Konverze obrazového formátu GIF na BMP

František Šumšal

May 5, 2018

1 Struktura GIF a BMP souborů

Jelikož cílem tohoto projektu byla implementace knihovny a aplikace pro konverzi obrázků ve formátu GIF do formátu BMP, budou tyto formáty v následujících odstavcích stručně popsány.

1.1 Graphics Interchange Format – GIF

Soubory ve formátu GIF se mohou vyskytnout ve dvou verzích – a to ve verzi *87a* a *89a*. Tento projekt implementuje podporu pouze verze *89a* [1] a to v její 8-bitové variantě (1 až 7-bitové varianty nejsou podporovány). Navíc se předpokládá, že daný GIF je komprimován algoritmem LZW [2].

Struktura GIFu samotného se skládá z několika částí. Pro stručnost budou detailněji zmíněny jen ty podstatné.

1.1.1 Header Block

Header Block obsahuje signaturu GIF souboru (která je vždy *GIF*) a jeho verzi – v našem případě vždy *89a*.

1.1.2 Logical Screen Descriptor

Následuje blok dat popisující velikost obrázku. Z tohoto bloku jsou důležité bajty *canvas width* a *canvas height* určující velikost obrázku a dále bity s informacemi o globální tabulce barev (*GCT* [3]) – jeden bit pro určení, zda-li daný GIF obsahuje GCT, a další tři bity určující *velikost* (pro získání skutečné velikosti GCT o velikosti N je třeba použít vzorec $2^{(N+1)}$).

1.1.3 Global Color Table

Pokud Logical Screen Descriptor obsahuje informaci o přítomnosti GCT, následuje pole hodnot této tabulky, které je reprezentováno trojicí bajtů pro každou barvu (složky R, G a B).

1.1.4 Graphics Control Extension

Tento blok obsahuje data, která jsou využita k nastavení transparentnosti a pro animované GIFy. Jelikož tato implementace podporuje pouze statické GIFy, je tento blok parsován, ale nevyužit.

1.1.5 Image Descriptor

Dalším blokem jsou samotná data obrázku. Tento blok může být přítomen vícekrát v animovaných GIFech. Každý obrázek může mít svou vlastní tabulku barev (*LCT* [4]), pro kterou platí stejná pravidla jako pro globální.

Data obrázku jsou uložena po blocích – první bajt obsahuje velikost následujícího bloku (0 – 255), dále následují data samotná. Ukončovací blok má velikost 0.

Samotné pixely obrázku jsou komprimovány algoritmem LZW, který je v GIF verzi lehce pozměněn – využívá proměnnou délku komprimačního kódu. V této variantě se kromě komprimačních kódu vyskytují navíc dva

[1] https://www.w3.org/Graphics/GIF/spec-gif89a.txt
[2] Lempel-Ziv-Welch
[3] Global Color Table
[4] Local Color Table

speciální kódy: *Clear Code*, který slouží pro inicializaci tabulky kódů a jeho hodnota je $2^{codesize}$, a *End of Information*, který značí konec obrazových dat (hodnota $2^{codesize} + 1$). Dekomprimace dat probíhá následovně:

1. Načtení prvního bajtu, který obsahuje minimální velikost kódu, a jeho inkrementaci o 1 (pro Clear Code a EOI) – code size (CS)
2. Dalších CS bitů by mělo obsahovat *Clear Code* – indikace pro inicializaci kódové tabulky, a to následovně:
   - Indexy 0 - $2^N - 1$: obsah GCT/LCT
   - Index 2^N: Clear Code
   - Index $2^N + 1$: End of Information
   - Další indexy slouží pro (de)komprimační kódy
3. Načtení dalších CS bitů – tento kód by měl být v kódové tabulce. V tomto kroku tento kód pouze vypíšeme na výstup a uložíme do proměnné pro uložení předchozího kódu
4. Dále v cyklu načítáme vždy CS bitů:
   - Pokud se načtený kód vyskytuje v kódové tabulce, vypíšeme obsah pro tento kód na výstup a do tabulky přidáme nový záznam, který bude obsahovat data předchozího kódu + první položku z dat aktuálního kódu
   - Pokud se kód v tabulce nevyskytuje, přidáme do tabulky data předchozího kódu + první položku z dat předchozího kódu. Zároveň výsledek vypíšeme
   - Pokud je nový kód roven maximální hodnotě aktuální velikosti kódu (CS), provedeme inkrementaci CS o 1 – avšak pouze v případě, že je hodnota CS menší než 12
   - Pokud je kód Clear Code, provedeme reinicializaci kódové tabulky a nastavíme CS na původní hodnotu

Následující bloky GIF souboru jsou určené pro volitelná rozšíření, která nejsou podstatná pro tento projekt.

1.2 Bitmap file format - BMP

Načtený GIF je konvertován do formátu BMP. Struktura tohoto formátu obsahuje hlavičku bitmapy (bitmap file header), která obsahuje základní informace o obrázku, jako jeho signaturu (zde vždy *BM*), velikost v bajtech a adresu, na které začínají samotné pixely.

Další hlavičkou je *DIB header* (zde vždy typu *BITMAPINFOHEADER*), která obsahuje velikost obrázku v pixelech, barevnou hloubku (8 bitů), velikost tabulky barev a další informace. Data obrázku mohou být komprimována, ale aktuální implementace ukládá data nekomprimovaně.

Pokud je barevná hloubka menší nebo rovna 8 bitům, neobsahují pixely BMP souboru informace o barvě, ale jsou indexem do tabulky barev, která je v tomto případě povinná. Lze zde jednoduše použít data z tabulky barev GIF souboru, jen je nutné změnit pořadí uložení dat – barevné složky jsou zde uloženy v pořadí B, G a R.

Následují samotné pixely. Standardně jsou pixely uloženy v obráceném pořadí než je zvyklé (odspodu nahoru), ale v případě nastavení záporné výšky obrázku v DIB hlavičce je možné data ukládat odshora dolů. Pixely jsou rozděleny do řádků, kde každý řádek musí být zarovnán na 32-bitů. Jak bylo zmíněno výše, v případě 8-bitové barevné hloubky je každý pixel index do tabulky barev.

2 Implementace

Knihovna a aplikace pro převod GIF obrázků do BMP je implementována v jazyce C++, avšak z C++ knihoven jsou využity jen kontejnery (hash tabulky a vektory) pro jednodušší manipulaci s daty.

Jak bylo popsáno výše, v případě GIF souborů je podporován pouze statický 8-bitový GIF verze 89a bez rozšíření, komprimovaný LZW algoritmem. BMP soubory jsou vytvářeny s negativní výškou, pro uložení dat odshora dolů, a jednotlivé pixely nejsou komprimovány.

2.1 Parametry aplikace

`./gif2bmp [-i infile] [-l logfile] [-o outfile]`, kde:

- **-i**: vstupní GIF soubor *infile* – pokud není zadán, použije se standardní vstup (stdin)
- **-l**: soubor pro uložení výstupní zprávy – pokud není zadán, nebude zpráva vypsána
  Zpráva je ve formátu:

  ```
  login = xsumsa01
  uncodedSize = <velikost GIF souboru v bajtech>
  codedSize = <velikost BMP souboru v bajtech>
  ```

- **-o**: výstupní BMP soubor *outfile* – pokud není zadán, použije se standardní výstup (stdout)