3D графика и трасиране на лъчи v.2.0

http://raytracing-bg.net/

Тема 2

Цветови пространства
Перцепция на цветовете
Рендериране със широк динамичен обхват

Съдържание

- Кратка история на представянето на цветовете в компютърните системи
- Светлината като физично явление
- Перцепция на цветовете – трикомпонентният модел
  - И разликата му с по-сложни модели
- Цветовото пространство RGB
- Други цветови пространства
- Рендериране с широк динамичен обхват (HDR)

История на графичния хардуер

- Монохромни графични карти
- Първият цветен адаптер за PC – CGA (1981г)
  - 320x200, 640x200
  - 16 предефинирани стандартни цвята (палитра); до 4 цвята едновременно на екрана за 640x200
  - Цифров интерфейс
- EGA
  - 640x350 @ 16 цвята
  - 16-те цвята се избират от палитра с 64 (2 бита на канал)

История на графичния хардуер

- VGA (1988г)
  - Революция в много отношения
  - Аналогова връзка
  - 640x480@16 цвята, 320x200@256 цвята, избрани от палитра с 2^{18} цвята (6 бита на канал)
    - Съвременните TN матрици на LCD мониторите също са едва 18-битови!
- След VGA, IBM изпускат контрола в областта
  - Има опити за стандартизация (VESA), но на практика започват да се ползват изключително видео-драйверите

История на графичния хардуер

- SVGA
  - 16 и 32-битов цвят; вече без палитри
  - 16.7 милиона цвята, срещу 10 милиона – видими от окото
  - 10 и 12 бита на канал в повечето видеокарти, но няма добра софтуерна поддръжка
    - Например, 30-битов цвят
- Бъдещето
  - High-contrast монитори (200,000:1)
  - High-DPI монитори (IBM T220: 3840x2400@22“)

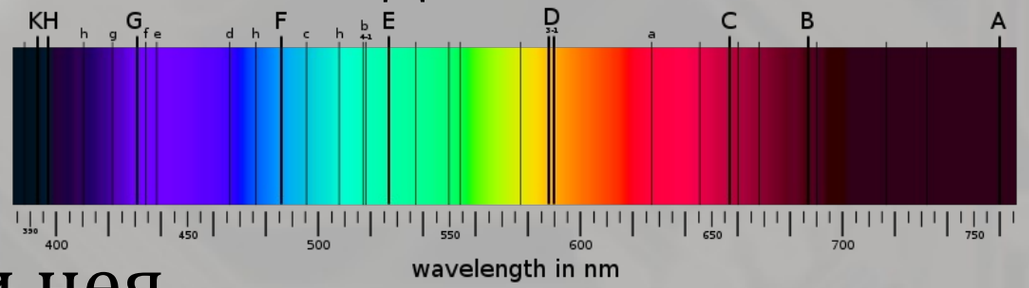
Светлината като физично явление

- Част от електромагнитния спектър, с дължина на вълната от 380nm÷750nm
  - Няма рязка граница, но чувствителността на окото рязко спада в краищата, докато е най-чувствително в областта на зеленото, около 555nm

  - Част е от един малък спектрален „прозорец“, който земната атмосфера пропуска
  - Хората възприемат цветовете чрез три вида фото-рецептори – червени, зелени и сини – и не могат да определят точния спектър на светлината, а само силата на дразнение на трите си фото-рецептора

Светлината като физично явление

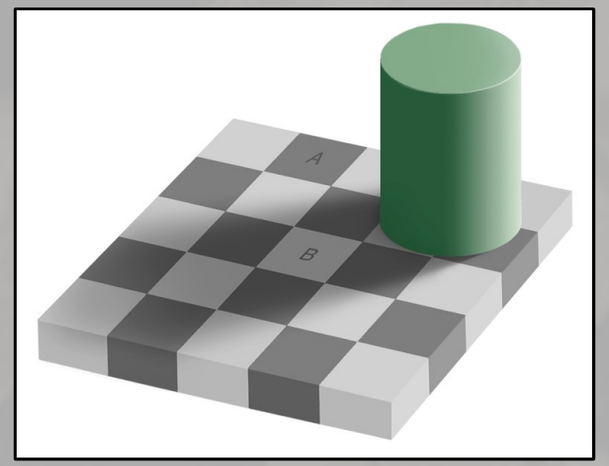
- Например, слънчевата светлина има много „дупки“ (спектрални линии), но ние не можем да ги засечем и я приемаме за бяла



- Монохроматична светлина („чист цвят“) – при нея, излъчването е концентрирано в един тесен участък
  - Лазерите излъчват монохроматична светлина
  - Окото разпознава различните монохроматични цветове много добре. Дори само 2 nm разлика между два монохроматични източника е осезаема

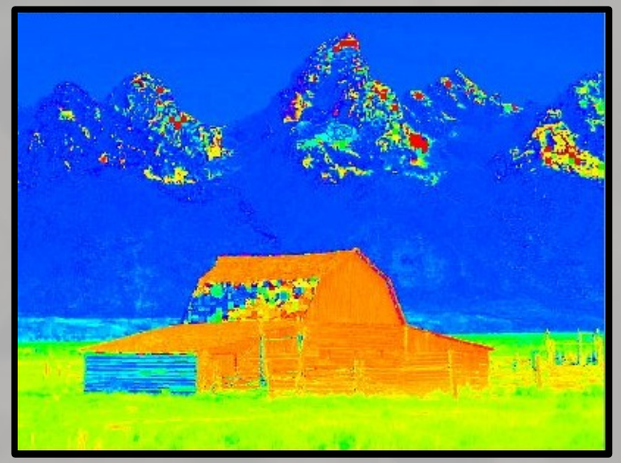
Човешкото зрение

- След подходяща акомодация, клетките в окото реагират дори на единичен фотон
- Огромен обхват на яркостите, които могат да се възприемат – 1:1,000,000,000 (най-тъмна до най-ярка)
- Окото не „мери“ с абсолютни стойности, а засича контраста между съседните части на изображението
  - Демонстрацията вдясно се базира на това: квадратчетата A и B са с еднакъв цвят!
  - Окото засича до 2% контраст



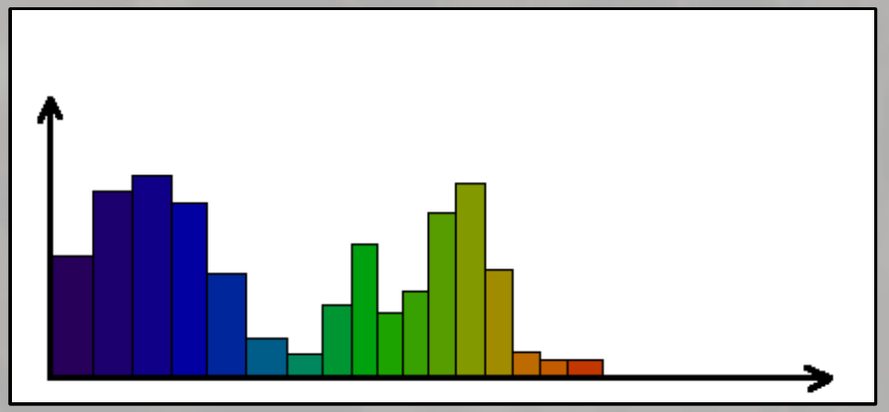
Човешкото зрение

- Човек е много по-чувствителен към яркостта, отколкото към цвета
- Чувствителността зависи от дължината на вълната – най-голяма е при зелено-жълтите цветове



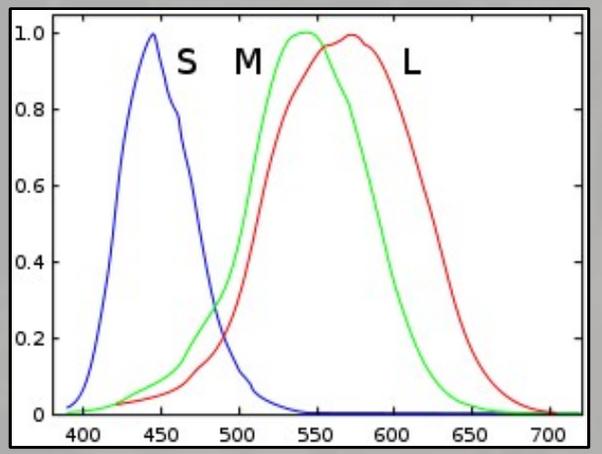
Представяне на светлината

- За да запишем пълната информация за дадена светлина, може да разбием видимия спектър на много малки парчета и да измерим излъчването във всеки от тях (напр. във ватове на стерадиан)
- Но ще ни трябват прекалено много деления, заради споменатата чувствителност при монохроматична светлина



Представяне на светлината

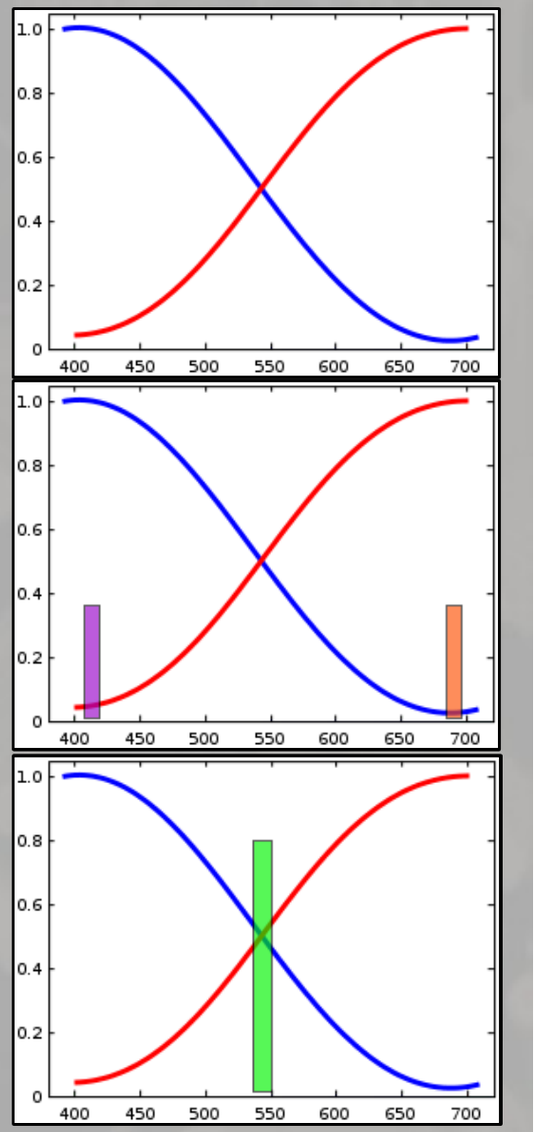
- Вместо това, можем да изградим цветова система, базирана на трите вида фото-рецептори в човешкото око
- Такава система е недостатъчна за някои специфични случаи
  - Симулиране на физичния ефект „дисперсия“ (напр., дисперсия през призма), за който е необходим пълния спектър
  - Ще дадем и други примери



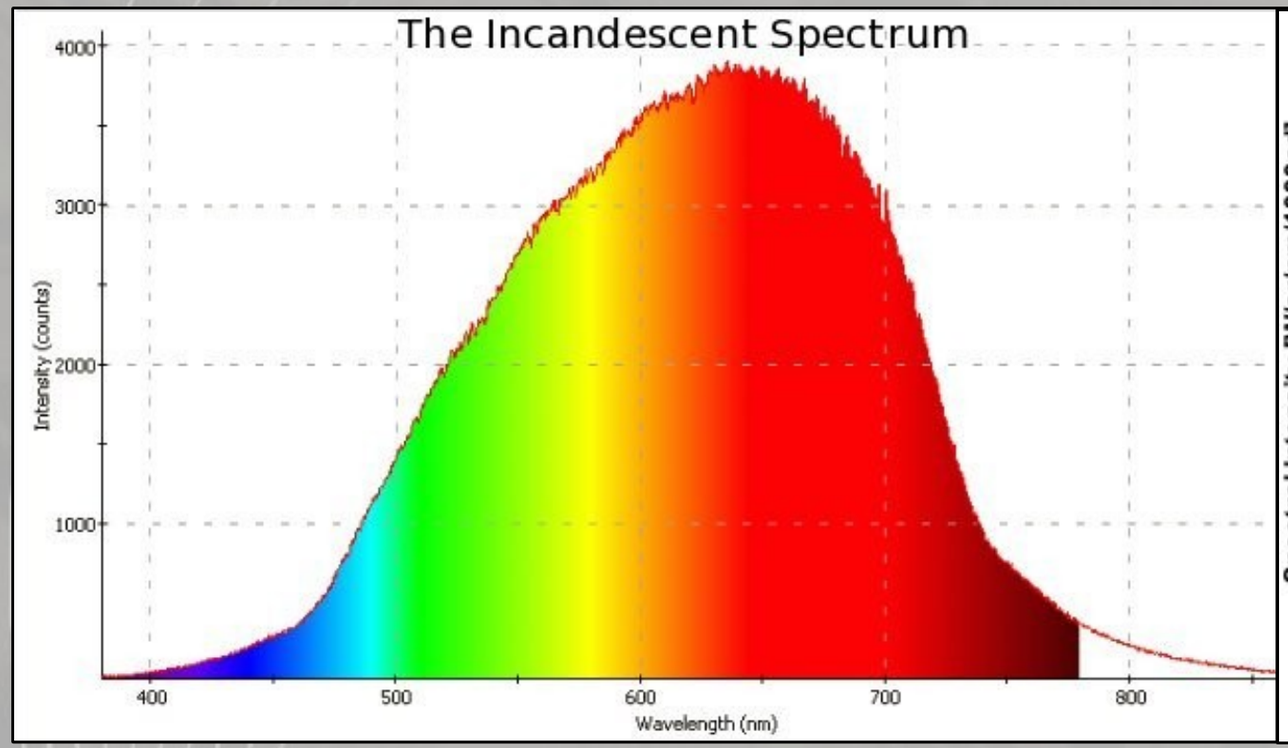
Чувствителност на трите вида фото-рецептори в човешкото око (нормирани)

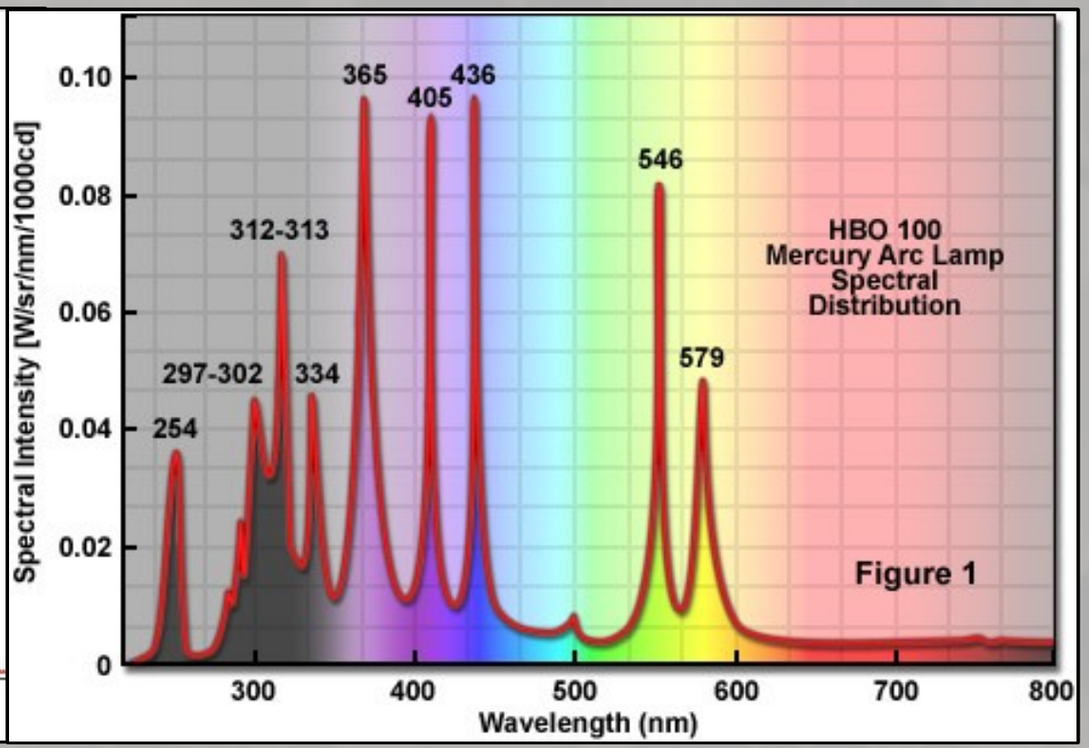
Недостатъци на трикомпонентния модел

- Недостатъчно представяне на информацията, когато светлинните източници имат тесен спектър на излъчване или остри „пикове“, както и при граничния случаи с монохроматична светлина.
  - Мисловен експеримент: ако човешкото око имаше само два вида рецептори, сини и червени, то монохромна зелена светлина щеше да изглежда еквивалентна на смес от монохромни синя и червена



Недостатъци на трикомпонентния модел

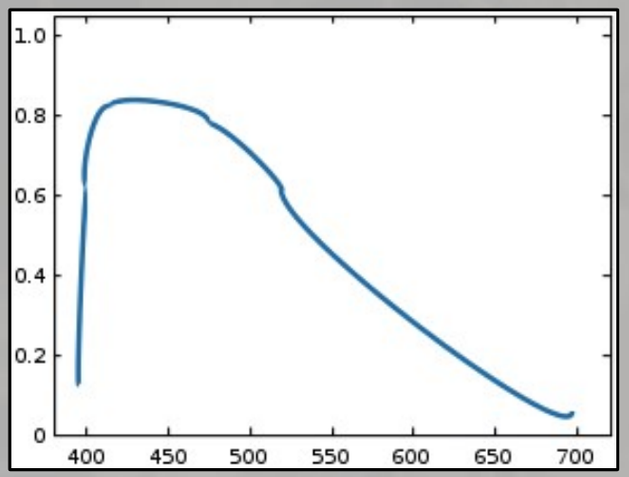


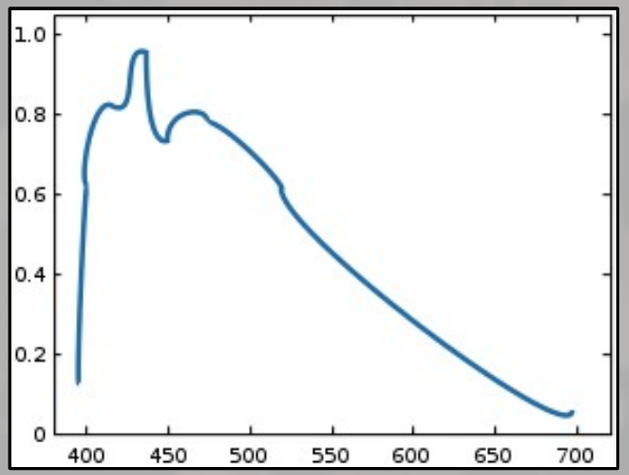


- Лампа с нажежаема жичка (вляво) и живачна (вдясно)

Недостатъци на трикомпонентния модел

- Отражателност на два вида боя
- Боядисваме нещо с двата типа боя
  - При нажежаемата жичка, заради гладкия спектър, различните бои ще изглеждат еднакво
  - При живачната лампа, те ще са различни, заради съвпадението на пиковете на излъчването на лампата и на отражението на боята





Недостатъци на трикомпонентния модел

- Същия вид проблеми може да имаме и при заснемане на обектите с фотоапарат. Там RGB сензорите може да имат различна функция на чувствителност спрямо „човешките“
- Използването на RGB модела е стандартно в повечето приложения на рейтрейсинг алгоритмите. Но ако все пак искаме да избегнем споменатите проблеми, има и решение за тях, но изисква повече сметки
- Нашия рейтрейсър все пак ще е RGB-базиран

RGB моделът

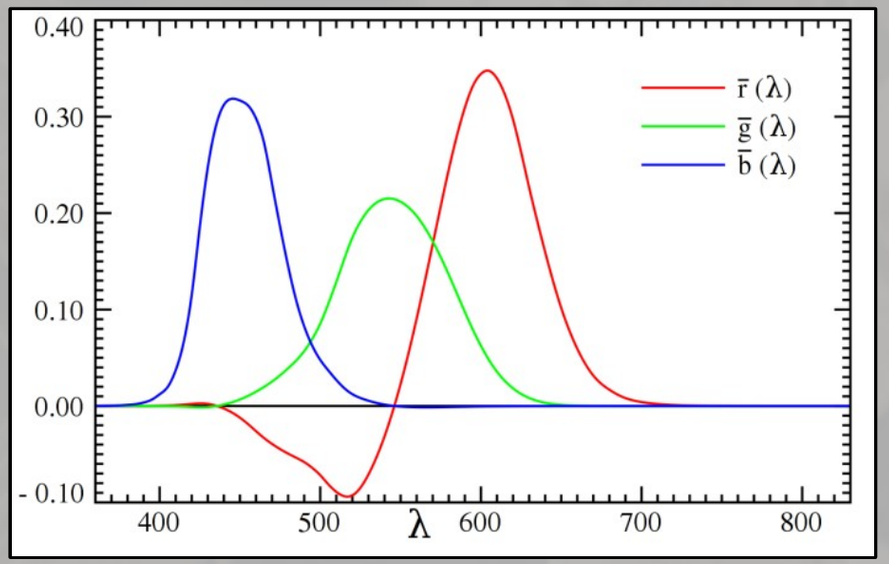
- Всеки цвят се представя като смесим определени количества от трите основни цвята
- Описва се с тройка (R, G, B) стойности
- Основните цветове не са твърдо фиксирани (зависят от възпроизвеждащото устройство)
- Прецизно дефиниран стандарт за цветовете е CIE RGB от 1931г.
  - Базиран на практически експерименти с хора

CIE RGB цветово пространство

- При него, използваните основни цветове R, G и B са твърдо фиксирани монохроматични светлини
- По време на експеримента, пред човек е сложен екран, в едната половина на която е тестовият цвят (този, на който искаме да намерим RGB стойностите), а в другата половина – генерираният RGB цвят, като са на разположение три контрола, за силата на всеки един от R, G и B компонентите.
- Експериментаторът наглася R, G и B, докато двете половини „съвпаднат“

CIE RGB цветово пространство

- Така, за всеки монохроматичен източник могат да се намерят RGB стойности
- Графиката по-долу показва участието на трите основни цвята за всяка дължина на вълната от видимия спектър
- Графиката е специфична за CIE RGB; при друг избор на основните цветове ще изглежда различно!



CIE RGB цветово пространство

- Има и отрицателни части от графиката: където не е било възможно да се постигне съвпадение чрез настройването само на RGB цвета, добавяли са RGB цвят към „тестовата“ част, като добавеният интензитет се интерпретира като отрицателен в графиката
- Следователно, не можем да постигнем всички възможни цветове само с RGB стойности в [0.0, 1.0]
  - Практически пример: повечето монитори не могат да изобразят наситено жълто

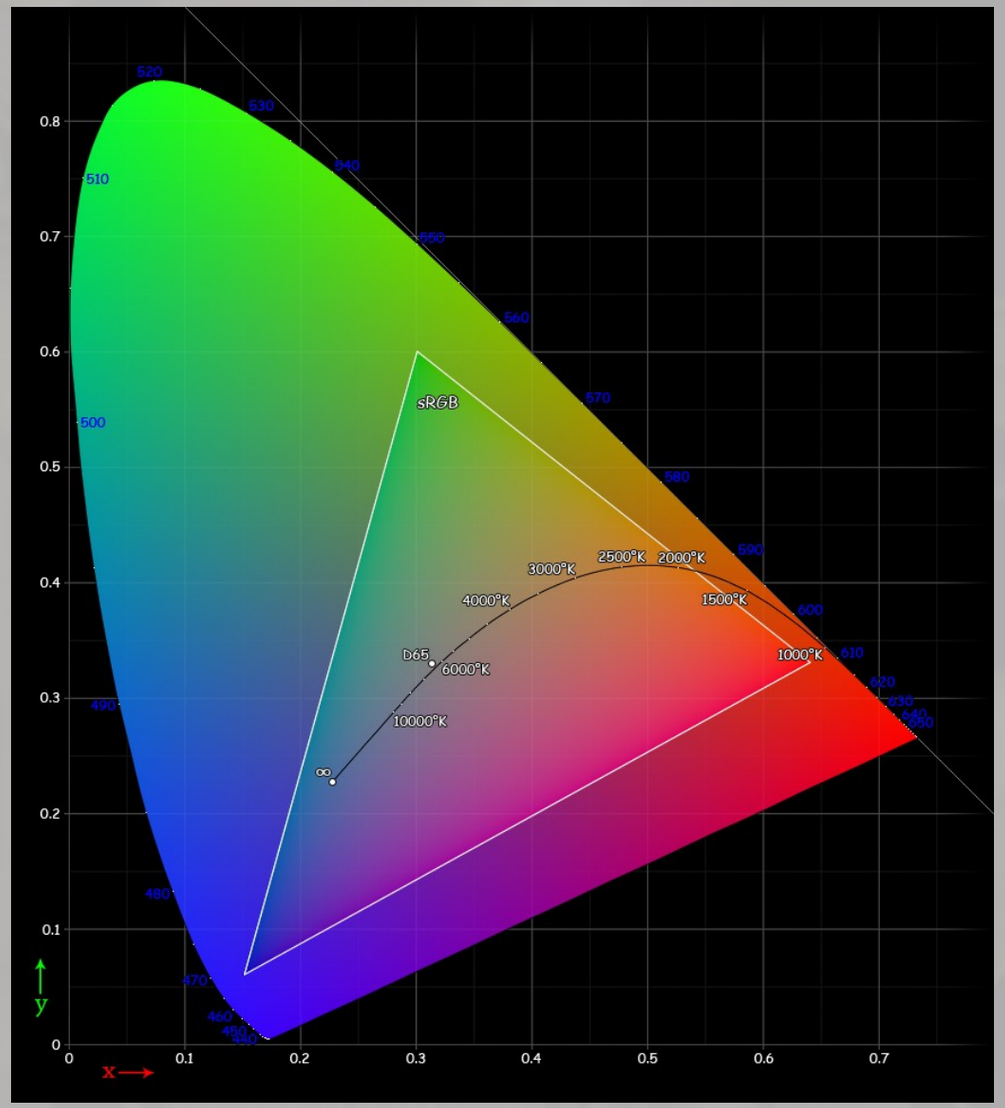
RGB в практическите устройства

- Заради споменатите недостатъци, CIE разработват и тяхното XYZ цветово пространство (XYZ ≈ RGB)
  - Няма отрицателни части
  - Всички цветове могат да бъдат представени, но за сметка на това, X, Y и Z цветовете са „изкуствени“ – излизат извън видимите от човек
- На практика повечето възпроизвеждащи устройства дефинират основните си цветове в XYZ пространството и го ползват за еталон за точно възпроизвеждане на цветовете

Цветови триъгълник

- На практика, оттенъка на даден цвят има две степени на свобода (RGB(0.8, 0.8, 0) и RGB(0.5, 0.5, 0) все е жълто, само второто е по-тъмно). Затова „чист“ оттенък имаме когато R+G+B = 1.
- Можем да съпоставим „чистите“ оттенъци с координати в 2D триъгълник (върховете на който са чисто червено/зелено/синьо). Този триъгълник обикновено се ползва при дефиниране на конкретно RGB цветово пространство (CIE 1931, sRGB и пр.)

Цветови триъгълник



Цветови триъгълник

- Това, което се вижда от цветовия триъгълник е, че за конкретно RGB пространство, когато чистите R, G и B цветове са реални, то винаги има оттенъци, които не могат да се изобразят в това пространство
  - Освен ако не разрешаваме стойности извън [0..1]

sRGB цветово пространство

- sRGB се използва за повечето обичайни устройства (монитори, фотоапарати, принтери...)
- Добре дефинирани основни цветове
- Gamma correction – заради особености на CRT кинескопите и човешкото зрение, яркостта не е линейно зависима от подаваното напрежение, а се мени приблизително по закона $\text{Output} \propto \text{Input}^{\gamma}$, където γ е гама-стойността; обикновено $\gamma = 2.2$
  - При sRGB, стойностите на R, G и B са предварително компенсирани за гама (неутрализират ефекта при CRT-тата)

sRGB цветово пространство

- Реалната информация, записана в картинка от подходящ формат (напр., JPEG файл) е нелинейна – има gamma компенсация; след като я декомпресираме и прочетем (например, чрез libjpeg), тя вече е прехвърлена в линейното RGB пространство
  - Т.е. няма нужда да се интересуваме от gamma компенсацията; тя се върши автоматично (от libjpeg, при четене) и пак автоматично (от монитора, ако е нужно, при „писане“)

Линейно RGB пространство

- В графичния софтуер, sRGB пространството е неудобно заради нелинейността си (усложнява се смесването на цветовете и пр.), затова се работи в линейно RGB пространство, а превръщането към sRGB се оставя като последна стъпка преди визуализация
- В междинните сметки може да работим със стойности извън [0.0, 1.0]
  - Но трябва да отрежем стойностите преди да ги покажем на екрана

Операции в линейното RGB пространство

- Събиране
  - $C = A + B \Leftrightarrow (R_c, G_c, B_c) = (R_a+R_b, G_a+G_b, B_a+B_b)$
  - Например, смесването от светлините на два прожектора
- Умножение
  - $C = A \cdot B \Leftrightarrow (R_c, G_c, B_c) = (R_a*R_b, G_a*G_b, B_a*B_b)$
  - Например, светлината от жълт прожектор, отразена от синьо-зелен лист е зелена

Операции в линейното RGB пространство

- Умножение по коефициент
  - $B = A * k \Leftrightarrow (R_b, G_b, B_b) = (R_a*k, G_a*k, B_a*k)$
  - Например (при k = 0.5) – светлина, след като е минала през филтър, който поглъща 50%
- Често използвана комбинация от умножението по коефициент и събирането е операцията „разделяне“:
  - $A = C * \alpha$; $B = C * (1 - \alpha)$ $(0 \leq \alpha \leq 1)$
  - Например, светлинен лъч среща стъкло. Светлината C се разделя на отразена (A) и пречупена (B)

Операции в линейното RGB пространство

- Намиране на интензитет на цвета
  - Директен подход: $I = (R + G + B) / 3$
  - Перцептуален подход: $I = (R * 0.299 + G * 0.587 + B * 0.114)$
- Близост на два цвята
  - $D = |R_a - R_b| + |G_a - G_b| + |B_a - B_b|$
    - Но това не е перцептуална близост; по-нататък ще дадем пример за по-добър подход

RGB цветово пространство

- В 3D графиката, RGB стойности извън обхвата [0.0, 1.0] са разрешени за светлината; но са невалидни за *множители*, като например за цвета на даден обект
  - Ако интензитета на цвета на обекта е над 1, то светлината, след отражение, ще се увеличава, вместо да намалява; получава се светлинна експлозия
  - На практика, дори стойности близки до 1 са опасни, защото предизвикват твърде много трасиране на светлинните лъчи

RGB цветови пространства

- Съществуват много дефиниции на RGB цветови пространства
- Повечето са линейни и могат да се превръщат едно в друго чрез просто матрично умножение
- Ако не разрешаваме стойности извън [0.0, 1.0], нито едно от тях не може да изобрази всички цветове, видими от човека
- Има и други линейни пространства, които не са RGB

Пример – YUV цветово пространство

- Пример за не-RGB линейно пространство (още се нарича YC_rC_b)
- Y-компонентът пази яркостта, а другите – цвета
  - $Y \approx (R + G + B)/3$
  - $U \approx G - R$
  - $V \approx G - B$
- Ползва се в JPEG и MPEG компресиите, като обикновено U и V каналите се пазят отделно от Y и то с по-ниска разделителна способност
- При чернобели изображения, U и V на практика отпадат

Пример – HSV цветово пространство

- Пример за нелинейно, не-RGB пространство
- HSV = Hue, Saturation, Value
  - Hue – оттенък (монохроматичен)
  - Saturation – наситеност (при ниски стойности, цвета е смесен със сиво)
  - Value – яркост
- Лесно се превръща до RGB и обратно
- Подходящо за вход от потребителя, тъй като е по-близо до човешкото усещане за състава на цвета

Пример – L*a*b* цветово пространство

- L*a*b* (CIELAB) е пример за перцептуално цветово пространство
- Подобно на YUV, L каналът показва яркостта, а другите два са цветоразликови
- Перцептуалните му качества се състоят в това, че разликата между два L*a*b* цвята отговаря на човешкото усещане за разлика между цветовете

L*a*b* цветово пространство

- Например: ако имаме G и G' – два оттенъка на зеленото; B и B' – два оттенъка на синьото; и разликата в цветовете G и G' изглежда подобна на тази между B и B', то в L*a*b*, тези цветове ще са на равно евклидово разстояние
- L*a*b* освен това взема в предвид, че при много тъмни и много светли цветове, човешката чувствителност за контраст пада
- L*a*b* е базирано на XYZ и може да представя всички видими от човека цветове

HDR

- HDR може да означава две различни неща (в зависимост от контекста)
  - HDRI: High Dynamic Range Images
  - HDRR: High Dynamic Range Rendering

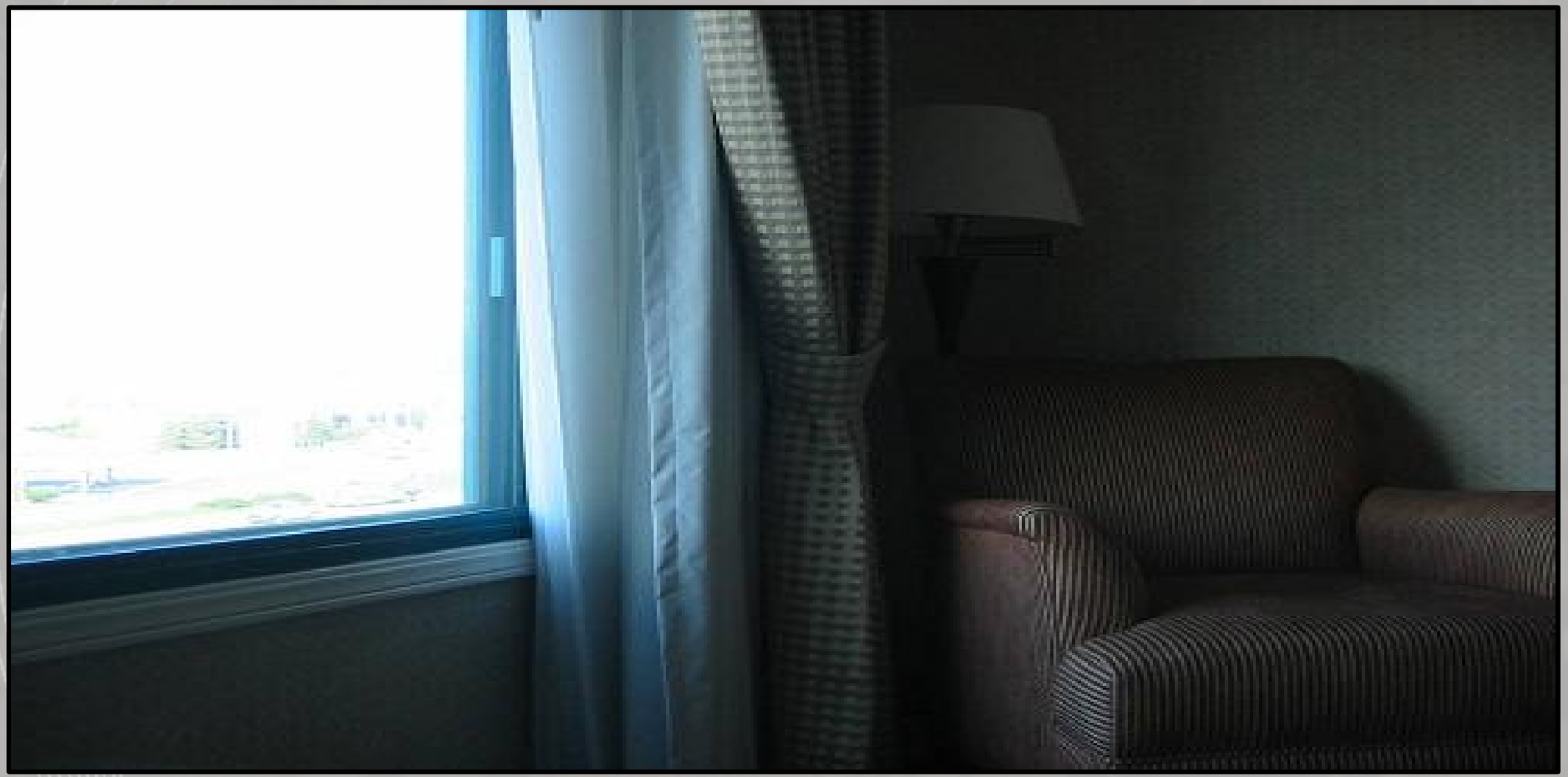
HDRI

- Най-кратко, HDRI е използването на изображения съдържащи цветове с плаваща запетая (floating point)
- Традиционно, дисплей технологиите (монитори/принтери/плотери) могат да показват фиксиран обхват от цветове
  - Най-често **Truecolor**, т.е. по 8 бита за компонента (R8G8B8)
- С други думи, ако единствената функция на едно изображение е да бъде показано на екрана, няма смисъл да се пазят повече от 8 бита за компонента
  - ... и това правят почти всички файлови формати

HDRI (2)

- Обаче, 8 бита за компонента означава, че разликата между най-тъмния пиксел и най-светлия пиксел в изображението може да е максимално 255:1
- В реалния свят тази разлика може да е до 50000:1
- В резултат на което...

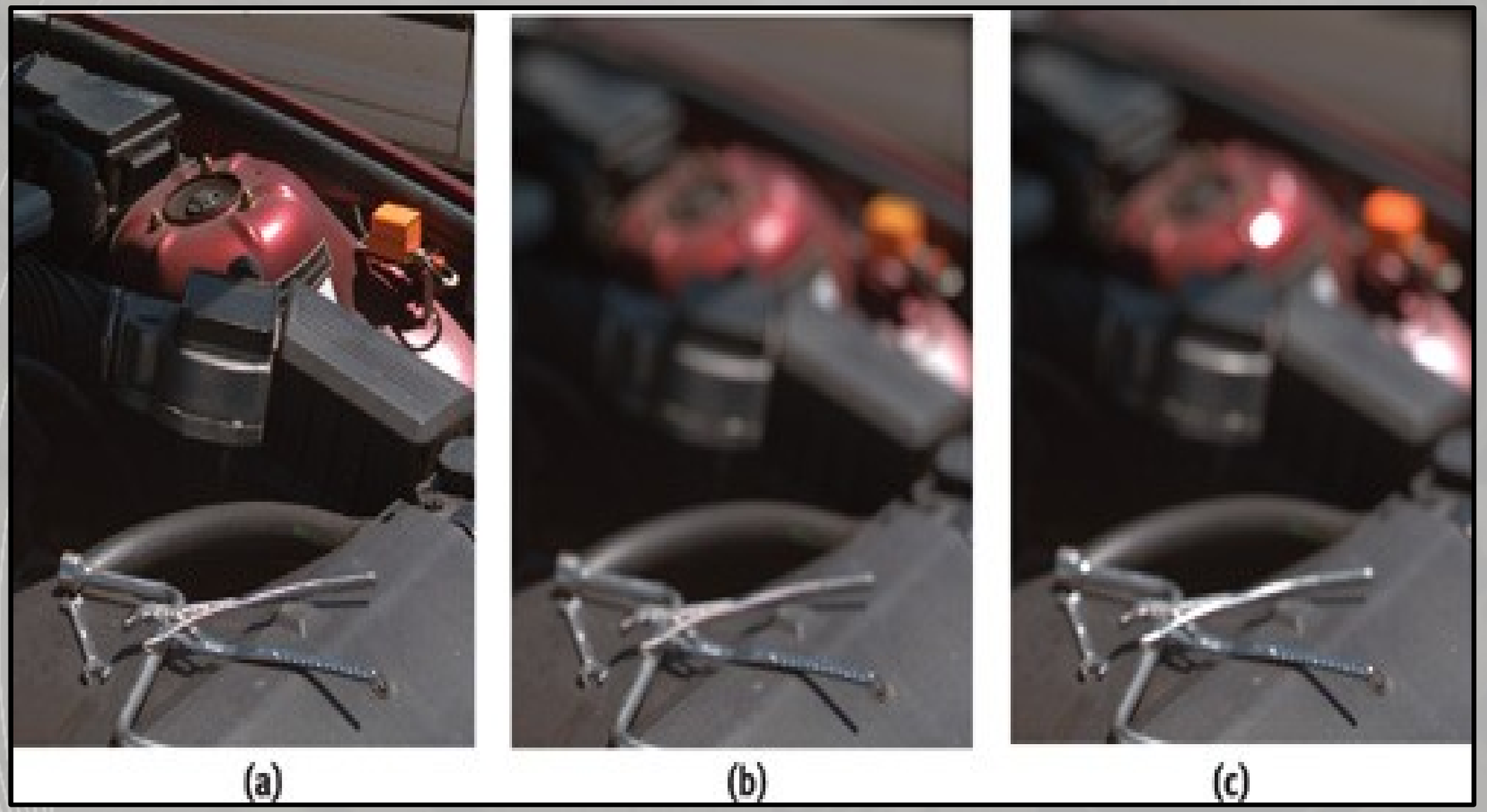
HDRI (3)



HDRI (4)

- HDR файловите формати позволяват да се съхрани цялата информация
  - Без отрязване на много светли или много тъмни пиксели
  - Без да се губят цветови детайли
- Защо това е важно?
  - Postprocessing
  - High Dynamic Range Rendering
  - Debugging

HDRI (5)



HDRI (6)

- HDR файлови формати:
  - Radiance RGBE, .hdr
  - TIFF, .tiff
  - **OpenEXR, .exr**
- Ако трябва да изберете един, изберете OpenEXR
  - Разработен през 1999 от Industrial Light and Magic (ILM) за техния вътрешен софтуер и пуснат opensource през 2003
  - Чисто C++ API (и имплементация)
    - Също C, Python, Ruby, Perl, Lua, Haskell, etc.
  - Поддържа се от всички графични приложения (вече)

HDRI (7)

- Числата с плаваща запетая идват в два вкуса
  - Single precision: float
    - 32bit - sign 1 bit, exponent 8 bits, mantissa 23 bits (+1 hidden bit)
  - Double precision: double
    - 64bit - sign 1 bit, exponent 11 bits, mantissa 52 bits (+1 hidden bit)
- Понякога 32 бита са прекалено много
  - С други думи, може да се мине с по-малко прецизност, без особена загуба на качество

HDRI (8)

- Чисто нов вкус!
  - Half precision: half
    - 16 bit – sign 1 bit, exponent 5 bits, mantissa 10 bits (+1 hidden bit)
  - Първоначално създаден от ILM за OpenEXR, след това е IEEE стандартизиран
  - Подържа се от графичните API (OpenGL, Direct3D) и техните shading езици (HLSL, GLSL, Cg)
  - Half текстури заемат 2x по-малко място без визуална загуба

HDRR

- За High Dynamic Range Rendering (HDRR) и Image Based Lighting (IBL) ще говорим след няколко лекции
- Понеже IBL е една от основните мотивации за HDRI, ще споменем няколко думи

Image Based Lighting

- Осветлението може да не идва от лампи, а само от „обкръжението“, чрез картинка. За да симулираме ярките („светещи“) части в самата картинка, трябва тя да е с широк динамичен обхват

Image Based Lighting

- Следващата страница показва самото „снимане“ на такава картинка, за употреба при IBL

Image Based Lighting (2)

Imaged Base Lighting (3)

- Употреба на получения environment HDRI