

Компьютерная графика

Лекция 11: HDR, tone mapping, gamma correction, sRGB, color banding, dithering

2023

- Проблема: после вычисления освещённости мы можем получить значения компонент цвета большие, чем 1, и при записи на экран/фреймбуфер они будут обрезаны до 1

- Проблема: после вычисления освещённости мы можем получить значения компонент цвета большие, чем 1, и при записи на экран/фреймбуфер они будут обрезаны до 1
- В реальном мире интенсивность света ничем не ограничена, и между солнечным днём и тёмной ночью может меняться на 15 порядков

- Проблема: после вычисления освещённости мы можем получить значения компонент цвета большие, чем 1, и при записи на экран/фреймбуфер они будут обрезаны до 1
- В реальном мире интенсивность света ничем не ограничена, и между солнечным днём и тёмной ночью может меняться на 15 порядков
- При рендеринге объект может освещаться тусклым ambient освещением, а может сразу десятком ярких источников света

HDR

- High Dynamic Range (HDR) – термин, означающий большой (не ограниченный $[0, 1]$) диапазон интенсивностей света

- High Dynamic Range (HDR) – термин, означающий большой (не ограниченный $[0, 1]$) диапазон интенсивностей света
- HDR текстура (например, environment map) – содержит значения, выходящие за диапазон $[0, 1]$ (например, 32-bit floating-point, или normalized 16-bit + фиксированный множитель)

- High Dynamic Range (HDR) – термин, означающий большой (не ограниченный [0, 1]) диапазон интенсивностей света
- HDR текстура (например, environment map) – содержит значения, выходящие за диапазон [0, 1] (например, 32-bit floating-point, или normalized 16-bit + фиксированный множитель)
- HDR рендеринг – позволяет отобразить HDR-освещение

Tone mapping

- Чтобы поддержать HDR рендеринг, Нужно превратить диапазон освещённостей $[0, \infty)$ в диапазон $[0, 1]$

Tone mapping

- Чтобы поддержать HDR рендеринг, Нужно превратить диапазон освещённостей $[0, \infty)$ в диапазон $[0, 1]$
- В принципе, подойдёт любая монотонно возрастающая функция $[0, \infty) \rightarrow [0, 1]$ – *tone-mapping curve*:
 - $x \mapsto \frac{x}{1+x}$ – Reinhard operator
 - $x \mapsto \arctan(x)$
 - $x \mapsto \frac{2}{1+e^{-x}} - 1$

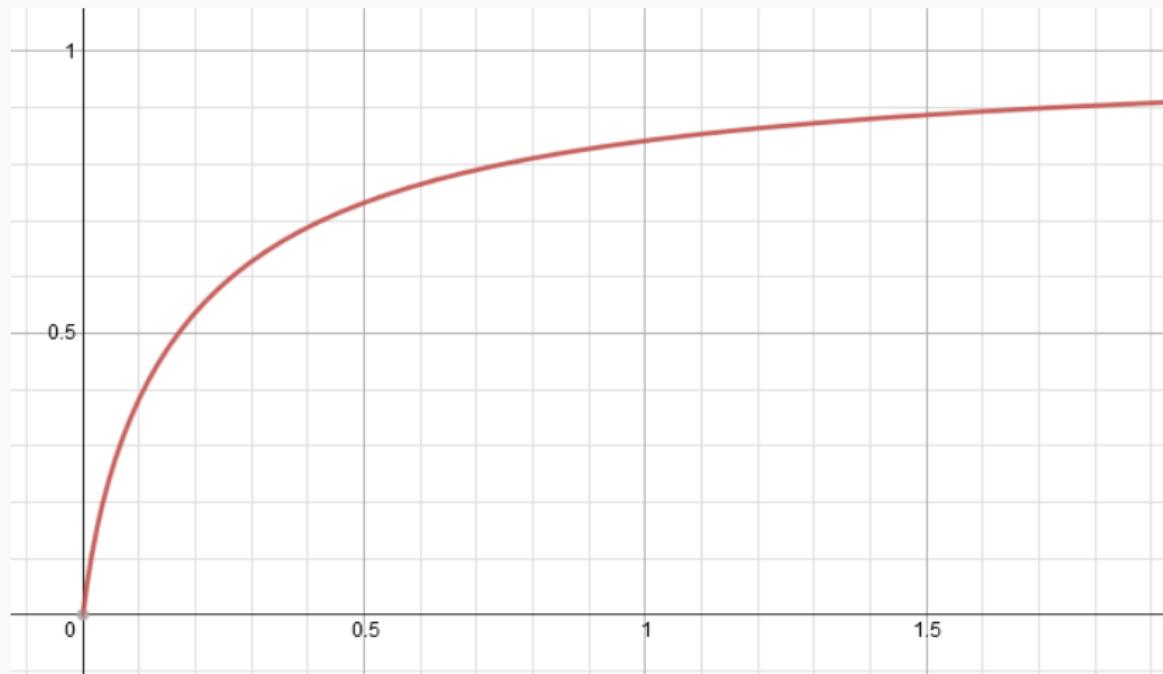
Tone mapping

- Чтобы поддержать HDR рендеринг, Нужно превратить диапазон освещённостей $[0, \infty)$ в диапазон $[0, 1]$
- В принципе, подойдёт любая монотонно возрастающая функция $[0, \infty) \rightarrow [0, 1]$ – *tone-mapping curve*:
 - $x \mapsto \frac{x}{1+x}$ – Reinhard operator
 - $x \mapsto \arctan(x)$
 - $x \mapsto \frac{2}{1+e^{-x}} - 1$
- Иногда используют более сложные функции, взятые из кинематографа (filmic tone mapping)

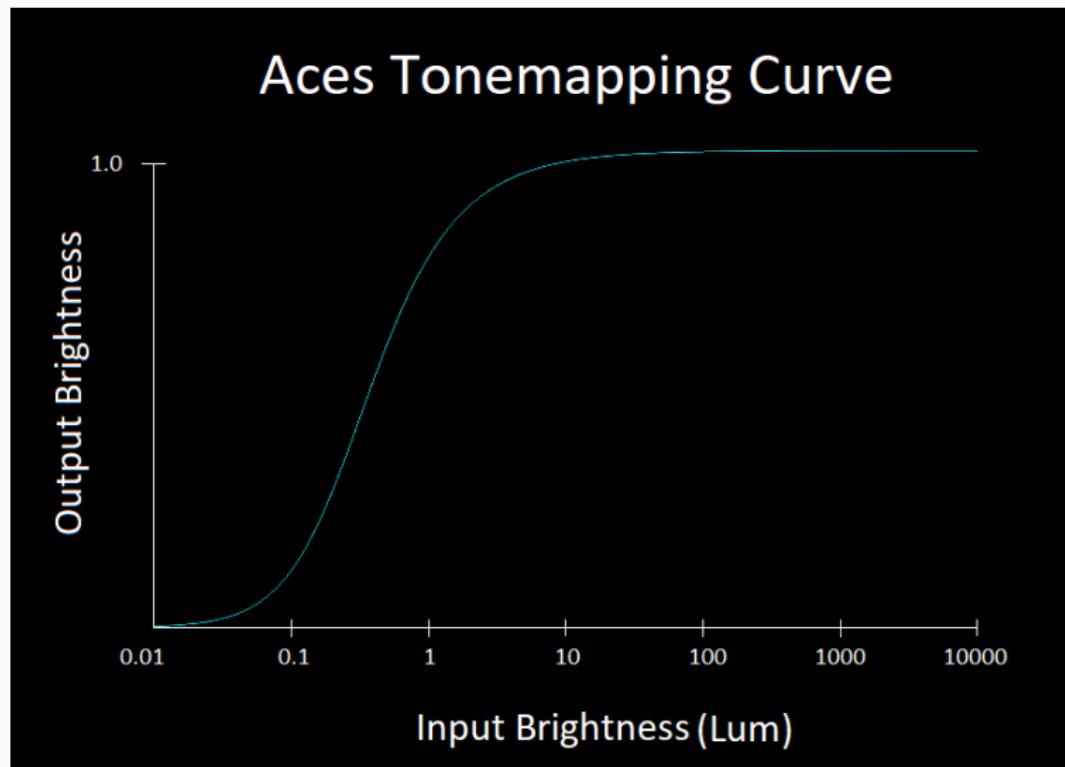
Tone mapping

- Чтобы поддержать HDR рендеринг, Нужно превратить диапазон освещённостей $[0, \infty)$ в диапазон $[0, 1]$
- В принципе, подойдёт любая монотонно возрастающая функция $[0, \infty) \rightarrow [0, 1]$ – *tone-mapping curve*:
 - $x \mapsto \frac{x}{1+x}$ – Reinhard operator
 - $x \mapsto \arctan(x)$
 - $x \mapsto \frac{2}{1+e^{-x}} - 1$
- Иногда используют более сложные функции, взятые из кинематографа (filmic tone mapping)
- Иногда всё равно ограничивают диапазон входных интенсивностей каким-то максимальным значением W (white), и делают нелинейное преобразование $[0, W] \rightarrow [0, 1]$

Reinhard operator



Filmic tone mapping curve (ACES)



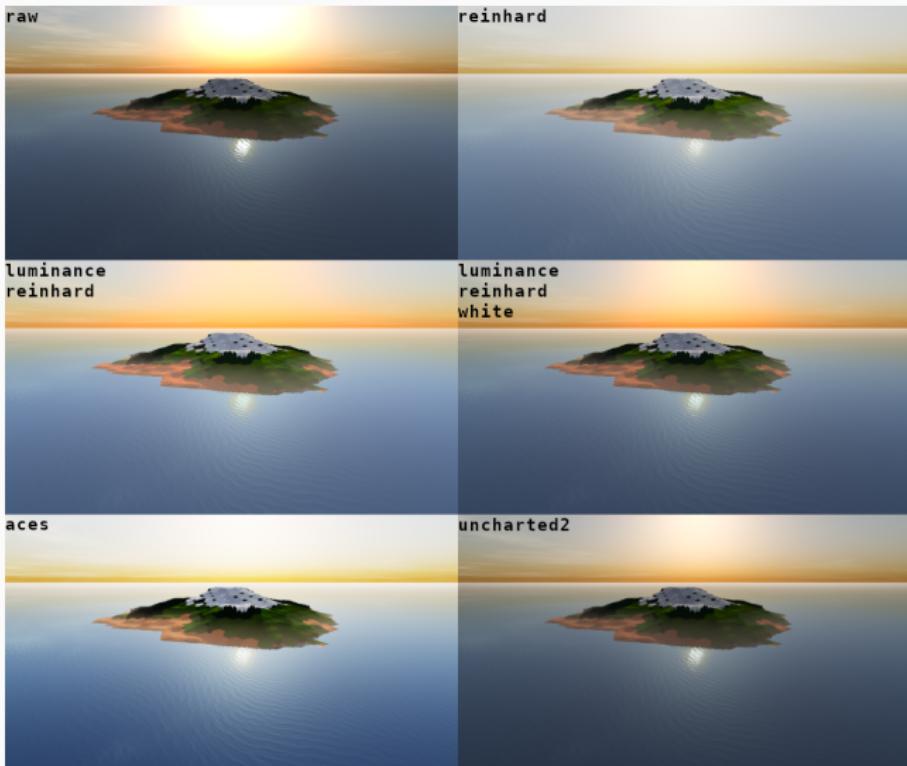
Uncharted 2 tone mapping

```
vec3 TonemapRaw(vec3 x)
{
    float A = 0.15;
    float B = 0.50;
    float C = 0.10;
    float D = 0.20;
    float E = 0.02;
    float F = 0.30;

    return ((x*(A*x+C*B)+D*D)/(x*(A*x+B)+D*D))-E/F;
}

vec3 Uncharted2Tonemap(vec3 color)
{
    float W = 11.2;
    return TonemapRaw(color) / TonemapRaw(vec3(W));
}
```

Сравнение разных tone-mapping кривых



Tone mapping

- Tone-mapping функции часто позволяют настраивать среднюю/максимальную видимую освещённость сцены либо выдержку (*exposure*)

Tone mapping

- Tone-mapping функции часто позволяют настраивать среднюю/максимальную видимую освещённость сцены либо выдержку (*exposure*)
- Чем выше средняя/максимальная освещённость, тем *темнее* сцена

Tone mapping

- Tone-mapping функции часто позволяют настраивать среднюю/максимальную видимую освещённость сцены либо выдержку (*exposure*)
- Чем выше средняя/максимальная освещённость, тем *темнее* сцена
- Чем выше выдержка, тем *светлее* сцена

Tone mapping

- Tone-mapping функции часто позволяют настраивать среднюю/максимальную видимую освещённость сцены либо выдержку (*exposure*)
- Чем выше средняя/максимальная освещённость, тем *темнее* сцена
- Чем выше выдержка, тем *светлее* сцена
- Выдержку можно подгонять под конкретную сцену, а можно автоматически подстраивать под среднюю яркость видимых камерой объектов (*dynamic exposure, auto-exposure*)

Witcher 3: high exposure



Witcher 3: low exposure



Как применять tone mapping

- При обычном forward рендеринге, когда все источники света учитываются циклом в шейдере, а также при forward tiled/clustered shading'e, tone-mapping можно применять в этом же шейдере, и писать значения цвета сразу на экран

Как применять tone mapping

- При обычном forward рендеринге, когда все источники света учитываются циклом в шейдере, а также при forward tiled/clustered shading'e, tone-mapping можно применять в этом же шейдере, и писать значения цвета сразу на экран
- При deferred shading'e, применяющем каждый источник света по одному с аддитивным блендингом, применять tone-mapping в этом же месте нельзя: сумма tone-mapped цветов не равна значению tone map от суммы (т.е. tone-mapping функция нелинейна)

Как применять tone mapping

- При обычном forward рендеринге, когда все источники света учитываются циклом в шейдере, а также при forward tiled/clustered shading'e, tone-mapping можно применять в этом же шейдере, и писать значения цвета сразу на экран
- При deferred shading'e, применяющем каждый источник света по одному с аддитивным блендингом, применять tone-mapping в этом же месте нельзя: сумма tone-mapped цветов не равна значению tone map от суммы (т.е. tone-mapping функция нелинейна)
- В этом случае обычно результирующий цвет пикселя аккумулируется в HDR-текстуре (32-bit floating point, или normalized 16-bit с фиксированным множителем), и потом отдельным проходом рисуется на экран уже с tone-mapping'ом

Как применять tone mapping

- При обычном forward рендеринге, когда все источники света учитываются циклом в шейдере, а также при forward tiled/clustered shading'e, tone-mapping можно применять в этом же шейдере, и писать значения цвета сразу на экран
- При deferred shading'e, применяющем каждый источник света по одному с аддитивным блендингом, применять tone-mapping в этом же месте нельзя: сумма tone-mapped цветов не равна значению tone map от суммы (т.е. tone-mapping функция нелинейна)
- В этом случае обычно результирующий цвет пикселя аккумулируется в HDR-текстуре (32-bit floating point, или normalized 16-bit с фиксированным множителем), и потом отдельным проходом рисуется на экран уже с tone-mapping'ом
- Некоторые эффекты пост-обработки лучше работают именно в HDR диапазоне (т.е. до применения tone mapping) – размытие, свечение (bloom), и т.п.

HDR & tone mapping: ссылки

- learnopengl.com/Advanced-Lighting/HDR
- What is tone mapping
- What is HDR tone mapping
- ACES filmic tone mapping curve
- Filmic tone-mapping operators

Гамма

- Обычно, интенсивность света I , излучаемого монитором, нелинейно зависит от значения V , записанного в пикселе

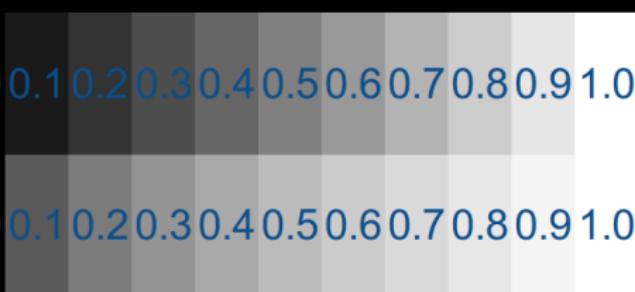
Линейное значение пикселя vs линейная интенсивность излучения

Linear
encoding

$$V_S = 0.00\ 0.10\ 0.20\ 0.30\ 0.40\ 0.50\ 0.60\ 0.70\ 0.80\ 0.90\ 1.0$$

Linear
intensity

$$I = 0.00\ 0.10\ 0.20\ 0.30\ 0.40\ 0.50\ 0.60\ 0.70\ 0.80\ 0.90\ 1.0$$



Гамма

- Обычно, интенсивность света I , излучаемого монитором, нелинейно зависит от значения V , записанного в пикселе
- Это лучше соответствует восприятию света человеком и даёт больше точности тёмным цветам, которые человек различает лучше

Гамма

- Обычно, интенсивность света I , излучаемого монитором, нелинейно зависит от значения V , записанного в пикселе
- Это лучше соответствует восприятию света человеком и даёт больше точности тёмным цветам, которые человек различает лучше
- Почти всегда используется показательная функция:

$$I \sim V^\gamma \tag{1}$$

Гамма

- Обычно, интенсивность света I , излучаемого монитором, нелинейно зависит от значения V , записанного в пикселе
- Это лучше соответствует восприятию света человеком и даёт больше точности тёмным цветам, которые человек различает лучше
- Почти всегда используется показательная функция:

$$I \sim V^\gamma \quad (1)$$

- γ обычно равна 2.2 (некоторые компьютеры Macintosh использовали 1.8)

Проблемы гаммы

- Картинка может издалека выглядеть ярче, чем её усреднённый вариант (e.g. mipmap)

Серый (цвет=0.5) квадрат и квадрат с мелкой шахматной раскраской



Проблемы гаммы

- Картинка может издалека выглядеть ярче, чем её усреднённый вариант (e.g. `tiptap`)
- Искажается восприятие относительных яркостей, особенно при реалистичном рендеринге (e.g. объект в два раза ярче не будет выглядеть в два раза ярче)

Проблемы гаммы

- Картинка может издалека выглядеть ярче, чем её усреднённый вариант (e.g. `tiptap`)
- Искажается восприятие относительных яркостей, особенно при реалистичном рендеринге (e.g. объект в два раза ярче не будет выглядеть в два раза ярче)
- Неправильно выглядит освещение, наложение источников света, и т.д.

Коррекция гаммы (gamma-correction)

- Коррекция гаммы – общий термин для применения любых нелинейных преобразований над интенсивностью пикселя

Коррекция гаммы (gamma-correction)

- Коррекция гаммы – общий термин для применения любых нелинейных преобразований над интенсивностью пикселя
- В рендеринге под гамма-коррекцией обычно подразумевают применение обратного к $V^{2.2}$ преобразования, чтобы получить линейную зависимость выходящего излучения от значения пикселя

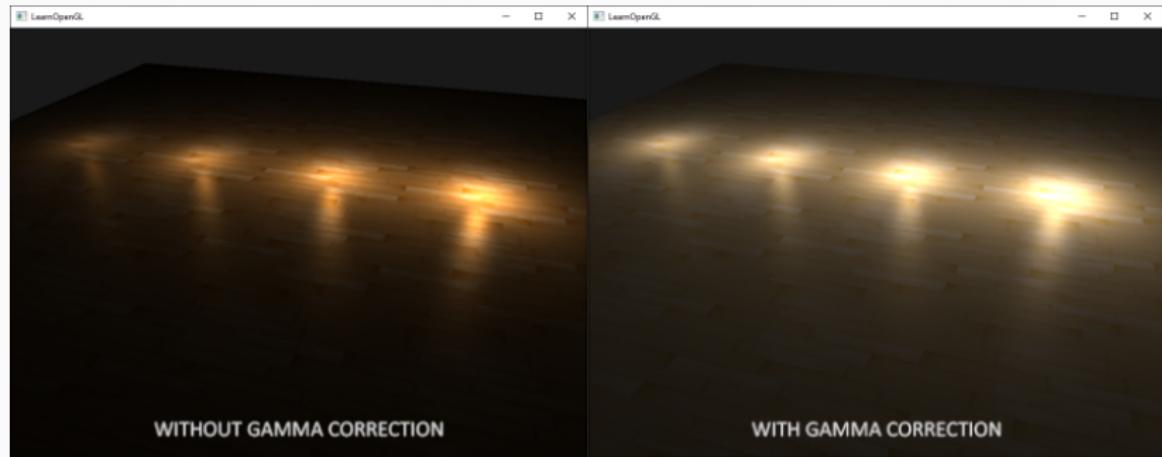
Коррекция гаммы (gamma-correction)

```
// Вычислили цвет пикселя  
// с учётом освещения  
vec3 color = ...;  
  
color = pow(color, vec3(1.0 / 2.2));
```

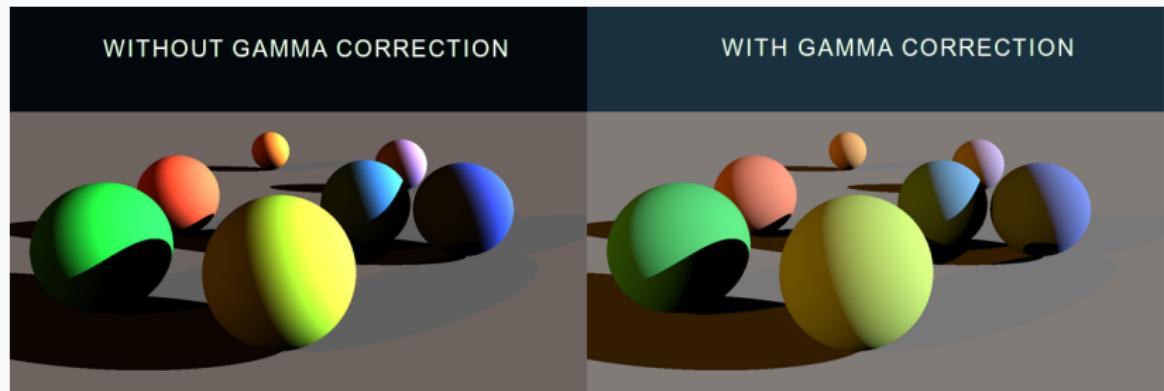
Эффекты коррекции гаммы



Эффекты коррекции гаммы



Эффекты коррекции гаммы



Эффекты коррекции гаммы

- Типичные эффекты гамма-коррекции:

Эффекты коррекции гаммы

- Типичные эффекты гамма-коррекции:
 - Картинка становится ярче (чем темнее цвет, тем больше прирост яркости)

Эффекты коррекции гаммы

- Типичные эффекты гамма-коррекции:
 - Картинка становится ярче (чем темнее цвет, тем больше прирост яркости)
 - Картинка становится менее контрастной

Эффекты коррекции гаммы

- Типичные эффекты гамма-коррекции:
 - Картинка становится ярче (чем темнее цвет, тем больше прирост яркости)
 - Картинка становится менее контрастной
 - Освещение выглядит правильнее

Эффекты коррекции гаммы

- Типичные эффекты гамма-коррекции:
 - Картинка становится ярче (чем темнее цвет, тем больше прирост яркости)
 - Картинка становится менее контрастной
 - Освещение выглядит правильнее
 - Переход от освещённой к неосвещённой области объекта выглядит более резким (как в реальности)

Эффекты коррекции гаммы

- Типичные эффекты гамма-коррекции:
 - Картинка становится ярче (чем темнее цвет, тем больше прирост яркости)
 - Картинка становится менее контрастной
 - Освещение выглядит правильнее
 - Переход от освещённой к неосвещённой области объекта выглядит более резким (как в реальности)
- Некоторые tone-mapping кривые уже включают в себя гамма-коррекцию!

Коррекция гаммы: ссылки

- en.wikipedia.org/wiki/Gamma_correction
- What every coder should know about gamma
- Linear-space lighting (i.e. gamma)
- Learnopengl.com tutorial

- По тем же причинам, по которым существует нелинейная гамма у мониторов (отдать больше бит под тёмные цвета), хочется хранить текстуры не в сыром формате, а в гамма-преобразованном: значение пикселя V зависит от желаемой интенсивности I как $V \sim I^{\frac{1}{\gamma}}$

sRGB

- По тем же причинам, по которым существует нелинейная гамма у мониторов (отдать больше бит под тёмные цвета), хочется хранить текстуры не в сыром формате, а в гамма-преобразованном: значение пикселя V зависит от желаемой интенсивности I как $V \sim I^{\frac{1}{\gamma}}$
- Такой формат хранения цвета называется sRGB
 - N.B. Точная формула преобразования sRGB отличается от $I^{\frac{1}{2.2}}$, но очень близка к ней

sRGB

- По тем же причинам, по которым существует нелинейная гамма у мониторов (отдать больше бит под тёмные цвета), хочется хранить текстуры не в сыром формате, а в гамма-преобразованном: значение пикселя V зависит от желаемой интенсивности I как $V \sim I^{\frac{1}{\gamma}}$
- Такой формат хранения цвета называется sRGB
 - N.B. Точная формула преобразования sRGB отличается от $I^{\frac{1}{2.2}}$, но очень близка к ней
- Обычно изображения хранятся именно в таком формате

sRGB

- По тем же причинам, по которым существует нелинейная гамма у мониторов (отдать больше бит под тёмные цвета), хочется хранить текстуры не в сыром формате, а в гамма-преобразованном: значение пикселя V зависит от желаемой интенсивности I как $V \sim I^{\frac{1}{\gamma}}$
- Такой формат хранения цвета называется sRGB
 - N.B. Точная формула преобразования sRGB отличается от $I^{\frac{1}{2.2}}$, но очень близка к ней
- Обычно изображения хранятся именно в таком формате
- Проверить формат можно любым редактором изображений или программой `identify` пакета `imagemagick`

sRGB

- При чтении из sRGB-текстуры нужно возводить прочитанное значение в степень 2.2

sRGB

- При чтении из sRGB-текстуры нужно возводить прочитанное значение в степень 2.2
- При записи в sRGB-текстуру нужно возводить записываемое значение в степень 1/2.2

sRGB

- При чтении из sRGB-текстуры нужно возводить прочитанное значение в степень 2.2
- При записи в sRGB-текстуру нужно возводить записываемое значение в степень 1/2.2
- В OpenGL есть поддержка sRGB для текстур и фреймбуферов

sRGB-текстуры

- Специальное значение `internal format` для текстуры (`GL_SRGB8` или `GL_SRGB8_ALPHA8`) означает, что текстура хранит sRGB-значения – они будут *автоматически* переведены в линейные значения (т.е. возведены в степень 2.2) при чтении из шейдера

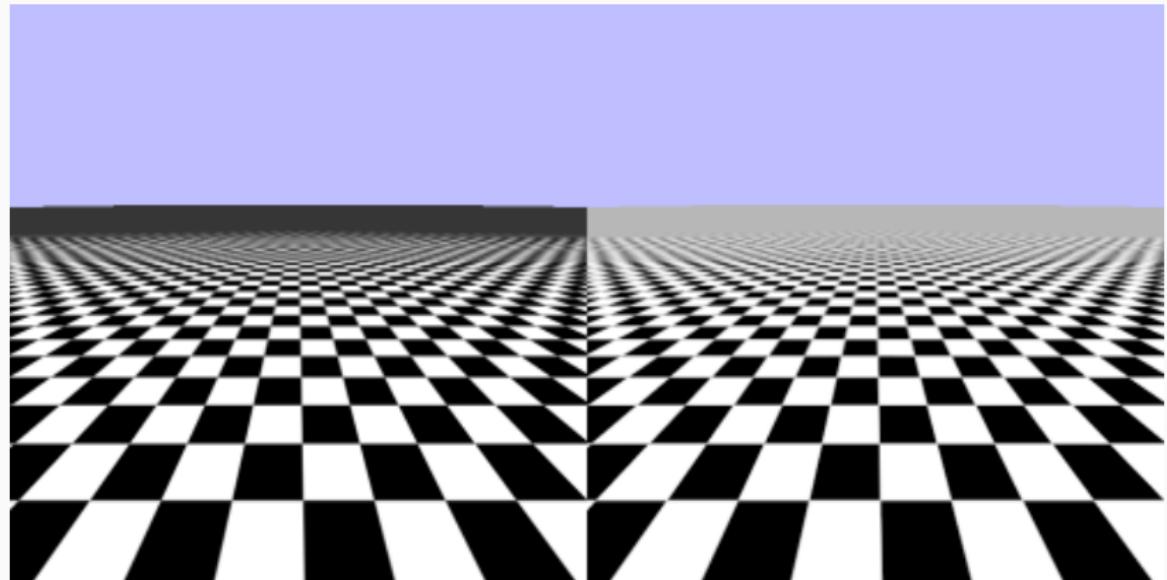
sRGB-текстуры

- Специальное значение `internal format` для текстуры (`GL_SRGB8` или `GL_SRGB8_ALPHA8`) означает, что текстура хранит sRGB-значения – они будут *автоматически* переведены в линейные значения (т.е. возведены в степень 2.2) при чтении из шейдера
- `glEnable(GL_FRAMEBUFFER_SRGB)` включит автоматическое обратное преобразование (возведение в степень 1/2.2) при рисовании в sRGB-текстуру

sRGB-текстуры

- Специальное значение `internal format` для текстуры (`GL_SRGB8` или `GL_SRGB8_ALPHA8`) означает, что текстура хранит sRGB-значения – они будут *автоматически* переведены в линейные значения (т.е. возведены в степень 2.2) при чтении из шейдера
- `glEnable(GL_FRAMEBUFFER_SRGB)` включит автоматическое обратное преобразование (возведение в степень 1/2.2) при рисовании в sRGB-текстуру
- **N.B.:** `glGenerateMipmap` не обязан правильно обрабатывать sRGB-текстуры, и может делать просто линейное усреднение
⇒ лучше мипмап'ы для sRGB-текстур генерировать руками (на CPU, или на GPU с помощью рендеринга в мипмап-уровни текстуры)

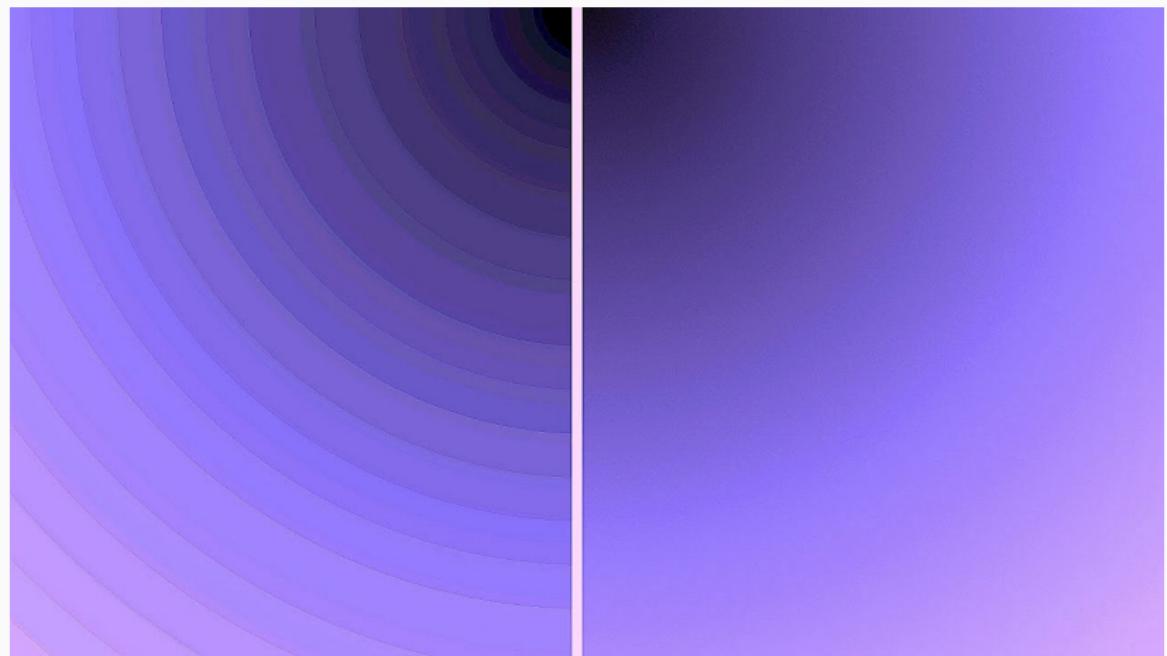
sRGB-correct mipmap



sRGB: ссылки

- en.wikipedia.org/wiki/SRGB
- Stackoverflow post on sRGB
- sRGB framebuffers

Color banding



Color banding

- Артефакт, когда чётко видна граница между областями, заполненными одним цветом

Color banding

- Артефакт, когда чётко видна граница между областями, заполненными одним цветом
- Обычно не сильно заметен, особенно после наложения текстур и пост-обработки

Color banding

- Артефакт, когда чётко видна граница между областями, заполненными одним цветом
- Обычно не сильно заметен, особенно после наложения текстур и пост-обработки
- Хорошо заметен на монотонных поверхностях (напр. небо)

Color banding

- Артефакт, когда чётко видна граница между областями, заполненными одним цветом
- Обычно не сильно заметен, особенно после наложения текстур и пост-обработки
- Хорошо заметен на монотонных поверхностях (напр. небо)
- Сильнее заметен с тёмными цветами (из-за особенностей человеческого зрения)

Color banding

- Артефакт, когда чётко видна граница между областями, заполненными одним цветом
- Обычно не сильно заметен, особенно после наложения текстур и пост-обработки
- Хорошо заметен на монотонных поверхностях (напр. небо)
- Сильнее заметен с тёмными цветами (из-за особенностей человеческого зрения)
- Может проявляться из-за ограниченности вычислений (напр. маленькое число сэмплов при SSAO)

Color banding

- Артефакт, когда чётко видна граница между областями, заполненными одним цветом
- Обычно не сильно заметен, особенно после наложения текстур и пост-обработки
- Хорошо заметен на монотонных поверхностях (напр. небо)
- Сильнее заметен с тёмными цветами (из-за особенностей человеческого зрения)
- Может проявляться из-за ограниченности вычислений (напр. маленькое число сэмплов при SSAO)
- Может усиливаться при неаккуратной работе с форматами хранения и нелинейными преобразованиями

Color banding: пример возникновения

- Рисуем сцену, записываем линейную интенсивность в текстуру в формате GL_RGB8

Color banding: пример возникновения

- Рисуем сцену, записываем линейную интенсивность в текстуру в формате GL_RGB8
- Выводим сцену на экран с гамма-коррекцией: значения из текстуры возводятся в степень 1/2.2

Color banding: пример возникновения

- Рисуем сцену, записываем линейную интенсивность в текстуру в формате GL_RGB8
- Выводим сцену на экран с гамма-коррекцией: значения из текстуры возводятся в степень 1/2.2
- Значение текстуры 0 переходит в значение 0 на экране

Color banding: пример возникновения

- Рисуем сцену, записываем линейную интенсивность в текстуру в формате GL_RGB8
- Выводим сцену на экран с гамма-коррекцией: значения из текстуры возводятся в степень $1/2.2$
- Значение текстуры 0 переходит в значение 0 на экране
- Значение текстуры 1 переходит в значение $255 \cdot \left(\frac{1}{255}\right)^{\frac{1}{2.2}} = 21$ на экране

Color banding: пример возникновения

- Рисуем сцену, записываем линейную интенсивность в текстуру в формате GL_RGB8
- Выводим сцену на экран с гамма-коррекцией: значения из текстуры возводятся в степень $1/2.2$
- Значение текстуры 0 переходит в значение 0 на экране
- Значение текстуры 1 переходит в значение $255 \cdot \left(\frac{1}{255}\right)^{\frac{1}{2.2}} = 21$ на экране
- Колossalная потеря точности: значения от 1 до 20 не используются!

Color banding: пример возникновения

- Рисуем сцену, записываем линейную интенсивность в текстуру в формате GL_RGB8
- Выводим сцену на экран с гамма-коррекцией: значения из текстуры возводятся в степень $1/2.2$
- Значение текстуры 0 переходит в значение 0 на экране
- Значение текстуры 1 переходит в значение $255 \cdot \left(\frac{1}{255}\right)^{\frac{1}{2.2}} = 21$ на экране
- Колossalная потеря точности: значения от 1 до 20 не используются!
- Для избавления можно увеличить битность промежуточной текстуры до GL_RGB16

Color banding

- Banding, связанный с потерей точности, обычно решают увеличением точности (8-bit → 16-bit)

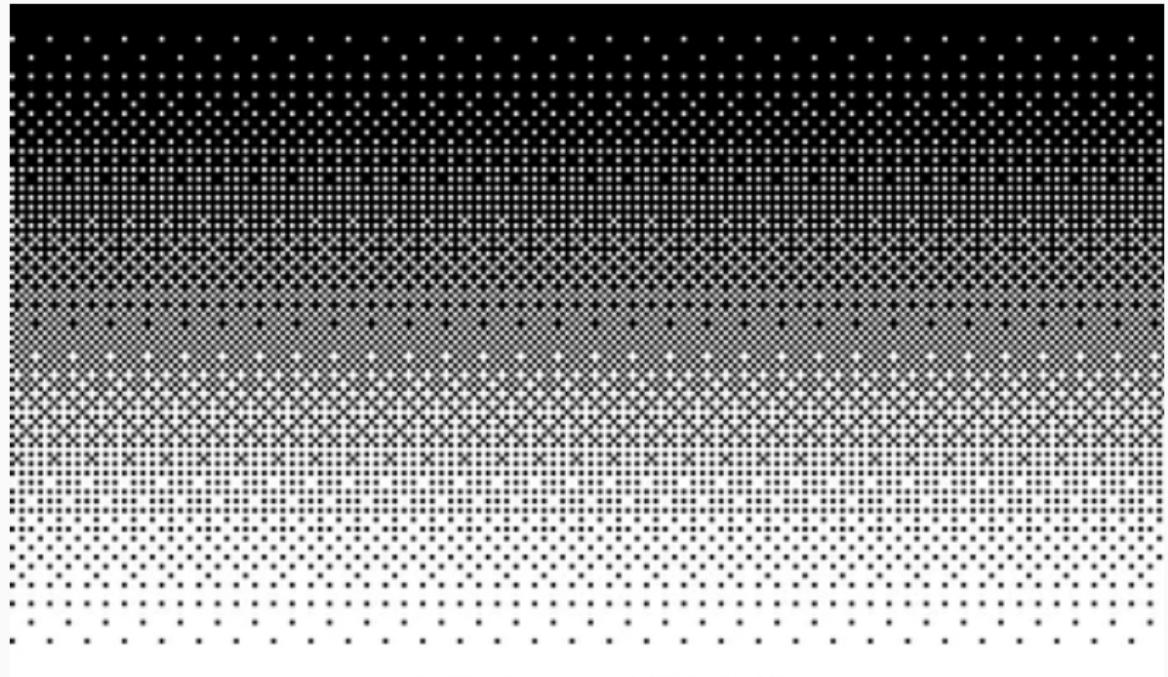
Color banding

- Banding, связанный с потерей точности, обычно решают увеличением точности ($8\text{-bit} \longrightarrow 16\text{-bit}$)
- Banding, связанный ограниченными вычислениями, обычно решают рандомизацией, разменивая banding на шум

Color banding

- Banding, связанный с потерей точности, обычно решают увеличением точности ($8\text{-bit} \longrightarrow 16\text{-bit}$)
- Banding, связанный ограниченными вычислениями, обычно решают рандомизацией, разменивая banding на шум
- Banding, связанный с ограниченной точностью самого экрана, обычно решают с помощью дизеринга

Dithering



Dithering

- Способ рисования полутона за счёт варьирования количества пикселей двух цветов

Dithering

- Способ рисования полутонаов за счёт варьирования количества пикселей двух цветов
- Работает только для больших (по размеру) градиентов

Dithering

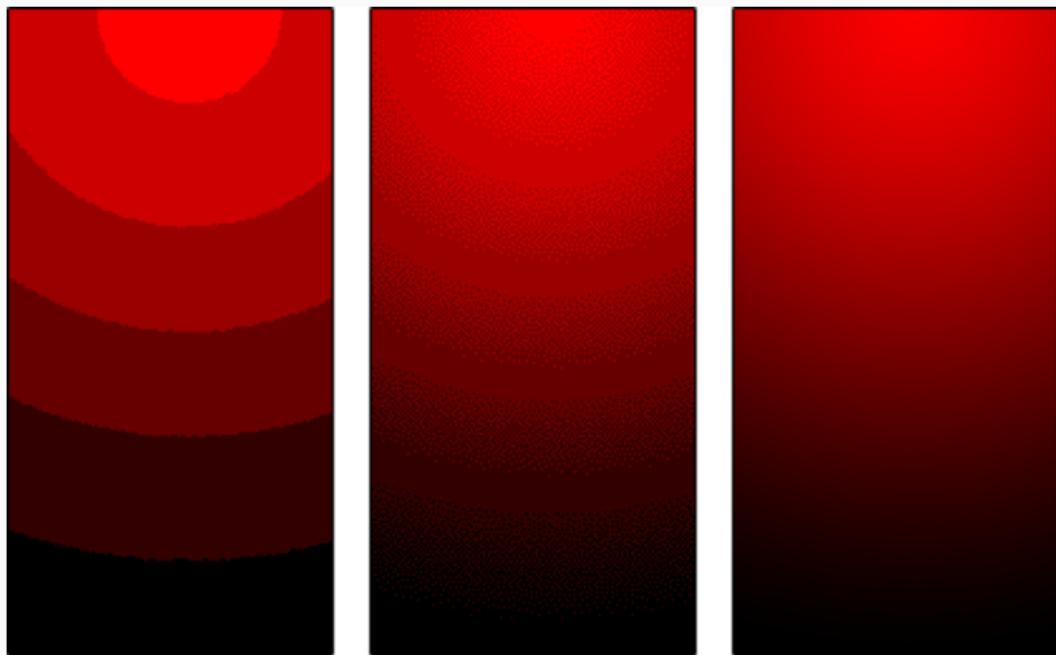
- Способ рисования полутона за счёт варьирования количества пикселей двух цветов
- Работает только для больших (по размеру) градиентов
- Работает при любой точности формата хранения, визуально увеличивая её

Dithering

- Способ рисования полутона за счёт варьирования количества пикселей двух цветов
- Работает только для больших (по размеру) градиентов
- Работает при любой точности формата хранения, визуально увеличивая её
- Может использоваться для рисования полутона между областями, окрашенными в соседние представимые цвета

- Способ рисования полутона за счёт варьирования количества пикселей двух цветов
- Работает только для больших (по размеру) градиентов
- Работает при любой точности формата хранения, визуально увеличивая её
- Может использоваться для рисования полутона между областями, окрашенными в соседние представимые цвета
- Может использоваться как художественный приём (имитация пиксель-арт графики, имитация очень старых игр)

Dithering



8-bit gradient

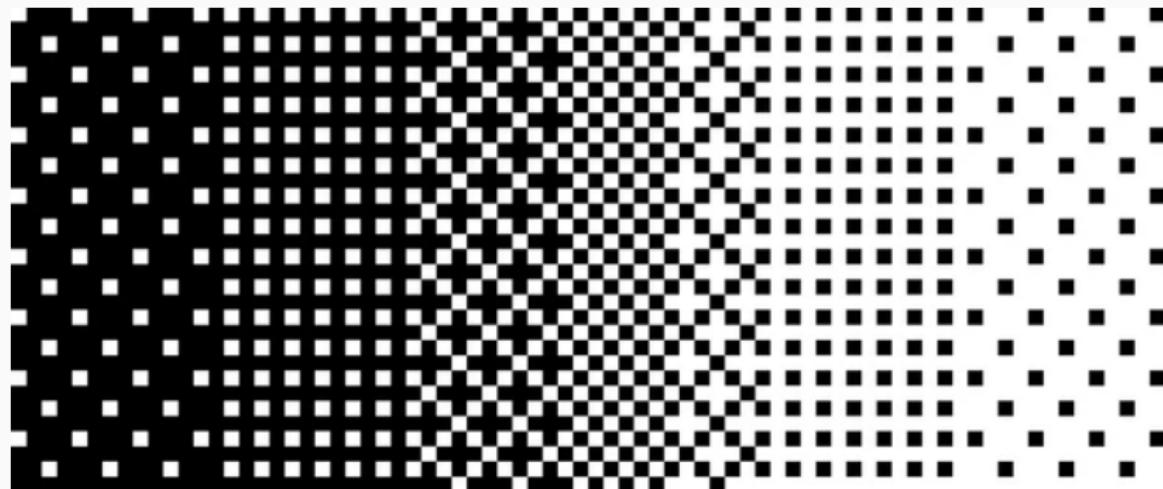
8-bit gradient,
dithered

24-bit gradient

Dithering: реализация

- Допустим, мы рисуем в однобитное (чёрно-белое) изображение, и хотим добиться серого цвета с помощью дизеринга

Dithering



Dithering: реализация

- Допустим, мы рисуем в однобитное (чёрно-белое) изображение, и хотим добиться серого цвета с помощью дизеринга

Dithering: реализация

- Допустим, мы рисуем в однобитное (чёрно-белое) изображение, и хотим добиться серого цвета с помощью дизеринга
- Нам нужен способ определить, какой из пикселей закрасить чёрным, а какой – белым

Dithering: реализация

- Допустим, мы рисуем в однобитное (чёрно-белое) изображение, и хотим добиться серого цвета с помощью дизеринга
- Нам нужен способ определить, какой из пикселей закрасить чёрным, а какой – белым
- При одноразовой обработке изображения можно оптимизировать паттерн дизеринга под конкретное изображение

Dithering: реализация

- Допустим, мы рисуем в однобитное (чёрно-белое) изображение, и хотим добиться серого цвета с помощью дизеринга
- Нам нужен способ определить, какой из пикселей закрасить чёрным, а какой – белым
- При одноразовой обработке изображения можно оптимизировать паттерн дизеринга под конкретное изображение
- В real-time графике это обычно делается с помощью т.н. dither mask / dither matrix

Dithering: реализация

- Dither mask – одноканальная текстура, хранящая для каждого пикселя пороговое значение, начиная с которого его надо рисовать белым

Dithering: реализация

- Dither mask – одноканальная текстура, хранящая для каждого пикселя пороговое значение, начиная с которого его надо рисовать белым
- Например, для серого цвета с яркостью 0.25 мы нарисуем белыми те пиксели, для которых значение в dither map ≤ 0.25

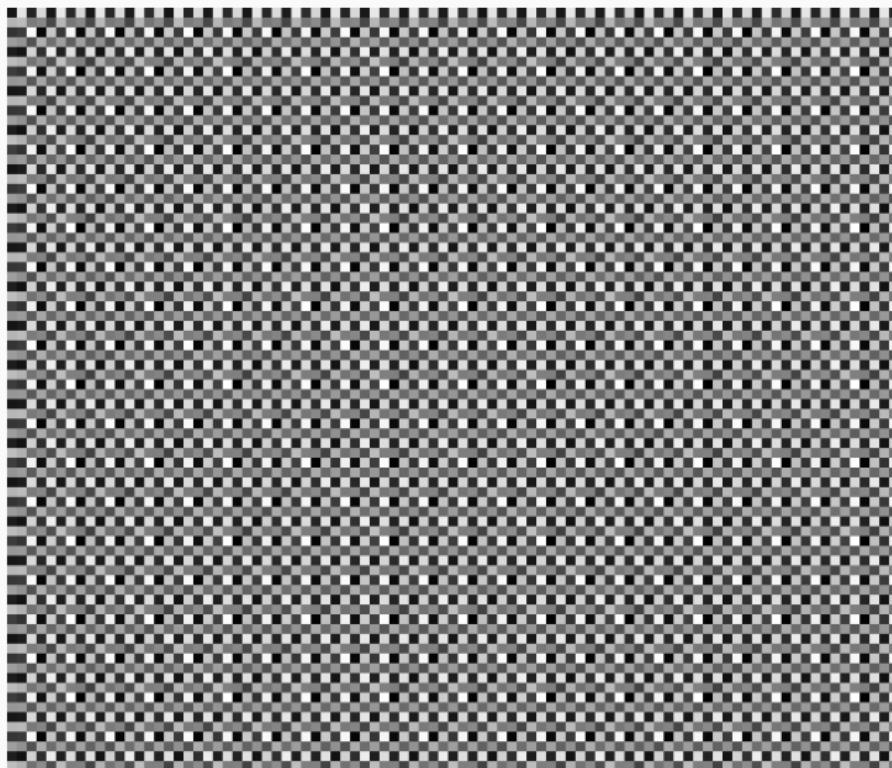
Dithering: реализация

- Dither mask – одноканальная текстура, хранящая для каждого пикселя пороговое значение, начиная с которого его надо рисовать белым
- Например, для серого цвета с яркостью 0.25 мы нарисуем белыми те пиксели, для которых значение в dither map ≤ 0.25
- Например, для серого цвета с яркостью 0.5 мы нарисуем белыми те пиксели, для которых значение в dither map ≤ 0.5

Dithering: реализация

- Dither mask – одноканальная текстура, хранящая для каждого пикселя пороговое значение, начиная с которого его надо рисовать белым
- Например, для серого цвета с яркостью 0.25 мы нарисуем белыми те пиксели, для которых значение в dither map ≤ 0.25
- Например, для серого цвета с яркостью 0.5 мы нарисуем белыми те пиксели, для которых значение в dither map ≤ 0.5
- Например, для серого цвета с яркостью 0.75 мы нарисуем белыми те пиксели, для которых значение в dither map ≤ 0.75

Dither mask



Dithering: реализация

- Для дизеринга при рисовании в 8-битный буфер:
 - Хотим записать значение $v \in [0, 1]$
 - Вычисляем $v' = 255 \cdot v$
 - Вычисляем $v_0 = \text{floor}(v')$
 - Вычисляем $v_1 = v_0 + 1$
 - Вычисляем $dv = v' - v_0$
 - Если значение в dither mask меньше dv , рисуем значение $v_1/255$, иначе $v_0/255$

Dithering: реализация

- Для дизеринга при рисовании в 8-битный буфер:
 - Хотим записать значение $v \in [0, 1]$
 - Вычисляем $v' = 255 \cdot v$
 - Вычисляем $v_0 = \text{floor}(v')$
 - Вычисляем $v_1 = v_0 + 1$
 - Вычисляем $dv = v' - v_0$
 - Если значение в dither mask меньше dv , рисуем значение $v_1/255$, иначе $v_0/255$
- Дизеринг нужно делать для каждого канала (RGB) по отдельности, **до** применения tone mapping и гамма-коррекции

Dithering: реализация

- Для дизеринга при рисовании в 8-битный буфер:
 - Хотим записать значение $v \in [0, 1]$
 - Вычисляем $v' = 255 \cdot v$
 - Вычисляем $v_0 = \text{floor}(v')$
 - Вычисляем $v_1 = v_0 + 1$
 - Вычисляем $dv = v' - v_0$
 - Если значение в dither mask меньше dv , рисуем значение $v_1/255$, иначе $v_0/255$
- Дизеринг нужно делать для каждого канала (RGB) по отдельности, **до** применения tone mapping и гамма-коррекции
- Обычно dither mask имеет маленький размер (напр. 16x16) и циклически (`GL_REPEAT`) повторяется на весь экран; её пиксели пробегают все возможные значения от 0 до 255

Dithering: реализация

```
vec3 color = ...;

vec3 c = color * 255.0;
vec3 c0 = floor(c);
vec3 c1 = c0 + vec3(1.0);
vec3 dc = c - c0;

float threshold = texture(dither_mask, pixel_coord);

color = c0 / 255.0;
if (dc.r > threshold) color.r = c1.r / 255.0;
if (dc.g > threshold) color.g = c1.g / 255.0;
if (dc.b > threshold) color.b = c1.b / 255.0;
```

Генерация dither mask

- Bayer (ordered) dithering – очень быстро генерируется (явная формула), могут быть заметны артефакты из-за регулярности паттерна

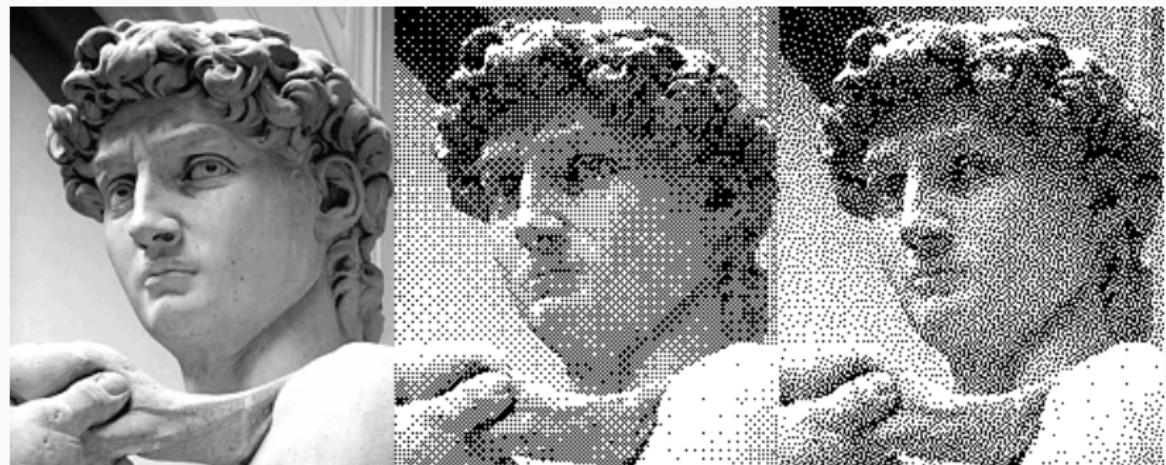
Генерация dither mask

- Bayer (ordered) dithering – очень быстро генерируется (явная формула), могут быть заметны артефакты из-за регулярности паттерна
- Blue noise dithering – использует синий шум (напр. алгоритм void-and-cluster) для генерации маски, лучше распределение пикселей, нет артефактов из-за регулярности

Генерация dither mask

- Bayer (ordered) dithering – очень быстро генерируется (явная формула), могут быть заметны артефакты из-за регулярности паттерна
- Blue noise dithering – использует синий шум (напр. алгоритм void-and-cluster) для генерации маски, лучше распределение пикселей, нет артефактов из-за регулярности
- Готовые текстуры с blue noise dither mask разных размеров можно легко найти онлайн

Сравнение ordered dither и blue noise dither



Screen-door transparency

- Дизеринг часто применяют как дешёвый аналог прозрачности

Screen-door transparency

- Дизеринг часто применяют как дешёвый аналог прозрачности
- Если значение в альфа-канале больше значения из dither mask, то рисуем пиксель, иначе – не рисуем (**discard**)

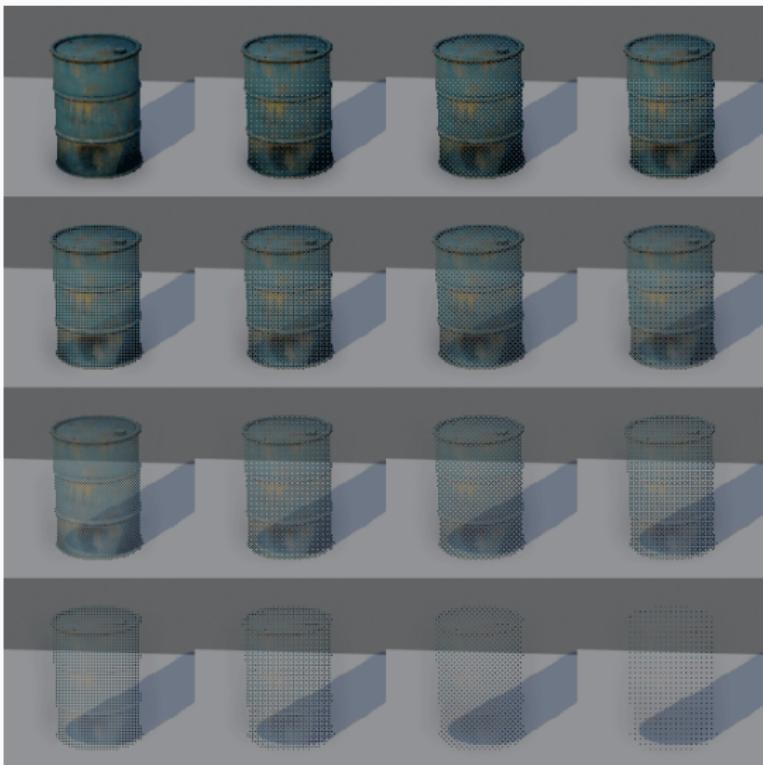
Screen-door transparency

- Дизеринг часто применяют как дешёвый аналог прозрачности
- Если значение в альфа-канале больше значения из dither mask, то рисуем пиксель, иначе – не рисуем (**discard**)
- Не требует сортировки объектов по расстоянию, работает с deferred shading'ом

Screen-door transparency

- Дизеринг часто применяют как дешёвый аналог прозрачности
- Если значение в альфа-канале больше значения из dither mask, то рисуем пиксель, иначе – не рисуем (**discard**)
- Не требует сортировки объектов по расстоянию, работает с deferred shading'ом
- Выглядит заметно хуже, часто применяется для временно прозрачных объектов (появляющихся/исчезающих), далёких объектов (деревья, кусты)

Screen-door transparency



Return of the Obra Dinn (2018)



Dithering: ссылки

- en.wikipedia.org/wiki/Dither
- surma.dev/things/ditherpunk
- alex-charlton.com/posts/Dithering_on_the_GPU
- Dithering in Return of the Obra Dinn
- Shadertoy с примером дизеринга
- Free blue noise textures