Grafické formáty GIF a PNG

Graphics formats GIF & PNG

Jiří Fůsek
ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: Jiří FŮSEK
Studijní program: B 3902 Inženýrská informatika
Studijní obor: Informační technologie
Téma práce: Grafické formáty GIF a PNG

Zásady pro vypracování:

1. Vytvořte literární rešerší na zadané téma. Ta bude obsahovat historii a vývoj rastrových grafických formátů. Zaměřte se především na formáty GIF a PNG.

2. Podrobně popište grafické formáty GIF a PNG. Zaměřte se zejména na jejich strukturu.

3. Ze získaných informací vytvořte prezentaci v Powerpointu, která by se dala použít při výučce.


5. Vytvořte program podle předchozího bodu zadání. Vytvořte dokumentaci k programu a zdrojové kódy doplňte komentáři, usnadňující pochopení tohoto kódu.

6. Koncepci programu navrhněte tak, aby se se načítání/ukládání formátů GIF a PNG dalo použít i v jiných programech.
ABSTRAKT

Cílem této bakalářské práce je vytvořit aplikaci v programovacím jazyce C/C++, která bude umět pracovat s grafickými formáty typu GIF a PNG (jde především o jejich načítání, ukládání a zobrazování), společně s prezentací těchto formátů pro potřeby výuky počítačové grafiky. Dále je důležité vytvořit k tomuto programu patřičnou dokumentaci a zdrojové kódy doplnit o komentáře, usnadňující pochopení celé aplikace. Teoretická část práce se zabývá historií a vývojem rastrových grafických formátů, přičemž důraz je kladen právě na grafické formáty GIF a PNG, u kterých je navíc podrobně zpracován popis jejich struktury.

Klíčová slova: GIF, PNG, FreeImage, C++

ABSTRACT

The aim of the bachelor’s thesis is to create application in programming language C/C++, which will deal with graphics formats – type GIF and PNG (their retrieving, saving and displaying) in conjunction with the presentation of these formats for education of computer graphic. Then is important to create for this programme an appropriate documentation and in source codes complete the commentary, which help us to understand the whole application. The theoretical part of the thesis is engaged in the history and the progress of the raster graphic formats, first of all graphics formats GIF and PNG, which structure is detaily described.

Keywords: GIF, PNG, FreeImage, C++
Poděkování:

Rád bych tímto poděkoval vedoucímu práce panu Ing. Pavlu Pokornému, Ph.D. za cenné rady v průběhu řešení této práce a hlavně za „betatesting“ programu a upozornění na řadu chyb či nápadů. Dále bych chtěl poděkovat rodičům a přátelům za jejich podporu při studiu.

Motto:

„U počítačů není nic nemyšlitelné, natož pak nemožné - kromě toho, co je potřeba.“

„Dítě, které dosáhne rukama na klávesnici, přijde hned při první příležitosti na kombinaci kláves, kterou se dá něco zničit. Existuje-li více možností, pak si vybere tu, která má nejkatastrofálnější následky.“

Murphyho počítačové zákony

Prohlašuji, že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně

Podpis diplomanta
OBSAH

ÚVOD .............................................................................................................................. 7
I TEORETICKÁ ČÁST .................................................................................................. 8
1 BITMAPOVÁ GRAFIKA ............................................................................................. 9
  1.1 DEFINICE OBRAZU V BITMAPOVÉ Grafice .................................................. 9
  1.2 VÝHODY A NEVÝHODY BITMAPOVÉ GRAFIKY ........................................ 10
  1.3 VYUŽITÍ BITMAPOVÉ GRAFIKY .................................................................. 11
  1.4 BITMAPOVÉ FORMÁTY ................................................................................. 12
2 HISTORIE A VÝVOJ GRAFICKÉHO FORMÁTU GIF A PNG ................... 13
  2.1 GIF ............................................................................................................... 13
      2.1.1 Základní charakteristika ........................................................................ 14
  2.2 PNG ............................................................................................................. 15
      2.2.1 Základní charakteristika ........................................................................ 15
3 ANATOMIE GRAFICKÉHO FORMÁTU GIF ................................................... 16
  3.1 INTERNÍ STRUKTURA FORMÁTU GIF...................................................... 16
      3.1.1 Pořadí bloků souboru tvořeném jedním nebo více rámci ....................... 17
  3.2 BINÁRNÍ PODOBA OBRÁZKU TYPU GIF O VELIKOSTI 1X1 PIXEL ............... 18
      3.2.1 Význam jednotlivých sekvencí byteů ................................................. 19
4 ANATOMIE GRAFICKÉHO FORMÁTU PNG .................................................. 23
  4.1 OBECNÉ INFORMACE O STRUKTUŘE SOUBORŮ PNG ....................... 23
  4.2 HLAVIČKA SOUBORU TYPU PNG .............................................................. 23
  4.3 POPIS CHUNKŮ .......................................................................................... 26
      4.3.1 Obecná struktura chunků ....................................................................... 26
      4.3.2 Způsob pojmenování chunků .................................................................. 27
      4.3.3 Nejvýznamnější povinné chunky ......................................................... 28
          4.3.3.1 Chunk typu IHDR ........................................................................... 28
          4.3.3.2 Chunk typu IEND ........................................................................... 30
      4.3.4 Nejvýznamnější nepovinné chunky ..................................................... 30
  4.4 STRUKTURA SOUBORU PNG S OBRÁZKEM ULOŽENÝM VE STUPNÍCH ŠEDI 31
      4.4.1 Binární podoba obrázku uloženého ve stupních šedi ......................... 31
II PRAKTICKÁ ČÁST ............................................................................................... 34
5 VÝVOJ PROGRAMU .............................................................................................. 35
  5.1 VÝBĚR KNIHOVEN ....................................................................................... 35
      5.1.1 FreeImage Charakteristika ..................................................................... 35
      5.1.2 Základní vlastnosti knihovny FreeImage .............................................. 36
  5.2 TRÍDY V APLIKACI ....................................................................................... 37
  5.3 MFC ............................................................................................................. 40
  5.4 SPUSTĚNÍ PROGRAMU ............................................................................... 41
  5.5 IMPLEMENTACE KNIHOVNY FREEIMAGE V PROGRAMU .................. 41
      5.5.1 Otevření obrázku .................................................................................. 41
      5.5.2 Uložení obrázku .................................................................................... 43
      5.5.3 Obnova změn na obrázku ....................................................................... 44
| 5.5.4 | Zobrazení obrázku | 45 |
| 5.5.5 | Obsluha zpráv v programu | 45 |
| 5.6 | IMPLEMENTACE V JINÝCH PROGRAMECH | 47 |
| 5.7 | GENEROVANÁ DOKUMENTACE K PROGRAMU | 47 |
| 6 | UŽIVATELSKÁ PŘÍRUČKA K PROGRAMU | 48 |
| 7 | PREZENTACE PRO VÝUKU | 49 |
|  | ZÁVĚR | 50 |
|  | SEZNAM POUŽITÉ LITERATURY | 52 |
|  | SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK | 53 |
|  | SEZNAM OBRÁZKŮ | 54 |
|  | SEZNAM TABULEK | 55 |
|  | SEZNAM PŘÍLOH | 56 |
ÚVOD

Grafické formáty stanovují pravidla, podle kterých je obrázek uložen v souboru. Některé formáty mohou do souboru ukládat i další informace, např. náhled obrázku v malém rozlišení, informace o expozici, datum a čase pořízení a podobně.

Formát grafického souboru je formát, ve kterém jsou grafická data – data popisující grafickou předlohu – uložena ve formě souboru. Tyto soubory byly zavedeny z toho důvodu, že postupně vyvíjala potřeba ukládat, organizovat a znovu obnovovat grafická data efektivně a logicky.

Prakticky každá důležitá aplikace vytváří a ukládá nějaká grafická data. I ty nejjednodušší textové editory, pracující ve znakovém módu, dovolují vytvářet soubory, které obsahují čáry vytvořené z ASCII znaků nebo z escape – posloupností. GUI aplikace, které jsou v dnešní době velmi rozšířené, potřebují podporu hybridních formátů kvůli spolupráci s bitmapovými daty v textových dokumentech. Databáze rozšířené o grafické předlohy rovněž dovolují ukládat v jednom souboru bitmapová a textová data dohromady. Navíc, grafické soubory jsou důležitým transportním mechanismem, který podporuje vizuální data mezi aplikacemi a systémem počítače.

Tuto práci jsem si vybral především proto, abych zdokonalil své programovací schopnosti spočívající v návrhu aplikace, výběru co nejvhodnější knihovny a maximálního využití možností, které daná knihovna nabízí. Práce je rozčleněna na teoretickou a praktickou část.

V teoretické části je cílem podrobně popsat rastrové grafické formáty GIF a PNG. Stručně charakterizovat jejich historii, vývoj, využití, výhody a nevýhody a hlavně důkladně rozehrat jejich strukturu.

V praktické části jsem v programovacím jazyce C/C++ navrhl a naprogramoval s využitím knihovny FreeImage aplikaci, která umí pracovat s grafickými formáty GIF a PNG. Kromě standardních věcí jako je jejich načítání, zobrazování a ukládání umožňuje aplikace také základní transformace s načtenými obrázky jako je např. změna velikosti obrázku, otočení obrázku o zadaný počet stupňů nebo inverzi barev. Program je také řádně popsán a zdokumentován v praktické části textu. Součástí práce je ještě vygenerovaná dokumentace pomoci programu Doxygen, která detailně popisuje strukturu aplikace a také prezentace vytvořená programem Microsoft PowerPoint, popisující grafické formáty GIF a PNG, určená jako pomůcka pro výuku předmětu Počítačová grafika.
I. TEORETICKÁ ČÁST
1 BITMAPOVÁ GRAFIKA

Bitmapová grafika (rastrová grafika) je jeden ze dvou základních způsobů, jakým počítače ukládají a zpracovávají obrazové informace. Spolu s vektorovou grafikou představují dva základní způsoby ukládání obrázků. [1]  

Už podle názvu je patrné, že grafika je složena z rastru (tedy jakési pomyslné sítě bodů, tzv. bitmapy), kde každý bod má definovanou svou barvu a jas. Při určitém množství a jemnosti rastru začnou body opticky splývat a vytvoří obraz.

Obrázek č. 1 - Převod obrazu do bitmapové grafiky

1.1 Definice obrazu v bitmapové grafice

1.2 Výhody a nevýhody bitmapové grafiky


Hlavní nevýhodou bitmapové grafiky je její datová náročnost. Kvůli skutečnosti, že každý bod obrazu musí nést informaci o svém jasu (v případě černobílých bitmap), své barvě (v případě barevných bitmap), případně ještě další informaci o průhlednosti, zabírají rozměrné bitmapy na disku velký úložný prostor.

Obrázek č. 2 – Demonstrace zhoršení kvality bitmapového obrázku při jeho zvětšení

Druhou nevýhodou bitmapové grafiky je, že ji nelze bez snížení kvality zvětšovat. Při zvětšování dochází k interpolaci, kdy se pixely v podstatě roztahují a vyhlazují. V případě kvalitního zvětšovacího algoritmu u specializovaných programů lze dosáhnout zvětšení kvalitních rastrových podkladů (např. fotografii) až o 30 % bez výrazné degradace obrazu, ale spolu s klesající kvalitou zdrojových dat výrazně klesá i možnost dalšího zvětšení. [2]
1.3 Využití bitmapové grafiky

Bitmapová grafika vyniká tam, kde by byla vektorová grafika příliš komplexní (fotografie, složité ilustrace plné stínů a rozmanitých barev atp.) nebo když je třeba zdigitalizovat data, u nichž nelze provést jejich jednoduchou vektorizaci.

Bitmapová grafika má své využití napříč všemi počítačovými obory. Její využití sahá od drobných grafických prvků na internetových stránkách, přes bitmapové textury aplikované na 3D objekty, až po fotografie připravené pro DTP.

Jedním z nejpoužívanějších programů, využívaným pro tvorbu a úpravu rastrové grafiky pro internet a pro tisk, je v současné době Adobe Photoshop. Jeho nativní formát PSD podporuje ukládání rastrové grafiky ve vrstvách spolu s vektorovými objekty a editovatelným textem. Pro kvalitní přenos fotografii se nejčastěji používá rastrový formát TIF (příp. TIFF), který je však, stejně jako většina bezztrátových rastrových formátů, pro svou datovou náročnost nevhodný pro použití na webu či v digitálních fotoaparátech. Na webu je nejrozšířenějším rastrovým formátem GIF, JPEG a PNG. [2]
1.4 Bitmapové formáty

Obrazem rastrových (bitmapových) formátů je matice grafických elementů, bodů a pixelů, z nichž posledně jmenovaný má jediný atribut, a to barvu. Tento formát většinou zahrnuje komprimaci a z hlediska počtu barev jej lze použít pro monochromatické a barevné obrazy nebo obrazy ve stupních šedi. Mezi nejběžnější bitmapové formáty patří Windows BitMaP (BMP), Graphics Interchange Format (GIF), Tagged Image File Format (TIFF) a Joint Photographic Experts Group (JPEG). [3]

Soubor, ve kterém je obraz uložen, se skládá z hlavičky (identifikace, verze a informace o uloženém obrazu = pozice, rozměry, poměry stran, počet řádků předlohy, počet pixelů na řádku, hloubka pixelu (= počet možných barev) a způsob uložení grafických dat), dále z palety a samotných dat (informace o barvách pixelů - nejčastěji RGB).

Kreslení a úprava v rastrových editorech je realizována pomocí změny barvy bodů. K dispozici jsou základní geometrické tvary, různé typy čar, rozsáhlé možnosti výplně (s možností přechodu barev či vzorků), úpravy rastru (barev, velikostí), výřezy (kopírování, otočení, posun, zrcadlení), retušovací nástroje (zaostření, rozmazání, apod.), rastrové efekty a samotný export do rastrových formátů.

Rastrový formát je ideální pro předlohy z reálného světa, snadné vytváření z dat uložených do paměťového pole. Pixelové hodnoty mohou být měněny hromadně a dalším kladem je snadný přenos na rastrová výstupní zařízení. [3]
2 HISTORIE A VÝVOJ GRAFICKÉHO FORMÁTU GIF A PNG

2.1 GIF


Zásadní komplikací, jež významně ohrozila dominanci GIFu na poli internetové grafiky, se stala patentová kauza, vyvolaná společností Unisys. Ta totiž vlastní patent na LZW kompresi, použitou v rámci GIFu. V roce 1994 začal Unisys nutit výrobce komerčních programů, pracujících s GIFem, k platbě licenčních poplatků. V roce 1999 pak dokonce obdobným způsobem začal usilovat o platby z nekomerčního software či od uživatelů. Uvedená skutečnost vedla k tomu, že mnozí výrobci raději přestali používat GIF na prozatímně zřejmě podporovanou alternativu - formát PNG (podnětem byla také některá omezení GIFu), kampaně jako Burn All GIFs pak vyzývaly k náhradě GIFu jinými formáty. I přes uvedené tráble si ovšem GIF udržel své výsadní postavení. Krize kolem patentu navíc v podstatě končí, neboť práva Unisysu vypršela v letech 2003 (USA) a 2004 (Japonsko, Evropa, Kanada). [8]

Formát GIF podporuje osmibitovou grafiku, což znamená, že obrázek může mít maximálně 256 barev (každý z 8 bitů může nabývat hodnotu 0 nebo 1, pokud uvážíme všechny možné kombinace nul a jedniček v osmici bitů, dostaneme celkem 256 možností). Formát GIF samozřejmě podporuje i menší barevnou hloubku, například 64 nebo 32 barev.
2.1.1 Základní charakteristika

Hlavní vlastností, která napomohla GIFu k jeho popularitě, je malý objem obrázků, uložených v uvedeném formátu. Toho je dosaženo jednak zmíněnou LZW kompresí a dále tím, že obrázky v GIFu mohou obsahovat pouze několik málo barev, maximálně 256 (v módu RGB), přičemž použité barvy jsou k obrázku připojeny v podobě tzv. palety. (Existují i metody, jak s pomocí více palet připojených k obrázku vyjádřit v rámci GIFu TrueColor barvy, uvedená metoda ovšem není příliš optimální a ani příliš podporovaná.) Vzhledem k poslání formátu (zobrazení na obrazovce, nikoli kvalitní tisk) bývají obrázky v uvedeném formátu ukládány v nízkém rozlišení (typicky rozlišení obrazovky, tedy 72 či 96 dpi). Všechny tyto zmíněné vlastnosti činí GIF prakticky nepoužitelným v pre-pressu, i když odpovídající aplikace (sazba, editace bitmap apod.) dokází soubory v uvedeném formátu (například screenshoty apod.) zpracovat uspokojivým způsobem. [8]

Obrázek č. 5 – Ukázka „znitosti“ a ztráty plynulosti barevných přechodů u obrázku typu GIF

GIF nabízí i jiné vlastnosti, zohledňující specifika on-line zobrazení. Konkrétně se to týká tzv. progresivního zobrazení, využívajícího proklad (interlacing), dovolujícího vykreslení hrubého náhledu obrázku již po načtení části souboru. GIF také v omezené mře podporuje průhlednost (transparency), umožňující v obrázku vypustit zobrazení určité barvy (vede k různým zajímavým vizuálním efektům). Možnost uložení více obrázků do tétohož souboru je pak využívána k tvorbě animací v uvedeném formátu (zejména reklamní bannery). [8]

Někdo by se při pohledu na uvedený výčet vlastností mohl dotázat, proč vlastně GIF obstojí v konkurenci formátů jako je např. JFIF, JPEG atp. Hlavním důvodem je zde především rozdílná schopnost uvedeného formátu reprodukovat určité typy grafických motivů: JPEG je vhodný spíše pro "přírodní"obrázky, typicky digitální fotografie, zatímco GIF si dokáže lépe poradit s obrázky, obsahujícími pravidelné tvary (typicky elektronická grafika - tlačítka, čáry, loga, kliparty, screenshots apod.).
2.2 PNG

Impuls pro vytvoření formátu PNG přišel v roce 1994, kdy po dohodě firem Unisys a CompuServe došlo k licenčnímu zpoplatnění za použití formátu GIF. Po následných úpravách licence se začala vztahovat nejen na velké firmy, ale i na programátory vyvíjející freeware, shareware a také programy šířené pod volnou licencí (GPL). Tato politika vyvolala velkou vlnu odporu, která vyvrcholila akcí „Bun All GIFs“, které se zúčastnili jak velké softwarové firmy, tak samotní programátoři na svých soukromých stránkách.


Současně s vývojem GIF24, ale ne u žádné softwarové firmy, se začal vyvíjet naprosto nový formát, který neměl být vázán žádným patentem ani vztahem k žádné firmě a předčil by tehdejší grafické formáty (především GIF, JPEG). Původní název zněl PBF (Portable Bitmap Format), ale posléze byl změněn na PNG (Portable Network Graphics). [7]

Obrázek č. 6 – Příklad obrázku PNG s 8-bitovou průhledností

2.2.1 Základní charakteristika

PNG je grafický formát určený pro bezeztrátovou kompresi rastrové grafiky. Byl vyvinut jako zdokonalení a náhrada formátu GIF, který byl patentově chráněný (LZW84 algoritmus), dnes jsou patenty prošlé. PNG nabízí podporu 24 bitové barevné hloubky, nemá tedy jako GIF omezení na maximální počet 256 barev současně. PNG tedy do jisté míry nahrazuje GIF, nabízí více barev a lepší kompresi (algoritmus Deflate + filtry). Navíc obsahuje osmibitovou průhlednost (tzv. alfa kanál). To znamená, že obrázek může být v různých částech různě průhledný (tzv. RGBA barevný model). Nevýhodou PNG oproti GIF je praktická nedostupnost jednoduché animace, pro kterou sice existují 2 návrhy APNG a MNG, které se ale zatím neprosadily. PNG se stejně jako formáty GIF a JPEG používá na internetu. [7]
3 ANATOMIE GRAFICKÉHO FORMÁTU GIF

3.1 Interní struktura formátu GIF

V první řadě je třeba si vysvětlit, jakým způsobem jsou data v grafickém formátu GIF interně ukládána.

Celý soubor (resp. v případě přenosu po datových sítích "pouhý" tok bytů) je rozdělen do bloků, z nichž některé jsou povinné, některé se musí vyskytovat na určitém místě souboru, další se mohou opakovat apod. Povolené bloky, které se mohou v GIFu vyskytovat, jsou popsány v následující tabulce spolu s informací, zda se jedná o bloky povinné (tj. bloky, které se musí vyskytovat v každém validním souboru) a zda je možné bloky v rámci jednoho souboru opakovat (pro účely animací, slideshow apod.). Dále je u každého bloku uvedeno, zda se jeho definice vyskytovala už ve specifikaci GIF87a či až v novější specifikaci GIF89a. [5]

*Tabulka č. 1 – Tabulka povolených bloku vyskytujících se v souborech typu GIF*

<table>
<thead>
<tr>
<th>Název bloku</th>
<th>Povinná položka?</th>
<th>Lze opakovat?</th>
<th>Verze</th>
</tr>
</thead>
<tbody>
<tr>
<td>signatura GIFu</td>
<td>ano</td>
<td>ne</td>
<td>87a</td>
</tr>
<tr>
<td>verze souboru</td>
<td>ano</td>
<td>ne</td>
<td>87a</td>
</tr>
<tr>
<td>popis logické obrazovky</td>
<td>ano</td>
<td>ne</td>
<td>87a</td>
</tr>
<tr>
<td>globální barvová paleta</td>
<td>ne</td>
<td>ne</td>
<td>87a</td>
</tr>
<tr>
<td>rozšiřující grafický blok</td>
<td>ne</td>
<td>ano</td>
<td>89a</td>
</tr>
<tr>
<td>popis rámce</td>
<td>ano</td>
<td>ano</td>
<td>87a</td>
</tr>
<tr>
<td>lokální barvová paleta</td>
<td>ne</td>
<td>ano</td>
<td>87a</td>
</tr>
<tr>
<td>data rámce (pixely)</td>
<td>ano (pro rámec)</td>
<td>ano</td>
<td>87a</td>
</tr>
<tr>
<td>rozšiřující informace</td>
<td>ne</td>
<td>ano</td>
<td>89a</td>
</tr>
<tr>
<td>(textové, binární)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ukončovací znak GIF souboru</td>
<td>ano</td>
<td>ne</td>
<td>87a</td>
</tr>
</tbody>
</table>

Pro zjednodušení je dobré popis interní struktury GIFu ukázat na nejjednodušším prakticky použitelném souborem, který je možné vytvořit. Podrobný popis bude uveden v následující kapitole. Zde si však nejprve řekneme, jaké bloky je nutné i v tom nejjednodušším obrázku použít a v jakém pořadí.
3.1.1 Pořadí bloků souboru tvořeném jedním nebo více rámců

*Tabulka č. 2 - Tabulka znázorňující pořadí bloků v souborech typu GIF*

<table>
<thead>
<tr>
<th>Pořadí</th>
<th>Název bloku</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>signatura GIFu</td>
</tr>
<tr>
<td>2</td>
<td>verze souboru</td>
</tr>
<tr>
<td>3</td>
<td>popis logické obrazovky</td>
</tr>
<tr>
<td>4</td>
<td>globální barová paleta</td>
</tr>
<tr>
<td>5</td>
<td>popis rámce</td>
</tr>
<tr>
<td>6</td>
<td>data rámce (pixely)</td>
</tr>
<tr>
<td>7</td>
<td>ukončovací znak GIF souboru</td>
</tr>
</tbody>
</table>

Výše uvedeným způsobem jsou uloženy prakticky všechny statické GIFy, tj. GIFy neobsahující animaci ani optimalizaci s využitím více rámců. V případě, že se ukládají animované GIFy, je nutné každý snímek animace reprezentovat pomocí minimálně jednoho rámce a mezi tyto rámce vložit pauzu pomocí rozšířujícího grafického bloku. V případě, že se má animace opakovat, je nutné přidat i informaci o počtu opakování (popř. o nekonečné animaci), což se děje pomocí bloku aplikačního rozšíření. Animace tvořená třemi snímkami (rámců) bude mít následující strukturu: [5]

*Tabulka č. 3 - Tabulka znázorňující pořadí bloků v souboru typu GIF tvořeném třemi rámcem*

<table>
<thead>
<tr>
<th>Pořadí</th>
<th>Název bloku</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>signatura GIFu</td>
</tr>
<tr>
<td>2</td>
<td>verze souboru</td>
</tr>
<tr>
<td>3</td>
<td>popis logické obrazovky</td>
</tr>
<tr>
<td>4</td>
<td>globální barová paleta</td>
</tr>
<tr>
<td>5</td>
<td>aplikační rozšiřující blok se smyčkou</td>
</tr>
<tr>
<td>6</td>
<td>první rozšiřující grafický blok (nastavení zpoždění)</td>
</tr>
<tr>
<td>7</td>
<td>první popis rámce</td>
</tr>
<tr>
<td>8</td>
<td>první lokální barová paleta (pouze pokud se liší od globální)</td>
</tr>
<tr>
<td>9</td>
<td>první data rámce (pixely)</td>
</tr>
<tr>
<td>10</td>
<td>druhý rozšiřující grafický blok (nastavení zpoždění)</td>
</tr>
<tr>
<td>11</td>
<td>druhý popis rámce</td>
</tr>
<tr>
<td>12</td>
<td>druhý lokální barová paleta (pouze pokud se liší od předchozí)</td>
</tr>
<tr>
<td>13</td>
<td>druhý data rámce (pixely)</td>
</tr>
<tr>
<td>14</td>
<td>třetí rozšiřující grafický blok (nastavení zpoždění)</td>
</tr>
<tr>
<td>15</td>
<td>třetí popis rámce</td>
</tr>
<tr>
<td>16</td>
<td>třetí lokální barová paleta (pouze pokud se liší od předchozí)</td>
</tr>
<tr>
<td>17</td>
<td>třetí data rámce (pixely)</td>
</tr>
<tr>
<td>18</td>
<td>ukončovací znak GIF souboru</td>
</tr>
</tbody>
</table>
3.2 Binární podoba obrázku typu GIF o velikosti 1x1 pixel

Absolutně nejmenší obrázek typu GIF, který je možné zobrazit ve všech prohlížečích, obsahuje pouze globální barvovou paletu se dvěma položkami, jeden rámec o rozlišení 1×1 pixel a v něm uložený pouhopouhý jeden pixel. Délka souboru je rovna 35 bytům, což se sice na první pohled může zdát mnoho, ale "konkurenční" minimální PNG má celých 65 bytů. Teoreticky je sice možné GIF ještě zmenšit, například vynecháním globální barvové palety či dokonce celého rámce, ale takto vytvořený GIF mnoho prohlížečů odmítne zpracovat. Zde se tedy budeme zabývat 35-bytovým miniobrázkem, který má při zobrazení v některém z hexadecimálních editorů (KHexEdit, bvi, vim s xxd, Hiew, biew, Dos Navigátor nebo např. český PSpad obsahující hexadecimální editor) tento tvar: [5]

| 0000000: | 47 49 46 38 39 61 01 00 01 00 00 80 00 00 ff 80 00 |
| 0000010: | ff ff ff 2c 00 00 00 00 01 00 01 00 00 02 02 44 |
| 0000020: | 01 00 3b |

**Obrázek č. 7 - Ukázka nejmenšího možného obrázku typu GIF zobrazeného v multieditoru PSpad**

Jak už bylo uvedeno výše, obsahuje tento soubor sedm bloků: signaturu GIF, verzi souboru, popis logické obrazovky, globální barvovou paletu, popis rámce, data rámce a ukončovací znak GIF souboru. Všechny zmiňované bloky jsou uloženy za sebou, takže si předchozí zápis můžeme blíže ujasnit a rozepsat v následující tabulce.
3.2.1 Význam jednotlivých sekvencí bytů

<table>
<thead>
<tr>
<th>Offset (dec)</th>
<th>Byte či sekvence (hex)</th>
<th>Význam bytové sekvence</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>47 49 46</td>
<td>ASCII řetězec ’GIF’ – “magické” číslo souboru (signatura GIF)</td>
</tr>
<tr>
<td>03</td>
<td>38 39 61</td>
<td>ASCII řetězec ’89a’ – verze GiFu</td>
</tr>
<tr>
<td>06</td>
<td>01 00</td>
<td>šířka logické obrazovky: jeden pixel</td>
</tr>
<tr>
<td>08</td>
<td>01 00</td>
<td>výška logické obrazovky: jeden pixel</td>
</tr>
<tr>
<td>10</td>
<td>80</td>
<td>bitové pole: povolení globální barvové palety</td>
</tr>
<tr>
<td>11</td>
<td>00</td>
<td>index barvy pozadí</td>
</tr>
<tr>
<td>12</td>
<td>00</td>
<td>poměr výšky a šířky pixelu: 1:1</td>
</tr>
<tr>
<td>13</td>
<td>ff 80 00</td>
<td>první barva v paletě: oranžový odstín</td>
</tr>
<tr>
<td>16</td>
<td>ff ff ff</td>
<td>druhá barva v paletě: čistě bílá</td>
</tr>
<tr>
<td>19</td>
<td>2c</td>
<td>značka začátku rámce</td>
</tr>
<tr>
<td>20</td>
<td>00 00</td>
<td>x-ová pozice levého okraje rámce: nultý sloupec</td>
</tr>
<tr>
<td>22</td>
<td>00 00</td>
<td>y-ová pozice horního okraje rámce: nultý řádek</td>
</tr>
<tr>
<td>24</td>
<td>01 00</td>
<td>šířka rámce: jeden pixel</td>
</tr>
<tr>
<td>26</td>
<td>01 00</td>
<td>výška rámce: jeden pixel</td>
</tr>
<tr>
<td>28</td>
<td>00</td>
<td>bitové pole: bez barvové palety, bez prokládání a animace</td>
</tr>
<tr>
<td>29</td>
<td>02</td>
<td>počáteční velikost LZW kódu v bitech–1</td>
</tr>
<tr>
<td>30</td>
<td>02</td>
<td>velikost bloku zakódovaného pomocí LZW</td>
</tr>
<tr>
<td>31</td>
<td>44 01</td>
<td>zakódovaná data rámce (jeden pixel s nulovým indexem)</td>
</tr>
<tr>
<td>33</td>
<td>00</td>
<td>ukončovací znak bloku zakódovaného pomocí LWZ</td>
</tr>
<tr>
<td>34</td>
<td>3b</td>
<td>ukončovací znak GIF souboru: znak ‘;’</td>
</tr>
</tbody>
</table>

Tabulka č. 4 - Význam bytů v jednotlivých blocích souboru typu GIF

Rozpoznání jednotlivých bloků a informací v nich uložených si můžeme popsat ještě podrobněji:

1. Každý soubor typu GIF musí začínat jeho signaturou, což je sekvence bytů s hexadecimálními hodnotami 0x47, 0x49 a 0x46. Po převodu do ASCII můžeme tuto sekvenci číst jako řetězec ’GIF’.

2. Ihned po signatuře GiFu musí následovat další trojice bytů, která udává verzi GiFu. V současné době (a pravděpodobně ji v budoucnosti) je možné použít pouze dvě verze, první je GIF87a (sekvence bytů 0x38 0x37 0x61) a druhá GIF89a (sekvence bytů 0x38 0x39 0x61).
3. Po signatuře je uložen *popis logické obrazovky*. Nejprve je na dvou bytech uvedena šířka obrazovky v pixelech, následuje výška obrazovky, samozřejmě taktéž v pixelech. Stejně jako u dalších vícebytových polí, i u hodnot šířky a výšky jsou byty uspořádány v pořadí nížší-vyšší (little endian). V našem obrázku, který má rozlišení nastaveno na 1×1 pixel, tedy bude šířka i výška nastavena na hexadecimální hodnotu **0x01 0x00**.

4. Posléze následuje bitové pole, ve kterém je uvedeno, zda je použita globální barová paleta (sedmý bit), počet bitů na pixel-1 (bity 3-6), zda je barová paleta setříděna (bit 2) a konečně délka barové palety (bity 0 a 1). Skutečná délka palety se vypočte podle vzorce: $2^{\text{hodnota}+1}$. V našem obrázku je použita globální barová paleta a její délka je rovna 2 (zapisujeme jako nulu, protože $2^{0+1}=2$), tj. hodnota zapsaná do bitového pole je rovna **0x80**.

5. Další byte obsahuje index barvy, která je použita jako pozadí v případě, že by plocha rámců nevyplnila celý obrázek. My tuto hodnotu prozatím nevyužíváme, proto zde může být nula – **0x00**.

6. Následuje byte, ve kterém je uveden poměr mezi výškou a šířkou pixelu. Vztah pro výpočet je následující: Aspect Ratio=(Pixel Aspect Ratio+15)/64. Pokud je v tomto byte uložena nula (**0x00**), není poměr specifikovaný. U verze GIF87a není tento byte využit a podle specifikace zde musí být zapsána hodnota nula. Tímto bytem také končí blok s informacemi o logické obrazovce a následuje *globální barová paleta*.

7. Jelikož je délka globální barové palety rovna nule, obsahuje paleta pouze dvě barvy, každou reprezentovanou trojicí bytů. První barva, tj. barva s indexem nula, má barové složky RGB nastaveny na hodnoty **0xff 0x80 0x00**, tj. jedná se v tomto případě o oranžový odsín. Druhá barva má barové složky nastaveny na maximální hodnoty (**0xff 0xff 0xff**), tj. jedná se o čistě bílou barvu.
8. Po globální barvové paletě ihned následuje *hlavička rámce*. Ta je v souboru rozpoznána značkou začátku rámce, tj. bytem s hodnotou 0x2c. Odpovídající ASCII znak je ' (čárka), což je pro oddělovač vhodná mnemotechnická pomůcka.

9. Dva byty za značkou udávají x-ovou pozici levého okraje rámce, další dva byty pak y-ovou pozici horního okraje rámce. Vzhledem k tomu, že rámec začíná v levém horním rohu obrázku, jsou zde uloženy nulové hodnoty (0x00 0x00 0x00 0x00).

10. Hlavička rámce pokračuje dvěma byty se šířkou rámce a dalšími dvěma byty s jeho výškou. Obě hodnoty nastavíme na jedničku, protože rámec bude mít rozlišení 1x1 pixel (0x01 0x00 0x01 0x00).

11. Následuje bitové pole, ve kterém je uvedeno, zda se v rámci bude používat lokální barvová paleta (sedmý bit), zda je rámec prokládán (šestý bit), zda jsou barvy v lokální barvové paletě setříděny (pátý bit) a konečně délka lokální barvové palety (bity 0-2). Vzhledem k tomu, že lokální barvovou paletu nepotřebujieme, je v tomto bitovém poli uložena hodnota 0x00. Pokud by byla lokální barvová paleta přítomna, začínala by ihned za tímto bytem.


13. Následuje byte udávající velikost bloku zakódovaného pomocí LZW. Kód pro náš jediný pixel se kupodivu musí rozepsat na dva byty, proto je i zde uložena hodnota 0x02.

15. Náš jediný pixel je uložen, proto se celý blok ukončí takzvaným ukončovacím znakem, který má hodnotu \texttt{0x00}.

16. V obrázku je použitý pouze jeden rámec, je tedy na čase celý soubor uzavřít. K tomu slouží byte, jehož ASCII hodnota je rovna znaku ':' a hexadecimální kód je tedy \texttt{0x3b}. Teoreticky by za tímto kódem mohla následovat další data, prohlížeč by je však měl ignorovat. \[5\]

\begin{tabular}{|c|c|}
\hline
Bitový vzorek & Význam \\
\hline
000 & první index do barvové palety \\
001 & druhý index do barvové palety \\
100 & clear code (CC) – inicializace hashovací tabulky \\
101 & end of LZW code (EC) \\
\hline
\end{tabular}

\textit{Tabulka č. 5 – Význam jednotlivých trojic bitů komprimovaných dat souboru typu GIF}
4 ANATOMIE GRAFICKÉHO FORMÁTU PNG

4.1 Obecné informace o struktuře souborů PNG

Mnozí programátoři považují PNG za jeden z nejlépe navržených binárních souborových formátů. Autoři při návrhu uvažovali o několika variantách interní struktury tohoto grafického formátu.

Jednou z uvažovaných variant byl čistě textový popis obrazových dat ve stylu formátů PBM, PGM a PPM (Portable BitMap, Portable GrayMap, Portable PixelMap), přičemž by celý soubor byl před svým uložením na disk zkomprimován programem gzip či bzip2. Nakonec se od tohoto návrhu, který byl v několika ohledech problematický (čitelnost, zabezpečení, přeskakování bloků dat apod.), upustilo a přešlo se na návrh čistě binárního formátu, což se z dlouhodobého hlediska ukázalo jako krok správným směrem.

Tvůrci PNG se také poučili z mnoha chyb a nedostatků do té doby používaných binárních formátů a navrhlí konzistentní interní strukturu PNG, která je na jednu stranu velmi sofistikovaná a na druhou stranu přitom poměrně jednoduchá na implementaci (aplikace mají přesné informace o tom, která data musí načístat a která data mají menší význam). Celý binární soubor s uloženým obrázkem se skládá z hlavičky, která je neměnná, tj. neobsahuje ani číslo verze. Za hlavičkou se nachází libovolné množství takzvaných chunků, což jsou pojmenované bloky, z nichž každý je opatřen svou délkou, typem a kontrolním součtem CRC (Cyclic Redundancy Check/Cyclic Redundancy Code). [9]

4.2 Hlavička souboru typu PNG

Hlavička PNG byla vytvořena s ohledem na heterogenní prostředí Internetu. Při přenosu souborů v tomto heterogenním prostředí může docházet k nežádoucím modifikacím dat, které je nutné v případě binárního formátu detekovat a vadné soubory (resp. jejich části) nenačítat. Nežádoucí modifikace mohou být několika typů:
1. Při přenosu obrázku přes internet se ztratí nejvyšší bit ve všech bajtech (bit se většinou vynuluje). Tato nežádoucí modifikace může nastat například při přenosu souboru pomocí protokolu FTP nebo při posílání souborů špatně nastaveným e-mailovým klientem (tento problém však mohou způsobit i některé e-mailové servery).

2. Dojde k náhradě ASCII znaku CR (Cursor Return) za znak LF (Line Feed). Může se jednat o chybu při přenosu binárního souboru mezi různými platformami, nebo při editaci binárního souboru ve špatně nastaveném textovém editoru.

3. Dojde k náhradě znaku LF za znak CR. Jedná se o obdobu předchozí chyby.


5. Dojde k náhradě dvojice znaků CR+LF za znak LF či za znak CR. Jedná se o obdobu předchozí chyby.


Hlavička grafického formátu PNG je ze těchto důvodů vytvořena tak, aby byly všechny výše uvedené nežádoucí modifikace při načítání souboru detekovány a modifikované soubory odstraněny z dalšího zpracování. Hlavička PNG má délku pouhých osmi bytů, které mají vždy konstantní hodnoty (ve výpisu uvedené hexadecimálně):

<table>
<thead>
<tr>
<th>Byte</th>
<th>Význam bytu</th>
</tr>
</thead>
<tbody>
<tr>
<td>89</td>
<td>jedná se o byte s nastaveným nejvyšším bitem (detekce sedmibitového přenosu)</td>
</tr>
<tr>
<td>50 4e 47</td>
<td>řetězec 'PNG', spolu s prvním bytem jednoznačně detekuje typ souboru na platformách, které typ rozpoznávají z obsahu a ne z koncovky</td>
</tr>
<tr>
<td>Od 0a</td>
<td>znaky CR+LF, detekce náhrady za jinou sekvenci: CR, LF či LF+CR</td>
</tr>
<tr>
<td>1a</td>
<td>znak Ctrl+Z, pro příkaz type MS-DOSu</td>
</tr>
<tr>
<td>0a</td>
<td>znak LF, detekce náhrady za CR+LF či LF+CR</td>
</tr>
</tbody>
</table>

Tabulka č. 6 – Význam bytů v hlavičce souboru PNG
Zde je vidět, že v pouhých osmi bytech se tvůrcům PNG podařilo vytvořit sekvenci bytů, na kterých je možné otestovat, zda přenos s velkou pravděpodobností proběhl v pořádku. Hned první byte byl zvolen tak, aby byl větší než 0x7f, tj. aby měl nastavený nejvyšší bit na jedničku. To má dva významy: detekuje se defektní sedmibitový přenos a některé textové editory soubor správně detekují jako binární a ne ASCII. Dále následují tři znaky tvořící řetězec 'PNG', což způsobuje, že hlavička souboru je částečně čitelná při zběžném pohledu i pro člověka. Následuje dvojice znaků CR+LF, na kterých je možné detekovat problémy při přenosu souboru mezi různými platformami. Následuje znak Ctrl+Z, který při zadání příkazu: type obrázek.png v prostředí MS-DOS/MS-Windows způsobí výpis následujícího textu: „ëPNG“

Obrázek č. 8 – Pokus o otevření obrázku PNG příkazem „type“ v prostředí MS-DOS / WINDOWS

¡t. zobrazi se pouze čtyři znaky, za kterými je řádek i výpis ukončen. Nemůže tedy nastat situace, kdy se po zadání příkazu PNG obrazovka zaplní různými "paznaky". Jediným vypsaným paznakem je ono é (čí jiný znak v závislosti na lokalizaci), které by mělo uživatele varovat, že se pokouší vypsat obsah binárního souboru. Poslední znak v hlavičce je LN, který opět slouží pro detekci, zda nedošlo k jeho náhradě jiným znakem či jinou sekvencí znaků, typicky CR, CR+LF či LF+CR. Ihned za hlavičkou totiž následuje sekvence tří nulových bytů, které by byly náhradou znaku CR za dva znaky posunuty, což opět způsobí detekovatelnou chybu při čtení. [9]

Zde stojí za povšimnutí, že nikde není uvedena verze PNG. To je zcela zbytečné, protože informace o tom, která data v obrázku jsou pro korektní zpracování (zobrazení) zapotřebí a která ne, je uvedena přímo u jednotlivých chunků.

Pokud je hlavička korektně načtena, může prohlížeč či jiná aplikace pracující s PNG pokračovat v načítání dalších dat. Před jejich popisem je však třeba se ještě seznámit s významem takzvaných chunků, což jsou pojmenované a pomocí kontrolního součtu zabezpečené bloky dat.
4.3 Popis chunků

4.3.1 Obecná struktura chunků

Veškeré informace i metainformace o zpracovávaném obrázku jsou uloženy v blocích dat nazvaných chunky (přibližný překlad je blok). Každý chunk se skládá ze čtyř částí:

1. První část má konstantní velikost čtyř byty a obsahuje celkovou délku datové části chunku. Teoreticky je tedy maximální délka datové části rovna $2^{32}$-1 bytům; avšak pro snazší implementaci, například v těch programovacích jazycích, které nemají implementovaný bezznaménkový datový typ (Java), je maximální hodnota délky rovna "pouze" $2^{31}$-1 bytů. V praxi jsou však délky chunků o několik řádů menší.


3. Třetí část chunku je tvořena vlastními daty. Tato část má v některých případech, například u chunku nazvaného IDAT, nulovou délku.

4. Poslední čtvrtá část chunku má délku čtyř byty a obsahuje kontrolní součet (CRC) druhé a třetí části, tj. jména (typu) chunku a uložených dat. Vzhledem k tomu, že délka chunku není do kontrolního součtu zahrnuta, je možné CRC generovat přímo při zápisu dat bez nutnosti druhého průchodu v případě, že délka chunku není dopředu známá (například při komprimaci). Použitý polynom má tvar:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

Všechny vícebytové položky v chuncích (včetně jejich délky) jsou uloženy v pořadí big endian, tj. byte s nejvyšší váhou je v souboru na prvním místě. Použití tohoto uspořádání bytů je pochopitelné, protože většina internetových formátů a protokolů pracuje se stejným uspořádáním, kterému se proto také někdy říká network byte order. Na platformách Apple Macintosh, Sun, Atari ST a Amiga se jedná o nativní formát, na platformě i86 je nutné provést převod pomocí knihovních maker. [9]
4.3.2 Způsob pojmenování chunků

Jak už bylo uvedeno výše, jméno (resp. typ) chunku je uloženo ve čtyřech bytech. Vždy se jedná o posloupnost čtyř ASCII znaků malé či velké abecedy, tj. z celkového množství 232 jmen je jich možné použít \((2\times26)^4=380 \, 204 \, 032\), což by mělo po nějaký čas postačovat.

Za povšimnutí stojí fakt, že u kódu ASCII je rozdíl velkého a malého znaku dán pouze změnou jednoho bitu, konkrétně bitu číslo 5. Jinými slovy, znak malé abecedy (minuska) má ASCII kód o 32 vyšší než odpovídající znak velké abecedy (verzála). Vzhledem k tomu, že jméno chunku tvořeno čtyřmi znaky, máme k dispozici celkem čtyři bity, kterými můžeme rozlišit různou důležitost chunků. Význam pátých bitů každého ze znaků názvu je následující:

1. **Pátý bit prvního znaku názvu:** tento bit určuje, zda je daný chunk pro zpracování obrázku nezbytný. Příkladem chunku, který je nutné vždy zpracovat, je **IHDR**, mezi nepovinný chunk patří **tEXt**.

2. **Pátý bit druhého znaku názvu:** tento bit určuje, zda je typ chunku veřejný (určený specifikací a známý více aplikacím) nebo privátní, tj. dávající smysl pouze jedné či několika málo aplikacím. Díky tomuto bitu nikdy nenastanou kolize mezi oficiálními a aplikačními chunky.

3. **Pátý bit třetího znaku názvu:** je ve verzi PNG 1.0, PNG 1.1 i PNG 1.2 vždy vynulován, tj. třetí znak názvu chunku je vždy zapsán velkým písmenem.

4. **Pátý bit čtvrtého znaku názvu:** tento bit je určen zejména pro grafické editory a konvertory obrázků. Jeho hodnota určuje, zda se při změně obrázku (editaci) má daný chunk uložit zpět do souboru, či zda se se změnou obrázku chunk stane nepoužitelným. Příkladem může být chunk obsahující histogram (**hIST**) – při změně obrázku dojde i ke změně histogramu, a proto není korektní provést kopii chunku ze zdrojového obrázku do obrázku cílového (něco jiného ovšem je vytvoření chunku stejného typu, ale s jiným obsahem). [9]
4.3.3 Nejvýznamnější povinné chunky

Prakticky každý obrázek typu PNG obsahuje tři povinné chunky v následujícím pořadí:

*Tabulka č. 7 – Pořadí povinných chunků v souboru typu PNG*

<table>
<thead>
<tr>
<th>Typ chunku</th>
<th>Název chunku</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHDR</td>
<td>hlavička obrázku</td>
</tr>
<tr>
<td>IDAT</td>
<td>datová část – pixmapa</td>
</tr>
<tr>
<td>IEND</td>
<td>ukončující značka</td>
</tr>
</tbody>
</table>

U obrázků obsahujících barvovou paletu přibývá ještě jeden typ chunku (paletu) a formát je následující:

*Tabulka č. 8 – Pořadí povinných chunků u obrázku obsahující barvovou paletu v souboru typu PNG*

<table>
<thead>
<tr>
<th>Typ chunku</th>
<th>Název chunku</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHDR</td>
<td>hlavička obrázku</td>
</tr>
<tr>
<td>PLTE</td>
<td>barvová paleta</td>
</tr>
<tr>
<td>IDAT</td>
<td>datová část – pixmapa</td>
</tr>
<tr>
<td>IEND</td>
<td>ukončující značka</td>
</tr>
</tbody>
</table>

Dále je potřeba si vysvětlit formát dvou důležitých chunků, kterými jsou **IHDR** a **IEND**.

4.3.3.1 Chunk typu IHDR

V chunku typu **IHDR** je uložena hlavička obrázku. Ta je odlišná od výše uvedené hlavičky celého souboru, protože obsahuje základní metainformace o uloženém obrázku, zejména jeho rozlišení, bitovou hloubku, typ kódování barev a použitou filtraci. Vzhledem k tomu, že datová část hlavičky obrázku má vždy délku 13 bytů, začíná chunk posloupností **0x00 0x00 0x0d**. Hlavička obrázku musí být uvedena ihned za hlavičkou PNG, při čtení je tedy nutné otestovat, zda se ihned po přečtení hlavičky PNG objeví tato posloupnost. Pokud tomu tak není, mohlo dojít k poškození souboru. [9]
Datová část chunku **IHDR** obsahuje následující položky:

### Tabulka č. 9 – Význam offsetů IHDR chunku v souboru typu PNG

<table>
<thead>
<tr>
<th>Offset</th>
<th>Počet byte</th>
<th>Význam</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>4</td>
<td>šířka obrázku uvedená v pixelech</td>
</tr>
<tr>
<td>04</td>
<td>4</td>
<td>výška obrázku uvedená v pixelech</td>
</tr>
<tr>
<td>08</td>
<td>1</td>
<td>bitová hloubka pixelů v barvovém kanálu (povolené hodnoty jsou 1, 2, 4, 8 a 16)</td>
</tr>
<tr>
<td>09</td>
<td>1</td>
<td>typ kódování barev (viz další tabulka)</td>
</tr>
<tr>
<td>0a</td>
<td>1</td>
<td>použitá metoda komprimace (musí zde být 0-deflate)</td>
</tr>
<tr>
<td>0b</td>
<td>1</td>
<td>použitá metoda filtrace (musí zde být 0-adaptivní filtrace)</td>
</tr>
<tr>
<td>0c</td>
<td>1</td>
<td>prokládání obrázku (0-bez prokládání, 1-prokládání)</td>
</tr>
</tbody>
</table>

Metody komprimace i filtrace musí být v současnosti nastaveny na nulu. Pokud je obrázek prokládaný, je příslušný byte nastaven na 0x01, v opačném případě na 0x00. Dále je třeba vysvětlit vztah mezi bitovou hloubkou pixelů v barvovém kanálu a typem kódování barev.

V současnosti je možné využít následující kombinace:

### Tabulka č. 10 – Vztah mezi bitovou hloubkou pixelů a typem kódování barev u souboru typu PNG

<table>
<thead>
<tr>
<th>Typ kódování barev</th>
<th>Bitová hloubka</th>
<th>Popis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1,2,4,8,16</td>
<td>Obrázek ve stupních šedi (popř. černobílá pérovka)</td>
</tr>
<tr>
<td>2</td>
<td>8,16</td>
<td>Plnobarevný (true color) obrázek typu RGB</td>
</tr>
<tr>
<td>3</td>
<td>1,2,4,8</td>
<td>Obrázek s barvovou paletou (v souboru musí existovat chunk <strong>PLTE</strong>)</td>
</tr>
<tr>
<td>4</td>
<td>8,16</td>
<td>Obrázek ve stupních šedi a průhlednosti (alfa kanálem) – nepříliš používaná kombinace</td>
</tr>
<tr>
<td>6</td>
<td>8,16</td>
<td>Obrázek, kde každý pixel obsahuje všechny čtyři hodnoty RGBA (tj. kromě tří barvových složek i alfa kanál)</td>
</tr>
</tbody>
</table>
4.3.3.2 Chunk typu IEND

Chunk **IEND** by se měl nacházet na konci každého PNG souboru, aby byl načítací program informován, že může ukončit načítání a že se soubor přenesl celý. Tento chunk neobsahuje žádnou datovou část, proto je jeho bytová sekvence o délce dvanácti bytů vždy stejná (0x00 0x00 0x00 0x00 0x49 0x45 0x4e 0x44 0xae 0x42 0x60 0x82). [9]

*Tabulka č. 11 – Význam jednotlivých sekvencí bytů v chunku IEND u souboru typu PNG*

<table>
<thead>
<tr>
<th>Sekvéncie bytů</th>
<th>Význam</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 00 00 00</td>
<td>Délka datové části chunku: 0</td>
</tr>
<tr>
<td>49 45 4e 44</td>
<td>Sekvénci ASCII znaků 'IEND'</td>
</tr>
<tr>
<td>ae 42 60 82</td>
<td>CRC tohoto chunku</td>
</tr>
</tbody>
</table>

4.3.4 Nejvýznamnější nepovinné chunky

Zde je soupis nejvýznamnějších nepovinných chunků, které se mohou v grafických souborech PNG vyskytnout.

*Tabulka č. 12 – Nejvýznamnější nepovinné chunky v grafických souborech PNG*

<table>
<thead>
<tr>
<th>Typ chunku</th>
<th>Název chunku</th>
</tr>
</thead>
<tbody>
<tr>
<td>gAMA</td>
<td>Gamma Value</td>
</tr>
<tr>
<td>bKGD</td>
<td>Background Color</td>
</tr>
<tr>
<td>time</td>
<td>Timestamp</td>
</tr>
<tr>
<td>tEXT</td>
<td>zTXt Latin-1 Text Annotations</td>
</tr>
<tr>
<td>iTXt</td>
<td>International Text Annotations</td>
</tr>
<tr>
<td>hIST</td>
<td>Histogram</td>
</tr>
<tr>
<td>sPLT</td>
<td>Suggested Palette</td>
</tr>
<tr>
<td>sBIT</td>
<td>Significant Bits</td>
</tr>
<tr>
<td>pHYs</td>
<td>Physical Pixel Dimensions</td>
</tr>
<tr>
<td>sCAL</td>
<td>Physical Scale</td>
</tr>
<tr>
<td>oFFs</td>
<td>Image Offset</td>
</tr>
<tr>
<td>pCAL</td>
<td>Pixel Calibration</td>
</tr>
<tr>
<td>fRAC</td>
<td>Fractal Parameters</td>
</tr>
<tr>
<td>glFg</td>
<td>glFx GIF Conversion Info</td>
</tr>
<tr>
<td>glFt</td>
<td>GIF Plain Text</td>
</tr>
</tbody>
</table>
4.4 Struktura souboru PNG s obrázkem uloženým ve stupních šedi

Obrázky uložené ve stupních šedi mají v hlavičce obrázku, tj. v metainformacích popisujících obrázek, nastaven typ kódování barev na 0. Počet bitů na pixel může nabývat hodnot 1, 2, 4, 8 nebo 16; zde se budeme zabývat strukturou obrázku s pixely uloženými v bitové hloubce 8 bitů na pixel (bpp – bits per pixel), ve kterých je možné vzájemně rozlišit maximálně \(2^8 = 256\) odstínů šedi. Jedná se o prakticky nejběžnější formát tzv. grayscale obrázků, jiný počet bitů na pixel (např. 16) se uplatňuje ve speciálních případech, například v lékařství. Konkrétní binární podoba obrázku o rozlišení 1×1 pixel uloženého ve stupních šedi je popsána v následující podkapitole.

4.4.1 Binární podoba obrázku uloženého ve stupních šedi

Nejmenší obrázek uložený ve stupních šedi při 8 bpp (bitech na pixel) má rozlišení 1×1 pixel a jeho celková délka při uložení do formátu PNG je 67 bytů. To je skoro dvakrát více, než u formátu GIF, kde měl nejmenší obrázek s tímto rozlišením velikost pouhých 35 bytů. Rozdílná délka je způsobena zejména poměrně velkými hlavičkami chunků (12 bytů) u PNG oproti formátu GIF, kde mají některé bloky dat délku menší, někdy dokonce pouhý jeden byte (například ukončující blok). Při práci s obrázky většími než cca 30×30 pixelů se však velikostí souborů většinou obrácí, protože soubory PNG bývají díky filtrům a lepšímu komprimačnímu algoritmu menší. Vraťme se však k našemu 1-pixelovému obrázku uloženému ve formátu PNG. Při prohlížení binárních informací takového to obrázku pomocí hexadecimálního editoru (KHexEdit, bvi, vim s xxd, Hiew, biew, Dos Navigátor nebo např. český PSpad obsahující hexadecimální editor) uvidíme následující sekvenci bytů. Všechny další údaje, tj. jak offsetsy od začátku souboru, tak i jednotlivé byty, jsou zapsány hexadecimálně. [13]
Zde je třeba si význam jednotlivých bytů v PNG obrázku ozřejmit:

*Tabulka č. 13 – Význam jednotlivých sekvencí bytů u obrázku typu PNG uloženém ve stupních šedi*

<table>
<thead>
<tr>
<th>Sekvence bytů</th>
<th>Význam sekvence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hlavička souboru typu PNG</td>
<td>89 úvodní byte souboru typu PNG pro detekci sedmibitového přenosu</td>
</tr>
<tr>
<td>řetězec 'PNG' pro detekci typu souboru v některých operačních systémech</td>
<td>50 4e 47 znaky CR+LF, detekce náhrady za jinou sekvenci</td>
</tr>
<tr>
<td>znak Ctrl+Z pro ukončení výpisu obsahu souboru v MS-DOSu</td>
<td>0d 0a znaky CR+LF, detekce nechtěného ASCII přenosu</td>
</tr>
<tr>
<td>Hlavička obrázku (metainformace)</td>
<td>1a znak LF, detekce nechtěného ASCII přenosu</td>
</tr>
<tr>
<td>délka dat hlavičky je 13 bytů</td>
<td>00 00 00 0d</td>
</tr>
<tr>
<td>řetězec 'IHDR' – typ chunku</td>
<td>49 48 44 52</td>
</tr>
<tr>
<td>šířka obrázku je jeden pixel</td>
<td>00 00 00 01</td>
</tr>
<tr>
<td>výška obrázku je také jeden pixel</td>
<td>00 00 00 01</td>
</tr>
<tr>
<td>bitová hloubka je 8bpp (odpovídá 256 odstínů šedi)</td>
<td>08</td>
</tr>
<tr>
<td>typ obrázku 0 – stupně šedi</td>
<td>00</td>
</tr>
<tr>
<td>použitá metoda komprimace (0=deflate)</td>
<td>00</td>
</tr>
<tr>
<td>použitá metoda filtrace (0=adaptivní filtrace)</td>
<td>00</td>
</tr>
<tr>
<td>prokládání obrázku (0=bez prokládání)</td>
<td>00</td>
</tr>
<tr>
<td>CRC hlavičky obrázku</td>
<td>3a 7e 9b 55</td>
</tr>
<tr>
<td>Data obrázku</td>
<td>00 00 00 0a</td>
</tr>
<tr>
<td>délka chunku s daty je 10 bytů</td>
<td>49 44 41 54</td>
</tr>
<tr>
<td>řetězec 'IDAT' – typ chunku</td>
<td>78 da 63 60</td>
</tr>
<tr>
<td>10 bytů reprezentujících jeden pixel obrázku ve formátu grayscale</td>
<td>00 00 00 02</td>
</tr>
<tr>
<td>CRC dat obrázku</td>
<td>00 01</td>
</tr>
<tr>
<td>00 00 00 00</td>
<td>e5 27 de fc</td>
</tr>
<tr>
<td>CRC ukončujícího chunku</td>
<td>Ukončující chunk</td>
</tr>
<tr>
<td>délka datové části ukončujícího chunku je 0 bytů</td>
<td>00 00 00 00</td>
</tr>
<tr>
<td>řetězec 'IEND' – typ chunku</td>
<td>49 45 4e 44</td>
</tr>
<tr>
<td>CRC ukončujícího chunku</td>
<td>ae 42 60 82</td>
</tr>
</tbody>
</table>
Z tabulky je vidět, že celý soubor je rozdělený do čtyř bloků dat. V prvním bloku o délce osmi bytů je uložena hlavička souboru typu PNG, která má stále stejný obsah. V dalších 25 bytech je uložena hlavička obrázku obsahující nejdůležitější metainformace, které jsou potřebné pro každý program manipulující s obrázky (jedná se o chunk IHDR). Dalších 22 bytů je vyhrazeno pro datovou část, tj. vlastní informaci o uložených pixelech zabalenou v chunku IDAT. Posledních 12 bytů slouží pro uložení ukončujícího chunku IEND, který nemá datovou část, proto je jeho podoba stále stejná.

Ve skutečnosti, zejména u delších obrázků, může být rastrový obrázek rozdělen do více chunků IDAT. Efekt rozdělení je dvojí: obrázek je možné zobrazit již při přenosu (podle specifikace by se měla zobrazit pouze ta část obrázku, která má korektní CRC) a i poškozený obrázek je možné částečně zrekonstruovat. Vzhledem k tomu, že každý chunk IDAT zvyšuje velikost souboru o minimálně 12 bytů, je vhodné používat velikost bloku cca 50–100 kB. [13]
II. PRAKTICKÁ ČÁST
5 VÝVOJ PROGRAMU


5.1 Výběr knihoven

Kromě výběru vývojového prostředí, ve kterém budeme program psát, je před samotným vývojem aplikace velice důležitou otázkou také použití knihoven. Knihovny jsem se snažil vybírat tak, aby měly podporu grafických formátů, měly dostatečné množství operací, byly pokud možno použitelné pro co největší počet platform a hlavně nebyly příliš komplikované na pochopení a použití při programování.


Pro vývoj prostředí aplikace jsem použil knihovnu Microsoft Foundation Class (dále jen MFC). Tato knihovna je objektově orientovanou implementací Win32 API. Poskytuje komplexní prostředí pro vývoj aplikací typu klient-server a dokument/pohled kterou lze velmi jednoduše rozšířit pomocí agregace od tříd rozhraní MFC. Dnes již MFC nemá příliš velkou popularitu, jelikož jí nahradily jednodušší knihovny .NET, ale stále má své zastánce díky rychle vykonávajícímu se kódu.

5.1.1 FreeImage Charakteristika

FreeImage je OpenSource knihovna pro vývojáře s podporou známých grafických formátů jakými jsou GIF, PNG, BMP, JPEG, TIFF a další, které dnes naleznou využití v mnoha programech. Knihovna FreeImage je snadno použitelná, rychlá, kompatibilní se všemi 32-bitovými verzemi Windows, Linux a MacOS s podporou multithreadingu.

Knihovna je dostupná ve dvou verzích: DLL distribuce, která může být zkompilována na jakémkoliv WIN32 C/C++ kompilátoru a ve zdrojové distribuci. Knihovna má velké množství podporovaných formátů: BMP (čtení, zápis), Dr. Halo (čtení), DDS (čtení), EXR (čtení, zápis), Raw Fax G3 (čtení), GIF (čtení, zápis), HDR (čtení, zápis), ICO (čtení, zápis), IFF (čtení), JBIG (čtení, zápis), JNG (čtení), JPEG/JIF (čtení, zápis), JPEG-2000 (čtení, zápis), KOALA (čtení), MNG (čtení), PCX (čtení), PBM (čtení, zápis), PGM (čtení, zápis), PNG (čtení, zápis), PPM (čtení, zápis), Sun RAS (čtení), SGI (čtení), TARGA (čtení, zápis), TIFF (čtení, zápis), WBMP (čtení, zápis), XBM (čtení), XPM (čtení, zápis).

### 5.1.2 Základní vlastnosti knihovny FreeImage

Snadnost použití - Knihovna byla navržena, aby byla jednoduchá při používání.

Neomezený přístup k souborům - Struktura FreeImageIO modulu umožňuje otevřít obrázky prakticky odkudkoliv. Je možné otevřít jak samostatné soubory, tak soubory uložené v paměti, zkomprimované soubory nebo soubory z internetu bez nutnosti opětovné kompilace.

Modulární - Vnitřní struktura je modulární. Snadno lze vytvořit nový plug-in, který je možný uložit buď v DLL knihovně, nebo vložit přímo do aplikace.

Barevná konverze - FreeImage poskytuje mnoho funkcí pro převody obrázků z jedné bitové hloubky do jiné. Knihovna podporuje konverze mezi 1, 4, 8, 16, 24 a 32 bitovými obrázky.

Jednoduché transformační funkce - Poskytuje jednoduché rozhraní pro transformace s obrázky jako rotaci, změnu velikosti nebo změnu barvené hloubky obrázku.
5.2 Třídy v aplikaci

V aplikaci máme dvě základní nosné třídy. Každá z tříd zaštítuje funkčnost programu. CImageViewWnd je třída, která se stará o správnou funkčnost okna a třída CImageViewApp o správnou funkci aplikace samotné. Jelikož nebylo využito tzv. technologie dokument/pohled, zastává třída (CImageViewWnd) jak roli pohledu, tak dokumentu, takže obsahuje data specifická pro danou aplikaci (což je handle na obrázek).

Pro zobrazení dialogů, jsou v aplikaci vytvořeny tři třídy, odvozené od třídy CDialog z třídy MFC. Třída CDialog poskytuje prostředí pro správné vytvoření, zobrazení a výměnu dat dialogu. Předáním identifikátoru zdroje konstruktoru třídy CDialog vytvoříme vazbu mezi zdrojem a dialogem a zavoláním metody DoModal třídy CDialog zobrazíme dialog. Odvozené třídy jsou zde pouze pro výměnu dat mezi dialogem a třídou. Třída CDialog má virtuální metodu DoDataExchange, která slouží k výměně dat mezi třídou a dialogem. Přepsáním této metody můžeme vyměňovat vyplněná data v komponentách dialogu. Pro výměnu dat v těle metody se používá metoda DDX_Text, která si bere jako parametr ukazatel na třídu CDataExchange, identifikátor dané komponenty v dialogu a referenci na proměnnou, která bude uchovávat danou hodnotu.

V aplikaci jsou čtyři dialogy.
Třída CInformaceDialog slouží pro zobrazení dialogu s informacemi o právě otevřeném obrázku.

**Obrázek č. 11 – Dialogové okno - Vlastnosti**

Konstruktor CInformaceDialog si bere jako parametr ukazatel na právě zobrazený dialog, ze kterého následně v těle virtuální metody DoDataExchange získává data o daném obrázku. K získání šířky, resp. výšky obrázku slouží funkce FreeImage_GetWidth resp. FreeImage_GetHeight. K získání barevné hloubky použije funkci FreeImage_GetColorsUsed, která vrací počet barev v obrázku a v případě true color vrací nulu, což je ošetřeno druhou mocninou počtu bitů na pixel, takže barevná hloubka je dána \(2^{24} = 16777216\) barev. K umocňování je použita funkce pow, definovaná v hlavičkovém souboru math.h

Třída COAplikaci slouží pouze pro zobrazení dialogu s informacemi o aplikaci.

**Obrázek č. 12 – Dialogové okno - O aplikaci**

Poslední kompilace: Apr 19 2008
Verze kompilátoru 1200
Verze MFC: 1536
Virtuální metoda `DoDataExchange` přepisuje svého předchůdce ve třídě `Dialog` a nastavuje tři komponenty v dialogu. První komponenta Static Text je den poslední kompilace programu. Datum poslední kompilace je dáno makrem `__DATE__`, které vrací řetězec s datem, kdy byla aplikace naposledy kompilována. Pro zjištění verze kompilátoru slouží makro `_MSC_VER` a ke zjištění verze MFC makro `_MFC_VER`.

Třída `CRotaceDialog` slouží k získání úhlu, o kolik chceme otočit právě otevřený obrázek.

![Obrázek č. 13 – Dialogové okno Rotace](image)

Tato třída má definovanou metodu `GetUhel`, která vrací úhel zadaný uživatelem v dialogu. Dále má třída `CRotaceDialog` přepsanou metodu předka `DoDataExchange`, která provádí výměnu dat s hodnotami zadaných úhlů.

Třída `CVelikostDialog` slouží ke zobrazení dialogu, kde má uživatel možnost zadat novou výšku a šířku obrázku v pixelech.

![Obrázek č. 14 – Dialogové okno Změna velikosti](image)
První dva atributy slouží k nastavení nové velikosti obrázku, ve třetím je možnost výběru z několika typů grafických filtrů.

Třída CVelikostDialog má definované tři metody, které vrací hodnoty atributů m_iVelikostX, m_iVelikostY a m_eFiltr. Atributy m_iVelikostX, m_iVelikostY a m_eFiltr obsahují dimenze nového obrázku a filtr pro případné rozostření/zaostření obrázku a nebo zmenšení, resp. zvětšení obrázku. Tyto atributy tedy obsahují dimenze:

- FILTER_BOX
- FILTER_BILINEAR
- FILTER_BSPLINE
- FILTER_BICUBIC
- FILTER_CATMULLROM
- FILTER_LANCZOS3

5.3 MFC

Pro správný běh aplikace, která zobrazuje okna je potřeba vytvořit dvě základní třídy odvozené od tříd MFC.

Primárně musí být v aplikaci třída odvozená od třídy MFC nazvaná CWinApp, která zajišťuje spuštění aplikace a případné vytvoření tříd okna/okén. Odvozená třída CImageViewApp přepisuje ve svém předkově virtuální metodu InitInstace, která se stará o vytvoření okna. Tato metoda je volána v okamžiku vytvoření aplikace.

Další třída je třída okna, která může být odvozena buď od třídy CWnd, která zajišťuje rozhraní pro vytvoření jednoduchého okna bez rámců (tzn. klientská oblast má stejnou barvu, jako má celý zbytek okna, klientská oblast je celé okno) nebo CFrameWnd, která poskytuje rozhraní pro zobrazení jednoduchého okna s rámcem (má klientskou oblast s odlišnou barvou, klientská oblast je oblast bez rámce okna). V aplikaci je třída okna CImageViewWin odvozena od třídy CFrameWnd. Stejně jako u třídy aplikace je potřeba vytvořit okno, tak u třídy okna je potřeba předat parametry pro vytvoření okna a vytvořit okno samotné. V konstruktoru třídy CImageViewApp voláme metodu třídy CFrameWnd nazvanou Create, která se postará o vytvoření okna. Pro zobrazení okna je třeba už jen zavolat v metodě třídy CImageviewApp::InitInstance po vytvoření třídy (konstruktor zavolá metodu Create, takže okno už máme vytvořené) okna zavoláme metody ShowWindow(SW_SHOW) a UpdateWindow, které nám zajistí, že se nám okno zobrazí.
5.4 Spuštění programu

Spuštění programu je poněkud atypické oproti Win32 aplikaci. V každé aplikaci tvořené pomocí MFC není funkce main, ani WinMain. V MFC je třída WinMain implementována, tak, že volá jakýkoliv globální objekt třídy, nebo odvozené třídy od třídy CWinApp. Takže pro spuštění aplikace je třeba vytvořit globální (ve skutečnosti se může jednat o jakýkoliv statický objekt, který běží po celou dobu běhu aplikace) instanci třídy odvozené, nebo samotné třídy CWinApp. Třída CWinApp předá prostředí MFC ukazatel na běžící instanci funkci WinMain a WinMain zavolá metodu InitInstance.

5.5 Implementace knihovny FreeImage v programu


5.5.1 Otevření obrázku

Ve třídě okna CImageViewWin je metoda OnSouborOtevrit, která je volána metodou třídy aplikace CImageViewApp::OnSouborOtevrit a obsluhuje zprávy na otevření souboru. Metoda CImageViewApp::OnSouborOtevrit, kromě agregace své role na metodu CImageVirewWin::OnSouborOtevrit provádí zobrazování dialogu s požadavkem na otevření souboru.

Obrázek č. 15 – Dialogové okno pro otevírání obrázků – Otevřít
Samotné tělo metody zavolá sofistikovanější metodu ImageOpener, která se stará o otevření souboru pomocí knihovny FreeImage a vydá požadavek na překreslení clientské oblasti okna, tzn. aktualizaci a zobrazení nově otevřeného obrázku.

Metoda ImageOpener slouží ke zjištění o jaký typ souboru se jedná, otevření souboru a inicializaci handle na otevřený soubor do handle m_image.

S otevíráním souborů souvisí metoda třídy pohledu IsImageOpened, která vrací dvouhodnotový typ boolean, v závislosti na tom, zda je nějaký soubor otevřen. Jestliže je nějaký soubor otevřen, tak handle je nenulový a metoda vrací true, jinak vrací false.


**Obrázek č. 16 – Průchod funkcemi při otevírání obrázku**
5.5.2 Uložení obrázku

Při požadavku na uložení souboru je odeslána aplikaci zpráva ID_SOUBOR_ULOZIT, která je obsloužena dle definice v mapě zpráv metodou třídy aplikace OnSouborUlozit. Metoda OnSouborUlozit zobrazí odpovídající dialog s požadavkem kam soubor uložit a jak soubor pojmenovat. Když uživatel v dialogu klikne na OK zavolá metoda OnSouborUlozit ve třídě CImageViewApp metodu třídy CImageViewWin nazvanou OnSouborUlozit a předá mu parametrem řetězec s názvem a cestou souboru, který chce uživatel uložit. Metoda CImageViewWin::OnSouborUlozit už se sama stará o uložení souboru odpovídajícími metodami na disk.


5.5.3 Obnova změn na obrázku

Ve třídě okna, která zároveň zastává třídu dokumentu, je atribut m_Previous zvaný zásobník operací. Objekt m_Previous je instance šablounové třídy STL zásobníku. Jako parametr šablony si bere ukazatel na handle obrázku.

```cpp
std::stack<FIBITMAP *> m_Previous;
```

Při použití jakékoliv metody pro provádění modifikací na obrázku (např. rotace) je uložen před provedením změny duplikát obrázku na zásobník operací. V případě potřeby resp. když uživatel potřebuje, zavolá se metoda OnUpravyZpet, která nahraje duplikát z vrcholu zásobníku operací (tzn. předchozí operaci) do atributu třídy m_image a překreslí (aktualizuje) klientskou oblast okna.

![Obrázek č. 18 – Ukázka uchovávání tří duplikátů na zásobníku operací. Jako první proběhla operace změna velikost, poté inverze a naposledy rotace](image1)

<table>
<thead>
<tr>
<th>Operace</th>
<th>Poslední operace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotace</td>
<td>2.operace</td>
</tr>
<tr>
<td>Inverze</td>
<td>1.operace</td>
</tr>
<tr>
<td>Změna velikosti</td>
<td></td>
</tr>
</tbody>
</table>

Pokud uživatel potřebuje vrátit změny zpět, tedy použije tlačítko z nabídky, odešle tímto zprávu aplikaci ID_UPRAVY_ZPET. V mapě zpráv je definována jako obslužná metoda zprávy OnUpravyZpet ve třídě aplikace. Tato metoda volá pouze kompetentní metodu třídy okna OnUpravyZpet. Metoda třídy okna OnUpravyZpet nahraje vrchol zásobníku do atributu m_image, odstraní poslední operaci z vrcholu a překreslí klientskou oblast okna.

![Obrázek č. 19 – Průchod funkcemi při zpětném vrácení obrázku](image2)

```cpp
ID_UPRAVY_ZPET

CImageApp::OnUpravyZpet

CImageWin::OnUpravyZpet

m_image = m_Previous.top();
m_image = m_Previous.pop();
```
5.5.4 Zobrazení obrázku

Pro zobrazování obrázků nemá knihovna FreeImage přímou podporu pro operační systém Windows (až na výjimku, její objektově orientovaná nadstavba má implementaci, která vrací handle na otevřený obrázek, ale v programu není tento modul zahrnut). Pro zobrazení obrázku z handlu obrázku knihovny FreeImage do okna se používá speciální funkce SetDIBitsToDevice, která zobrazí každý bit obrázku na kontext zařízení. Funkci SetDIBitsToDevice stačí předat šířku, výšku v pixelech a pole s jednotlivými byty obrázku, což vše dokáže pokrýt knihovna. Pro získání šířky slouží funkce FreeImage_GetWidth, pro výšku FreeImage_GetHeight a pro pole bitů FreeImage_GetBits.

Metoda, která se stará v programu o vykreslování má implementaci ve třídě CImageViewWin a jmenuje se OnPaint. Metoda OnPaint se stará o překreslování okna pokračdě, když je okno programu překryto jiným (včetně vlastních dialogů), změněna velikost (včetně minimalizace a maximalizace okna), nebo otevřen soubor. Při každém volání metody OnPaint se zavolá funkce SetDIBitsToDevice s odpovídajícími parametry právě otevřeného obrázku a vykreslí ho na kontext zařízení.

5.5.5 Obsluha zpráv v programu

Když uživatel klikne na nějakou z nabídek programu (např. Rotace v nástrojové liště), tak je vygenerována zpráva WM_COMMAND, která dále obsahuje zprávu o tom, jaký ovládací prvek byl použit. Právě informace o tom, jaký ovládací prvek byl použit, je podstatná pro obsluhu zpráv. V souboru definic (tzn. soubor CImageViewApp.cpp) je tzv. mapa zpráv což je mechanismus, který nahrazuje proceduru okna ve Win32 API, a navíc snižuje paměťovou náročnost na vytváření virtuálních tabulek [14]. Na začátku souboru před definice třídy CImageViewApp jsou vložena makra definovaná knihovnou MFC nazvané BEGIN_MESSAGE_MAP a END_MESSAGE_MAP, mezi něž vložením makra ON_COMMAND s parametry názvu odeslané zprávy se zprávou WM_COMMAND (např. při kliknutím na Rotace se odešle zpráva WM_COMMAND se zprávou ID_OBRAZEK_ROTACE) a obslužnou metodou třídy nechám obsloužit zprávy programu určitou metodou.
Např. obsluhu kliknutí na tlačítko Rotace z panelu nástrojů, nebo nabídky lze zapsat následujícím způsobem:

BEGIN_MESSAGE_MAP(CImageViewApp, CWinApp)
    // ...
    ON_COMMAND(ID_OBRAZEK_ROTACE, OnObrazekRotace)
    // ...
END_MESSAGE_MAP()

- kde program bude reagovat na odeslání zprávy WM_COMMAND společně se zprávou ID_OBRAZEK_ROTACE a tato zpráva bude obsloužena metodou OnObrazekRotace třídy aplikace.
5.6 Implementace v jiných programech

Tento program nelze použít nikde jinde, než v MFC, protože třída okna (CImageViewWin) zároveň obsahuje dokument, což je objekt představující vlastnosti obrázku a obrázek samotný. Pro ještě větší universálnost by se tato data musela umístit zvlášť do speciální třídy, které se právě odborně říká dokument a ta by se potom dala vkládat do jiných programů. Ovšem při tvorbě programu bylo zásadní, aby byl kód co nejkratší, proto byly vytvořeny pouze dvě třídy jedna pro program a jedna pro okno, která zároveň obstarává roli dokumentu. Třída okna má k dispozici všechny vlastnosti obrázku, včetně handle, které by jinak mohl obsahovat dokument. Kód je proto čitelnější, kratší a jednodušší.

5.7 Generovaná dokumentace k programu

Dokumentační CD obsahuje také generovanou dokumentaci k programu. Ke generování dokumentace byl použit open-source program Doxygen [11], který vytváří ze zdrojových souborů Javy, C, C++, PHP, Delphi nebo Python dokumentace na základě syntakticky správně okomentovaných částí kódu. Doxygen na základě komentářů vstupních zdrojových souborů umí vygenerovat html, chm nebo pdf dokumentaci k funkcím, třídám, metodám a atributům. Komentáře mohou být dvojího druhu a to buď stručný, který ve stručnosti charakterizuje funkcii/metodu a podrobný, který podrobně rozebírá, co daná funkce dělá, co vrací a jaké parametry si bere. Komentáře se dávají buď před deklarace, nebo definice jednotlivých metod, atributů, nebo tříd, s tím, že stručný komentář se označí:

```
//!
```

A podrobný komentář začíná
```
/*
A končí znaky
*/
```

Pokud chceme do dokumentace vložit vlastnosti návratové hodnoty, vložíme do kódu podrobného komentáře řetězec:
```
\return charakteristika návratové hodnoty
```

A chceme-li okomentovat parametry funkce, vložíme do kódu podrobného komentáře řetězec:
```
\param jménoParametru charakteristikaParametru
```
6 UŽIVATELSKÁ PŘÍRUČKA K PROGRAMU

1. Otevření obrázku typu *.gif nebo *.png z disku Ctrl + O
2. Uložení právě zobrazeného obrázku na disk Ctrl + U
3. Informace o obrázku Ctrl + I
4. Otočení obrázku o zadaný počet stupňů Ctrl + R
5. Změna velikosti obrázku Ctrl + V
6. Inverze barev obrázku Ctrl + N
7. Tlačítko „zpět“ Ctrl + BACKSPACE

- zavření aplikace -> soubor -> zavřít nebo kl. zkratka - Ctrl + Z
- dialog O aplikaci -> nápověda nebo kl. zkratka - Ctrl + H
7 PREZENTACE PRO VÝUKU

Prezentaci jsem vytvořil podle zadaní v programu MS PowerPoint. Popsal jsem v ní pro potřeby výuky grafické formáty GIF a PNG, které jsem rozebral podrobněji v teoretické části práce. Tímto jsem splnil poslední bod zadání bakalářské práce.

Obrázek č. 21 – Ukázka prezentace pro výuku

**Základní charakteristika formátu GIF**

- Při přenosu rastrových obrázků po pomalých přenosových linkách (modemy atd.) se uplatnila možnost prokládání řádků v jednotlivých rámcích. Prokládání funguje tak, že se nejdříve přenese každý osmý řádek, posléze každý čtvrtý resp. druhý řádek (kromě těch přenesených), a tak dále do posledního kroku, ve kterém jsou přeneseny všechny ostatní - liché řádky.

<table>
<thead>
<tr>
<th>nerekládáno</th>
<th>prokládáno</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>64</td>
<td>128</td>
</tr>
</tbody>
</table>

- U GIFu může mít každý rámec svou lokální barvovou paletu s libovolným počtem barev v rozsahu 2–256. Počet bitů na pixel je také volitelný a může dosahovat libovolné hodnoty 1-8.

**Prokládání pixelů ve formátu PNG**

- sedm kroků vedoucích k vykreslení všech pixelů v obrázku
ZÁVĚR

Cílem této bakalářské práce bylo vytvořit literární rešerši, zabývající se historií a vývojem rastrových (bitmapových) grafických formátů GIF a PNG a dále naprogramovat aplikaci zobrazující uvedené formáty a uzpůsobit zdrojové kódy tak, aby je bylo možno implementovat do jiné aplikace.

V teoretické části jsem se před samotným popisem zmíněných bitmapových grafických formátů krátce věnoval bitmapové grafice obecně. Uvedl jsem její základní vlastnosti a stručně popsal, jakým způsobem jsou data v bitmapové grafice zobrazena. Dále jsem shrnul jak výhody bitmapové grafiky, mezi které patří především široká podpora co do počtu různých druhů formátů nebo nezávislost na obsahu obrázku, tak nevýhody týkající se datové náročnosti a ztráty kvality obrázku při jeho zvětšení. Poté jsem ještě naznačil způsob využití bitmapové grafiky, který se týká např. fotografií nebo složitých ilustrací. Následuje obecný popis bitmapového formátu, obsahující výčet nejzákladnějších prvků, které každý bitmapový formát obsahuje. V dalších kapitolách se již přímo věnuji popisu grafických formátu GIF a PNG. U každého formátu je popsána jeho historie, stručná charakteristika, výhody a nevýhody, struktura ukládání dat, členění souboru a možnosti uplatnění.

V praktické části jsem vyvinul aplikaci zobrazující oba formáty. Kromě možností uložit / načíst / zobrazit tyto formáty jsem navíc přidal pár funkcí, umožňující základní transformace s obrázky jako je např. změna velikosti obrázku, otočení obrázku o zadaný počet stupňů nebo inverzi barev. Aplikace má jednoduché grafické rozhraní a intuitivní ovládání, přičemž tu je také možnost využití klávesových zkratek, které usnadní a urychlí celkovou práci s obrázky. Program jsem také úspěšně otestoval jak na operačním systému Windows XP (32-bit / 64-bit), tak i na Windows Vista (32-bit / 64-bit) a mohu konstatovat, že program fungoval bez jakéhokoliv problému na obou zmíněných OS. Jedinou nevýhodou programu je to, že nelze použít nikde jinde, než v MFC z důvodu ne zcela šťastného navržení celé aplikace. Program je také řádně popsán a zdokumentován v praktické části textu.

Na CD přiloženém k této práci jsou umístěny zdrojové soubory aplikace a vygenerovanou dokumentace pomocí programu Doxygen, která detailně popisuje strukturu aplikace a dále prezentaci, určenou jako pomůcka pro výuku předmětu Počítačová grafika.
ZÁVĚR V ANGLIČTINĚ

The aim of the bachelor’s thesis is to create the literature search, which is interested in the history and the progress of the raster graphics formats GIF and PNG and then programme the application, which displays these formats and modificate the source codes to be able to implement them to another application.

In the theoretical part I was interested before the description of the bitmap graphics formats in bitmap graphic in general. I introduced the basic characteristics and shortly described how are the dates in the bitmap graphic displayed. Then I showed both the advantages of the bitmap graphic (which included the wide support of the different kinds of formats or the independence from the capacity of the picture) and the disadvantages refers to the dates intensity and the loss of quality of the picture by zooming. Then I showed the way how use the bitmap graphic, which refers to for example photographs or the composite illustrations. It is followed by the general description of the bitmap format, which includes the enumeration of the most fundamental items, which each bitmap format includes. In the other heads I straight turned to the description of the graphics formats GIF and PNG. In each format is described its history, description brief, advantages and disadvantages, the structure of the dates saving, the structure of the file and facility of use.

In the practical part I worked up the application displaying both formats. In addition to possibility save/fetch/display these formats I added several basic functions, which are able to transform the picture, for example the change of the size of picture, turning round the picture of the said number of degrees or color inversion. The application has simple graphics interface – line and intuitive operating. There is the possibility to use keyboard shortcuts, which helps and speeds up the work with pictures. The programme was tested both on the operating system Windows XP (32-bit / 64-bit) and Windows Vista (32-bit / 64-bit). I can state that the programme worked without any problems on both operating systems. The only disadvantage of the programme is that it couldn’t be use nowhere else than in MFC for reason of no bright design of whole application. The programme is correctly described and documented in the practical part of the text.

On the CD enclosed to this thesis are placed the source files and generated documentation assists in programme Doxygen, which detaily describes the structure of the application and presentation, which is designed for the education of the subject Computer graphic.
SEZNAM POUŽITÉ LITERATURY


### SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATKŮ

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Ohraničení</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Programovací jazyk ANSI C</td>
</tr>
<tr>
<td>C++</td>
<td>Programovací jazyk ISO C++ vzešlý z jazyka ANSI C</td>
</tr>
<tr>
<td>RGB</td>
<td>Barevný model RGB neboli červená-zelená-modrá je aditivní způsob míchání barev používaný u většiny monitorů či projektorů</td>
</tr>
<tr>
<td>RGBA</td>
<td>Barevný model vycházející z modelu RGB, který je rozšířen o tzv. alfa kanál</td>
</tr>
<tr>
<td>GIF</td>
<td>Graphics Interchange Format – grafický formát vyvinutý firmou CompuServe</td>
</tr>
<tr>
<td>PNG</td>
<td>Portable Network Graphics – grafický formát v mnoha ohledech nahrazující GIF</td>
</tr>
<tr>
<td>APNG</td>
<td>Animated Portable Network Graphics – PNG formát rozšířený o podporu animací</td>
</tr>
<tr>
<td>MNG</td>
<td>Multiple-image Network Graphics - grafický formát pro animované obrázky</td>
</tr>
<tr>
<td>BMP</td>
<td>Microsoft Windows Bitmap – grafický formát vyvinutý firmou Microsoft</td>
</tr>
<tr>
<td>JFIF</td>
<td>JPEG File Interchange Format – metoda ztrátové kompresi používané pro ukládání počítačových obrázků ve fotorealistické kvalitě</td>
</tr>
<tr>
<td>JPEG</td>
<td>Joint Photographic Experts Group - název konsorcia, které navrhlo JFIF kompresi</td>
</tr>
<tr>
<td>TIFF</td>
<td>Tag Image File Format – grafický formát vyvinutý v roce 1986 firmou Aldus</td>
</tr>
<tr>
<td>DTP</td>
<td>Desktop Publishing - publikování na stole potažmo publikování pomocí počítače</td>
</tr>
<tr>
<td>PSD</td>
<td>Photoshop Dokument-formát firmy Adobe užívaný programem Adobe Photoshop</td>
</tr>
<tr>
<td>LZW</td>
<td>Lempel-Ziv-Welch - bezeztrátový kompresní algoritmus vyvinutý Abrahadem Lempelem, Jacobem Zivem a Terry Welchem</td>
</tr>
<tr>
<td>GPL</td>
<td>General Public Licence - Všeobecná veřejná licence</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange - americký standardní kód pro výměnu informací</td>
</tr>
<tr>
<td>MFC</td>
<td>Microsoft Foundation Class - objektově orientovaná implementace Win32 API</td>
</tr>
<tr>
<td>STL</td>
<td>Standard Template Library - Standardní knihovna jazyka C++ obsahující řadu užitečných tříd a rozhraní, podstatně rozšiřujících schopnosti základního C++</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface - rozhraní pro programování aplikací</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic Link Library</td>
</tr>
</tbody>
</table>
SEZNAM OBRÁZKŮ

Obrázek č. 1 - Převod obrazu do bitmapové grafiky .................................................................9
Obrázek č. 2 - Demonstrace zhoršení kvality bitmapového obrázku při jeho zvětšení......................10
Obrázek č. 3 - Ukázka rastrové grafiky se souřadnicemi bodů a jejich RGB a HEX barevnými kódy….11
Obrázek č. 4 - Aditivní míchání barev u barevného modelu RGB ..................................................12
Obrázek č. 5 - Ukázka „zmitosti“ a ztráty plynulosti barevných přechodů u obrázku typu GIF ........14
Obrázek č. 6 - Příklad obrázku PNG s 8-bitovou průhledností .........................................................15
Obrázek č. 7 - Ukázka nejmenšího možného obrázku typu GIF zobrazeného v multieditoru PSpad ....18
Obrázek č. 8 - Pokus o otevření obrázku PNG příkazem „type“ v prostředí MS-DOS / WINDOWS …25
Obrázek č. 9 - Základní třídy a jejich dědiční vztahy ................................................................37
Obrázek č. 10 - Základní dialogové třídy a jejich předkové ............................................................37
Obrázek č. 11 - Dialogové okno - Vlastnosti ..................................................................................38
Obrázek č. 12 - Dialogové okno - O aplikaci ................................................................................38
Obrázek č. 13 - Dialogové okno - Rotace .......................................................................................39
Obrázek č. 14 - Dialogové okno - Změna velikosti .........................................................................39
Obrázek č. 15 - Dialogové okno pro otevírání obrázků – Otevřít .......................................................41
Obrázek č. 16 - Průchod funkcemi při otevírání obrázku .................................................................42
Obrázek č. 17 - Průchod funkcemi při ukládání obrázku .................................................................43
Obrázek č. 18 - Ukázka uchovávání tří duplikátů na zásobníku operací .............................................44
Obrázek č. 19 - Průchod funkcemi při zpětném vrácení obrázku ....................................................44
Obrázek č. 20 - Zprávy generované kliknutím na tlačítko z panelu nástrojů .................................46
Obrázek č. 21 - Ukázka prezentace pro výuku ................................................................................49
SEZNAM TABULEK

Tabulka č. 1 - Tabulka povolených bloku vyskytujících se v souborech typu GIF.........................16
Tabulka č. 2 - Tabulka znázorňující pořadí bloků v souborech typu GIF.............................................17
Tabulka č. 3 - Tabulka znázorňující pořadí bloků v souboru typu GIF tvořeném třemi rámcemi........17
Tabulka č. 4 - Význam bytů v jednotlivých bločích souboru typu GIF..................................................19
Tabulka č. 5 - Význam jednotlivých trojic bitů komprimovaných dat souboru typu GIF...............22
Tabulka č. 6 - Význam bytů v hlavičce souboru PNG.................................................................24
Tabulka č. 7 - Pořadí povinných chunků v souboru typu PNG ..........................................................28
Tabulka č. 8 - Pořadí povinných chunků u obrázku obsahující barovou paletu v souboru typu PNG... 28
Tabulka č. 9 - Význam offsetů IHDR chunku v souboru typu PNG......................................................29
Tabulka č. 10 - Vztah mezi bitovou hloubkou pixelů a typem kódování barev u souboru typu PNG..... 29
Tabulka č. 11 - Význam jednotlivých sekvencí bytů v chunku IEND u souboru typu PNG...............30
Tabulka č. 12 - Nejvýznamnější nepovinné chunky v grafických souborech PNG.........................30
Tabulka č. 13 - Jednotlivých sekvencí bytů u obrázku typu PNG uloženém ve stupních šedi............32
SEZNAM PŘÍLOH

PI Dokumentační CD obsahující elektronickou verzi této bakalářské práce, prezentaci vytvořenou v programu MS PowerPoint a program včetně zdrojových souborů.