engine. Yet i cant tell or know what is done to them. We have a so called Darkroom where we can set all kind of camera settings like iso, f-stop, gamme etc etc. So this is post work and thus the images are probably altered there.

i dont know much about what is done exactly at that point. There is reversed gamma correction applied i read. The workflow is done linear and that is all i know.

Cessen (Cessen) 2018-11-19 04:46:41 UTC #69

> troy_s:
>
> The main issue is that folks forget that *RGB* is a model with no inherent meaning other than "it's three lights". XYZ, for example, is also three lights.

Just jumping in to say that if you mean the **CIE XYZ colorspace**, that's not correct. X, Y, and Z in CIE XYZ don't represent primaries. They don't even directly represent the spectral responses of the cones in human eyes (~~although XYZ is derived from the~~ [EDIT: no it's not, see troy's correction below] CIE LMS colorspace, which does).

Also, just for kicks, I'll point out that doing rendering in *any* RGB space is technically wrong. Tristimulus values aren't how color works, they're just how human color *perception* works.

To do light transport correctly (in terms of color, at least), you need to do spectral rendering. Not even for fancy spectral effects, just for proper color handling. The paper "**Physically Meaningful Rendering Using Tristimulus**

Colours" by Meng et al. covers some–but not all–of the issues with RGB-based rendering. In short, when you use RGB for rendering, you're pretending that light spectrums behave like human color vision, but they absolutely don't. It is in many respects much like confusing scene-referred and display-referred color.

Thankfully, in practice RGB-based rendering still works well enough for most purposes–at least when the goals are artistic rather than simulation-oriented. But since we're going down rabbit holes anyway, I thought this would be fun. 😉

---

troy_s (troy_s) 2018-11-19 05:07:04 UTC #70

> Cessen:
>
> Just jumping in to say that if you mean the **CIE XYZ colorspace**, that's not correct. X, Y, and Z in CIE XYZ don't represent primaries.

sigh

It's a three light system. They don't represent physically plausible colours, but it is a three light system nonetheless.

> Cessen:
>
> They don't even directly represent the spectral responses of the cones in human eyes

False. Do your homework before jumping in. The Wright Guild experiment is they basis for the CMFs which derive the XYZ to spectral response connection. Heck, it is called a spectral locus for a reason.

> Cessen:
>
> Also, just for kicks, I'll point out that doing rendering in *any* RGB space is technically wrong. Tristimulus values aren't how color works, they're just how human color *perception* works.

sigh

It's me Nathan! Hello!?

Cessen (Cessen) 2018-11-19 05:44:50 UTC #71

Sorry, I didn't intend my post to be confrontational, rather as adding some fun additions to the discussion. The rabbit hole goes very deep, so I wanted to peel back another layer. Apologies if I came across otherwise.

> troy_s:
>
> > They don't even directly represent the spectral responses of the cones in human eyes
>
> False. Do your homework before jumping in. The Wright Guild experiment is they basis for the CMFs which derive the XYZ to spectral response connection. Heck, it is called a spectral locus for a reason.

You're confusing XYZ with **LMS**. I did the same for a long time, it's an easy mistake to make. In LMS the three values *directly* represent the spectral responses of the human eye's three types of cones (sensitive to "long", "medium", and "short" wavelengths of light). You can easily transform between XYZ and LMS, and they both represent the entire range of human-visible color. But XYZ is not a *direct* representation of the human cone responses like LMS is. That's what I was getting at.

> troy_s:
>
> It’s a three light system. They don’t represent physically plausible colours, but it is a three light system nonetheless.

Well… so this is interesting: you actually can realize lights with emission spectrums that match the spectral curves of X, Y, or Z. But (as I understand it) it wouldn’t be especially useful to do so. Which is what I meant in more precise terms. This is in contrast to e.g. the Rec709 primaries which are not just physically realizable, but are also *useful* to physically realize because they produce the intended colors for a human viewing them (which is precisely what sRGB displays attempt to do).

We may be speaking at cross-purposes, though. I think we might mean subtly different things by “three lights”, so this is likely a terminology mismatch (which I may indeed be in the wrong about), but unless you also disagree with my paragraph above, I don’t think we have any disagreement of substance on this point.

> troy_s:
>
> It’s me Nathan! Hello!?

Yes Troy, I know! I strongly recommend reading the paper I linked in my previous post, if you haven’t already, as that’s the main paper that started me down the rabbit hole about the shortcomings of RGB (or tristimulus in general) in rendering.

EDIT:
Having said all of this, I was indeed incorrect in saying that XYZ is derived from LMS. It was derived from CIE RGB (or rather the CMF’s, as you correctly noted). But CIE RGB also doesn’t directly represent the cone responses.
