



ZÁKLADY POČÍTAČOVÉ GRAFIKY

Redukce barevného prostoru

Ing. Michal Vlnas

ivlnas@fit.vutbr.cz

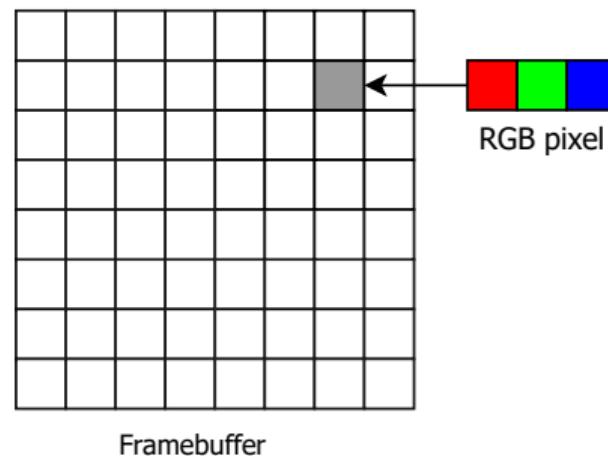


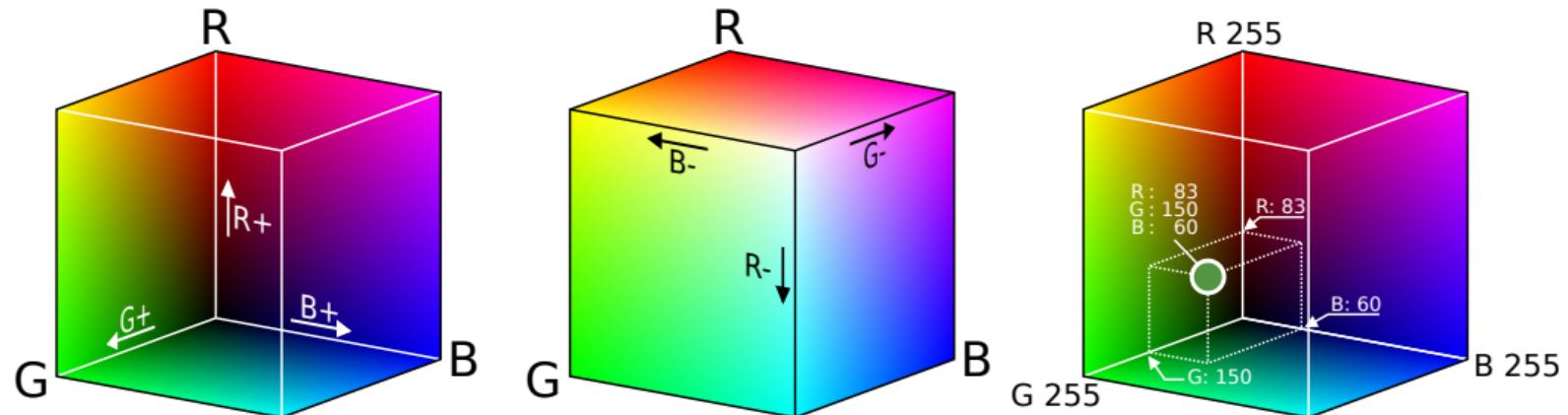
VYSOKÉ UČENÍ FAKULTA
TECHNICKÉ INFORMAČNÍCH
V BRNĚ TECHNOLOGIÍ

2023

- Implementace v jazyku C/C++ s využitím CMake.
- Využita knihovna **SDL 2.0**.
 - Knihovna pro multiplatformní programování médií (zvuk, grafika, vstup) s HW akcelerací.
- Implementace úkolů **pouze** v souboru **student.cpp**.
- Knihovna SDL je součástí kostry a **není potřeba** ji stahovat.

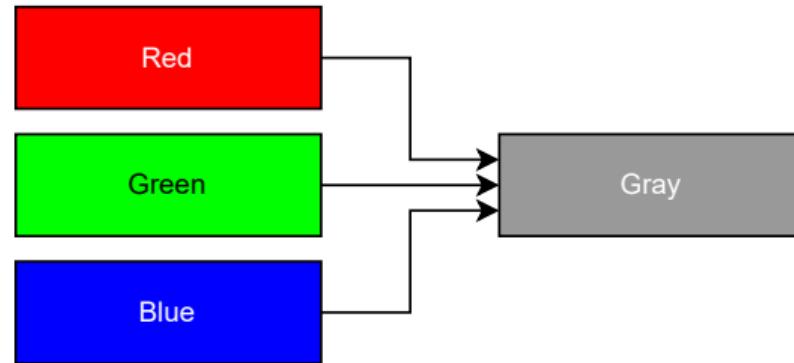
- Pole pixelů reprezentující obraz
- Pixel = pix (in the sense “pictures”) + el(ement), 1965-70
- Vnímáme jako 2D pole
- **V IZG implementaci pomocí 1D pole (efektivnější)**





```
struct RGB
{
    uint8_t r;
    uint8_t g;
    uint8_t b;
};
```

- Frambuffer je reprezentován jako `RGB* frame_buffer`, tedy 1D pole
- Většina operací je založena na přímém zápisu do paměti framebufferu pomocí již implementovaných metod



Vstup

- RGB: $[0, 255] \times [0, 255] \times [0, 255]$

Výstup

- Grayscale: $[0, 255]$
- Dle vztahu: $I = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$

V RGB:

- Všechny složky stejné: $R = G = B = I$



Úkol: implementovat funkci (0.5b)

- void ImageTransform::grayscale()

Pomocné proměnné

- uint32_t cfg->w
- uint32_t cfg->h

Pomocné metody a funkce

- void setPixel(uint32_t x, uint32_t y, RGB color)
- RGB getPixel(uint32_t x, uint32_t y)
- std::round(x) např.: unsigned x = std::round(42.45f);

Testování

- Klávesa "L/K"načte obrázek, "G"převede obrázek do stupňů šedi

Nutno vycházet z šedotónového obrazu

Různé metody:

- Prahování (thresholding)
- Náhodné rozptýlení
- Maticové rozptýlení
- Distribuce chyby

Implementováno (náhodné rozptýlení):

- Klávesa "R" provádí algoritmus náhodného rozptýlení
- Metoda k nahlédnutí v souboru [student.cpp](#)



Úkol: implementovat funkce (1b)

- void ImageTransform::evaluateThreshold()
- void ImageTransform::threshold()

Pomocné proměnné

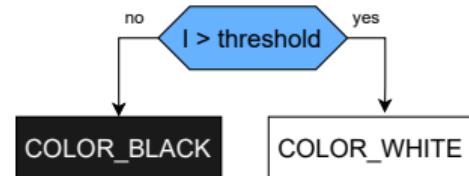
- COLOR_WHITE
- COLOR_BLACK
- uint32_t cfg->w
- uint32_t cfg->h

Pomocné metody a funkce

- void ImageTransform::grayscale()

Testování

- "L/K"načtení obrázku, "T"spustí algoritmus



Jak vypočítat práh (threshold): (**POZOR: mění se pro každou skupinu**)

Z průměru předchozího řádku

Implementovat do funkce

- void evaluateThreshold()

nebo

- void evaluateThreshold(int32_t x, int32_t y)

Lze využít funkci, která vrátí pixel z read-only grayscale bufferu:

- RGB getPixelGrayScale(uint32_t x, uint32_t y)

Převod na černobílý obraz – ukázka



Prahování podle rozptylovací matice. Pro každý pixel najdeme práh v matici prahů.

Výstupní intenzita

Matice prahů

$$M = \begin{bmatrix} 0 & 204 & 51 & 255 \\ 68 & 136 & 187 & 119 \\ 34 & 238 & 17 & 221 \\ 170 & 102 & 153 & 85 \end{bmatrix}$$

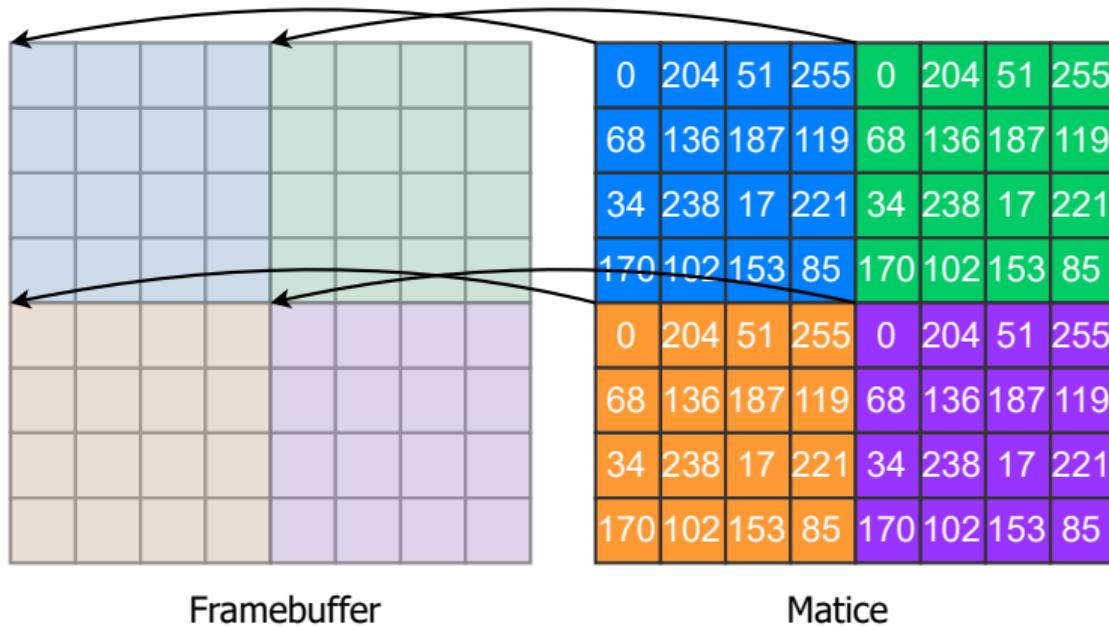
$$G(x, y) = \begin{cases} I_{\max} & I(x, y) > M(i, j) \\ I_{\min} & \text{jinak} \end{cases}$$

Index pro matici M

$$i = x \bmod n$$

$$j = y \bmod n$$

Prahování podle rozptylovací matice. Pro každý pixel najdeme práh v matici prahů.



Úkol: implementovat funkci (0.5b)

- void ImageTransform::orderedDithering()

Pomocné proměnné

- COLOR_WHITE
- COLOR_BLACK
- uint32_t cfg->w
- uint32_t cfg->h
- uint32_t m_side

Pomocné metody a funkce

- void ImageTransform::grayscale()

Testování

- "L/K"načtení obr., "O"spustí algoritmus

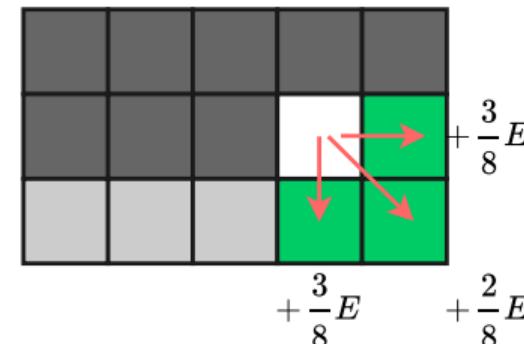


- Všechny dosavadní techniky pracují pouze s jedním pixelém
- Pro lepší kvalitu je vhodné sledovat okolí

$$G(x, y) = \begin{cases} I_{max} & I(x, y) > T \\ I_{min} & \text{jinak} \end{cases}$$

$$E = \begin{cases} I(x, y) - I_{max} & \text{pokud } G(x, y) = I_{max} \\ I(x, y) & \text{pokud } G(x, y) = I_{min} \end{cases}$$

- 1 Thresholding (použijte fixní threshold = 127)
- 2 Výpočet chyby
- 3 Aktualizace okolí chybou
- 4 Nastavení hodnoty pixelu



Úkol: implementovat funkci (1b)

- void ImageTransform::errorDistribution()
- a pomocnou metodu: updatePixelWithError()

Pomocné proměnné

- COLOR_WHITE
- COLOR_BLACK
- uint32_t cfg->w
- uint32_t cfg->h

Pomocné metody a funkce

- void ImageTransform::grayscale()
- std::round(x), std::min(a, b), std::max(a, b)

Testování

- "L/K" načtení obr., "E" spustí algoritmus



Samostatná práce