VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PYTHON IMAGING LIBRARY

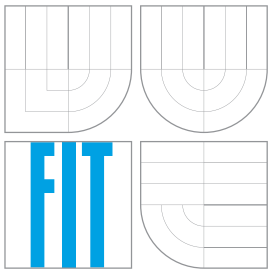
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN MÁGR

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PYTHON IMAGING LIBRARY

PYTHON IMAGING LIBRARY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN MÁGR

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL VYSKOČIL

BRNO 2008

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací editoru grafů, který je propojen s konkrétním informačním systémem protokolem XML-RPC. Aplikace je vytvořena pomocí jazyka Python a jeho grafické knihovny Python Imaging Library. Pro realizaci grafického uživatelského rozhraní je využívána knihovna GTK+ resp. její aplikační programové rozhraní pro Python, PyGTK.

Klíčová slova

Python, Python Imaging Library, GTK+, PyGTK, protokol XML-RPC, grafická knihovna, editor grafů

Abstract

This bachelor thesis is engaged in concept and implementation of graph editor, which is connected with concrete information system using protocol XML-RPC. The application is created using programming language Python and its graphic library Python Imaging Library. Graphic user interface is realized by GTK+ or more precisely by PyGTK, its application program interface for Python.

Keywords

Python, Python Imaging Library, GTK+, PyGTK, protokol XML-RPC, graphic library, graph editor

Citace

Martin Mágr: Python Imaging Library, bakalářská práce, Brno, FIT VUT v Brně, 2008

Python Imaging Library

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Vyskočila. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Mágr
12. května 2008

Poděkování

Rád bych poděkoval vedoucímu této práce panu Ing. Michalovi Vyskočilovi za jeho rady a pomoc s návrhem. Další poděkování patří mým nadřízeným. Jmenovitě se jedná o pány Libora Michalčíka a Michala Koždoně, za umožnění propojení aplikace s firemním informačním systémem, dále Aleše Křupku DiS. a Petra Menšíka za shovívavost ohledně práce na aplikaci v pracovní době.

Obsah

1 Úvod **2**

2 Použité technologie **3**
2.1 Python . . . 3
2.1.1 Srovnání s obdobnými jazyky . . . 4
2.2 Python Imaging Library . . . 4
2.2.1 Vlastnosti knihovny . . . 5
2.2.2 Struktura knihovny . . . 6
2.2.3 Použité moduly . . . 8
2.3 Protokol XML-RPC . . . 11

3 Vlastní práce **12**
3.1 Návrh . . . 12
3.1.1 Specifikace požadavků . . . 12
3.1.2 Grafické uživatelské rozhraní . . . 12
3.2 Implementace . . . 14
3.2.1 Struktura aplikace . . . 14
3.2.2 Datové struktury . . . 16
3.2.3 Třída Draw modulu painter . . . 19

4 Závěr **22**

Kapitola 1

Úvod

V dnešní době většina firem, menších či větších, využívá nějaký informační systém. Nemalý počet těchto firem se potýká s nutností uložená data v databázi informačního systému nějakým způsobem prezentovat. To je možné několika způsoby, avšak nejsrozumitelnější pro zákazníky se většinou jeví grafická reprezentace, tedy různé grafy, diagramy apod.

Postupy, kterými se docílí převedení dat na grafickou podobu, se liší v závislosti na finančních a softwarových možnostech konkrétní firmy. Nejrozšířenější jsou dvě varianty. První je generovat grafy přímo informačním systémem a druhá je export dat do formátu pro tabulkový procesor. Pomocí zvoleného tabulkového procesoru se poté z exportovaných dat vytvoří graf.

Každá z těchto metod je nějakým způsobem omezující. Generované grafy informačním systémem mají většinou pevně daný vzhled. Pokud je nutné změnit vzhled dokumentu, ve kterém vygenerované grafy hodláme použít, je možné, že grafy budou svým grafickým stylem rozdílné od stylu dokumentu. Metoda exportu dat do formátu pro tabulkový procesor je sice flexibilnější, ale na druhou stranu je značně časově náročná. Pokud je množství exportovaných dat velké, je tato metoda také nepřehledná a je zde riziko, že tvůrce grafu udělá chybu.

Na základě výše uvedené úvahy je cílem této bakalářské práce vytvořit aplikaci, která je přímo propojena s databází informačního systému a umožňuje vykreslovat data do grafů, jejichž vzhled si může uživatel libovolně měnit.

Kapitola 2 obsahuje rozbor jazyka Python, kterým bude aplikace implementována. Jsou zde popsány jeho vlastnosti. Jazyk je poté srovnán s několika obdobnými programovacími jazyky. V další části kapitoly je popsána knihovna Python Imaging Library. Jsou zde uvedeny její vlastnosti, struktura a podrobnější popis některých jejích částí. Konec této kapitoly je věnován protokolu XML-RPC.

Kapitola 3 se zabývá již samotnou aplikací. Obsahuje specifikaci požadavků a popis návrhu grafického uživatelského rozhraní aplikace. Dále se kapitola zaobírá achitekturou aplikace a datových struktur používaných v aplikaci. V této kapitole je také podrobně rozebráno, jakým způsobem je graf vykreslován.

Závěr je věnován zhodnocení přínosů této práce a grafické knihovny Python Imaging Library.

Kapitola 2

Použité technologie

V této kapitole je uvedena charakteristika programovacího jazyka, ve kterém byla aplikace napsána, a stručné srovnání tohoto jazyka s ostatními programovacími jazyky. V kapitole jsou také uvedeny stěžejní technologie, na kterých je vytvořená aplikace postavena, a shrnuty jejich klady a zápory.

2.1 Python

Python je programovací jazyk vyvinutý Giudo van Rossumem. Byl navržen jako objektově orientovaný jazyk a jeho model podporuje vlastnosti objektově orientovaného programování, jako je polymorfismus, přetěžování a vícenásobná dědičnost (viz [1]).

Jazyk má základní sadu interních datových typů a struktur známou z vyšších jazyků. Sada je ale navíc rozšířena o asociativní pole, které jsou v Pythonu nazývané slovníky. Slovníky jsou účinnou pomůckou při tvorbě programu a v mnoha případech usnadňují řešení problému, jelikož lze indexovat jakýmkoliv datovým typem. Vzhledem k tomu, že každý objekt v Pythonu obsahuje slovník a že všechno v Pythonu je objekt (včetně čísel nebo funkcí), je zřejmé, že slovníky jsou základním stavebním prvkem jazyka. Další interní datová struktura charakteristická pro Python je n-tice. Na první pohled se n-tice zdá být stejná jako datová struktura seznam. Zdaní ale klame. N-tice lze pouze vytvářet. Jednou vytvořená n-tice se již nedá modifikovat.

Syntaxe se velmi podobá ostatním vyšším jazykům. Stejně jako například jazyk C obsahuje Python rozhodovací konstrukce, podmíněné i nepodmíněné cykly. Jejich použití je ale do jisté míry jednodušší. Další výhodou jazyka je, že celkový kód programu napsaný v Pythonu je, oproti kódu napsanému v jiném jazyce, znatelně kratší. Kromě dobře navrženého objektového modelu, syntaxe a flexibilních datových struktur má na tom podíl také to, že v Pythonu není nutné používat žádné speciální znaky k uvození bloku. Začátek bloku se určí pouhým odsazením textu od levého okraje.

Vytvořený kód není kompilován do spustitelného souboru, ale překládá se do byte kódu, který je následně interpretován. Toto může být bráno jako nevýhoda, jelikož z hlediska rychlosti běhu programu se Python nevyrovná programu napsanému v kompilovaném jazyce (více v [2]). Na druhou stranu pokud vezmeme v potaz výkon dnešních personálních počítačů, je tato nevýhoda zanedbatelná. Interpret Pythonu je napsán v jazyce ANSI C, který je nezávislý na platformě. Z toho vyplývá, že je prakticky možné ho spustit v jakémkoliv operačním systému. Jmenujme například nejrozšířenější operační systém MS Windows, serverové operační systémy Unix nebo Linux, ale také v Evropě nepříliš

rozšířený operační systém MAC OS. Pokud programátor nepoužívá speciální knihovny vztahující se na konkrétní operační systém, je zdrojový kód velmi lehce přenositelný.

Správa paměti je plně automatická. Jazyk je dynamicky typovaný, proto není nutné složitě deklarovat typ či alokovat určitou velikost paměti konkrétnímu objektu. To se provádí na základě hodnoty nebo velikosti objektu přiřazenému k proměnné. Jinými slovy typy nejsou spojené s proměnnými, ale s objekty resp. hodnotami. Taktéž realokace paměti pro objekt nebo proměnnou za běhu programu je automaticky ošetřena a programátor se tím nemusí zabývat. Stejné je to i s uvolňováním paměti, které zajišťuje takzvaný garbage collector (česky sběrač odpadu). Ten již nepoužívané objekty a proměnné sám odstraní a tudíž programátor není nucen z paměti explicitně objekty a proměnné uvolňovat.

2.1.1 Srovnání s obdobnými jazyky

V následujících bodech jsou shrnuty a doplněny výše uvedené vlastnosti jazyka Python a srovnány s vlastnostmi obdobných vyšších jazyků jako je jazyk C, C++ a Java.

- **Interpretovaný jazyk**: Python je stejně jako Java jazyk interpretovaný. Nevýhodou tohoto řešení je pomalejší běh programu než při použití kompilovaných jazyků C nebo C++. Tento fakt můžeme ale díky dnešní síle výpočetní techniky zanedbat. Na druhou stranu právě to, že interpret jazyka je napsán v platformně nezávislém jazyku C, je příčinou přenositelnosti zdrojového kódu napsaného v Pythonu.
- **Jednodušší syntaxe**: Díky rozsáhlé standartní knihovně, ve které je používána velká míra abstrakce, je syntaxe Pythonu jednodušší a lépe čitelná. Začátek bloku v programu se určuje pomocí odsazení textu od levého okraje, což má nemalý podíl na tom, že v porovnání s jazyky C/C++ a Java je zdrojový kód několikanásobně kratší.
- **Dynamické typování**: Není nutné deklarovat typ proměnných. Kdykoliv se za běhu programu může změnit, jelikož typ je vázán na hodnotu nebo objekt přiřazený k proměnné. Tuto možnost žádný z jazyků C, C++ nebo Javy neumožňuje.
- **Standartní datové typy**: Python nabízí stejné standartní typy jako výše uvedené jazyky. Navíc ale přidává jako standartní datové struktury asociativní pole a takzvané n-tice.
- **Automatická správa paměti**: O alokaci, realokaci i dealokaci se stará interpret jazyka sám. Programátor je zbaven starosti o správu paměti. Problém takzvaný „memory leak“ odpadá.

Kompletní dokumentaci jazyka Python lze nalézt na [6]

2.2 Python Imaging Library

Pro jazyk Python je na internetu možné stáhnout nemalé množství knihoven, které určitým způsobem rozšiřují možnosti jazyka. Jednou z těchto knihoven je také Python Imaging Library (dále jen PIL). Tato knihovna přidává jazyku možnost programově zpracovávát rastrové obrázky.

2.2.1 Vlastnosti knihovny

Prostřednictvím knihovny lze načítat a ukládat obrázky ve většině rastrových formátů. Některé další formáty je modul schopen pouze přečíst či pouze zapsat, což ale nemusí být překážkou. Systém pro podporu formátů je uživatelsky rozšiřitelný. Napsat si tedy vlastní kodér a dekodér je při troše usilí možné. Otevření rastrového obrázku je velice dobře vyřešeno a optimalizováno k otevírání více souborů najednou bez zbytečného zpomalení programu. Modul při otevření souboru načte pouze hlavičku otevíraného souboru, ve které jsou údaje o velikosti obrázku a o použitém barevném modelu. Zbytek souboru se načte až když je potřeba s grafickými daty (pixely) manipulovat. Níže je uveden výpis formátů souborů, se kterými je možno pracovat:

- **Plně podporované formáty**: BMP, GIF, IM, JPEG, MSP, PCX, PNG, PPM, SPIDER, TIFF, XBM
- **Formáty, které je možné pouze přečíst**: CUR, DCX, FLI, FLC, FPX, GBR, GD, ICO, IM, IPTC/NAA, MCIDAS, MIC, PCD, PIXAR, PSD, SGI, TGA, WAL, XPM
- **Formáty, které je možné pouze zapsat**: EPS, PALM, PDF
- **Formáty, které je možné pouze identifikovat**: BUFR, FITS, GRIB, HDF5, MPEG, WMF

Při práci s PIL je mimo formátu nutné znát barevný model. Knihovna je schopna rozeznat a pracovat s několika barevnými modely, v PIL nazývané módy. V následujícím seznamu jsou uvedené podporované barevné modely a jejich stručný popis:

- **Mód "1"**: Při práci pouze s černou a bílou barvou využijeme tento mód (1 bit/pixel)
- **Mód "L"**: Mód "L" představuje 256 stupňů šedi (8 bitů/pixel)
- **Mód "P"**: Používá se při mapování do jiného módu. K tomu je používána paleta barev. Konkrétní barva se určí pomocí ukazatele do palety (8 bitů/pixel)
- **Mód "RGB"**: Barva pixelu se určuje pomocí poměru červené, zelené a modré barvy (3 x 8 bitů/pixel)
- **Mód "RGBA"**: Podobný mód jako předchozí. K hodnotám zakladních barev je přidána míra průhlednosti pixelu (4 x 8 bitů/pixel)
- **Mód "CMYK"**: Známý model se subtraktivním mícháním barev (4 x 8 bitů/pixel)
- **Mód "I"**: Použití 32bitových celočíselných hodnot barev
- **Mód "F"**: Stejná velikost hodnot barev, ale použití reálných čísel

Práce s barvami není omezena pouze na použití všech barevných kanálů najednou. Je možné si například obrázek v "RGB" módu rozdělit na tři nezávislé obrázky, tedy jeden obrázek pro barevný kanál červené barvy, jeden pro kanál zelené barvy a jeden pro kanál modré barvy. S jednotlivými kanálovými obrázky pak lze provádět rozdílné úpravy a po ukončení úprav zase kanálové obrázky spojit zpět do jednoho obrázku.

Téměř všechny nástroje knihovny, vyjma jednoho na vytváření postscriptových souborů, považují v obrázku počátek souřadnicového systému levý horní roh.

2.2.2 Struktura knihovny

Nástroje knihovny jsou rozděleny do několika modulů v závislosti na jejich funkci. Níže jsou uvedeny názvy jednotlivých modulů a jejich stručný popis.

- `Image`:
  Prostřednictvím toho modulu je k dispozici třída stejného jména, která reprezentuje obrázek. Z toho je zřejmé, že funkcemi tohoto modulu se otevírají obrázky ze souboru, ukládájí do souboru nebo vytvářejí nové. Třída *Image* poskytuje metody pro transformaci jak celého obrázku, tak i zvoleného výřezu. Jedná se o otočení, překlopení či změnu velikosti. Třída disponuje také metodou umožňující použít na obrázek či výřez obrázku volitelný filtr. Dále jsou v této třídě naimplementovány metody pro rozkládání obrázku na barevné kanály a metoda pro složení těchto kanálů zpět do jednoho obrázku

- `ImageChops`:
  Tento modul obsahuje nástroje provádějící aritmetické operace s obrázky. Tyto operace se využijí hlavně při implementaci různých grafických efektů nebo například k algoritmickému kreslení. Jako příklad lze uvést sčítání, odčítání nebo násobení dvou obrázků. Jsou zde ale také funkce pro invertování, zesvětlení a ztmavení obrázku.

- `ImageColor`:
  Zde lze nalézt převáděcí funkce z různých formátů určující barvu. Převádí se do formátu tří čísel uložených v n-tici. Jednotlivá čísla představují hodnoty základních barev z barevného modelu RGB. Tento formát byl jediný použitelný v PIL do verze 1.1.4. Od této verze je možno v knihovně používat několik druhů specifikace barvy v řetězcovém formátu. Jednou z nich je specifikace barev používaná v HTML (*"#rrggbb"*). Další způsob je uvedení jména barvy (*"red"*). Modul podporuje 140 standartních jmen barev a nerozlišuje se zde velikost písmen. Dále je možné uvést procentuální poměr barev (*"rgb(100%,0%,0%)"*). Poslední možnost je uvedení odstínu barvy (tedy barvy, nasycení a jasu). Pořadí je nutné dodržet následovně: barva, nasyceni v %, jas v %. Například *"hsl(0,100%,50%)"* představuje sytě červenou barvu.

- `ImageDraw`:
  Metody třídy Draw umožňují vykreslovat do obrázku grafická 2D primitiva, jako je bod, čára, obdelník, elipsa nebo kruhová výseč. Do obrázku lze vkládat mimo grafických primitiv také text.

- `ImageEnhance`:
  V modulu *ImageEnhance* je k dispozici několik tříd určených k zprostředkování změny parametrů obrázku. Pomocí metod těchto tříd je možno změnit vyvážení barev, jas, kontrast nebo obrázek zaostřit.

- `ImageFile`:
  ImageFile je součástí modulu *Image*. Právě v něm jsou implementovány funkce pro načtení a uložení obrázku. Zpřístupněn individuálně je zřejmě z toho důvodu, že obsahuje třídu *Parser*, jejíž metody slouží k načtení souboru po částech. Tuto vlastnost je možno použít např. ke zpracovávání právě stahovaného obrázku.

- `ImageFileIO`:
  Tento modul je již zavržen. V podstatě měl stejnou funkci jakou má nyní třída *Parser* modulu *ImageFile*.

- `ImageFilter`:
  Zde jsou definovány filtry, které se používají v modulu *Image*. Jako příklad lze uvést filtry *BLUR* nebo *EMBOSS*, z jejichž jména je patrné jakým způsobem obrázek modifikují[1]. Dále jsou v tomto modulu zpřístupněny funkce na úpravu stávajících filtrů. Programátor tak není omezen na filtry ze základní sady, ale může si vytvořit své vlastní.

- `ImageFont`:
  Modul *ImageFont* obsahuje třídu se stejným názvem. Instance této třídy reprezentují načtené písmo ze souboru. Podpora formátů písem je široká. Modul je schopen načíst, jak formáty pro X Window (formáty BDF a PCF), tak truetypová písma (formát TTF), která využívá MS Windows. Navíc je možno fonty uložené ve formátu BDF a PCF převést na vnitřní formát fontu PIL.

- `ImageGrab`:
  Funkce modulu *ImageGrab* jsou schopny sejmout obrazovku a uložit jí do objektové reprezentace obrázku. Uvedený modul je schopen mimo obrazovky také načíst obrázek uložený ve schránce. Bohužel se výše popsané funkce dají použít pouze v operačním systému MS Windows.

- `ImageMath`:
  Od verze 1.1.6 knihovny je k dispozici tento modul pro vyhodnocování standartních Pythonovských výrazů nad obrázky. Modul je schopen vyhodnotit například funkce abs(image), float(image), max(image1, image2) nebo min(image1, image2). Parametry funkcí jsou, jak již bylo uvedeno výše, objektové reprezentace obrázku.

- `ImageOps`:
  Vzhledem k tomu, že se jedná zatím o experimentální modul je funkčnost omezena pouze na barevné módy *"L"* a *"RGB"*. Je to v podstatě sada funkcí, které jsou schopny různě graficky zpracovat obrázek. Pomocí tohoto modulu je možné například deformovat obrázek, ořezávat okraje, obarvit černobílý obrázek nebo invertovat obrázek podle zvoleného prahu.

- `ImagePalette`:
  V tomto modulu nalezneme třídu, která reprezentuje paletu barev. Vytvořenou paletu lze poté naplnit barvami a připojit ke konkrétnímu obrázku. Tento obrázek však musí být otevřen v módu "P" (viz kapitola 2.2.1).

- `ImagePath`:
  Funkce modulu *ImagePath* slouží k vytváření dvoudimenzionálních vektorů. Tyto vektory se poté mohou použít v modulu *ImageDraw* k určení souřadnic vykreslovaných grafických primitiv.

- `ImageQt`:
  Knihovna obsahuje tři moduly pro podporu toolkitů pro tvorbu grafického uživatelského rozhraní (dále jen GUI). *ImageQt* je jedním z nich. Umožňuje vytvářet objekty *QImage* z objektů *Image*. Objekty *QImage* jsou součástí toolkitu PyQt4, který slouží k tvorbě desktopového prostředí KDE. Modul je přístupný až od verze knihovny 1.1.6.

[1] filtr BLUR obrázek rozostří a filtr EMBOSS vyryje světlé části obrázku do černobílého podkladu

- `ImageSequence`:
  Díky modulu *ImageSequence* je možno zpracovávat animované soubory GIF nebo jiné sekvenčně animované formáty (např. FLI). Třída *Iterator*, jak je již z názvu patrné, umožňuje iterovat nad jednotlivými rámci animace.

- `ImageStat`:
  Pomocí tohoto modulu jsme schopni vypočítat statistické hodnoty z obrázku. Jedná se například o celkový počet nebo součet všech pixelů. Další funkce jsou průměr, medián úrovně, extrémy pixelů atd.

- `ImageTk`:
  Další modul určený k podpoře toolkitů pro tvorbu GUI. Tentokrát se jedná o toolkit *Tkinter*, který je součástí distribučního balíku jazyka Python. Tento modul tedy převádí objekty třídy *Image* na objekty třídy *BitmapImage* nebo *PhotoImage*, které jsou už toolkitem zpracovatelné.

- `ImageWin`:
  Modul *ImageWin* je třetím a posledním modulem zprostředkovávájícím podporu konkrétního GUI toolkitu. *ImageWin* je určen pro vytváření a zobrazování obrázků v prostředí operačního systému MS Windows.

- `PSDraw`:
  V tomto modulu je jako počátek souřadnicového systému považován levý dolní roh. Modul *PSDraw* slouží k vytváření postscriptových souborů

2.2.3 Použité moduly

V této části je věnována pozornost používaným modulům a metodám v implementovaném editoru grafů. Najdeme zde jejich přiblížení, avšak všechny funkce použitých modulů zde popsány nejsou. Příklady použití a detailní popis použitých a ostatních modulů lze nalézt na [4].

Modul Image

Jak již bylo uvedeno v kapitole 3.2.1, slouží tento modul k vytvoření objektové reprezentace obrázku. S objektem se poté dále pracuje. K načtení obrázku ze souboru použijeme funkci *open* následovně:

```
import Image
obrazek = Image.open('/home/obrazek.jpg')
```

K vytvoření nového obrázku se použije funkce *new*. Funkci je nutno předat rozměry obrázku a barevný model:

```
import Image
obrazek = Image.new('RGB', (800, 600))
```

Jako třetí parametr můžeme předat barvu pozadí. Pokud si žádnou nezvolíme, bude barva pozadí černá. Pokud jako barvu předáme objekt *None*[2], nebude obrázek zinicializován, jinými slovy se nevytvoří žádné pozadí.

[2]Objekt None představuje v Pythonu to samé jako NULL v jazyce C/C++

Ukladání obrázku do souboru je v PIL vyřešeno metodou *save* objektu obrázku. Jako parametr se této funkci předá cesta k souboru. Formát se určí automaticky z přípony názvu souboru, nebo ho lze explicitně určit parametrem. V příkladě se načte obrázek ze souboru v JPG formátu a následně uloží do souboru v PNG formátu:

```
import Image
obrazek = Image.open('/home/obrazek.jpg')
obrazek.save('/home/novy.png', 'PNG')
```

Obrázky lze převádět mezi barevnými modely pomocí metody *convert*. Parametrem metody je barevný model, do kterého chceme obrázek převést. Tuto metodu je možno využít k jednoduchému převedení obrázku na černobílou variantu pomocí práhování. Pokud totiž použijeme mód *"1"*, tak se všechny pixely s hodnotou nad 127 změní na černé. Naopak pixely s hodnotou pod 127 se změní na bílé. Následující příklad znázorňuje načtení obrázku ze souboru a převedení jeho barevného modelu na model *"CMYK"*:

```
import Image
obrazek = Image.open('/home/obrazek.jpg')
obrazek.convert('CMYK')
```

Změnu rozměrů obrázku zprostředkovává metoda *resize*. Parametry má dva, jeden povinný a jeden volitelný. Povinný parametr je dvoumístná n-tice s novými rozměry obrázku. Druhý parametr je filtr, který se použije při zvětšování obrázku. Varianty, mezi kterými si můžeme zvolit, jsou následující: $NEAREST$(metoda nejbližšího souseda), $BILINEAR$(lineární interpolace maticí 2x2), $BICUBIC$(interpolace pomocí matice 4x4) a $ANTIALIAS$(antialiasing). Níže se načtenému obrázku změní rozměry na 800x600 s použitím filtru $BILINEAR$:

```
import Image
obrazek = Image.open('/home/obrazek.jpg')
novy = obrazek.resize((800,600), Image.BILINEAR)
```

Metoda *paste* je určena k vkládání obrázků. Této metodě je nutné předat dva parametry. První parametr je objekt obrázku, který chceme vložit. Druhý parametr je n-tice obsahující souřadnice na které se má obrázek vložit. Pokud má vkládaný obrázek jiný barevný model než obrázek, do kterého se vkládá, je barevný model vkládaného obrázku převeden tak, aby souhlasil s modelem „hostitelského“ obrázku. Následuje názorný příklad, kdy na souřadnice (10,10) vložíme obrázek:

```
import Image
prvni = Image.open('/home/obrazek.jpg')
druhy = Image.open('/home/obrazek2.jpg')
prvni.paste(druhy, (10, 10))
```

Modul ImageDraw

ImageDraw je v editoru grafů využíván především, jelikož veškeré vykreslování je realizováno jím. V první řadě je nutné vytvořit instanci třídy *Draw*, která umožňuje vykreslovat do obrázku:

```
import Image, ImageDraw
obrazek = Image.open('/home/obrazek.jpg')
kresli = ImageDraw.Draw(obrazek)
```

Poté je možno volat následující metody této instance třídy *Draw*:

- `arc(xy, start, end, outline)`:
  Metoda *arc* slouží k vykreslování oblouků. Oblouk je v podstatě část elipsy vepsané do obdelníku. Parametr *xy* určuje souřadnice levého horního a pravého dolního rohu tohoto obdelníka. Pomocí parametrů *start* a *end* se určí počáteční a koncový úhel oblouku. Nepovinný parametr *outline* představuje barvu oblouku.

- `ellipse(xy, outline, fill)`:
  Touto metodou se vykreslí elipsa. Rozměry elipsy se určí stejným způsobem jako v metodě *arc*. Nepovinným parametrem *outline* je možno změnit barvu okraje elipsy a nepovinným parametrem *fill* lze měnit barvu výplně.

- `line(xy, fill, width)`:
  Již z názvu je patrné, že metoda *line* slouží k vykreslení čáry. Parametr *xy* obsahuje souřadnice počátečního a koncového bodu. Nepovinným parametrem *fill* se mění barva čáry. Šířku čáry je možno určit pomocí nepovinného parametru *width*.

- `pieslice(xy, start, end, outline)`:
  Vyplněná výseč elipsy se vykresluje pomocí metody *pieslice*. Rozměry elipsy se předávají parametrem *xy*. Parametr *outline* určuje barvu okraje výseče a parametr *fill* barvu výplně. Stejnou funkci jako v metodě *arc* zde mají parametry *start* a *end*.

- `rectangle(box, outline, fill)`:
  Metodou *rectangle* se vykreslují obdelníky. Parametrem *box* se určí levý horní a pravý dolní roh obdelníka. Stejně tak jako v předchozích metodách určuje parametr *outline* barvu okraje a parametr *fill* barvu výplně.

- `text(position, string, font, fill)`:
  Tato metoda umožňuje vkládat text do obrázku. Parametrem *position* se určí souřadnice, na které se text vloží. Parametr *string* představuje řetězec znaků které budeme vkládat. Parametrem *font* se předává instance třídy *ImageFont*, která určuje písmo textu. Nepovinným parametrem *fill* lze změnit barvu písma.

Modul ImageColor

V modulu *ImageColor* jsou k dispozici pouze dvě funkce. Funkce *getrgb* vrací třímístnou n-tici. Tato n-tice obsahuje jednotlivé složky barevného modelu RGB. Vstupem funkce je barva v jednom ze tří formátů popsaných v kapitole 3.2.1. Tato funkce tedy převádí barvy na číselnou reprezentaci tří základních barev v barevném modelu RGB. Funkce *getcolor* je téměř stejná. Rozdíl je v tom, že je zde možnost určit barevný model ze kterého je vstupní barva.

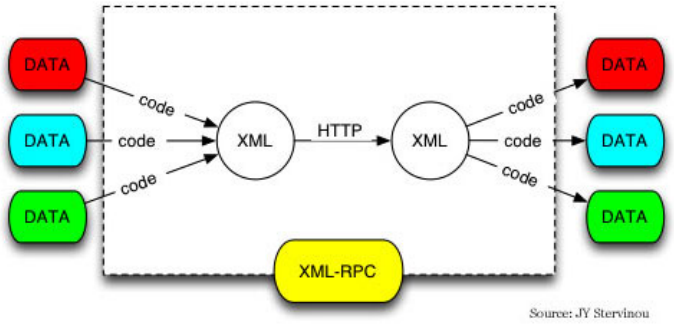
Modul ImageFont

K vytvoření instance třídy *ImageFont* se používají funkce *load* a *truetype*. Funkcí *load* se načítají písma ve formátu BDF, PCF a PIL. Této funkci je nutno jako parametr předat cestu k souboru s písmem. Funkce *truetype* je schopna načíst písmo ve formátu TTF. Mimo cesty k souboru s písmem je nutné druhým parametrem určit jaká velikost písma se bude načítat ze souboru. Knihovna PIL má své základní písmo, jehož instanci lze vytvořit funkcí *load_default*. Tato funkce nemá žádné vstupní parametry.

Vytvořená instance třídy *ImageFont* má k dispozici dvě metody. Jednou je *getsize* a druhou *getmask*. Metoda *getsize* vrací rozměry textu, který metodě předáme parametrem. Rozměry jsou vráceny ve formě n-tice, kde je uložena šířka a výška textu. Funkce *getmask* vrací předaný text ve formě bitmapy.

2.3 Protokol XML-RPC

Editor grafů je s informačním systémem propojen pomocí protokolu XML-RPC, což je protokol pro vzdálené volání procedur. Komunikace probíhá na bázi klient-server. Jinými slovy editor zasílá informačnímu systému požadavky na data. XML-RPC server informačního systému požadavky zpracuje a pošle editoru vyžádaná data z databáze. Pokud dojde na serveru k chybě, je chybová hláška a číslo chyby posláno klientovi místo dat. Obrázek 2.1 znázorňuje jakým způsobem jsou data přenášena ze serveru na klienta. Vyžádaná data



Obrázek 2.1: Přenos dat pomocí protokolu XML-RPC

jsou na straně serveru zapouzdřena pomocí značkovacího jazyka XML, a pomocí protokolu HTTP poslána klientovi. Klient poté z obdržené XML „zprávy“ přečte data. Více ohledně protokolu, včetně podrobné specifikace XML dotazů a XML odpovědí lze nalézt na [5].

Distribuční balík jakyka Python obsahuje knihovnu *xmlrpclib*, která zprostředkovává protokol XML-RPC pro tento programovací jazyk. Tato knihovna je použita pro realizaci klienta. Pro realizaci serveru je použita knihovna *Incutio XML-RPC Library*, která zprostředkovává protokol pro programovací jazyk PHP[3].

[3]XML-RPC server je součástí informačního systému, který je implementován v jazyce PHP. Více o knihovně *Incutio XML-RPC Library* lze nalézt na [7]

Kapitola 3

Vlastní práce

3.1 Návrh

3.1.1 Specifikace požadavků

Cílem této bakalářské práce je vytvořit editor grafů pro konkrétní informační systém. Data, na jejichž základě se budou vykreslovat průběhy do grafu, nebude nucen uživatel zadávat ručně, nýbrž se tyto údaje automaticky stáhnou z databáze. Uživatel si pouze vybere typ dat. Načtení vykreslovaných dat by mělo být možné také pokud nebude k dispozici připojení k síťi, resp. k databázi. Data by se v tomto případě načítala z XML souboru uloženého na pevném disku.

Vzhledem k tomu, že aplikace bude v budoucnu používána na více operačních systémech, je nutné aby kód byl přenositelný. Vhodné by bylo, aby aplikace byla spustitelná na operačním systému MS Windows XP a na operačním systému Linux.

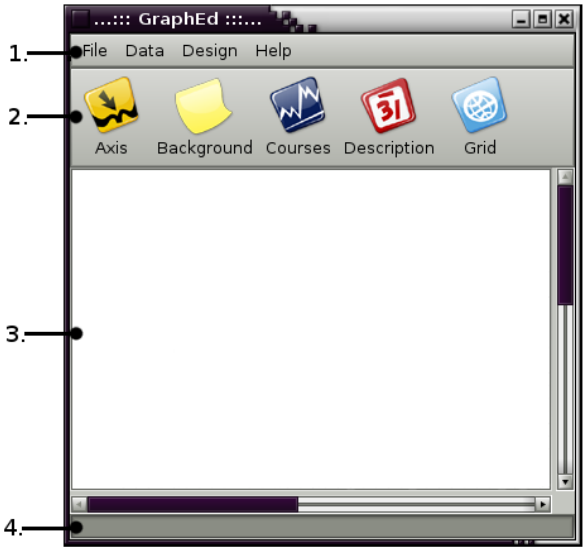
Aplikace by měla být schopna vykreslit tři typy grafů, a to spojnicový, sloupcový a koláčový. Uživatel by měl mít možnost měnit vzhled jednotlivých částí grafu. Jmenovitě se jedná o:

- **pozadí**: změna barvy pozadí
- **osy**: změna šířky a délky os
- **mřížku**: možnost vypnout zobrazení mřížky
- **vykreslované průběhy**: změna barev jednotlivých průběhů

Následně by mělo být umožněno vytvořený vzhled grafu uložit do šablony, aby se uživatel nemusel opakovaně propracovávat k požadovanému efektu. Jinými slovy by mělo být možné použít uložené vzhledové šablony pro další grafy, aby se urychlila sériová tvorba grafů.

3.1.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní aplikace (dále jen GUI) je realizováno pomocí knihovny PyGTK, což je aplikační programové rozhraní knihovny GTK+ pro jazyk Python. Knihovnu PyGTK jsem si zvolil, jelikož se mi jevila jednodušší než například PyQT či wxPython. I přesto, že je knihovna PyGTK resp. GTK+ primárně určena pro prostředí GNOME operačního systému Linux, je možné ho spustit na ostatních operačních systémech jako MS Windows nebo MAC OS X. Podrobnou dokumentaci knihovny lze nalézt na [3].



Obrázek 3.1: Rozmístění jednotlivých prvků grafického uživatelského rozhraní: 1. hlavní menu, 2. panel nástrojů, 3. náhled grafu, 4. stavová lišta

GUI aplikace je rozděleno do čtyř částí. První část je hlavní menu, druhá část je panel nástrojů, třetí část je náhled grafu a čtvrtá část stavová lišta. Na obrázku 3.1 je znázorněno rozmístění jednotlivých prvků GUI.

1. hlavní menu: Pomocí hlavního menu se ovládá celá aplikace kromě volby vzhledu grafu.
2. panel nástrojů: Pomocí panelu nástrojů se mění vzhled grafu.
3. náhled grafu: Náhled grafu zobrazuje aktuální vzhled grafu.
4. stavová lišta: Stavová lišta zobrazuje provádějící se operace.

Interaktivní části jsou pouze první dvě, tedy hlavní menu a panel nástrojů. To z toho důvodu, že není nutné, aby měl uživatel možnost měnit cokoliv v náhledu grafu. Náhled by se totiž měl automaticky obnovit, pokud se jakkoliv změní specifikace vzhledu vytvářeného grafu.

Hlavní menu

Rozdělení hlavního menu a funkce jednotlivých položek v menu nabídkách je následující:

- File
  - New graph: Pokud jsou již načteny data z databáze nebo ze souboru, vytvoří se nový graf. Pokud nejsou načteny data, je uživatel vyzván, aby jsi zvolil jestli chce data načíst z databáze či souboru.
  - Save graph: Uloží se vytvořený graf.
  - Exit: Ukončí se aplikace

- Data
  - Download: Zobrazí okno s nabídkou dat k načtení z databáze
  - Load data from file: Načte data ze souboru
  - Browse data: Zobrazení načtených dat
- Design
  - Load design: Načtení vzhledové šablony grafu a aplikování na vytvořený graf
  - Save design: Uložení vytvořené vzhledové šablony do souboru
- Help
  - HowTo: Zobrazení stručné nápovědy
  - About: Zobrazení informací o aplikaci

Panel nástrojů

Panel nástrojů, jak již je uvedeno výše, slouží k nastavení vzhledu grafu. Jednotlivá tlačítka otevírají okna se sadami nástrojů, které slouží k úpravě vzhledu jednotlivých částí grafu.

- Axis: úprava délky, šířky a barvy os
- Background: nastavení pozadí grafu
- Courses: změna barev a šířky průběhů vykreslovaných do grafu
- Description: nastavení písma a barvy popisků v grafu
- Grid: zapnutí/vypnutí mřížky

3.2 Implementace

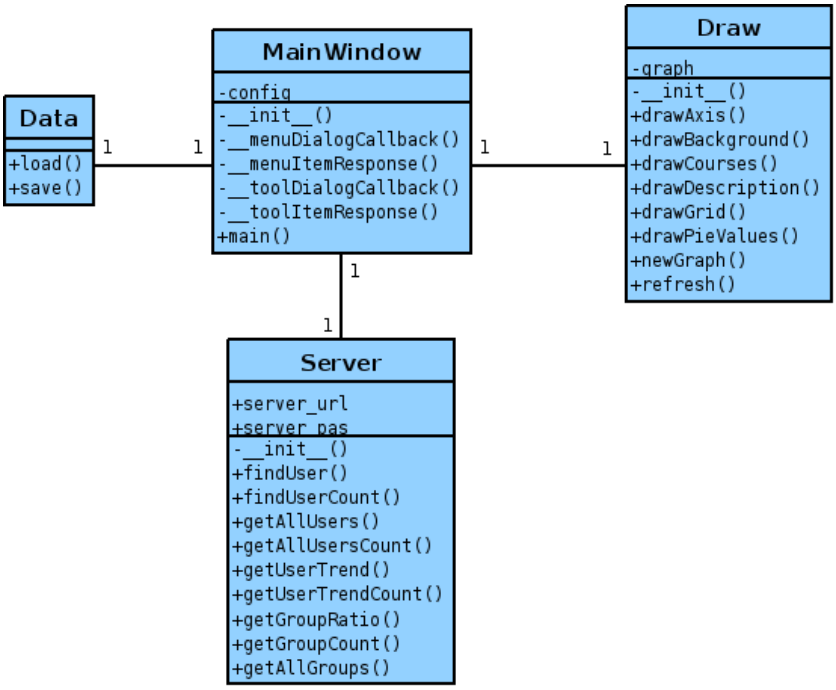
3.2.1 Struktura aplikace

Struktura aplikace byla rozdělena na čtyři hlavní části. Rozdělení aplikace je znázorněno na obrázku 3.2. Každá část má specifickou funkci.

Třída *MainWindow* je obsažena v modulu *gui*. Tato třída zprostředkovává grafické uživatelské rozhraní. Vytvořením instance se vytvoří všechny čtyři části rozhraní popsané v kapitole 3.1.2. Tato třída také zpracovává veškeré události a tím pádem zároveň řídí běh programu. Pomocí třídy MainWindow se však vytvoří pouze hlavní okno aplikace (obrázek 3.1). K vytváření ostatních částí grafického uživatelského rozhraní slouží modul *guiHelp*, který obsahuje sadu funkcí pro naplnění dialogových oken specifickými widgety [1].

Třídu *Server* nalezneme v modulu *rpcclient*. Použitím této třídy aplikace zasílá XML-RPC serveru požadavky na data z databáze. Volání jejích metod je v podstatě zrdcadlové volání metod, které nabízí XML-RPC server informačního systému. V modulu *rpcHelp* je naimplementována fukce, která tuto třídu využívá. Tato funkce slouží ke stažení konkrétních dat vybraných uživatelem. Jednotlivé hodnoty jsou následně vloženy do datové struktury slovník a pokud je hodnot více jsou jednotlivé slovníky dále vloženy do datové struktury seznam.

[1] Jednotlivé prvky grafického uživatelského rozhraní se v knihovně GTK nazývají *widgety*



Obrázek 3.2: Struktura aplikace

K uložení stáhnutých dat do XML souboru se využívá třída *Data*, jež je k nalezení v modulu *archiver*. Tato třída je jakýsi jednoduchý XML parser, určený k ukládání a načítání dat, které se budou vykreslovat do grafu. Vytvořený XML soubor má přesně danou strukturu, která je popsána v kapitole 3.2.2. Pokud nebude mít XML soubor danou strukturu, nebudou data k vykreslení načtena správně, respektive nebudou načtena vůbec.

Vykreslování celého grafu je realizováno pomocí třídy *Draw*. Tato třída obsahuje metody vykreslování jednotlivých částí nebo-li vrstev grafu. Vrstvy grafu je možné vykreslit jednotlivě pomocí metod třídy. Třída však obsahuje metodu pro vykreslení všech vrstev grafu v takovém pořadí, aby nedocházelo k překrývání těch vrstev, které se překrývat nemají. Správné vykreslování vrstev grafu je následující:

1. pozadí
2. mřížka
3. osy
4. průběhy
5. popisky

Vykreslování jednotlivých vrstev je závislé na vzhledové šabloně. Jedná se v podstatě o datovou strukturu slovník, která obsahuje nastavení vzhledu jednotlivých částí grafu, jako například tloušťka os, barva pozadí atd. Další informace o vzhledové šabloně lze nalézt v kapitole 3.2.2. Třída *Draw* a její metody jsou podrobně popsány v kapitole 3.2.3.

3.2.2 Datové struktury

V této kapitole lze nalézt popis struktury XML souboru pro uložení stáhnutých dat z databáze informačního systému. Jsou zde také popsány jednotlivé prvky ve vzhledové šabloně.

XML soubor

XML soubor se stáhnutými daty je uvozen značkou `<graph>`, ve které jsou obsaženy dvě hlavní části. První částí je hlavička a druhou částí je tělo souboru. Hlavička obsahuje informace o objemu a typu dat, které jsou uloženy v těle souboru. Data jsou v těle dokumentu ukládána jakoby do tabulky. Tedy jedním záznamem je řádek. Jednotlivé hodnoty záznamu jsou organizovány do sloupců (buněk řádku).

Hlavička je značena tagem `<head>` a obsahuje počet řádků, které jsou v těle, a počet slouců obsažených v každém řádku. Položka `<rows>` představuje počet řádků, položka `<cols>` počet sloupců a položka `<type>` typ uložených dat. Názorný příklad hlavičky je zobrazen níže:

```
<head>
   <rows>1</rows>
   <cols>2</cols>
   <type>priklad</type>
</head>
```

Tělo souboru je značeno tagem `<data>`. Tělo obsahuje jednotlivé záznamy (řádky). Záznam je označen tagem `<row>`. Každý záznam obsahuje přesně daný počet hodnot (sloupců) udaný v hlavičce. Hodnoty jsou označeny tagem `<column>` a obsahují položky `<name>` a `<value>`. Položka `<name>` určuje druh hodnoty a položka `<value>` samotnou hodnotu. Struktura těla je tedy následující:

```
<data>
   <row>
      <column>
         <name>sloupec 1</name>
         <value>hodnota x</value>
      </column>
      <column>
         <name>sloupec 2</name>
         <value>hodnota y</value>
      </column>
  </row>
</data>
```

Vzhledová šablona

Jak již bylo uvedeno v kapitole 3.2.2, je vzhledová šablona uložena v datové struktuře slovník. Každá položka tohoto slovníku představuje nastavení určité části grafu. Všechny tyto položky je možné měnit prostřednictvím widgetů dialogových oken, které se otevírají tlačítky na panelu nástrojů. Níže je uveden seznam prvků vzhledové šablony a jejich popis.

Výpis prvků je rozdělen podle toho, kterým dialogovým oknem je možné prvek změnit:

- Axis
  - centerX, centerY:
    V těchto prvcích jsou uloženy souřadnice počátku grafu v obrázku.
  - axisXpos, axisXneg:
    Tyto dva prvky představují délku osy x v kladném a záporném směru.
  - axisYpos, axisYneg:
    Tyto dva prvky představují délku osy y v kladném a záporném směru.
  - axisXresolution, axisYresolution:
    V těchto dvou prvcích je uloženo, kolik dílků bude mít každá z os. Například pokud bude *axisXresolution* rovno desíti, bude na ose x v kladném směru deset dílků.
  - axisColor:
    Pomocí tohoto prvku se nastaví barva os. Barva musí být v řetězcovém formátu `"#rrggbb"`, kde `"rr"` je hodnota červené barvy v hexadecimálním tvaru, `"gg"` hodnota zelené barvy v hexadecimálním tvaru a `"bb"` hodnota modré barvy v hexadecimálním tvaru.
  - axisWidth:
    Tento prvek určuje šířku os v pixelech.
- Background
  - background:
    Tímto prvkem se nastavuje pozadí grafu. Vzhled pozadí grafu může být trojí. Buďto je možné vyplnit pozadí grafu pouze barvou. V tomto případě bude mít prvek formát `"#rrggbb"` (například `"#00FF00"`). Dále je možné do pozadí grafu vložit libovolný rastrový obrázek, který načteme ze souboru. Poté bude mít prvek formát `"@cesta_k_souboru"` (například `"@/home/pozadi.jpg"`). Třetí možností je, že na pozadí vytvoříme vertikální barevný přechod. Formát v tomto případě vypadá následovně: `"$gradient#rrggbb#rrggbb$x"`. Jednotlivé podřetězce `"#rrggbb"` určují barvy mezi kterými se vytvoří barevný přechod. V části `"x"` je určeno kolik pixelů se použije pro jednotlivé barvy v barevném přechodu.
- Courses
  - courseColor:
    Prvek courseColor obsahuje seznam barev ve formátu `"#rrggbb"`. Barvy jsou použity při vykreslování průběhů. Průběhy jsou odlišeny barvami uloženými v tomto seznamu.
  - courseWidth:
    Hodnotou uloženou v tomto prvku se určuje šířka křivek průběhů u spojnicového grafu nebo šířka sloupců u sloupcového grafu. Při vykreslování koláčového grafu nemá tento prvek žádný vliv na vzhled grafu.

- Description
  - headline:
    Obsahuje text který bude vykreslen jako nadpis grafu
  - headlineColor:
    Určuje barvu textu uvedeného v prvku *headline*. Barva musí být ve formátu `"#rrggbb"`.
  - headlineFont:
    Určuje písmo textu uvedeného v prvku *headline*. V prvku je obsažena n-tice, která je složena z názvu požadovaného písma a jeho velikosti.
  - headlineXposition, headlineYposition:
    V těchto prvcích jsou uloženy souřadnice nadpisu grafu.
  - descriptX, descriptY:
    Tyto prvky obsahují popis os. Prvek *descriptX* popisuje osu x a prvek *descriptY* popisuje osu y.
  - descriptXposition:
    Tímto prvkem se určí, zda popis osy x bude nad osou (`"above"`) nebo pod osou (`"below"`)
  - descriptYposition:
    Tento prvek určuje, zda popis osy y bude vlevo od osy (`"left"`) nebo vpravo od osy (`"right"`)
  - descriptColor:
    Určuje barvu popisků os. Barva musí být ve formátu `"#rrggbb"`.
  - desriptFont:
    V prvku desriptFont je uloženo písmo popisků os.
- Grid
  - gridEnable:
    Prvek gridEnable obsahuje n-tici, která je složena ze dvou řetězců. Oba řetězce mají formát `"xxxx"`, kde za `"x"` je možno dosadit `"1"` nebo `"0"`. První řetězec slouží k nastavení zobrazení mřížky osy x. Druhý řetězec slouží k nastavení zobrazení mřížky osy y. Princip je takový, že jednotlivé znaky v řetězci slouží k zapnutí (`"1"`) nebo vypnutí (`"0"`) mřížky v konkrétním kvadrantu pro konkrétní osu. Například použitím n-tice `("0011","1100")` způsobíme to, že v prvním a druhém kvadrantu grafu bude pouze horizontální mřížkování. Naopak ve třetím a čtvrtém bude pouze vertikální mřížkování.
  - gridColor:
    Určuje barvu popisků mřížky. Barva musí být ve formátu `"#rrggbb"`
  - gridWidth:
    Tento prvek určuje šířku čar mřížky v pixelech.

Věškeré nastavení vzhledu je tedy uloženo v jedné datové struktuře. Knihovna *cPickle*, která je součástí distribučního balíku jazyka Python, nabízí uložení a zpětné načtení jakékoliv datové struktury do/ze souboru. Právě tímto způsobem je realizováno ukládání a načítání vzhledové šablony.

3.2.3 Třída Draw modulu painter

Jak již bylo uvedeno v předchozích kapitolách, třída Draw zprostředkovává vykreslování grafu. Jejími metodami lze vykreslit jak jednotlivé vrstvy grafu, tak celý graf najednou. Při vytváření instance je nutné konstruktoru předat instanci třídy *Image*, do které chceme graf vykreslovat.

```
import Image, painter
obrazek = Image.new('RGB', (800, 600))
graf = painter.Draw(obrazek)
```

Vzhledová šablona se předává až jednotlivým volaným metodám. To z toho důvodu, aby se po každé změně vzhledové šablony nemusela vytvářet nová instance třídy Draw. Objekt *graf* z předchozího příkladu po vytvoření mění svými metody objekt *obrazek*, tedy vykresluje do něj graf. Metody poskytované instancí třídy Draw jsou následující:

- newGraph(design,data,dataType)
- refresh(design,data,dataType)
- drawAxis(design)
- drawBackground(design)
- drawCourses(design,data,dataType)
- drawDescription(design)
- drawGrid(design)

Všem metodám je nutné předat vzhledovou šablonu pomocí parametru *design*. Parametrem *data* se některým metodám předávají hodnoty, které budou vykresleny do grafu. Parametrem *dataType* se určí typ hodnot předávaných parametrem *data* (tedy typ stáhnutých dat z XML-RPC serveru).

Metoda newGraph

Tato metoda slouží k inicializaci vzhledové šablony. Vzhledová šablona je vyplněna počátečními hodnotami. Kupříkladu se nastaví počátek grafu doprostřed obrázku, zapne se zobrazení mřížkování (vertikálního i horizonálního) ve všech kvadrantech grafu, nastaví se délky os až téměř po okraje obrázku, atd. Následně se zavolá metoda *refresh*, jejíž funkce je popsána níže.

Metoda drawAxis

Metoda *drawAxis* slouží k vykreslování os grafu. Osy jsou vykreslovány jednotlivě od počátku grafu. Šířka os se určí podle prvku *axisWidth* vzhledové šablony.

První se vykreslí osa x v pozitivním směru. Její délka se určí z prvku *axisXpos*. Následně se vertikálními čárkami osa rozdělí na počet dílků, který je uveden v prvku *axisXresolution*. Jako další se vykreslí osa x v záporném směru stejným způsobem, jako osa x v kladném směru. Je zde však jeden rozdíl. Dílky na ose v záporném směru mají stejnou šířku jako v kladném směru. Pokud je osa v záporném směru kratší než osa v kladném směru, rozdělí se osa v záporném směru na menší počet dílků než udává prvek *axisXresolution*. Analogicky jsou vykresleny oba směry osy y.

Metoda drawBackground

Tato metoda vykresluje pozadí grafu. V závislosti na formátu řetězce uloženého v prvku *background*[2] se vykreslí jeden z následujících typů pozadí:

- **barevná výplň**: Do obrázku se vykreslí čtverec se stejnými rozměry jako má samotný obrázek. Barva výplně čtverce je určena prvkem *background*.
- **obrázek**: Načte se obrázek, jehož název a cesta k němu jsou uvedeny v prvku *background*. Načtenému obrázku se poté změní rozměry tak, aby byly shodné s rozměry obrázku s grafem. Při změně rozměrů se použije filtr *BILINEAR*. Nakonec se obrázek s pozadím vloží do obrázku s grafem.
- **barevný přechod**: Z prvku *background* se načtou barvy, mezi kterými se má barevný přechod provést. Také se načte hodnota určující kolik pixelů se použije pro jednotlivé barvy v barevném přechodu. Poté se obě barvy rozloží na jednotlivé RGB složky. Pro každou RGB složku se určí hodnota přídavku, který bude postupně přičítán k barevné složce první barvy. Hodnota se vypočítá podle následujícího vzorce:

  ```
  přídavek = ( složka_první_barvy - složka_druhé_barvy ) / výška_obrázku
  ```

  Poté se postupně po jednom pixelu vykreslují z vrchu obrázku horizontální čáry po celé šířce obrázku. Na vrchu je barva čáry stejná jako první určená barva v prvku *background*. Postupně se ke složkám přičítají jednotlivé přídavky. Přídavky se přičtou až tehdy, pokud se již vykreslil daný počet čar se stejnou barvou[3].

Metoda drawCourses

Tato metoda se stará o vykreslení hodnot do grafu. Postup vynášení hodnot grafu závisí na typu grafu, který je určen prvkem *type* vzhledové šablony. Graf může být trojího typu:

- **spojnicový**: Údaje z databáze jsou do grafu vynášeny jako body, každé dva po sobě jdoucí body jsou spojeny čarou. jejíž šířka se určí z prvku *courseWidth* a barva z prvku *courseColor*. V případě, že hodnoty grafu jsou složeny z více průběhů, je barva na začátku vykreslování každého průběhu změněna. Pokud ovšem bude v prvku *courseColor* uložena pouze jedna barva, všechny průběhy budou vykresleny touto barvou.
- **sloupcový**: Údaje jsou do grafu vynášeny jako výšky obdelníků. Šířka obdelníků je závislá na prvku *courseWidth* a barva výplně na prvku *courseColor*. Stejně jako u spojnicového grafu je při více průbězích barva výplně měněna podle barev uložených v tomto prvku. Jelikož jsou sloupce jednotlivých průběhů vykreslovány na sebe, vykreslují se postupně od největšího po nejmenší, aby žádný sloupec nebyl překrytý jiným.
- **koláčový**: Jednotlivé údaje v tomto grafu představují vyplněné výseče elipsy. Výseče jsou vykreslovány za sebou a tím vytvoří celý „koláč“. Ten je poté vykreslen několikrát nad sebe, čímž se docílí 3D efektu. Šířka elipsy tvořící koláč je závislá na prvku *axisXresolution* a výška na prvku *axisYresolution*.

[2] Jednotlivé formáty řetězce jsou popsány v kapitole 3.2.2.
[3] počet čar je daný právě hodnotou "x" v *background* (viz 3.2.2)

Pokud data, která jsou vykreslována do grafu, obsahují více průběhů, jsou popisky jednotlivých průběhů ukládány do prvku *courseDesc* vzhledové šablony. Zároveň je kromě popisku uložena také barva průběhu a u vykreslování spojnicového grafu také souřadnice poslední hodnoty průběhu.

Metoda drawDescription

Tato metoda slouží k vykreslování veškerého textu, který je nutno do grafu vykreslit. Nejdříve se načte písmo nadpisu a písmo popisků os. Písmo popisků os je použito také jako písmo popisků průběhů. Bohužel PyGTK widget na výběr písma vrací pouze jméno a velikost zvoleného písma. K načtení písma do PIL je nutné znát název a cestu k souboru s písmem. V aplikaci se soubor s písmem vyhledává podle jména písma. Pokud je soubor s písmem pojmenován jinak než je název písma, požadované písmo se nenalezne a místo něho se načte základní písmo.

Po načtení písem se vykreslí hodnoty dílků na osách. Hodnoty dílků osy x jsou vykresleny vždy pod osou. Hodnoty dílků osy y jsou vyresleny vždy vlevo od osy. Pokud má graf více jak jeden průběh, tak se postupně vykreslí popisky průběhů, které jsou uloženy v prvku *courseDesc*. Při vykreslování spojnicového grafu se souřadnice popisku načítají z prvku *courseDesc*. Při vykreslování koláčového a sloucového grafu se popisky průběhů vykreslují pod sebe do šedého okna v pravém horním rohu obrázku. Následně se vykreslí nadpis grafu. Text nadpisu je uložen v prvku *headline* a souřadnice nadpisu se určí z prvků *headlineXposition* a *headlineYposition*. Poté se vykreslí popisky os. Popisek osy se vykreslí ke konci delšího směru osy. Tedy pokud je záporný směr osy delší než kladný směr osy, vykreslí se popisek ke konci záporného směru. Ovlivnit lze pouze na jaké straně osy bude popisek vykreslen[4].

Metoda drawGrid

Touto metodou se do obrázku vykresluje mřížka grafu. Vykreslení mřížky je možné povolit či zakázat v jednotlivých kvadrantech. Je možné zároveň zvolit možnost vykreslení pouze vertikálních či horizontálních čar mřížky. Jednotlivé čáry mřížky se vykreslují postupně podle kvadrantů. Nejprve se vykreslí vertikální mřížkování a poté horizontální mřížkování.

Nastavení vykreslování mřížky se určuje pomocí prvku *gridEnable* ve vzhledové šabloně a jeho přesný formát je popsán v kapitole 3.2.2. Mezery mezi jednotlivými čarami mřížky v horizontálním i vertikálním směru jsou závislé na počtu dílků na ose. Jinými slovy mřížkování má stejné rozlišení jako rozdělení os. Šířka čar mřížky se určuje z prvku *gridWidth* vzhledové šablony. Barva čar mřížky se určí z prvku *gridColor*

Metoda refresh

Metoda *refresh* volá postupně všechny metody vykreslující jednotlivé vrsty grafu. Metody jsou volány v tomto pořadí: *drawBackground*, *drawGrid*, *drawAxis*, *drawCourses*, *drawDescription*

[4] viz prvky *descriptXposition* a *descriptYposition* v kapitole 3.2.2

Kapitola 4

Závěr

Při realizaci této bakalářské práce jsem se naučil programovat v jazyce Python. Díky tomuto programovacímu jazyku jsem se také zlepšil v objektově orientovaném programování. Také jsem se při návrhu propojení editoru s informačním systémem naučil vytvářet spojení server-klient na základě protokolu XML-RPC.

Díky této práci jsem se seznámil s knihovnou Python Imaging Library, která je ve většině případů používána pro úpravu rastrových obrázků. Tento projekt je ukázkou toho, že knihovna je užitečným nástrojem nejen pro úpravu ale i tvorbu rastrových obrázků. Nabízí sadu flexibilních funkcí pro vykreslování grafických primitiv. Tyto funkce mi výrazně ulehčily realizaci vykreslování grafů.

Výsledkem mé práce je funkční editor grafů, který je schopen si automaticky stáhnout do grafu vynášená data z databáze informačního systému. Editor byl otestován na operačním systému MS Windows a Linux. Vzhled grafu je pomocí editoru poměrně slušně nastavitelný. Není sice možné nastavit vzhled veškerých částí grafu, to ale bude v další verzi programu umožněno. Jelikož se ale změnili požadavky na funkci informačního systému a obsahu databáze, bude zároveň nutné upravit implementaci XML-RPC serveru a klienta.

Literatura

[1] Mark Lutz a David Ascher. *Naučte se Python.* Grada Publishing a.s., Praha, 2003. ISBN 80-247-0367-X.

[2] Daryl Harms a Kenneth McDonald. *Začínáme programovat v jazyce Python.* Computer press, Brno, 2006. ISBN 80-7226-799-X.

[3] The GNOME Project and PyGTK Team. Pygtk 2.0 reference manual. `http://www.pygtk.org/docs/pygtk`.

[4] PythonWare. Python imaging library handbook. `http://www.pythonware.com/library/pil/handbook`.

[5] UserLand Software. Xml-rpc specification. `http://www.xmlrpc.com/spec`.

[6] Guido van Rossum. Python library reference. `http://docs.python.org/lib/lib.html`.

[7] Simon Willison. Incutio xml-rpc library for php. `http://scripts.incutio.com/xmlrpc/`.

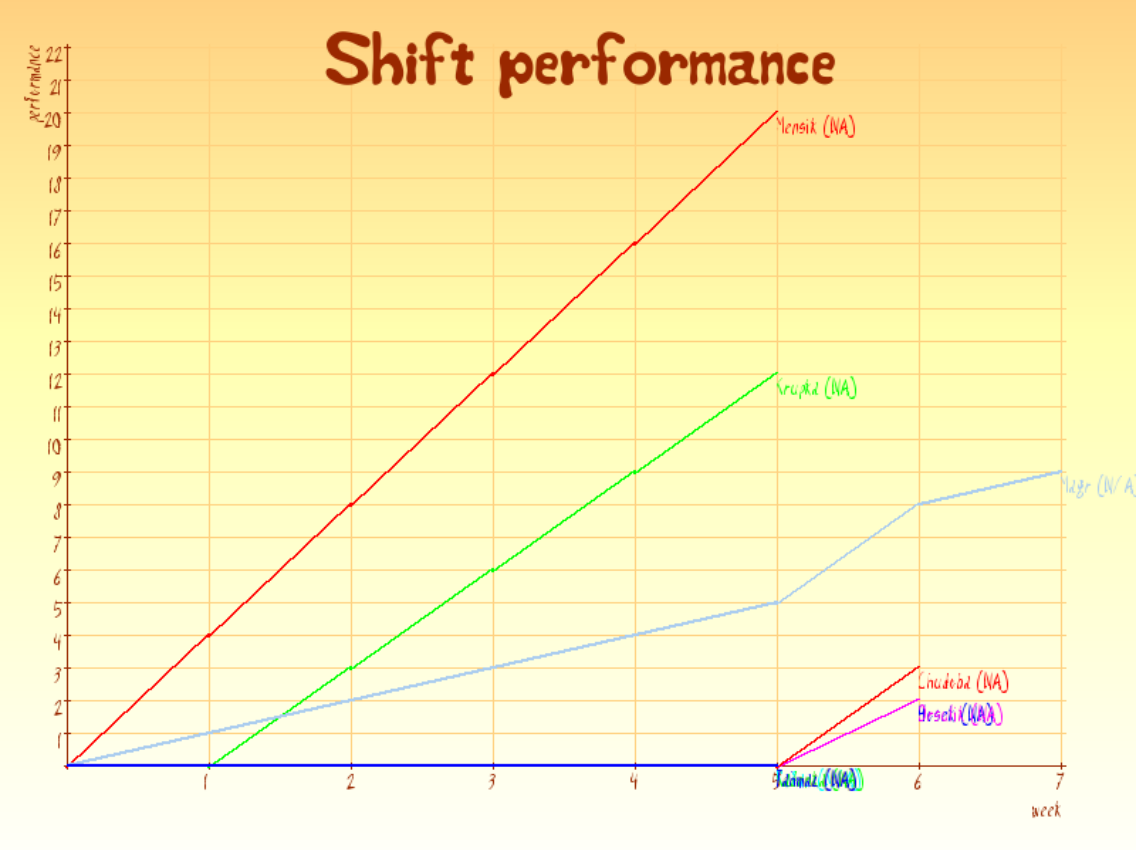
Seznam příloh

A Ukázky grafů

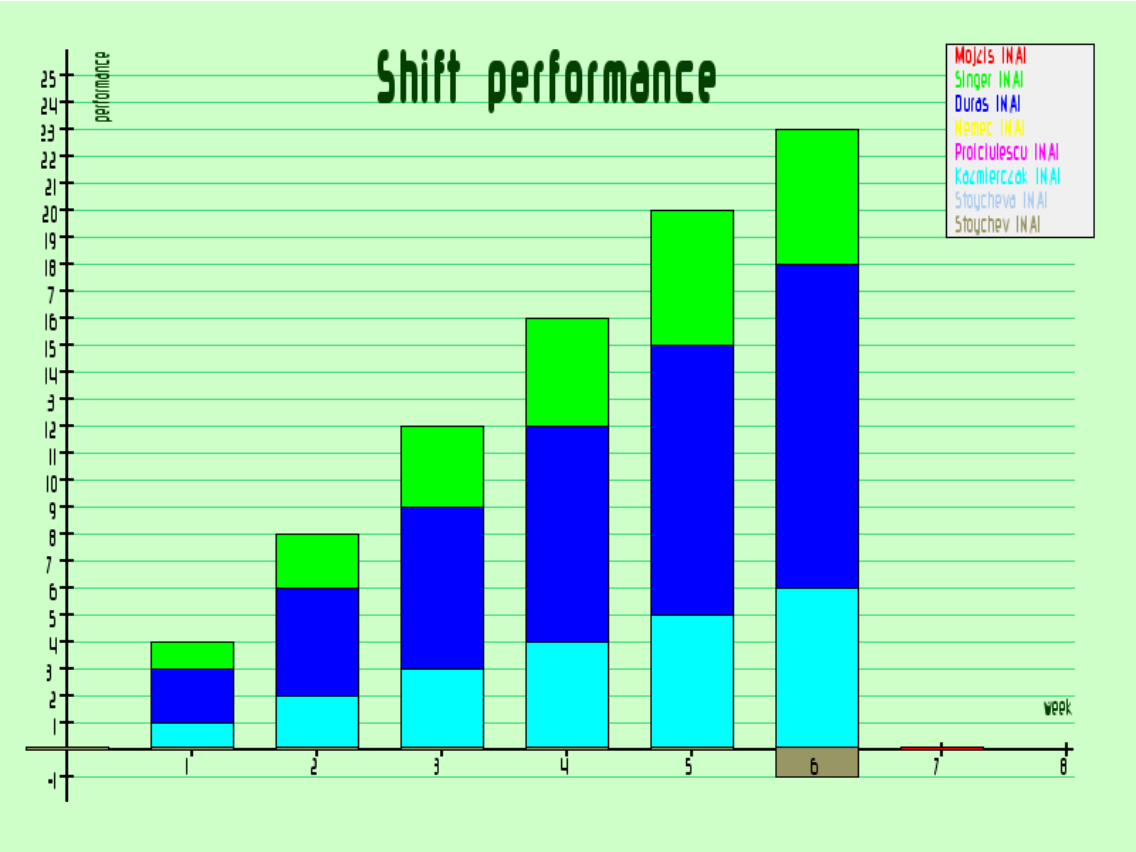
B Datový nosič CD s kompletní implementací včetně XML-RPC serveru

Příloha A - Ukázky grafů

Následující grafy byly vytvořeny pomocí editoru grafů, který lze nalézt v příloze B.



Obrázek 4.1: Spojnicový graf



Obrázek 4.2: Sloupcový graf