Demonstrační a dokumentační aplikace pro SW knihovnu wxWidgets

Demonstration and documentation application for the wxWidgets software library

Bc. Ján Chudý

Diplomová práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ján CHUDÝ**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Demonstrační a dokumentační aplikace pro SW knihovnu wxWidgets**

Zásady pro vypracování:

1. **S využitím programovacího jazyka ANSI C++ a SW knihovny wxWidgets vytvořte demonstrační a dokumentační aplikaci knihovny wxWidgets.**
2. **Aplikace bude provázána s HTML nápovědou knihovny wxWidgets a bude umožňovat efektní a intuitivní prohlížení komentovaných zdrojových kódů vzorových projektů dodávaných společně s knihovnou, jejich spouštění a prohlížení referenčních materiálu knihovny wxWidgets.**
3. **Aplikace bude připravena na nový formát nápovědy knihovny wxWidgets 2.9.0 a vyšší založené na HTML kódu generovaném nástrojem Doxygen.**
4. **Inspirujte se demonstračními programy softwarových knihoven Qt a wxPython.**
5. **Snažte se o co možná nejvyšší možnou univerzálnost a konfigurovatelnost aplikace pomocí externích konfiguračních souborů a skriptů.**
6. **Výsledná aplikace musí být spustitelná v prostředích MS Windows, Linux a případně také MacOS.**

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **BLIŽŇÁK, Michal. Systémové programování. 1. vyd. Zlín: UTB ve Zlíně, 2005. 202 s. ISBN 80-7318-364-1.**
2. **HARMS, Daryl. Začínáme programovat v jazyce Python. 1. vyd. Praha: Computer Press, 2003. 456 s. ISBN 80-7226-799-X.**
3. **LUTZ, Mark. Naučte se Python. 1. vyd. Praha: Grada Publishing, 2003. 339 s. ISBN 80-247-0367-X.**
4. **MASTERS, Jon. Linux profesionálně: programování aplikací. 1.vyd. Brno: Zoner Press, 2008. 539 s. ISBN 978-80-86815-71-8.**
5. **SMART, Julian, HOCK, Kevin. Cross-Platform GUI Programming with wxWidgets, Prentice Hall, 2006, ISBN 0-13-147381-6.**
6. **Dokumentace knihovny wxWidgets na WWW (http://docs.wxwidgets.org/stable/).**
7. **Webové stránky knihovny Qt (http://www.qtsoftware.com/products).**
8. **Webové stránky knihovny wxPython (http://www.wxpython.org/).**

Vedoucí diplomové práce: **Ing. Michal Bližňák, Ph.D.**
Ústav aplikované informatiky

Datum zadání diplomové práce: **20. února 2009**

Termín odevzdání diplomové práce: **27. května 2009**

Ve Zlíně dne 13. února 2009

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Cieľom tejto práce je vytvorenie demonštračnej a dokumentačnej aplikácie pre softwarovú knižnicu wxWidgets. Teoretická časť práce je zameraná na popis samotnej knižnice – od stručnej histórie jej vývoja, cez vlastnosti a výhody oproti iným podobne zameraným knižniciam až po popis dostupných vývojových nástrojov. Praktická časť sa venuje popisu aplikácie wxDemoViewer, ktorá vznikla ako súčasť tejto práce. Aplikácia ľahkým a zrozumiteľným spôsobom prezentuje vlastnosti aplikácií vytvorených pomocou knižnice wxWidgets. Obsahuje veľké množstvo vzorových aplikácií, ktoré môže užívateľ voľne spúšťať a prezerať ich zdrojové kódy spolu s kompletnou dokumentáciou knižnice a umožňuje tak začínajúcim aj pokročilým programátorom ľahšie pochopenie práce s touto knižnicou. Aplikácia je veľmi univerzálna – jej vzhľad, ako aj zobrazovaný obsah je možné zmeniť pomocou konfiguračných súborov – a môže tak byť použitá na prezentáciu mnohých ďalších programovacích jazykov a knižníc.

Kľúčové slová: wxWidgets, demo, C++, skin

ABSTRACT

The aim of this work is to develop a demonstration and documentation application for the wxWidgets software library. The theoretical part is aimed to a description of the Library – from a brief history of its development, through the features and advantages over other similar libraries, to description of available development tools. The practical part describes the demo application called wxDemoViewer, which was created as a part of this work. The application clearly presents characteristics of applications created using the wxWidgets library. It contains a large number of sample applications that the user can freely run and study their source code, along with complete documentation of the library. This approach allows both beginners and advanced programmers better understand the work with this library. The demo application is strongly customizable – a look and feel of the application, as well as a displayed content can be modified using several configuration files – and thus can be used for a presentation of many other programming languages and libraries.

Keywords: wxWidgets, demo, C++, skin

Na tomto mieste by som rád poďakoval vedúcemu mojej bakalárskej práce, pánovi Ing. Michalovi Bližňákovi, Ph.D., za inšpiráciu, cenné rady a čas venovaný mojej práci.

Ďalej by som chcel poďakovať mojej rodine, vďaka ktorej mi bolo umožnené študovať na Univerzite Tomáše Bati ve Zlíně.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD

Spolu s rastúcim počtom rôznych operačných systémov začala existovať požiadavka na vývoj takzvaných multi-platformných aplikácií – aplikácií ktorých beh je možný na viacerých softwarových platformách. Vývojári samozrejme môžu vytvárať aplikácie pomocou natívnych nástrojov cieľového operačného systému, no tento vývoj je väčšinou oveľa náročnejší z dôvodu udržovania viacerých verzií. Začali tak vznikať rôzne softwarové knižnice, ktoré sa snažia vývoj multi-platformných aplikácií čo najviac uľahčiť. Jednou z týchto knižníc je aj knižnica wxWidgets, ktorej sa venuje teoretická časť tejto práce.

Teoretická časť odpovedá na základné otázky čím sú a čím nie sú wxWidgets, stručne popisuje históriu ich vývoja a vyzdvihuje ich výhody oproti iným podobne zameraným knižniciam. Jedna z častí sa potom venuje praktickému popisu vytvárania jednoduchej vzorovej aplikácie. V závere teoretickej časti sú popísané niektoré z dostupných vývojových nástrojov pre túto knižnicu.

Praktická časť sa venuje popisu aplikácie wxDemoViewer, ktorá vznikla ako súčasť tejto práce. Cieľom aplikácie je ľahkým a zrozumiteľným spôsobom prezentovať vlastnosti aplikácií vytvorených pomocou knižnice wxWidgets a na základe veľkého množstva vzorových programov ktoré môže užívateľ voľne spúšťať a prezerať ich zdrojové kódy priblížiť začínajúcim aj pokročilým programátorom princípy tvorby aplikácií pomocou knižnice. Aplikácia sa snaží o čo najväčšiu univerzálnosť – jej vzhľad, ako aj zobrazovaný obsah je možné zmeniť pomocou konfiguračných súborov čo umožňuje jej použitie s mnohými inými programovacími jazykmi a knižnicami. Praktická časť tiež popisuje štruktúru aplikácie z programátorského hľadiska a základné ovládanie aplikácie. V záverečnej časti potom na praktickom príklade popisuje celý postup tvorby jednoduchého demo súboru – súboru ktorý je aplikáciou používaný na popis jej vzhľadu a obsahu.

I. TEORETICKÁ ČASŤ

1 WXWIDGETS

Slovo „widget“ pochádza z doby, keď sa vo svete začali objavovať prvé nástroje na tvorbu užívateľského rozhrania (Xt, GTK+, Qt…) a popisuje objekt, ovládací prvok, ktorý môže byť zobrazený v užívateľskom rozhraní ako napríklad okná, tlačítka, posuvníky, text boxy atď., no môže sa tiež jednať o prvky ktoré nemusia byť pre užívateľa viditeľné. [3]

1.1 Čo sú wxWidgets

Aplikácie typicky zobrazujú okná obsahujúce štandardné ovládacie prvky, prípadne vykresľujú obrázky a grafiku. Tieto prvky reagujú na rôzne vstupy akými sú napríklad myš alebo klávesnica. Môžu taktiež komunikovať s inými procesmi, alebo ovládať iné aplikácie. wxWidgets poskytujú programátorovi nástroje, pomocou ktorých môže relatívne jednoducho vytvoriť aplikáciu, ktorá bude obsahovať všetky tieto obvyklé prvky. [6]

wxWidgets predstavuje sadu softwarových knižníc vytvorených pre programovací jazyk C++, ktoré dovoľujú kompiláciu programov na niekoľkých rôznych softwarových platformách a prekladačoch s minimálnymi nutnými zmenami zdrojového kódu. Pre každú podporovanú platformu (Windows, GTK+, Motif, Mac OS, FreeBSD, Windows CE…) existuje samostatná verzia tejto knižnice. Pôvodným cieľom vývojárov bolo, aby aplikácie vytvorené pomocou tejto knižnice boli nerozpoznateľné od aplikácií vytvorených pomocou natívnych nástrojov daného operačného systému.

Okrem základných API funkcií pre tvorbu grafického rozhrania (GUI) poskytuje knižnica wxWidgets naviac ďalšie prvky vhodné pre prístup k ostatným technológiám (funkcionalite) cieľového operačného systému (začínajúc súborovými operáciami a končiac riadením procesov a vláken aplikácie). Knižnica naviac obsahuje ďalšie triedy, ktoré nemalou mierou zjednodušujú tvorbu programu a implementáciu často používaných technológií a algoritmov (napr. sieťové operácie, XML, 2D a 3D grafika, užívateľsky konfigurovateľné grafické rozhranie, zabudovaná podpora zobrazovania HTML a pod).

Je ale potreba si uvedomiť, že wxWidgets nie sú prekladačom natívneho zdrojového kódu z jednej platformy do druhej (nedá sa napríklad zobrať zdrojový kód ľubovoľného programu pre GTK+ a preložiť ho pre MS Windows) – užívateľ sa musí naučiť nové API. Toto API je však natoľko jednoduché a intuitívne že nie je väčším problémom ho so základnými znalosťami programovania pomocou jazyka C++ a so znalosťou princípu

vytvárania aplikácií pre správami riadené operačné systémy (napr. MS Windows) vo veľmi krátkej dobe zvládnuť.

Knižnica wxWidgets je vyvíjaná už viac ako 12 rokov a je šírená ako open source projekt. Existuje preto veľké množstvo dokumentácie, vývojových nástrojov a rozširujúcich tried ktoré je možné voľne získať a zdarma používať. V súčasnosti je možné knižnicu používať nielen v kombinácii s jazykom C++, ale aj s inými bežne dostupnými programovacími jazykmi akými sú napríklad Python (wxPython), C#, Perl (wxPerl), Ruby (wxRuby), Java ale tiež JavaScript (wxJS). [1]

1.2 História wxWidgets

Vývoj wxWidgets sa začal v roku 1992 na Inštitúte aplikácie umelej inteligencie na Univerzite v Edinburgu kde programátor Julian Smart pracoval na nástroji na vytváranie grafov s názvom Hardy. Jeho úlohou bolo vytvoriť aplikáciu, ktorá by mohla bežať pod operačným systémom Windows ako aj pod operačnými systémami založenými na Unixe. Rozhodol sa preto pre použitie multi-platformnej knižnice. Pretože množstvo multi-platformných nástrojov v tej dobe bolo veľmi obmedzené a taktiež pretože jeho ústav nemal na použitie týchto nástrojov dostatočný rozpočet, rozhodol sa pre vytvorenie knižnice vlastnej. V septembri roku 1992 mu Univerzita umožnila uložiť knižnicu wxWidgets (vtedy ešte pod názvom wxWindows) verzie 1.0 na FTP server ústavu. Spočiatku knižnica podporovala iba operačné systémy MS Windows (použitie knižnice MFC) a Linux (XView). Neskôr pre sťažnosti užívateľov prekladača jazyka C++ spoločnosti Borland na potrebu použitia MFC, bola knižnica prepísaná s použitím čistého Win32 API. Taktiež z dôvodu nahradenia XView nástrojmi Motif bol čoskoro vytvorený Motif port wxWidgets. [6] [3, str. 331]

S postupom času sa okolo wxWidgets vytvorila malá komunita nadšencov a došlo k vytvoreniu emailovej konferencie. Užívatelia začali zasielať svoje príspevky a opravy. O wxWidgets sa začalo zaujímať stále viac a viac ľudí z celého sveta – jednotlivci, vládne oddelenia ale hlavne užívatelia z korporátnej sféry, ktorý zistili že wxWidgets im ponúka viac ako ostatné komerčne dostupné produkty.

V roku 1997 bolo s pomocou Markusa Holzema vytvorené nové wxWidgets 2 aplikačné rozhranie. Wolfram Gloger navrhol, aby boli wxWidgets portované na GTK+ – nádejnú

sadu widgetov upravenú pre desktopové prostredie GNOME. Vznikol tak port wxGTK, ktorý je doteraz hlavným portom wxWidgets pre operačný systém Unix / Linux.

V roku 1998 boli zlúčené porty wxWidgets pre Windows a GTK+ a umiestnené pod kontrolu CVS. K projektu sa pripojil Vadim Zeitlin, ktorý k prispel veľkým množstvom kódu a Stefan Csomor začal pracovať na porte pre operačný systém Mac OS.

V roku 1999 pridáva Václav Slavík triedy pre podporu HTML (wxHTML) a prehliadač nápovedy podporujúci tento formát. V roku 2000 sponzorovala spoločnosť SciTech Inc. počiatočný vývoj portu wxUniversal – súbor widgetov pre platformy, ktoré vlastné widgety vôbec neobsahujú. V roku 2002 potom Julian Smart a Robert Roebling vytvorili port wxX11 len s použitím tried wxUniversal. Tento port vyžaduje ľubovoľný na Unixe založený operačný systém a X11 vďaka čomu je vhodný pre akýkoľvek systém vrátane systémov s obmedzenými zdrojmi.

V roku 2003 boli vytvorené porty wxWidgets pre mobilné zariadenia – port pre Windows CE a Robert Roebling predviedol wxGTK aplikácie bežiace na GPE mobilnom Linuxe.

Po sťažnosti spoločnosti Microsoft v roku 2004 ohľadne podobnosti doterajšieho názvu wxWidgets – wxWindows s ich obchodnou značkou bol celý projekt premenovaný na wxWidgets. V priebehu roku 2004 taktiež Stefan Csomor spoločne s ostatnými prispievateľmi kompletne prepracovali port wxMac pre Mac OS X. Výsledkom tak bolo výrazné vylepšenie vzhľadu a funkcionality aplikácií bežiacich pod týmto operačným systémom. V apríli 2005 potom vychádza verzia 2.6, ktorá významným spôsobom vylepšuje všetky porty. V súčasnosti je poslednou stabilnou verziou verzia označená číslom 2.8.10. [6] [7]

Plány do budúcnosti:

- Nástroj pre správu balíčkov pre jednoduchšiu integráciu komponent tretích strán.
- Alternatívny mechanizmus ošetrenia udalostí.
- wxHTML 2 s plnou podporou súčasných možností webu na všetkých platformách.
- Vylepšenie kompatibility so štandardami ako napríklad STL.

1.3 Prečo používať práve wxWidgets?

Aj keď v dnešnej dobe existuje veľké množstvo nástrojov na tvorbu aplikácií s grafickým užívateľským rozhraním (MFC, .NET), wxWidgets sa od nich odlišujú hlavne tým, že sú

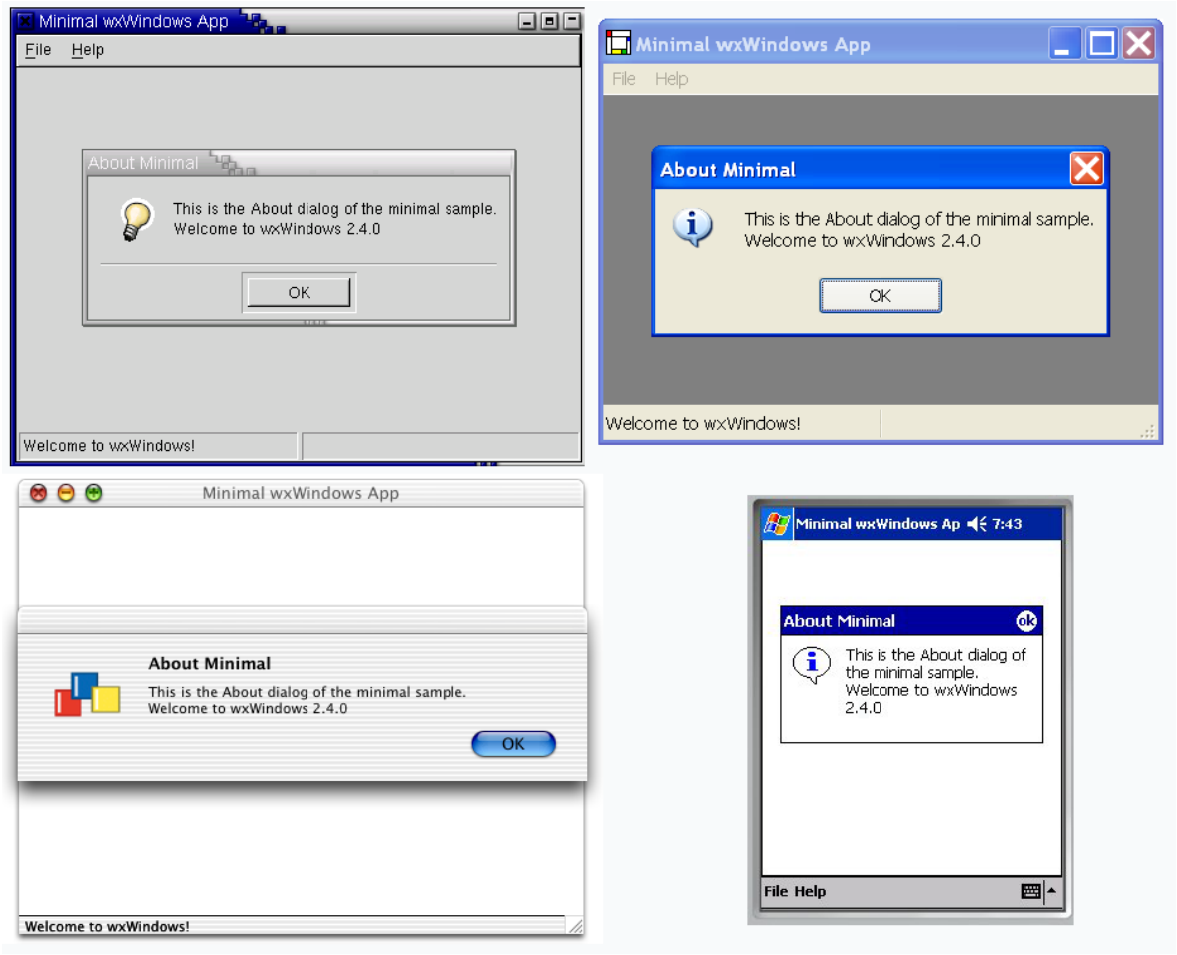
primárne multi-platformné. Aplikačné programové rozhranie (API) je na všetkých platformách rovnaké, alebo aspoň veľmi podobné. Znamená to, že môžeme napríklad napísať aplikáciu pod operačným systémom Windows a následne môžeme tento istý zdrojový kód (s prípadnými minimálnymi zmenami) preložiť na operačnom systéme Linux alebo Mac OS. Nemusíme tak zdĺhavo prepisovať zdrojový kód aplikácie pre každú platformu zvlášť. Takisto nie je potrebné sa pre každú platformu učiť jej vlastné natívne API. Súčasne s vývojom operačných systémov sa vyvíja aj knižnica wxWidgets, odpadá tak riziko že s príchodom novej verzie operačného systému nebude naďalej náš program spustiteľný.

Ďalšou skvelou vlastnosťou wxWidgets, ktorá ich odlišuje od ostaných knižníc je natívny vzhľad všetkých ovládacích prvkov. Niektoré knižnice používajú pre ovládacie prvky rovnaký programový kód na všetkých platformách, prípadne sú prvky naskinované tak, aby vyzerali natívne. wxWidgets sa vydali opačnou cestou – nielenže aplikácie vyzerajú a správajú sa natívne – aplikácie sú natívne. Toto je nesmierne dôležité, pretože aj najmenšia zmena od štandardného ovládania môže užívateľa odradiť od používania aplikácie. S použitím natívneho API daného operačného systému a jazyka C++ sú tiež aplikácie vytvorené pomocou wxWidgets oveľa rýchlejšie. Odpadajú taktiež rôzne závislosti na knižniciach a aplikáciách tretích strán (Java, .NET). Jediné čo tak užívateľ potrebuje k spusteniu aplikácie je samotná aplikácia. [6]

wxWidgets sú open source projekt (projekt s "otvoreným", voľne dostupným zdrojovým kódom). Znamená to že zdrojový kód môžeme zdarma získať a zdarma ho tiež používať vo svojich aplikáciách. Licencia wxWidgets (oficiálne licencia wxWindows) je typu L-GPL s dodatkom a hovorí, že s použitím knižnice wxWidgets môžeme vytvárať voľne dostupné aplikácie, ale aj aplikácie na komerčné použitie bez toho aby sme museli zverejniť zdrojové kódy nášho programu. Jedinou podmienkou je, že zmeny vykonané v knižnici samotnej by sme mali poskytnúť ostatným užívateľom. [7]

wxWidgets majú taktiež veľkú priemyslovú podporu. Zoznam užívateľov zahŕňa spoločnosti ako AOL, AMD, CALTECH, Lockheed Martin, NASA, Xerox a mnoho ďalších. wxWidgets zahrňuje celú radu vývojárov od veľkých spoločností až po jednotlivcov. wxWidgets sa tiež stali základom mnohých open source aplikácií, spomenúť môžeme napríklad audio editor Audacity alebo aplikáciu na návrh databáz pgAdmin III.

Dôvody pre použitie wxWidgets sa rôznia. Môže sa jednať o veľmi elegantnú náhradu za MFC, prípadne ako nástroj na prevedenie našej aplikácie na rôzne platformy. [6]



Obr. 1. Ukážka natívneho vzhľadu aplikácie na rôznych platformách

1.4 Porovnanie wxWidgets s inými knižnicami

1.4.1 Qt

- Knižnica Qt rovnako ako wxWidgets obsahuje množstvo tried ktoré nesúvisia s tvorbou grafického užívateľského rozhrania (napríklad triedy pre prácu s dátumom a časom, sieťou, OpenGL a pod).
- Knižnica Qt je dostupná pod komerčnou licenciou a pod licenciami LGPL v2.1 a GPL v3.0. V prípade prvej musíme za použitie knižnice zaplatiť, v druhom prípade musíme zverejniť všetky zmeny ktoré boli v knižnici vykonané (licencia wxWidgets ponecháva toto rozhodnutie na užívateľovi). V poslednom prípade musíme zverejniť kompletné zdrojové kódy našej aplikácie.

- Qt na rozdiel od wxWidgets nemá úplne natívne porty. Aj keď sa môže zdať že aplikácie vyzerajú natívne, Qt používa svoje vlastné zdrojové kódy pre vykresľovanie ovládacích prvkov (wxWidgets využívajú podobný prístup v porte wxUniversal). Na platformách, ktoré využívajú prostredie KDE, alebo na mobilných linuxových platformách založených na Qt, je knižnica úplne natívna.
- Qt je použité v niektorých veľkých projektoch ako napríklad KDE, alebo internetový prehliadač Opera. Na druhej strane wxWidgets sú použité v projektoch ako napríklad AOL Communicator.
- Qt používa veľké množstvo virtuálnych funkcií, čo dáva knižnici lepší objektovo orientovaný návrh oproti wxWidgets, ktoré uprednostňujú prístup používaný v MFC – použitie makier. Pre nás to znamená, že pri použití Qt napíšeme menej riadkov zdrojového kódu, no aplikácie budú na rozdiel od wxWidgets bežať o niečo pomalšie. [8] [10]

1.4.2 Java

- Java je programovací jazyk, ktorý je možné skombinovať s množstvom nástrojov na tvorbu užívateľského grafického rozhrania (napríklad Qt, Swing, AWT, ale aj wxWidgets), zatiaľ čo wxWidgets predstavujú len API pre najrôznejšie programovacie jazyky.
- Aplikácie vytvorené v Jave sú pri prvom štarte skompilované z byte kódu. Znamená to že takéto aplikácie potrebujú viacej času na svoj štart a taktiež prvých pár chvíľ používania programu bude aplikácia mierne spomalená. wxWidgets sú namiesto toho kompilované priamo do strojového kódu čo výrazne zrýchľuje štart a používanie aplikácie.
- Na spustenie Java aplikácií je potrebné mať nainštalovaný virtuálny stroj – JVM (Java Virtual Machine) ktorý zaisťuje ich beh. Užívatelia ktorý majú na svojich počítačoch nainštalované staršie verzie JVM tak môžu trpieť spomaleným behom aplikácií, prípadne rôznymi bezpečnostnými problémami.
- Pre jazyk Java existuje implementácia wxWidgets nazvaná wx4j(wxWidgets 4 (pre) Javu). Je tak možné využívať jazyk Java, ale zároveň využiť rýchlosť programu vytvoreného v jazyku C++. [8]
- Z dôvodu aby bol jazyk Java multi-platformný, sú vlastnosti ktoré sú dostupné len na niektorých platformách z jazyka vynechané (ako príklad môžeme uviesť prácu

s panelom úloh pod operačným systémom Windows, alebo manipuláciu s atribútmi súborov pod operačnými systémami ktoré sú založené na Unixe). wxWidgets v tomto smere ponúka možnosť ľubovoľne používať špeciálne vlastnosti daného OS – pomocou konštrukcií ifdef a ifndef – jazyk Java k týmto konštrukciám nemá ekvivalent. Takisto sa wxWidgets snažia emulovať všetky vlastnosti daného OS ktoré nie sú dostupné v ostatných OS (napríklad vyhľadávacie pole, ktoré je natívne pod operačným systémom Mac OS X no nenachádza sa v ostatných OS).

- Programy napísané v Jave spotrebujú viacej operačnej pamäte ako ich ekvivalenty vytvorené v jazyku C++. [8]

1.4.3 GTK+

- GTK+ predstavuje knižnicu na tvorbu GUI napísanú v jazyku C pre Unixové operačné systémy.
- GTK+ bolo portované na OS Windows, VMS a iné operačné systémy s použitím rovnakého API. Primárne sa však knižnica sústreďuje na Unixové operačné systémy.
- Knižnica je primárne použitá pri návrhu užívateľského rozhrania v prostredí GNOME.
- Na rozdiel od wxWidgets je celá knižnica napísaná v jazyku C.
- Pomocou GTK+ dosiahneme rovnaký vzhľad aplikácií na všetkých platformách (okrem Windows XP).
- Pretože wxWidgets využívajú práve GTK na Unixových OS, existuje len veľmi malý dôvod použiť knižnicu GTK na vytvorenie multi-platformnej aplikácie. [8]

1.4.4 MFC

- Knižnica MFC je zdarma dostupná len pre OS Windows, licencie pre ostatné operačné systémy sú veľmi nákladné.
- MFC na rozdiel od wxWidgets vytvára menšie spustiteľné súbory.
- Pre MFC je dostupné veľké množstvo komerčných komponent.
- Hierarchia tried vo wxWidgets je viac intuitívna. [8]

1.5 Porty wxWidgets

Na nasledujúcom obrázku môžeme vidieť hlavné porty wxWidgets rozdelené do štyroch vrstiev. Úplne navrchu sa nachádza verejné aplikačné rozhranie wxWidgets, pod ním môžeme vidieť všetky hlavné porty nasledované API použitej platformy a na konci používaný operačný systém. [6]

wxWidgets API									
Port	wxMSW	wxGTK	wxX11	wxMotif	wxMac	wxCocoa	wxOS2	wxPalmOS	wxMGL
API	Win32	GTK+	Xlib	Mofif	Carbon	Cocoa	PM	Palm OS Protein API	MGL
OS	Windows, Windows CE	Unix/Linux			Mac OS 9, Mac OS X	Mac OS X	OS/2	Palm OS	Unix, DOS
					 Mac				

Obr. 2. Porty wxWidgets [6]

1.5.1 wxMSW

Port určený pre všetky 32 a 64 bitové verzie operačného systému Microsoft Windows. Podporovanými operačnými systémami sú Windows 95, Windows 98, Windows ME, Windows NT, Windows 2000, Windows XP, Windows 2003 a Windows Vista. Tento port je taktiež možné preložiť pod operačným systémom Linux s využitím knižnice Winelib. K dispozícii je tiež konfigurácia pre Windows CE (viď port wxWinCE). Tento port je taktiež možné nakonfigurovať tak aby bolo použité rozhranie wxUniversal namiesto natívnych ovládacích prvkov.

1.5.2 wxGTK

wxGTK port wxWidgets môže využívať ovládacie prvky z GTK+ verzie 1.x, ako aj z verzie 2.x. Tento port môže bežať na akejkoľvek variante Unixového operačného systému, ktorý podporuje X11 a GTK+ (napríklad Linux, Solaris, FreeBSD, OpenBSD a veľa ďalších). Je tiež možné používať ho na embedded systémoch s dostatočným výkonom.

1.5.3 wxX11

Port wxWidgets pre X11 používa sadu ovládacích prvkov z wxUniversal a beží priamo na knižnici Xlib bez akýchkoľvek natívnych ovládacích prvkov. Tento port je mimoriadne vhodný pre embedded systémy a systémy s malým výkonom na ktorých by použitie GTK+ nebolo príliš vhodné. Port je podporovaný na každom Unixovom operačnom systéme s podporou X11. [6] [7]

1.5.4 wxMotif

Port využívajúci knižnice Motif, OpenMotif a Lesstif na väčšine Unixových operačných systémoch. Motif je však pomerne často nahrádzaný prostredím GNOME a GTK+ čo robí tento port pre užívateľov pomerne nezaujímavým.

1.5.5 wxMac

Už podľa názvu portu je hneď zrejmé, že tento port je určený pre operačný systém Mac OS. Podporované sú konkrétne verzie Mac OS 9 (od verzie 9.1), Mac OS X (od verzie 10.2.8). Pre Mac OS X je taktiež dostupný wxWidgets port wxCocoa.

1.5.6 wxWinCE

Port wxWinCE je určený pre operačný systém Microsoft Windows CE 3.0 a vyšší. Na rozdiel od ostatných embedded portov používa tento port natívne ovládacie prvky všade tam kde je to len možné. Jadro tohto portu je založené na porte wxMSW a modifikované pre menšiu platformu. [6] [7]

1.6 Tvorba aplikácie pomocou wxWidgets

V nasledujúcej kapitole sa pokúsime o vytvorenie jednoduchej vzorovej aplikácie pomocou knižnice wxWidgets. Bude sa jednať o jednoduchú aplikáciu typu „Hello World“ ktorá bude demonštrovať základné princípy tvorby aplikácie pomocou tejto knižnice. Aplikácia bude obsahovať hlavné menu s možnosťou ukončenia aplikácie a s možnosťou zobrazenia informácií o programe. Ďalej sa tu bude nachádzať statusbar a uprostred okna centrovaný text „Hello World“ – veľkosť toho okna budeme môcť meniť – pri zmene veľkosti sa text vykreslí tak aby bol vždy centrovaný.

Aplikácia sa skladá z viacerých prvkov. Tieto prvky môžeme rozdeliť na dve kategórie: ovládacie prvky a kontajnery. Ovládacie prvky reprezentujú viditeľné objekty v

užívateľskom rozhraní, pomocou ktorých môže užívateľ ovládať aplikáciu (napríklad tlačítka, posuvníky, text boxy atď). Kontajnery sú potom použité na organizáciu týchto prvkov. Môžeme pomocou nich napríklad zarovnať ovládacie prvky do mriežky s určitým počtom riadkov a stĺpcov. Kontajnerom môže byť napríklad hlavné okno aplikácie, alebo stránky so záložkami ktoré na každej strane obsahujú iné ovládacie prvky atď. Kontajnery môžu taktiež obsahovať iné kontajnery, čo nám dovoľuje vytvoriť veľmi zložitú štruktúru ovládacích prvkov. [3] [7]

Vzorová aplikácia bude priamo používať niektoré základné ovládacie prvky – jeden z nich je predstavovaný triedou `wxStaticText`. Tento prvok má len jednu funkciu – zobrazenie reťazca znakov. Reťazec, ako aj jeho umiestnenie a veľkosť v okne aplikácie môže byť definovaný už pri vytváraní prvku – pomocou argumentov konštruktora triedy `wxStaticText`. Pomocou členských funkcií tejto triedy potom môžeme zobrazovaný text meniť (alebo získať) dynamicky za behu aplikácie.

Trieda `wxStaticText` tak ako veľa ďalších tried vo wxWidgets je vytvorená pomocou dedičnosti z iných tried. V prípade triedy `wxStaticText` sú nimi triedy v tomto poradí: `wxObject`, `wxEvtHandler`, `wxWindow` a trieda `wxControl`. Ako môžeme vidieť, každá ďalšia trieda v hierarchii viac špecializuje triedu predchádzajúcu. Trieda `wxObject` nám poskytuje úplne najabstraktnejšiu funkcionalitu, ktorá je potrebná u všetkých prvkov vo wxWidgets. Trieda `wxWindow` potom predstavuje najväčšiu triedu v tejto hierarchii, čo odráža skutočnosť že väčšina ovládacích prvkov sú v skutočnosti okná – je zodpovedná hlavne za správu vlastností okien (ako napríklad veľkosť, pozícia, viditeľnosť a pod). Trieda `wxControl` nám poskytuje rozhranie pomocou ktorého môžu ovládacie prvky prijímať príkazy ktoré majú spracovať. A ako už bolo spomenuté, členské funkcie triedy `wxStaticText` nám umožňujú napríklad manipuláciu so zobrazeným textom.

Z kontajnerových tried bude naša vzorová aplikácia obsahovať inštancie dvoch tried: `wxFrame` a `wxBoxSizer`. Trieda `wxFrame` nám umožňuje vytvoriť hlavné okno aplikácie – okno s určitou veľkosťou, nadpisom, menu, statusbarom a pod. Trieda `wxBoxSizer` je potom zodpovedná za rozloženie ovládacích prvkov v hlavnom okne – v našom prípade prvku `wxStaticText`. [3] [6] [7]

Pre úspešný chod našej aplikácie je najskôr potrebné vytvoriť inštanciu triedy `wxApp`. Táto trieda predstavuje samotnú aplikáciu. Na začiatok odvodíme triedu z triedy `wxApp`:

```
1 class HelloWorldApp : public wxApp
2 {
3     public:
4         virtual bool OnInit();
5 };
```

Aj keď trieda `wxApp` obsahuje veľké množstvo virtuálnych funkcií, jedinou funkciou ktorú musí implementovať každá aplikácia je funkcia `OnInit()`. Táto funkcia bude zavolaná vo chvíli, keď sú wxWidgets pripravené vykonať náš kód – ekvivalent funkcie main v jazyku C. V implementácii tejto funkcie máme potom priestor pre vytvorenie hlavného okna aplikácie, spracovanie parametrov príkazového riadka a ostatné inicializačné úlohy potrebné pre beh aplikácie. Návratová hodnota z tejto funkcie určuje, či má chod aplikácie pokračovať (hodnota `true`) alebo má byť ukončený (hodnota `false`). Implementácia najjednoduchšej formy tejto funkcie môže vyzerať nasledovne:

```
1 bool HelloWorldApp::OnInit()
2 {
3     // vytvoríme hlavné okno aplikácie
4     HelloWorldFrame* frame = new HelloWorldFrame(wxT("Hello
      World"));
5
6     // zobrazíme toto okno
7     frame->Show();
8
9     // vrátime hodnotu true
10    return true;
11 }
```

Ako môžeme vidieť, v tele funkcie je vytvorená inštancia triedy `HelloWorldFrame`, ktorá predstavuje hlavné okno aplikácie. Po tom čo bolo okno vytvorené, zobrazíme ho na obrazovke pomocou metódy `Show()` – riadok 7. Následne vrátime hodnotu `true`, a umožníme tak ďalší beh aplikácie. Za povšimnutie tiež stojí zápis textov v zdrojovom kóde, ale aj v celej knižnici wxWidgets do makra `wxT()` – s jeho pomocou sú všetky texty prevedené na príslušný dátový typ, čo nám umožňuje skompilovať aplikáciu s použitím ale aj bez použitia Unicode. Konverzia prebieha v čase kompilácie a nepredstavuje tak riziko spomalenia aplikácie. [3] [6] [7]

Možno si kladiete otázku, kde v aplikácii je vytvorená inštancia našej novo vytvorenej triedy `HelloWorldApp`? Ako vás pravdepodobne napadne, wxWidgets vytvárajú túto

inštanciu automaticky – najskôr však musíme určiť ktorú triedu majú wxWidgets použiť. Urobíme tak pomocou makra `IMPLEMENT_APP()`:

```
1 IMPLEMENT_APP(HelloWorldApp);
```

V tele funkcie `OnInit()` je vytvorená inštancia triedy okna `HelloWorldFrame`. Na to, aby sme túto triedu mohli použiť je potrebné ju najskôr deklarovať. Urobíme tak rovnakým spôsobom, akým sme vytvorili triedu `HelloWorldApp` z triedy `wxApp` – odvodením z triedy `wxFrame`. Trieda `wxFrame` predstavuje okno ktoré má nadpis, môže obsahovať ovládacie prvky, menu, statusbar a pod. Deklarácia triedy `HelloWorldFrame` je nasledujúca:

```
1  class HelloWorldFrame: public wxFrame
2  {
3      public:
4          // konštruktor
5          HelloWorldFrame(const wxString& title);
6
7          // obsluha udalostí
8          void OnAbout(wxCommandEvent& event);
9          void OnExit(wxCommandEvent& event);
10
11     private:
12         // deklarácia tabuľky udalostí
13         DECLARE_EVENT_TABLE()
14 };
```

Naša trieda obsahuje konštruktor, ktorý ako svoj jediný parameter očakáva názov okna. Ďalej tu máme dve obsluhy udalostí ktoré budú previazané s položkami menu. Na konci sa potom nachádza makro ktoré hovorí že wxWidgets majú pre túto triedu vytvoriť tabuľku udalostí. Pomocou tabuľky udalostí môžeme previazať jednotlivé funkcie obsluhy udalostí s akciami ktoré vykonal užívateľ, prípadne vznikli na inom mieste v aplikácii alebo mimo nej. V našej vzorovej aplikácii chceme sledovať kliknutia na jednotlivé položky menu. K tomu musíme prepojiť identifikátory týchto položiek s ich obsluhami. Urobíme tak nasledovným spôsobom:

```
1 // tabuľka udalostí pre triedu HelloWorldFrame
2 BEGIN_EVENT_TABLE(HelloWorldFrame, wxFrame)
3     EVT_MENU(wxID_ABOUT, HelloWorldFrame::OnAbout)
4     EVT_MENU(wxID_EXIT,  HelloWorldFrame::OnExit)
5 END_EVENT_TABLE()
```

Ako vidíme makro `EVT_MENU` priradí k identifikátoru `wxID_ABOUT` členskú funkciu `OnAbout` triedy `HelloWorldFrame` a k identifikátoru `wxID_EXIT` funkciu `OnExit`. Tieto identifikátory sú už pred vytvorené vo wxWidgets, no nič nám nebráni vytvoriť si svoje vlastné pomocou konštánt, enumerácií alebo pomocou definícií preprocesora. Makro `EVT_MENU` je len jedným z mnohých makier slúžiacich na pripojenie obslúh udalostí (spomenúť môžeme napríklad makro `EVT_SIZE` ktoré určuje funkciu volanú pri zmene veľkosti okna, alebo makro `EVT_BUTTON` spájajúce identifikátor tlačítka s jeho obsluhou). Na nasledujúcom výpise môžeme vidieť definíciu oboch obslúh udalostí v našej aplikácii:

```
 1 void HelloWorldFrame::OnAbout(wxCommandEvent& event)
 2 {
 3     // zobrazenie správy
 4     wxMessageBox(wxT("Víta vás aplikácia Hello World!"), wxT("O
 5     Programe"), wxICON_INFORMATION);
 6 }
 7
 8 void HelloWorldFrame::OnExit(wxCommandEvent& event)
 9 {
10     // ukončenie aplikácie
11    Destroy();
12 }
```

Funkcia `OnAbout` je volaná vo chvíli keď užívateľ klikne na položku *„O Programe”* v hlavnom menu aplikácie. Pomocou funkcie `wxMessageBox` je potom zobrazená správa *"Víta vás aplikácia Hello World!"* a aplikácia bude čakať na stlačenie tlačítka OK. Podobne funkcia `OnExit` je volaná vo chvíli keď užívateľ vyberie z menu položku *“Ukončiť program”*. Táto funkcia ukončí beh hlavného okna aplikácie a pretože aplikácia má len jedno okno, je ukončený beh celého programu. V prípade že užívateľ ukončí aplikáciu štandardnou cestou (napríklad kliknutím na X v rohu aplikácie), wxWidgets poskytnú pre spracovanie tejto akcie vlastnú obsluhu udalostí. Samozrejmosťou je, že pre každú akciu môžeme poskytnúť vlastnú obsluhu udalostí – je tak možné sa napríklad užívateľa spýtať, či chce aplikáciu naozaj ukončiť pred jej ukončením. Týmto spôsobom je vo wxWidgets vyriešená obsluha väčšiny udalostí – udalosti, ktoré pre nás nemajú význam necháme na starosti wxWidgets, naopak pre udalosti ktoré nás zaujímajú môžeme poskytnúť vlastnú obsluhu. [6] [7]

Vráťme sa späť k triede hlavného okna aplikácie. Posledná funkcia ktorá zatiaľ zostala bez definície je konštruktor triedy:

```
1  HelloWorldFrame::HelloWorldFrame(const wxString& title) :
   wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(320, 200))
2  {
3      // menu súbor
4      wxMenu *fileMenu = new wxMenu;
5      fileMenu->Append(wxID_EXIT, wxT("&Ukončiť program\tAlt-X"),
       wxT("Ukončí program"));
6
7      // menu pomoc
8      wxMenu *helpMenu = new wxMenu;
9      helpMenu->Append(wxID_ABOUT, wxT("&O Programe...\tF1"),
       wxT("Zobrazí informácie o programe"));
10
11     // pripojenie menu do hlavného menu
12     wxMenuBar *menuBar = new wxMenuBar();
13     menuBar->Append(fileMenu, wxT("&Súbor"));
14     menuBar->Append(helpMenu, wxT("&Pomoc"));
15
16     // priradenie hlavného menu oknu aplikácie
17     SetMenuBar(menuBar);
18
19     // vytvorenie statusbaru a nastavenie úvodného textu
20     CreateStatusBar(1);
21     SetStatusText(wxT("Víta vás aplikácia Hello World!"));
22
23     // statický text zobrazovaný v okne aplikácie
24     wxStaticText *text = new wxStaticText(this, wxID_ANY, wxT("Hello
       World"));
25
26     // sizer ktorý tento text vycentruje
27     wxBoxSizer *sizer = new wxBoxSizer(wxVERTICAL);
28
29     // umiestnenie prvku wxStaticText medzi dve pružné medzery
30     sizer->AddStretchSpacer();
31     sizer->Add(text, 0, wxALIGN_CENTER);
32     sizer->AddStretchSpacer();
33
```

```
34      // priradenie sizeru oknu aplikácie
35      SetSizer(sizer);
36  }
```

Na začiatok si môžeme všimnúť, volanie konštruktora rodičovskej triedy `wxFrame`, ktorého parametre sú definované nasledovne:

```
1  wxFrame(wxWindow* parent, // ukazateľ na rodičovské okno
2          wxWindowID id, // identifikátor okna
3          const wxString& title, // názov okna
4          const wxPoint& pos = wxDefaultPosition, // pozícia okna
5          const wxSize& size = wxDefaultSize, // veľkosť okna
6          long style = wxDEFAULT_FRAME_STYLE, // štýl okna
7          const wxString& name = "frame");
```

Tomuto konštruktoru predávame ukazateľ na rodičovské okno (naše okno nemá rodiča, preto je tu hodnota `NULL`). Ako identifikátor okna zadávame hodnotu `wxID_ANY`, ktorá hovorí že o hodnote identifikátora okna rozhodnú samotné wxWidgets. Nasleduje názov okna predaný ako parameter pri vytváraní inštancie odvodenej triedy. Pozíciu okna ponecháme na starosti wxWidgets (hodnota `wxDefaultPosition`). Parameter `size` určuje veľkosť okna – v našom prípade je to veľkosť 320 x 200 pixelov. Ostatné parametre ponecháme nevyplnené a wxWidgets použijú na ich mieste defaultné hodnoty.

Konštruktor hlavného okna má na starosť vytvorenie všetkých ovládacích prvkov a určenie ich rozloženia. Ako prvé vytvárame položky menu. Pri ich vytváraní definujeme ich identifikátor nasledovaný zobrazovaným textom. Ako posledný parameter zadávame pomocný text, ktorý bude zobrazený v statusbare aplikácie pri zvýraznení tejto položky. Všimnúť si tiež môžeme písmena za znakom & v texte po ktorého stlačení na klávesnici je táto položka menu zvýraznená. Ďalej je tu možné vidieť definíciu klávesovej skratky, ktorá sa nachádza za znakom tabulátora ‘\t’. Po jej použití je vykonaná rovnaká akcia ako keby užívateľ vybral túto akciu pomocou menu. Po vytvorení menu je vytvorený statusbar aplikácie s prednastaveným úvodným textom *„Víta vás aplikácia Hello World!“*. Ako posledný ovládací prvok vytvárame inštanciu triedy `wxStaticText`. Konštruktor tejto triedy rovnako ako konštruktory mnohých ďalších tried ovládacích prvkov vo wxWidgets je veľmi podobný konštruktoru okna aplikácie:

```
1  wxStaticText(wxWindow* parent, // ukazateľ na rodičovské okno
2               wxWindowID id, // identifikátor okna
3               const wxString& label, // názov okna
```

```
4              const wxPoint& pos = wxDefaultPosition, // pozícia
5              const wxSize& size = wxDefaultSize, // veľkosť okna
6              long style = 0, // stýl okna
7              const wxString& name = "staticText");
```

O odstránenie všetkých vytvorených prvkov sa postará samotné rodičovské okno vo chvíli, keď bude toto okno zatvorené. Ovládací prvok chceme následne umiestniť uprostred okna aplikácie, preto v nasledujúcom kroku vytvoríme tzv. sizer. Prototyp konštruktora pre použitý `wxBoxSizer` je veľmi jednoduchý:

```
1 wxBoxSizer(int orientation);
```

Parameter `orientation` (orientácia) môže nadobúdať hodnotu `wxHORIZONTAL`, alebo `wxVERTICAL`. Vertikálny sizer umiestňuje jednotlivé prvky smerom odhora nadol, zatiaľ čo horizontálny sizer smerom zľava doprava. V našom prípade vytvárame sizer vertikálny. Pomocou členských metód potom do sizer-u vkladáme jednotlivé bunky:

```
1 Add(wxWindow* window, // ukazateľ na ovládací prvok
2     int proportion = 0, // proporcia
3     int flag = 0, // nastavenia – okraje, zarovnanie...
4     int border = 0, // veľkosť okraja okolo ovládacieho prvku
5     wxObject* userData = NULL);
```

```
1 AddStretchSpacer(int prop = 1); // proporcia
```

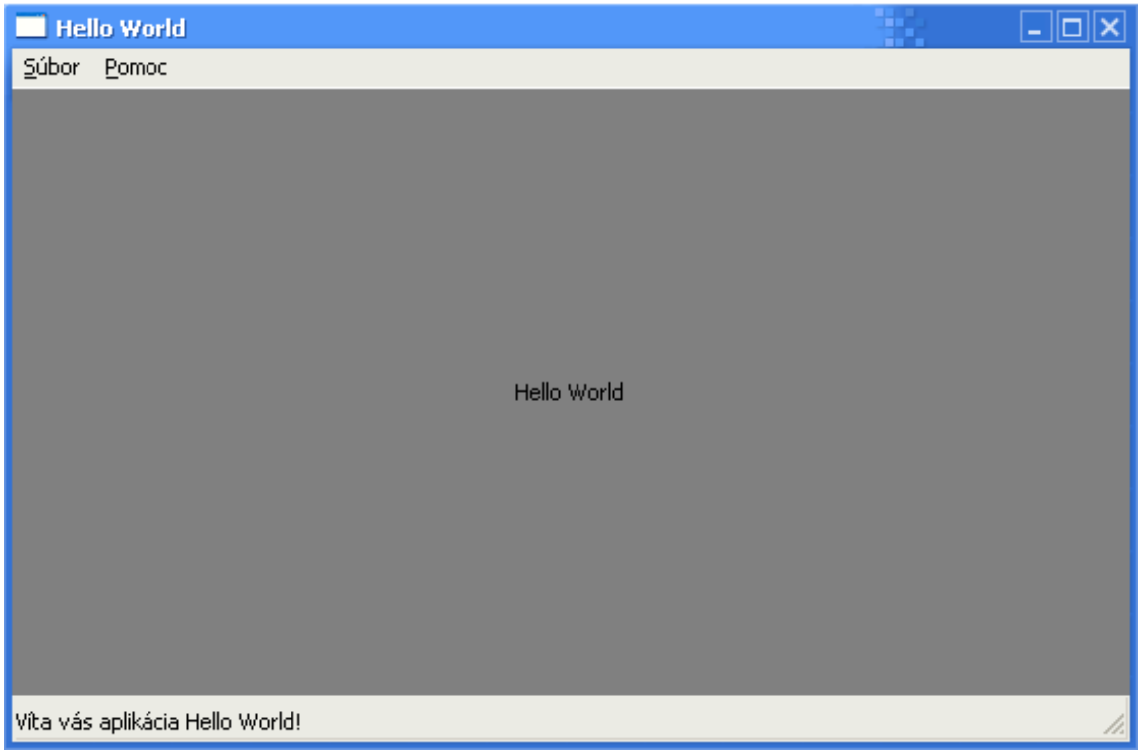
Pomocou argumentu proporcie môžeme určiť aký veľký priestor bude ovládací prvok zaberať z celkového priestoru vyhradeného pre sizer. Ak je jeho hodnota rovná nule zaberá prvok minimálnu možnú plochu, naopak ak je hodnota rovná jednej zaberá prvok plochu maximálnu. Pomocou tohto mechanizmu tak môžeme napríklad dosiahnuť rozloženie kde sa dva stĺpce (riadky) budú rovnomerne prispôsobovať veľkosti okna (proporcia rovná jednej), zatiaľ čo tretí si bude zachovávať konštantné rozmery (proporcia rovná nule). V našom prípade spôsobí nasledujúci kód:

```
30     sizer->AddStretchSpacer(1); // pružná medzera
31     sizer->Add(text, 0, wxALIGN_CENTER, 0);
32     sizer->AddStretchSpacer(1); // pružná medzera
```

vloženie ovládacieho prvku wxStaticText určeného ukazateľom `text` medzi dve rovnako veľké (proporcia rovná jednej) pružné medzery. Hodnota `wxALIGN_CENTER` hovor,í že sizer má vycentrovať obsah tejto bunky. Posledným krokom je potom už len prideliť tento sizer oknu aplikácie:

```
35     SetSizer(sizer);
```

Kompletný výpis zdrojového kódu vzorovej aplikácie môžete nájsť v prílohe číslo 1.



Obr. 3. Screenshot okna vzorovej aplikácie

1.7 Vývojové nástroje pre wxWidgets

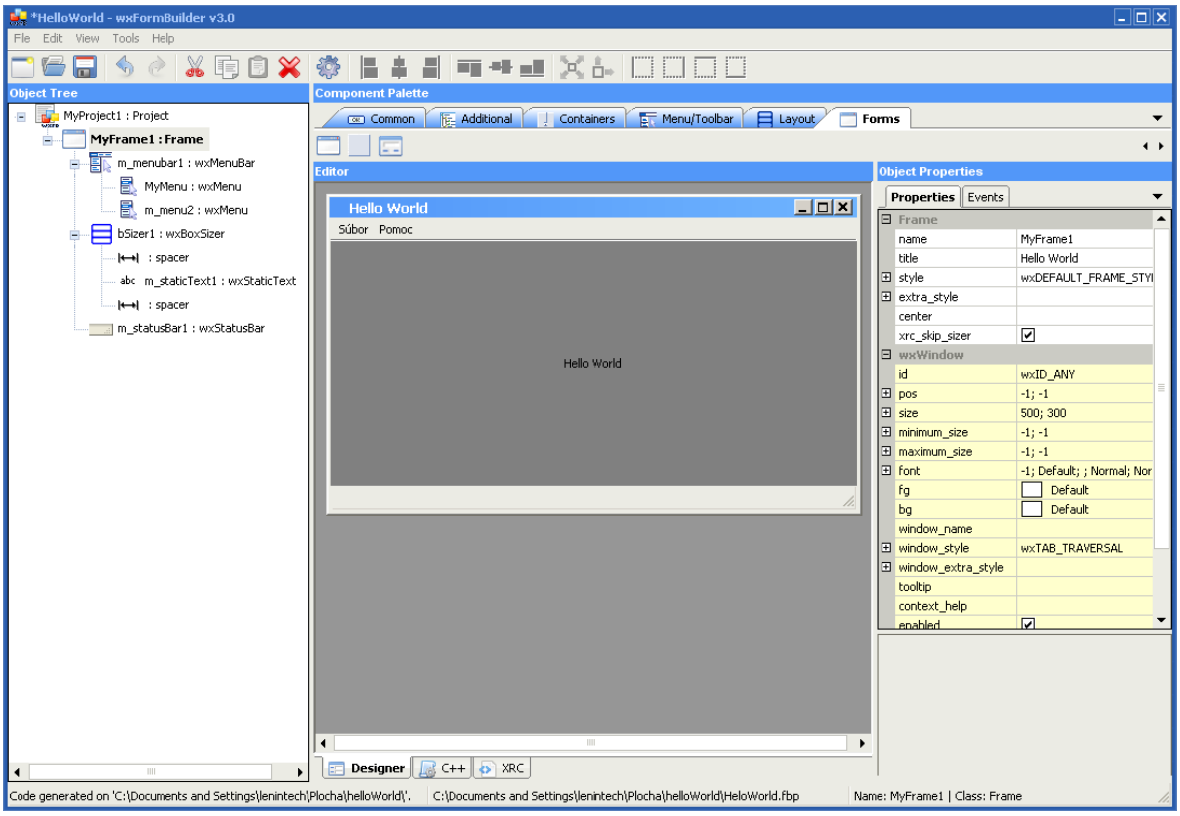
Pre wxWidgets vzniklo za dobu ich existencie veľké množstvo nástrojov uľahčujúcich vývoj aplikácií. Jedná sa napríklad o aplikácie na návrh užívateľského rozhrania pomocou ktorých môžeme aj bez znalosti programovania navrhnúť kompletné GUI, alebo aplikácie uľahčujúce kompiláciu a tvorbu projektov vo wxWidgets. V nasledujúcej kapitole si niektoré z nich popíšeme.

1.7.1 wxFormBuilder

wxFormBuilder je aplikácia s otvoreným zdrojovým kódom, ktorej hlavnou funkciou je vizuálna tvorba užívateľského rozhrania. Môžeme tak napríklad len s pomocou myši vytvoriť okno aplikácie spolu so všetkými ovládacími prvkami a z výsledného návrhu vygenerovať zdrojový kód niektorého z podporovaných jazykov. Prednosťami aplikácie wxFormBuilder sú:

- Open source – zdarma.
- Podpora veľkého množstva ovládacích prvkov.
- Možnosť vkladať negrafické ovládacie prvky.

- Prepojenie ovládacích prvkov s obsluhami udalostí – pomocou metódy Connect() alebo pomocou tabuľky udalostí.
- Rozšíriteľnosť pomocou pluginov.
- Z návrhu môžeme vygenerovať zdrojový kód jazyka C++ a XRC (XML zrdoje wxWidgets).
- Plná podpora Unicode.
- Beží na operačných systémoch MS Windows, Linux a Mac OS X.



Obr. 4. Aplikácia wxFormBuilder

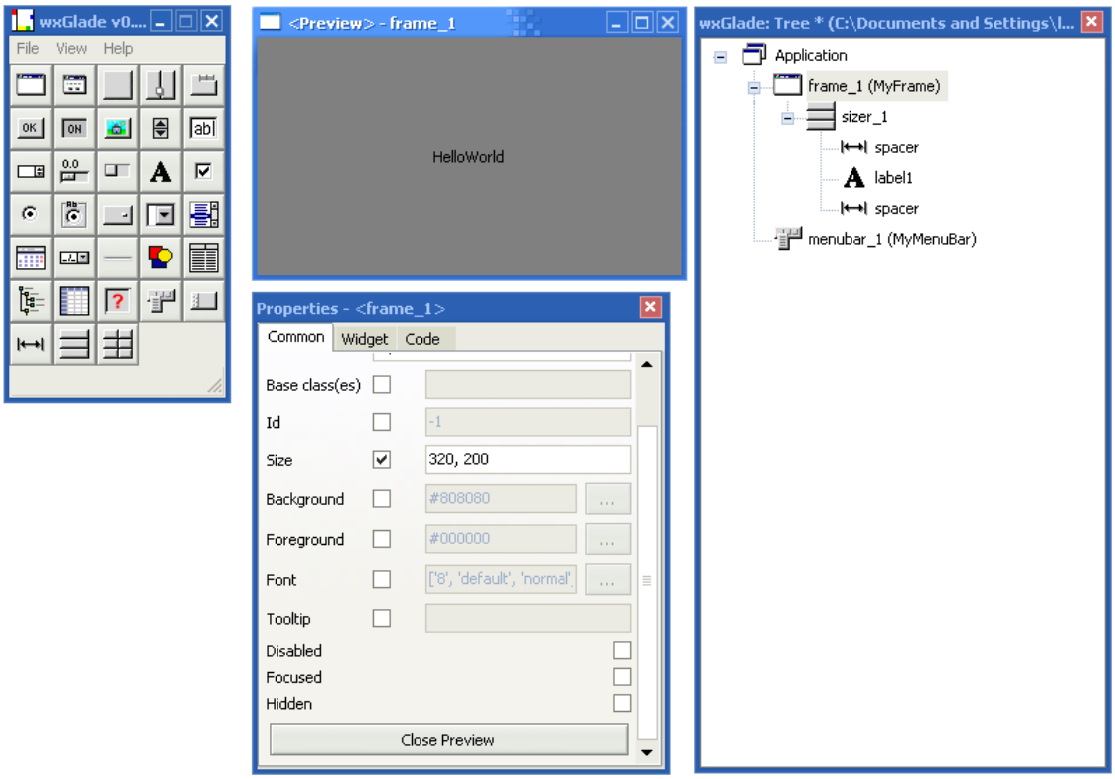
1.7.2 wxGlade

Aplikácia wxGlade je podobne ako aplikácia wxFormBuilder určená na vizuálnu tvorbu užívateľského rozhrania. Je vytvorená v jazyku Python s použitím rozšírenia wxPython.

Vlastnosti:

- Open source aplikácia – zdarma.
- Všetky ovládacie prvky musia byť umiestnené pomocou sizerov.
- Obsahuje menšie množstvo ovládacích prvkov.
- Umožňuje vygenerovať zdrojový kód pre jazyky Python, C++, Perl, Lisp a XRC.

- Beží na všetkých platformách, ktoré podporujú beh programov vytvorených vo wxPython-e.



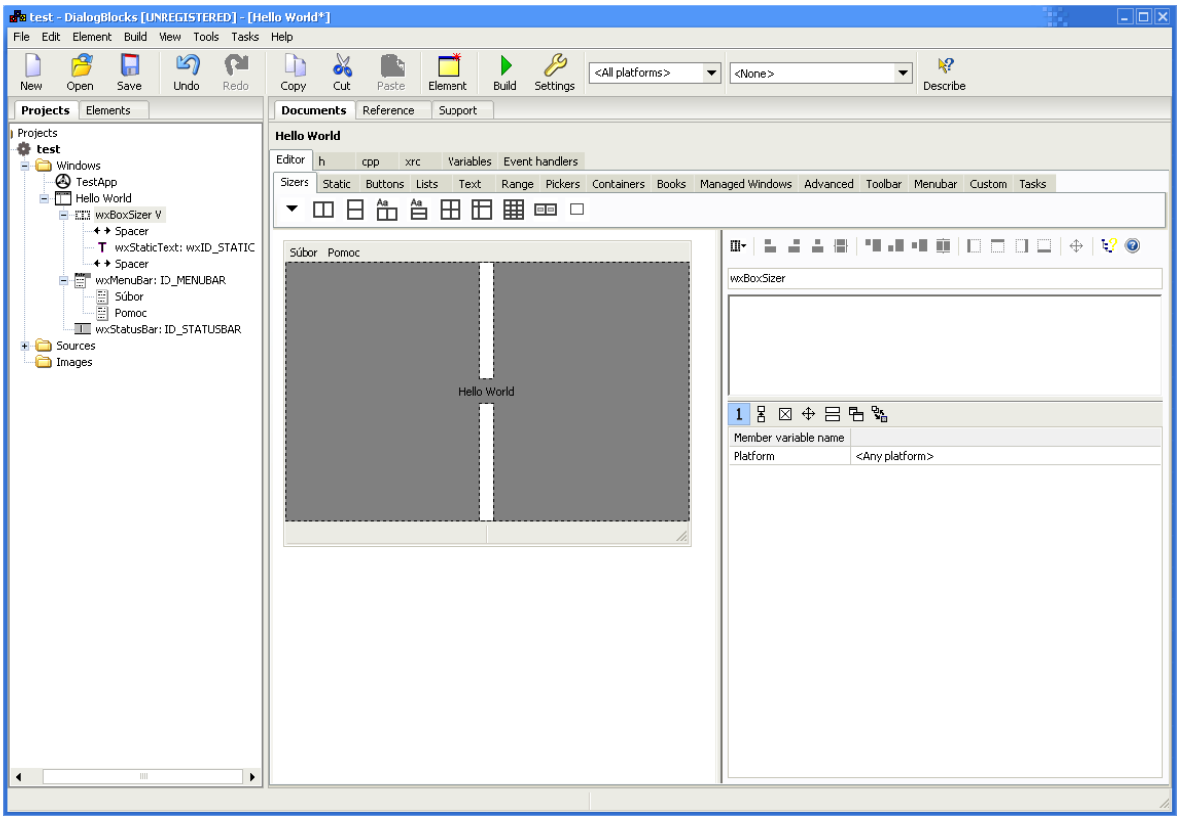
Obr. 5. Aplikácia wxGlade

1.7.3 DialogBlocks

Vývojové prostredie umožňujúce vytvoriť kompletnú aplikáciu vo wxWidgets vytvorené lídrom projektu wxWidgets – Julianom Smartom. Na rozdiel od predchádzajúcich aplikácií je však za jej použitie nutné zaplatiť. Hlavnými prednosťami aplikácie sú:

- Možnosť kompilácie a spustenia projektu priamo z aplikácie.
- Inteligentný editor vygenerovaného zdrojového kódu umožňujúci aplikáciu naďalej vizuálne upravovať bez straty vlastného kódu.
- Generovanie zdrojového kódu pre jazyky C++ a XRC.
- Vytváranie make súborov pre všetky najpoužívanejšie kompilátory a projektov pre MS Visual Studio.
- Vytváranie tabuliek a obslúh udalostí.
- Aplikácia je dostupná na platformách Windows, Linux, FreeBSD, Solaris x86 a Mac.

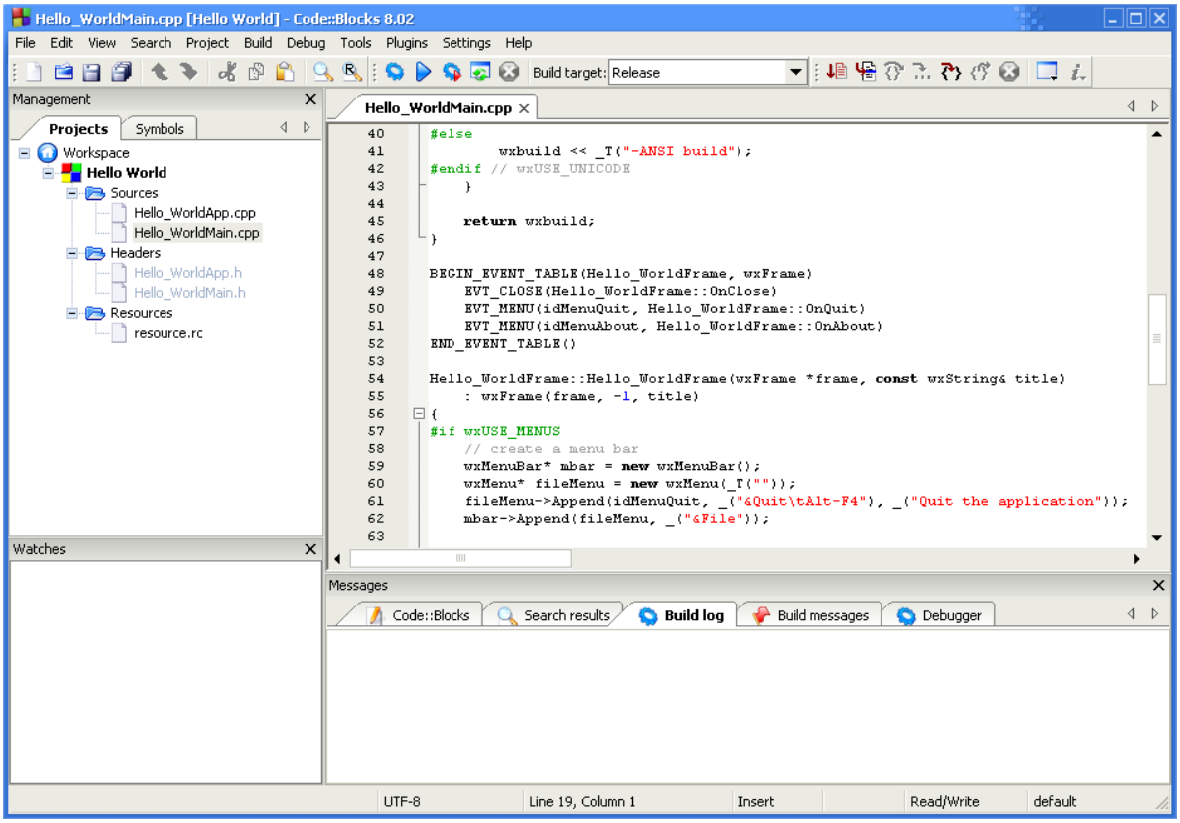
- Analýza projektu odhalí najznámejšie problémy ako sú napríklad prázdne sizere a zastarané štýly.
- Cena 56 EUR (90 USD).



Obr. 6. Aplikácia DialogBlocks

1.7.4 Code::Blocks

- Univerzálne multi-platformné vývojové prostredie – podporovanými platformami sú MS Windows, Linux a Mac.
- Open source projekt – zdarma.
- Rozšíriteľnosť pomocou pluginov.
- Podpora najznámejších kompilátorov, debugger.
- Zvýrazňovanie syntaxe zdrojového kódu, dopĺňanie kódu.
- Obsahuje jednoduchý vizuálny editor užívateľského rozhrania pre wxWidgets – wxSmith.
- Umožňuje vygenerovať projekt a kostru aplikácie pre wxWidgets.



Obr. 7. Aplikácia Code::Blocks

II. PRAKTICKÁ ČASŤ

2 DEMONŠTRAČNÁ A DOKUMENTAČNÁ APLIKÁCIA PRE SW KNIŽNICU WXWIDGETS

Úlohou praktickej časti tejto práce je vytvoriť aplikáciu, ktorá by ľahkým a zrozumiteľným spôsobom prezentovala vlastnosti a princípy tvorby aplikácií pomocou knižnice wxWidgets a umožnila tak začínajúcim aj pokročilým vývojárom ľahšie pochopenie práce s touto knižnicou.

2.1 Požiadavky na aplikáciu

- Aplikácia bude vytvorená s použitím jazyka ANSI C++ a knižnice wxWidgets.
- Jednoduché a intuitívne prehliadanie a spúšťanie vzorových aplikácií ktoré sú dodávané spoločne s knižnicou vrátane prehliadania ich zdrojových kódov.
- Previazanie s dokumentáciou knižnice wxWidgets vytvorenou pomocou nástroja Doxygen.
- Veľká univerzálnosť a konfigurovateľnosť pomocou externých súborov a skriptov.
- Funkčnosť pod operačnými systémami MS Windows a Linux, prípadne Mac OS.

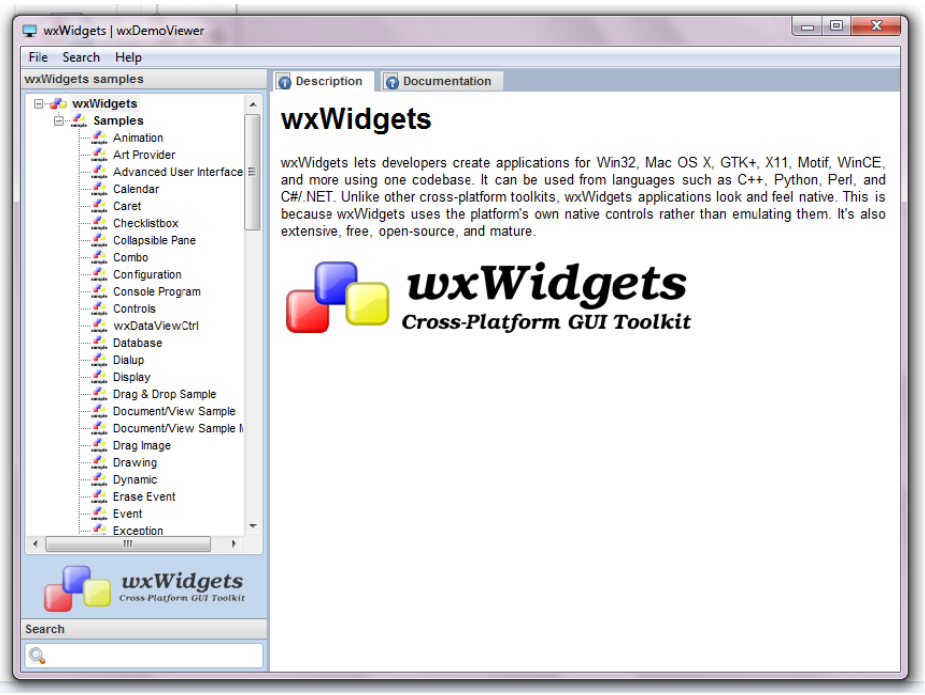
2.2 Vlastnosti aplikácie

Na základe týchto požiadavkou vznikla aplikácia wxDemoViewer. Jej vzorom sa stala demonštračná aplikácia knižnice wxPython [9], ktorá oplývala intuitívnym užívateľským rozhraním a podobnou funkcionalitou ako požadovaná aplikácia.

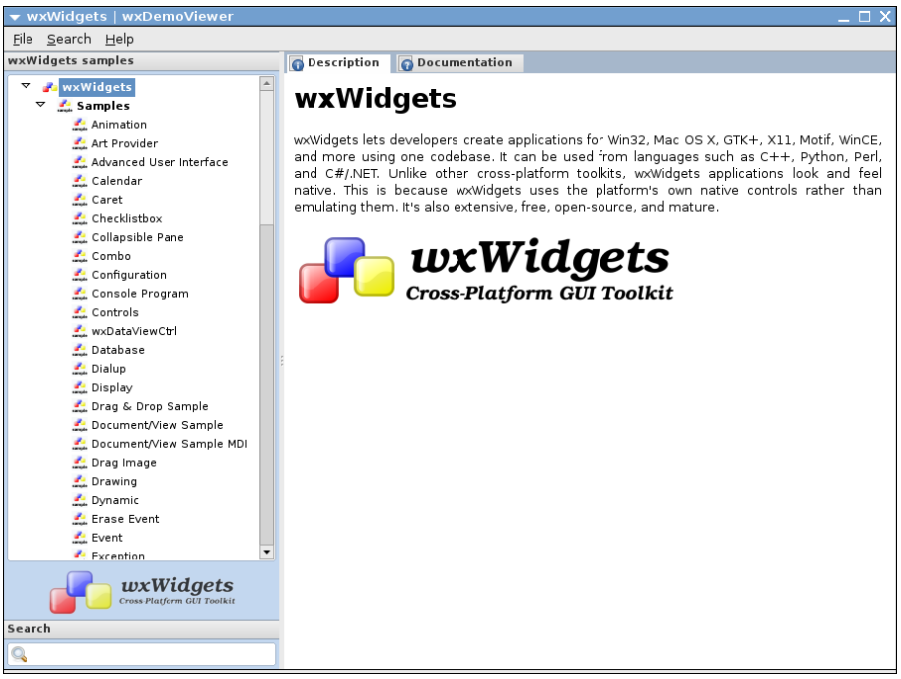
Hlavné vlastnosti aplikácie wxDemoViewer:

- Podpora skinov – pomocou externých súborov je možné zmeniť vzhľad aplikácie.
- Možnosť pridávať vlastný obsah, vrátane rôznych jazykových mutácií.
- Prehliadanie zdrojových kódov so zvýraznením syntaxe.
- Možnosť vybrať si prezeraný zdrojový súbor, ak vzorová aplikácia obsahuje viac zdrojových súborov.
- Možnosť definovať kľúčové slová ku každej vzorovej aplikácii, ktoré môžu byť vyhľadané v zdrojovom kóde alebo dokumentácii.
- Vyhľadávanie v zdrojovom kóde a v zozname vzorových aplikácií.
- Vstavaný prehliadač dokumentácie.

- Môže byť použitá ako prezentačná / dokumentačná aplikácia pre iné softwarové knižnice vytvorené pomocou rôznych programovacích jazykov ako napríklad Python, prípadne na prezentáciu obsahu ktorý nemá s programovaním žiadnu súvislosť.
- Aplikácia obsahuje vzorové aplikácie distribuované spoločne s knižnicou a kompletnú dokumentáciu wxWidgets.



Obr. 8. Aplikácia wxDemoViewer na OS Windows Vista



Obr. 9. Aplikácia wxDemoViewer na OS Linux

3 ŠTRUKTÚRA APLIKÁCIE

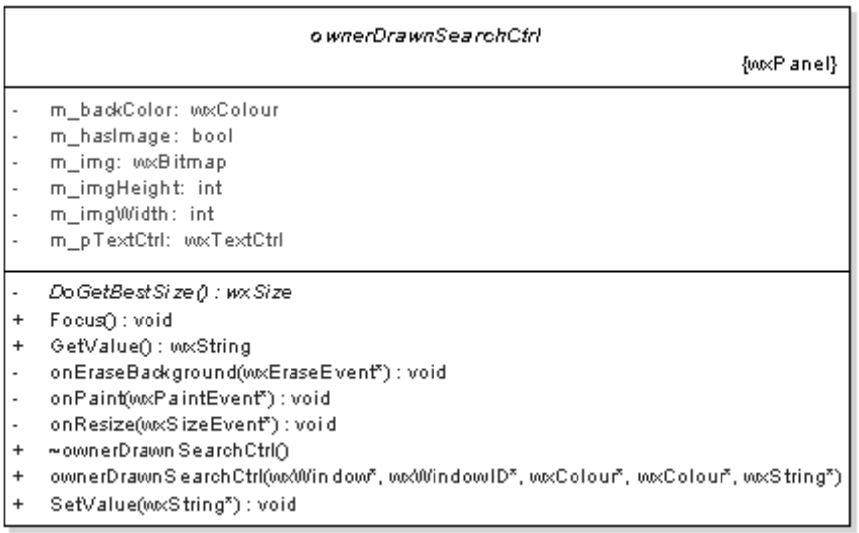
Štruktúra aplikácie je tvorená niekoľkými triedami. Medzi tieto triedy patria napríklad triedy skinovateľných ovládacích prvkov – prvky ktorých vzhľad môže byť zmenený zmenou v konfiguračných súboroch, alebo triedy slúžiace na načítanie a uchovávanie dát z externých súborov. Štruktúra týchto tried bude popísaná v nasledujúcich kapitolách.

3.1 Triedy skinovateľných ovládacích prvkov

Keďže hlavných cieľom aplikácie wxDemoViewer je prezentácia jej obsahu, je dôležité aby použité užívateľské rozhranie určitým spôsobom upútalo užívateľa. Je preto potrebné použiť ovládacie prvky ktorých vzhľad môže byť jednoduchou formou zmenený a umožniť tak tvorcovi dema zladiť vzhľad aplikácie s konkrétnym obsahom. V tomto ohľade však knižnica wxWidgets poskytuje len veľmi malé množstvo ovládacích prvkov s požadovanou funkcionalitou – práve preto že knižnica sa snaží používať natívne ovládacie prvky všade tam, kde je to len možné. Preto bolo nutné navrhnúť pre aplikáciu triedy, ktoré by túto funkcionalitu poskytovali.

Trieda ownerDrawnSearchCtrl

Aj keď už wxWidgets obsahujú ovládací prvok určený na vyhľadávanie, tento má niekoľko nedostatkov, ako napríklad nemožnosť zmeniť farbu pozadia prvku. Bola tak vytvorená trieda `ownerDrawnSearchCtrl` ktorej štruktúru môžeme vidieť na nasledujúcom obrázku.

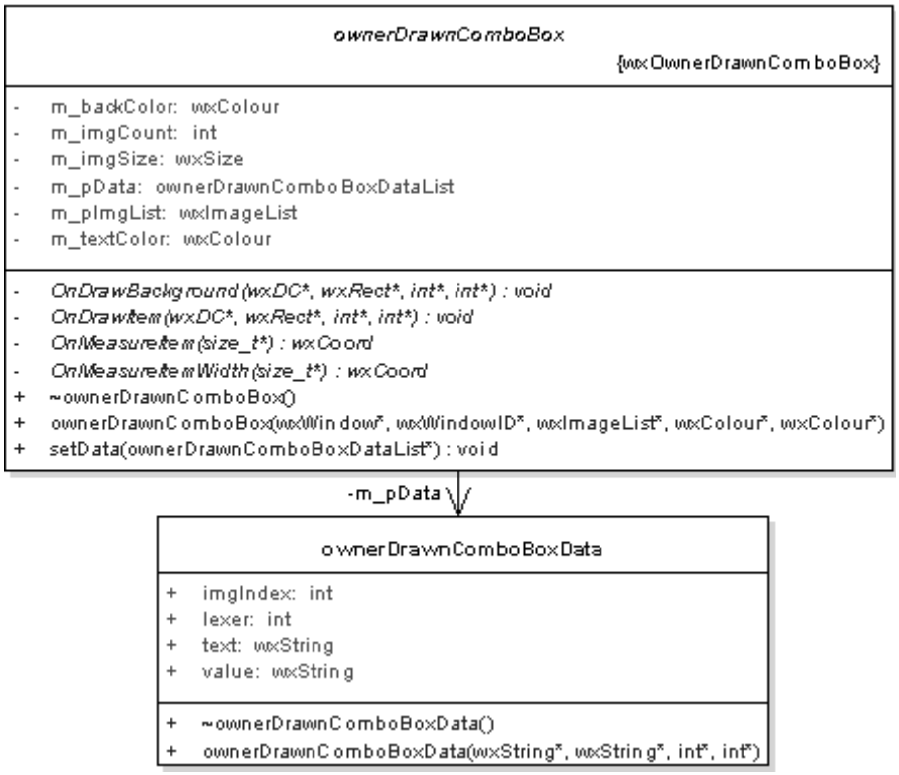


Obr. 10. Štruktúra triedy ownerDrawnSearchCtrl

Trieda implementuje ovládací prvok obsahujúci textové vyhľadávacie pole predstavované triedou `wxTextCtrl` u ktorého môžeme zmeniť farbu pozadia a textu, prípadne zobraziť ľubovoľnú ikonu vedľa tohto vyhľadávacieho poľa.

Trieda ownerDrawnComboBox:

Trieda reprezentujúca ovládací prvok Combo Box s vlastným kreslením jednotlivých položiek zoznamu a možnosťou priradiť každej položke inú ikonu. Vo wxWidgets sa už nachádza prvok s podobnou funkciou – `wxBitmapComboBox`, tento však neumožňuje zmeniť farbu pozadia a textu vyskakovacieho zoznamu. Na tento účel poskytujú wxWidgets triedu `wxOwnerDrawnComboBox`, z ktorej bola trieda `ownerDrawnComboBox` odvodená. Štruktúru tejto triedy je možné vidieť na nasledujúcom obrázku.



Obr. 11. Štruktúra triedy ownerDrawnComboBox

Ako môžeme vidieť z členských premenných triedy, trieda umožňuje nastavenie farby pozadia (`m_backColor`) a textu (`m_textColor`) a taktiež priradenie zoznamu obrázkov (`m_imgList`). Je tu tiež obsiahnutý zoznam inštancií triedy `ownerDrawnComboBoxData`, ktorá predstavuje jednu položku zoznamu Combo Boxu. Jej obsahom sú index obrázku do zoznamu obrázkov (`imgIndex`), zobrazovaný text (`text`) a textová hodnota (`value`). Nachádza sa tu taktiež index do zoznamu zvýrazňovačov zdrojového kódu (`lexer`).

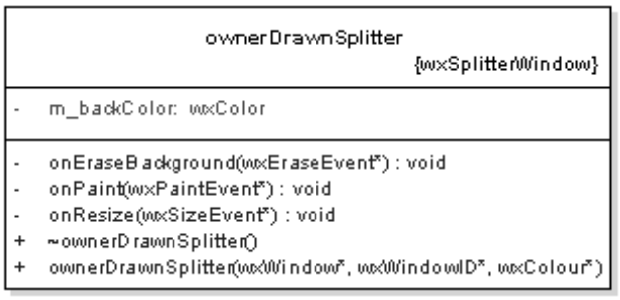
Trieda wxSTC

Trieda zapuzdrujúca ovládací prvok wxWidgets `wxStyledTextCtrl` predstavujúci textový ovládací prvok s možnosťou zvýrazňovania syntaxe mnohých programovacích jazykov. V členských premenných obsahuje trieda inštanciu triedy `styleLexer` ktorá určuje vzhľad zvýrazňovaného textu. Členské metódy potom umožňujú napríklad vyhľadávanie v texte alebo načítanie textu zo súboru. U prvku môžeme tiež nastaviť farbu pozadia, veľkosť písma a farbu označeného textu. Tieto hodnoty sa nachádzajú v štruktúre `wxSTCStyle`.

Obr. 12. Štruktúra triedy wxSTC

Trieda ownerDrawnSplitter

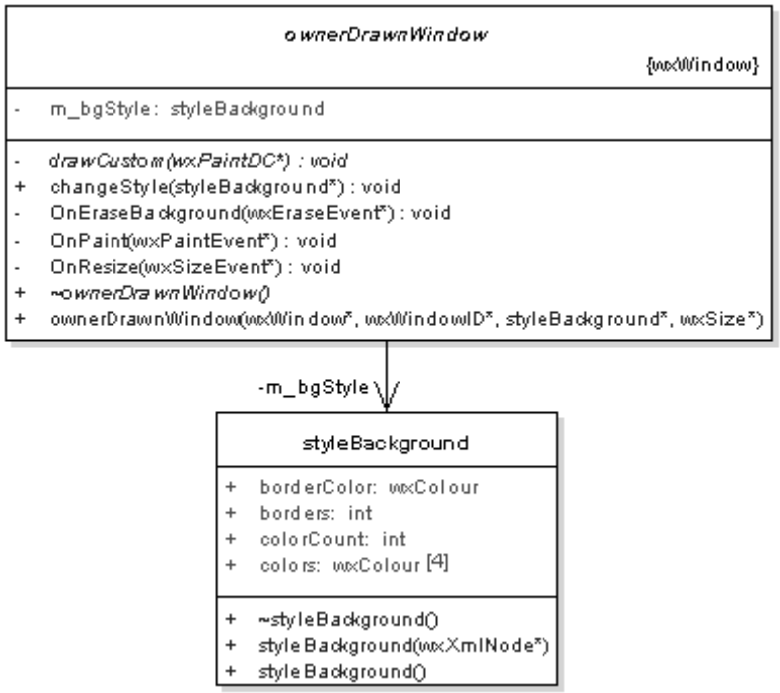
Táto trieda predstavuje predeľovač uprostred okna aplikácie, pomocou ktorého môžeme zmeniť pomer veľkosti ľavého panelu a obsahovej časti. U tohto prvku môžeme zmeniť farbu pozadia. Trieda má nasledujúcu štruktúru:



Obr. 13. Štruktúra triedy ownerDrawnSplitter

Trieda ownerDrawnWindow

Trieda predstavujúca základnú triedu z ktorej budú neskôr odvodené triedy ďalších ovládacích prvkov. Umožňuje vykresľovanie rôznych farebných prechodov na pozadí a zobrazenie vybraných okrajov okolo prvku. Môžeme si tu všimnúť jedinú členskú premennú ktorá určuje vzhľad tohto prvku. Obsah tejto premennej môže byť načítaný priamo z konfiguračného súboru. Viac informácií o triedach slúžiacich na uchovávanie a načítanie dát môžete nájsť v nasledujúcej kapitole.

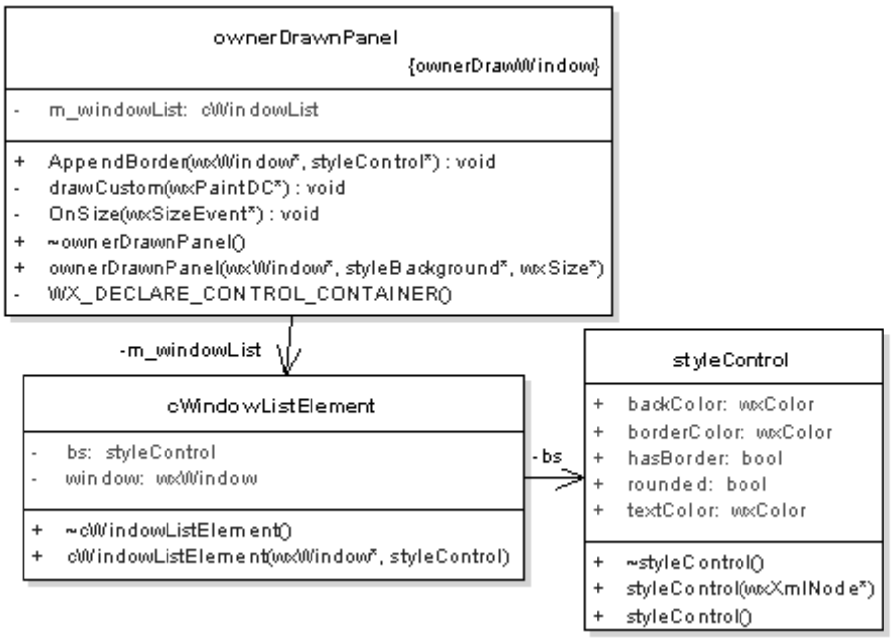


Obr. 14. Štruktúra triedy ownerDrawWindow

Trieda ownerDrawnPanel

Trieda odvodená z triedy `ownerDrawWindow` predstavujúca kontajnerový panel, na ktorý môžu byť umiestené ďalšie ovládacie prvky. Umožnuje taktiež vykreslenie klasických,

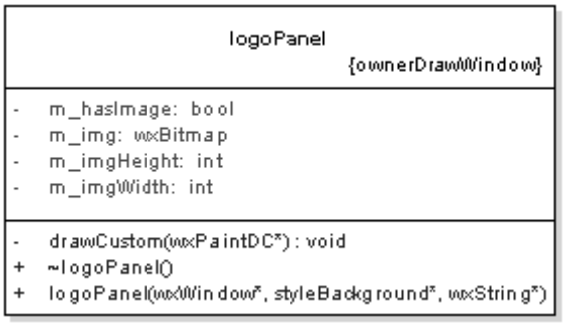
alebo zaoblených okrajov určitej farby okolo vybraných prvkov. Tieto prvky je možné pridať do zoznamu pomocou metódy `AppendBorder`, ktorá ako parametre očakáva ukazateľ na inštanciu ovládacieho prvku a požadovaný štýl okraja pre tento prvok.



Obr. 15. Štruktúra triedy ownerDrawnPanel

Trieda logoPanel

Ďalšia z tried odvodených z triedy `ownerDrawWindow`, ktorá okrem funkcionality základnej triedy umožňuje vykresliť obrázok – logo prezentovaného obsahu na pozadí prvku.

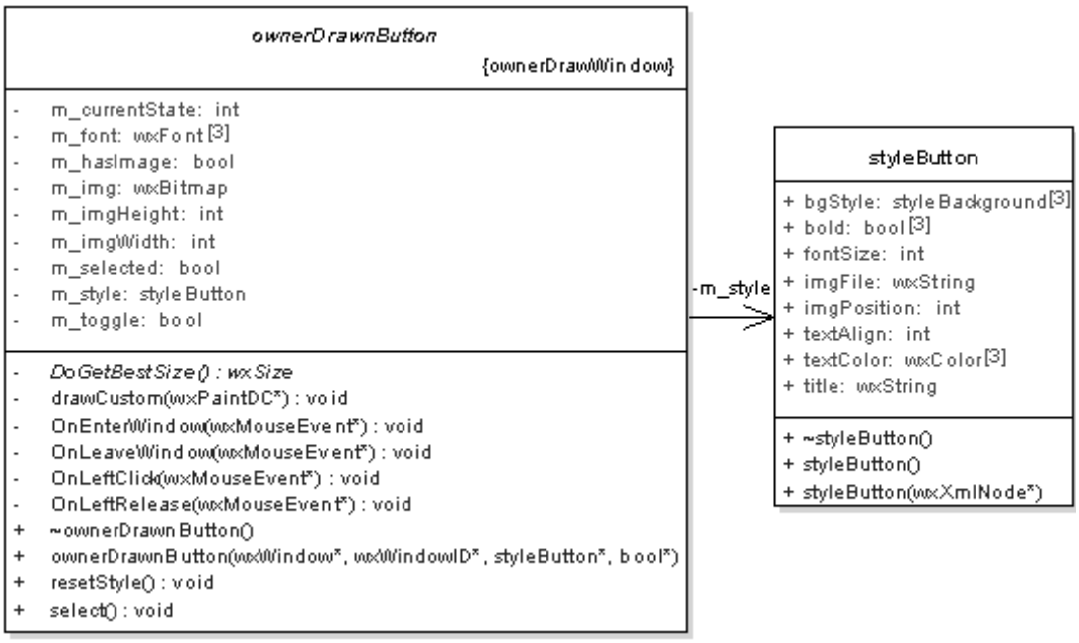


Obr. 16. Štruktúra triedy logoPanel

Trieda ownerDrawnButton

Trieda zapuzdrujúca funkcionalitu klasického tlačítka. Tu môžeme definovať štýl pozadia a textu pre jeho tri stavy: normálny stav kedy sa s tlačítkom nič nedeje, stav keď sa nad

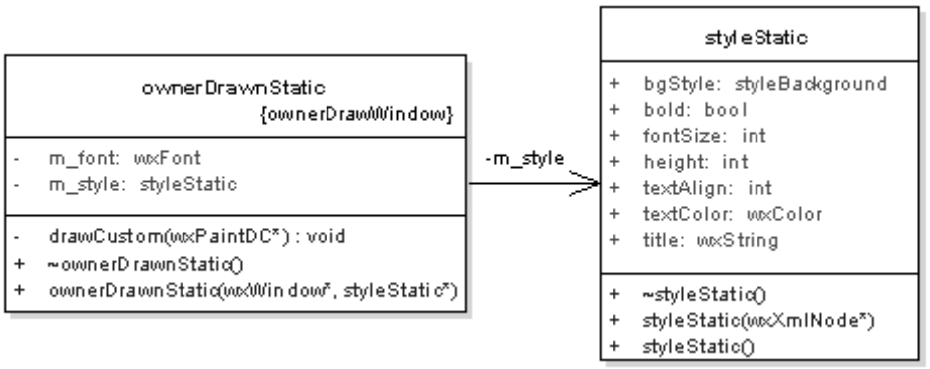
jeho povrchom nachádza kurzor myši a keď je tlačítko vybrané. Ďalej môžeme určiť ikonu a stranu na ktorej sa bude táto ikona nachádzať. Pri jeho vytváraní môžeme tiež určiť či bude mať tlačítko klasickú funkciu, alebo bude prepínacie.



Obr. 17. Štruktúra triedy ownerDrawnButton

Trieda ownerDrawnStatic

Trieda umožňujúca zobraziť text v okne aplikácie. U tohto textu môžeme definovať parametre ako veľkosť, zarovnanie, hrúbka alebo farba.



Obr. 18. Štruktúra triedy ownerDrawnStatic

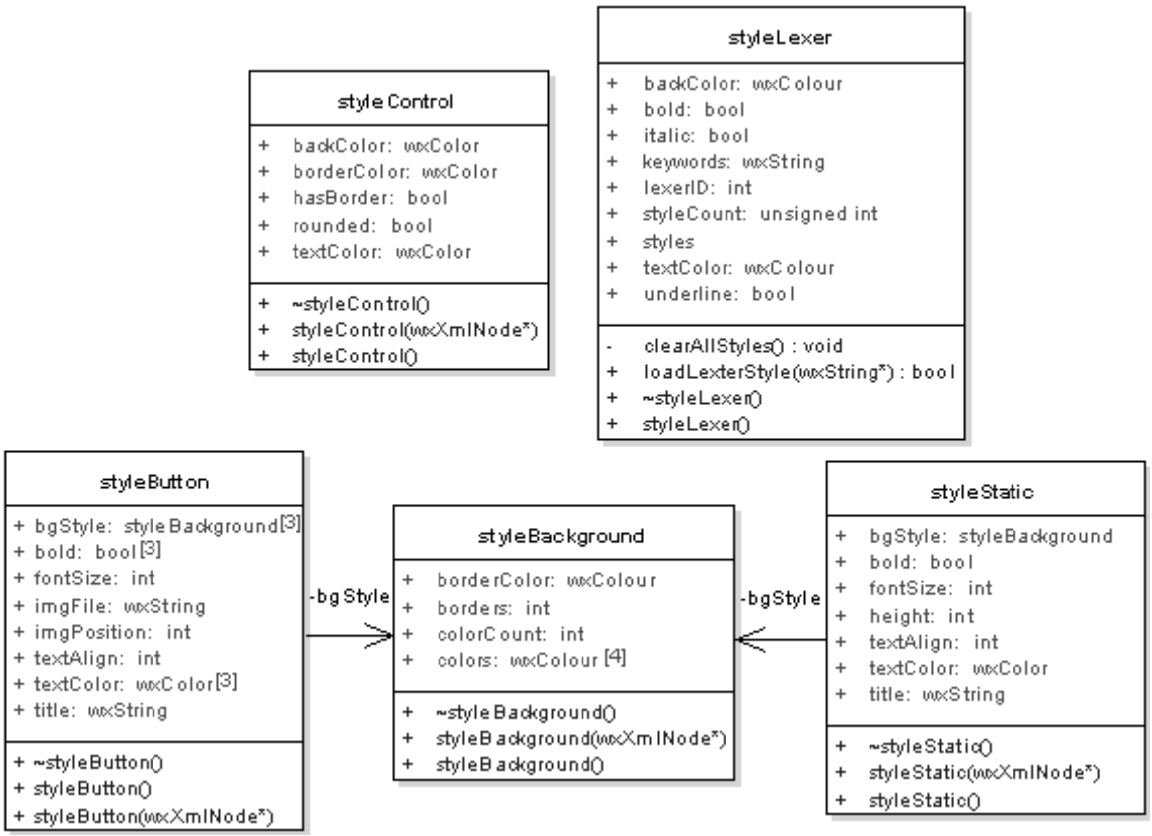
3.2 Triedy na uchovávanie a načítanie dát

Ako už bolo viackrát spomínané, aplikácia umožňuje načítať dáta určujúce jej vzhľad a obsah z externých súborov (v ďalšom texte nazývané ako demo súbory). Pre formát

týchto súborov bol vybraný jazyk XML. Jazyk XML je všeobecný značkovací jazyk ktorý bol vyvinutý a štandardizovaný konzorciom W3C. Umožňuje jednoduché vytváranie vlastných značiek pre rôzne účely. Tento formát bol vybraný hlavne z týchto dôvodov:

- Pratformne nezávislý, samopopisný, otvorený textový formát.
- Podpora Unicode – dáta tak môžu byť uložené v mnohých jazykoch.
- Možnosť definovať vlastné značky.
- wxWidgets už obsahuje triedy umožňujúce jednoduchú prácu s týmto formátom.

Dáta načítané z tohto formátu je však potrebné v aplikácii uchovávať. Boli tak vytvorené triedy, z ktorých každá uchováva rôzne špecifické dáta. Väčšina týchto tried je určená na načítanie a uchovávanie informácií o vzhľade jednotlivých ovládacích prvkov z formátu XML. Štruktúru týchto triedy môžeme vidieť na nasledujúcom obrázku:



Obr. 19. Štruktúra tried uchovávajúcich vzhľad ovládacích prvkov

Trieda styleControl

Trieda určuje vzhľad viacerých ovládacích prvkov. Umožňuje napríklad určiť farbu pozadia a textu prvku, prípadne určiť vzhľad a farbu okrajov (žiadny / klasický / zaoblený).

Trieda styleLexer

Trieda zapuzdrujúca vzhľad zvýrazňovača textu pre ovládací prvok `wxSTC`. Umožňuje definovať kľúčové slová daného programovacieho jazyka, určiť farbu a štýl textu apod.

Trieda styleBackground

Trieda určujúca vzhľad pozadia prvku. Môžeme definovať až štyri rôzne farby pozadia a farbu pre okraj u ktorého môžeme určiť na ktorých konkrétnych stranách bude zobrazený.

Trieda styleButton

Štýl pre ovládací prvok `ownerDrawnButton` predstavujúci tlačítko. Obsahuje tri inštancie triedy `styleBackground`, pre každý stav tlačítka jednu. Ďalej tu môžeme určiť zobrazovaný text a jeho štýl (zarovnanie, hrúbka, veľkosť, farba) a ikonu.

Trieda styleStatic

Trieda uchovávajúca vzhľad ovládacieho prvku `ownerDrawnStatic` určeného na zobrazenie textu v okne aplikácie. Umožňuje tiež určiť vzhľad pozadia a textu tohto ovládacieho prvku.

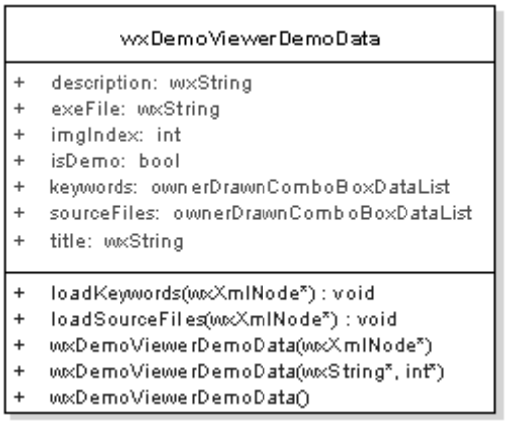
Ako si môžeme všimnúť, všetky tieto triedy obsahujú konštruktor, ktorý ako svoj jediný parameter očakáva ukazateľ na XML element. Vytváranie inštancií týchto tried potom prebieha v cykle v ktorom je z názvu XML elementu zistená konkrétna trieda umožňujúca správne načítanie dát, ktorej bude ako parameter konštruktora predaný ukazateľ na tento element. Konštruktor sa tak postará o správnu identifikáciu parametrov tohto elementu a ich uloženie do členských premenných triedy. Napríklad z nasledujúceho XML kódu budú vytvorené dve inštancie triedy `styleBackground` (element `background`) a jedna inštancia triedy `styleControl` (`control`), `styleStatic` (`static`) a `styleButton` (`button`):

```
1 <windowStyles>
2   <background name="mainTree" colorA="#DAE5F4" />
3   <background name="mainSearch" colorA="#C3D7EF" colorB="#D0E5FF" />
4
5   <control name="mainTree" backColor="#FFFFFF" borderColor="#99bbe8"
    hasBorder="true" rounded="false" />
6
```

```
 7   <static name="mainTitle" colorA="#D5E4F5" title="wxWidgets"
     textColor="#15428B" fontSize="10" bold="true" />
 8
 9   <button name="tabDescription" title="Description"
     icon="skinFiles/info.png" >
10     <stateNormal colorA="#dae6f4" textColor="#000000" />
11     <stateHover colorA="#b6cbe4" textColor="#000000" />
12     <stateSelected colorA="#b6cbe4" textColor="#000000" />
13   </button>
14 </windowStyles>
```

Trieda wxDemoViewerDemoData

Trieda slúžiaca na uchovávanie informácií o jednotlivých vzorových aplikáciách. Každej aplikácii môžeme priradiť jej názov, popis, meno spustiteľného súboru, ikonu, zoznam kľúčových slov a zdrojových súborov. Môžeme tiež určiť či sa jedná o vzorovú aplikáciu alebo o zložku v stromovej štruktúre. Štruktúru triedy môžeme vidieť na nasledujúcom obrázku:



Obr. 20. Štruktúra triedy wxDemoViewerDemoData

Trieda wxDemoViewerData

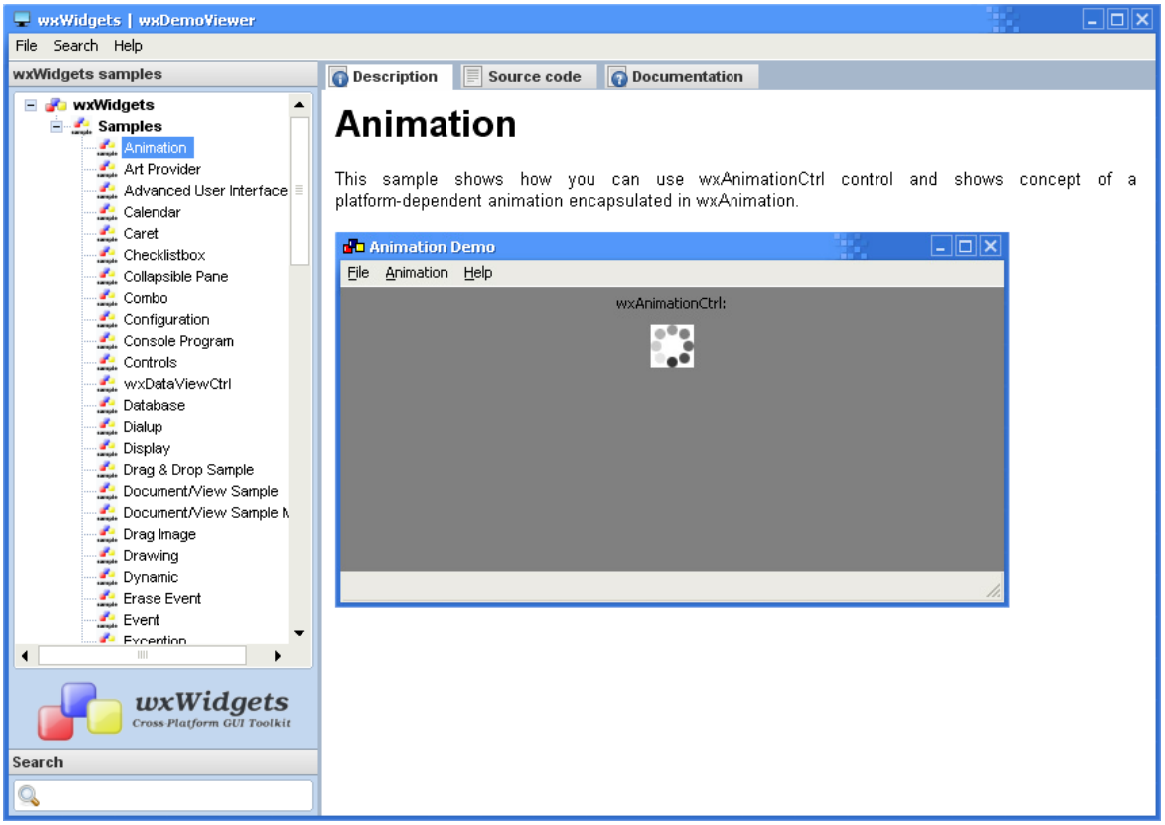
Trieda zapuzdrujúca všetky dáta načítané z XML súboru. Obsahuje inštancie všetkých vyššie spomínaných tried pre každý ovládací prvok, inštanciu triedy `wxDemoViewerDemoData` a pod. Stará sa o identifikáciu XML elementov, ku ktorým priraďuje správne triedy. Pre urýchlenie tohto procesu používa hashovaciu mapu pomocou ktorej k určitému reťazcu priradí jeho číselný identifikátor. Štruktúra triedy pre jej veľkosť nie je uvedená.

4 OVLÁDANIE APLIKÁCIE

Aplikáciu spustíme pomocou spustiteľného súboru wxDemoViewerApp. V prípade, že sa v adresári so spustiteľným súborom nachádza súbor *„autostart.conf“*, dôjde k automatickému otvoreniu demo súboru na ktorý odkazuje cesta uvedená v tomto konfiguračnom súbore. V opačnom prípade môžeme pomocou štandardného dialógu operačného systému pre výber súboru vybrať požadovaný demo súbor.

Po načítaní súboru vidíme na ľavej strane panel obsahujúci zoznam všetkých dostupných vzorových aplikácií. V spodnej časti sa potom nachádza vyhľadávacie pole pomocou ktorého môžeme v tomto zozname vyhľadávať. Na pravej strane sa môžeme pomocou záložiek v hornej časti prepínať medzi krátkym popisom wxWidgets (Description) a dokumentáciou (Documentation) (Obr. 8).

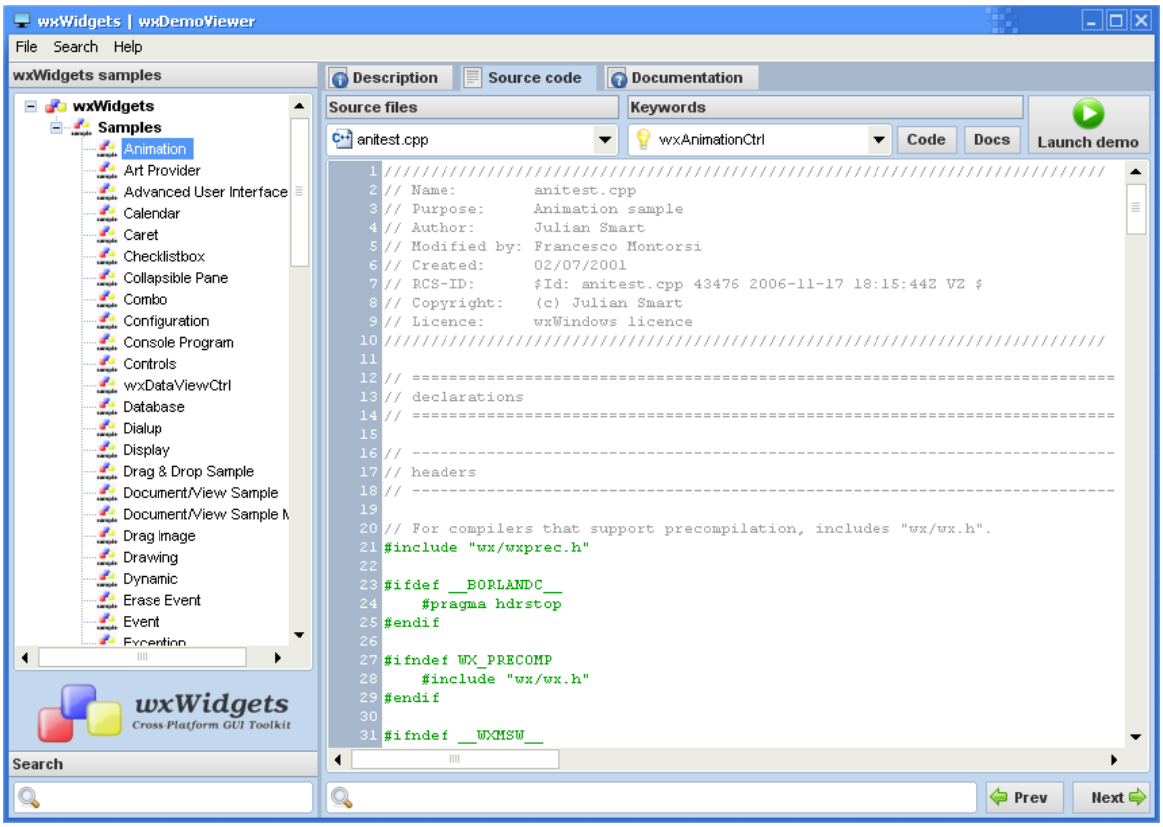
Po vybratí niektorej zo vzorových aplikácií v ľavej časti okna je zobrazený krátky popis a obrázok tejto aplikácie. Objaví sa tiež záložka s prehliadačom zdrojového kódu (Source code):



Obr. 21. wxDemoViewer – popis vzorovej aplikácie

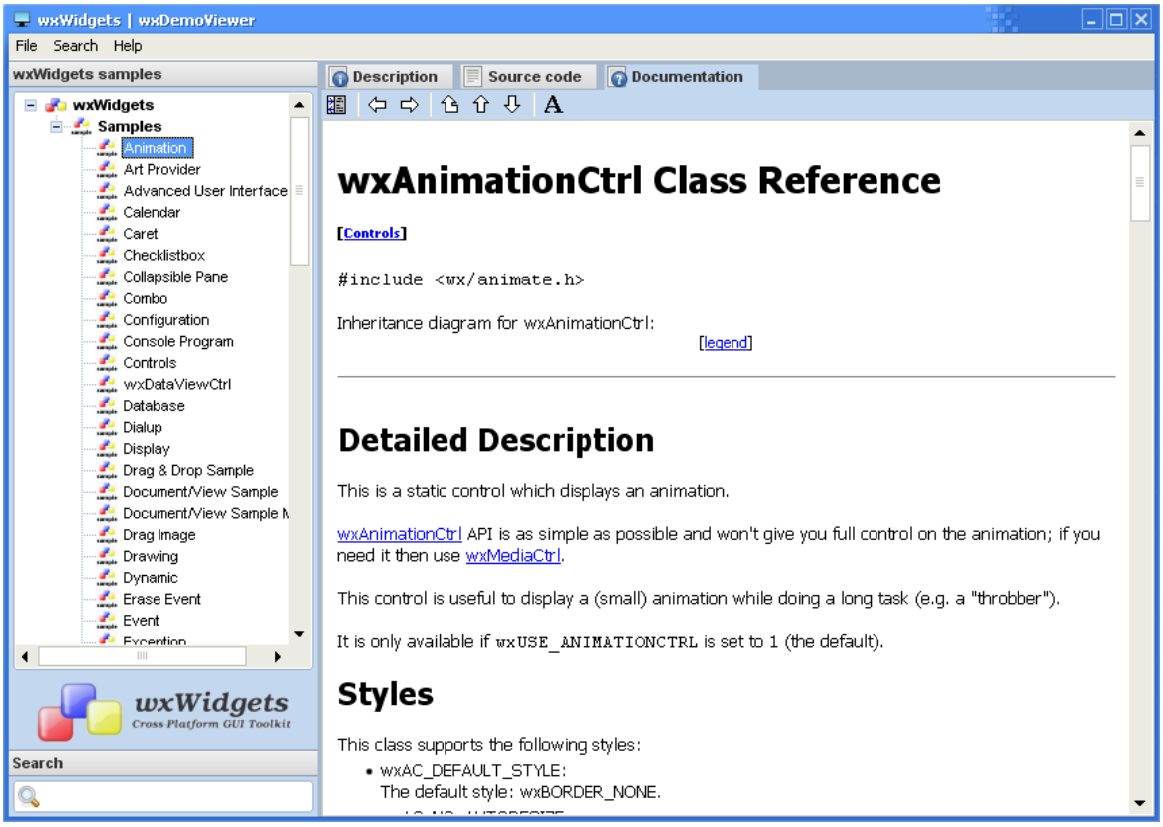
V prehliadači zdrojového kódu (Obr. 22.) môžeme v hornej časti zľava doprava vidieť pole s výberom aktuálne zobrazeného zdrojového súboru (Source files), pole na výber

niektorého z kľúčových slov (Keywords), tlačítka slúžiace na vyhľadanie zvoleného kľúčového slova v zdrojovom kóde (Code) alebo v dokumentácii (Docs). Úplne napravo sa nachádza tlačítko slúžiace na spustenie aktuálnej vzorovej aplikácie – v prípade ak je už spustená iná vzorová aplikácia, dôjde k jej ukončeniu. V strednej časti je potom zobrazený zdrojový kód zvoleného súboru. Kód je zobrazený so zvýraznením syntaxe daného programovacieho jazyka (v našom prípade jazyk C++) a určený len na čítanie. V kóde je možné po vybratí časti textu použiť položku hlavného menu Help -> Documentation, alebo klávesovú skratku F1 k vyhľadaniu tohto textu v dokumentácii. V spodnej časti môžeme nájsť pole určené na vyhľadávanie v zdrojovom kóde. Pre vyhľadanie nasledujúceho výskytu daného slova môžeme použiť tlačítko Next alebo klávesovú skratku F3, prípadne tlačítko Prev pre vyhľadanie predchádzajúceho výskytu.



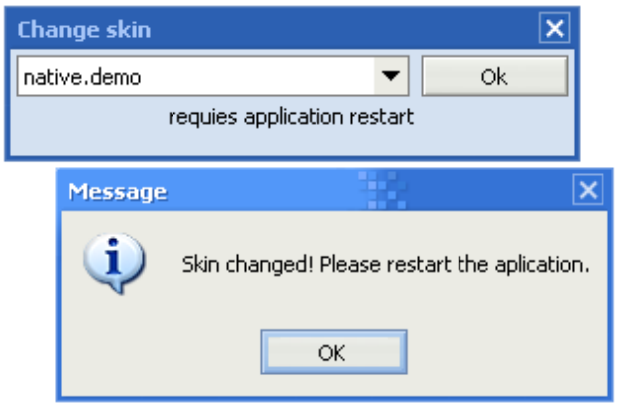
Obr. 22. wxDemoViewer – prehliadač zdrojového kódu

Pod poslednou záložkou (Documentation) sa nachádza prehliadač dokumentácie wxWidgets. Tu je možné po kliknutí na ľavé tlačítko na paneli nástrojov zobraziť stromovú štruktúru dokumentácie a vyhľadávač.



Obr. 23. wxDemoViewer – prehliadač dokumentácie

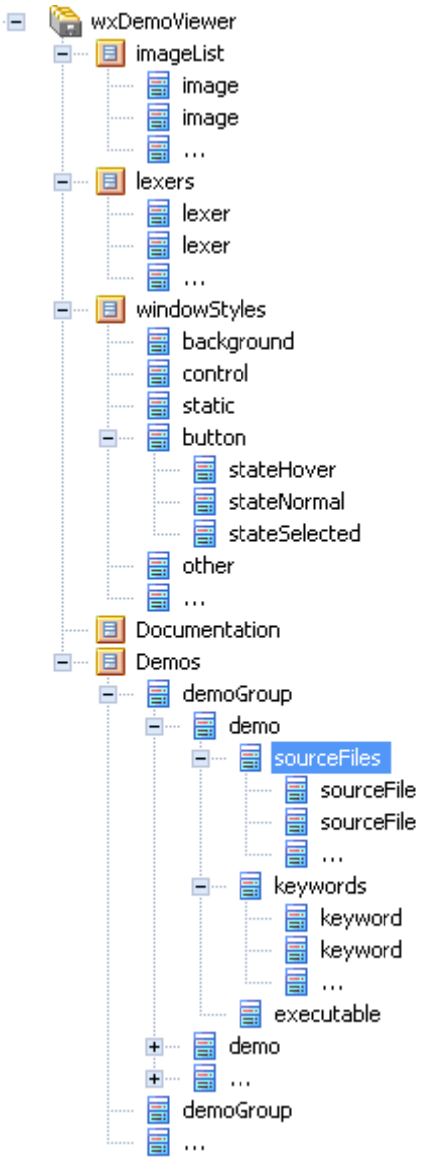
Pomocou menu File -> Change skin je tiež možné vybrať niektorý z dostupných vzhľadov aplikácie. K zmene vzhľadu dôjde pri nasledujúcom spustení aplikácie, na čo budeme upozornení.



Obr. 24. wxDemoViewer – zmena vzhľadu aplikácie

5 ŠTRUKTÚRA DEMO SÚBORU

Ako už bolo spomenuté, aplikácia získava dáta z externých súborov ktorých obsah je zapísaný pomocou značkovacieho jazyka XML. Dáta v týchto súboroch sú organizované do stromovej štruktúry, ktorej zjednodušenú podobu môžeme vidieť na nasledujúcom obrázku:



Obr. 25. Štruktúra demo súboru

V nasledujúcich kapitolách budú detailnejšie popísané jednotlivé XML elementy, ich parametre a funkcia. Parametre týchto elementov môžu obsahovať rôzne dátové typy ktorých popis sa nachádza v nasledujúcej tabuľke:

Tab. 1. Popis dátových typov používaných v demo súboroch

Dátový typ	Poznámka
Číslo	Celé číslo bez desatinnej čiarky / bodky.
Text	Ak je v texte potrebné použiť niektorý z kontrolných znakov jazyka XML, je nutné tento znak previesť na tzv. XML entitu. Napríklad pre znak < je použitá entita <, pre znak " entita " a podobne.
Logická	Logická hodnota Áno alebo Nie zapísaná v anglickom jazyku malými písmenami: Áno = true, Nie = false.
Farba	Farba zapísaná vo formáte HTML: #RRGGBB, teda napríklad #FFFC00 pre žltú farbu. Veľkosť písmen nie je podstatná.
Cesta	Cesta k súborom, zapísaná v tvare používanom na operačných systémoch založených na Unixe. Musí byť v relatívnom tvare, pričom ako koreňový adresár je použitý adresár v ktorom sa tento demo súbor nachádza.

Tvorbu XML súboru začneme vytvorením nového prázdneho súboru. Na začiatok tohto súboru vložíme deklaračnú značku XML, ktorá určuje kódovanie súboru a akú verziu jazyka XML súbor používa.

```
1 <?xml version="1.0" encoding="utf-8"?>
```

Následne za túto značku vložíme párovú značku `<wxDemoViewer>`, ktorá predstavuje koreňový element pre všetky dáta.

```
2 <wxDemoViewer>
3 </wxDemoViewer>
```

5.1 Definícia vzhľadu aplikácie – skinu

5.1.1 Zoznam obrázkov

Pomocou zoznamu obrázkov môžeme výrazne urýchliť ich načítanie a zároveň je s jeho použitím ušetrená veľká časť operačnej pamäte – obrázky ktoré sú v aplikácii použité viackrát sa touto cestou načítajú iba jeden raz. Poradie v akom sú obrázky uvedené nám potom určuje ich index – kde prvý obrázok začína indexom 0, druhý obrázok indexom 1 atď. U načítaných obrázkov potom platí, že môžu byť v ľubovoľnom formáte ktorý je podporovaný knižnicou wxWidgets (napríklad formát BMP, PNG, JPEG, GIF, PCX, TIFF, TGA alebo ICO). Obrázok taktiež môže obsahovať informáciu o priehľadnosti. Jedinou obmedzujúcou podmienkou je veľkosť obrázku ktorá musí byť rovná 16 x 16 pixelov. Medzi párovú značku `<imageList>` vložíme jednotlivé značky `<image />`, z ktorých každá predstavuje jeden obrázok.

Tab. 2. Parametre elementu image

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
file	Cesta	Žiadna	Cesta k súboru s obrázkom

V našom deme budeme používať niekoľko opakujúcich sa obrázkov (napr. v stromovej štruktúre vzorových aplikácií, v zozname zdrojových súborov a kľúčových slov), preto do demo súboru pridáme nasledujúce riadky a získame tak zoznam obsahujúci päť obrázkov s indexami 0 - 4:

```
4   <imageList>
5     <image file="skinFiles/wxWidgetsIcon.png" />
6     <image file="skinFiles/cppfile.png" />
7     <image file="skinFiles/hfile.png" />
8     <image file="skinFiles/keyword.png" />
9     <image file="skinFiles/sampleIcon.png" />
10  </imageList>
```

5.1.2 Zvýrazňovač zdrojového kódu

Pre zobrazenie zdrojového kódu so zvýraznením syntaxe určitého programovacieho jazyka používa aplikácia ovládací prvok predstavovaný triedou wxStyledTextCtrl, ktorá je obalom okolo komponenty nazvanej Scintilla (http://www.scintilla.org/). Určenie štýlov textu, ktoré má tento ovládací prvok používať vykonáme pomocou ďalšieho konfiguračného XML súboru. Najskôr je však potrebné zistiť parametre daného programovacieho jazyka ktoré tento ovládací prvok požaduje. Urobíme tak otvorením zdrojového súboru SciLexer.h ktorý môžeme nájsť v distribúcii komponenty Scintilla alebo v adresári s prídavkami k wxWidgets (wxWidgets28/contrib/src/stc/scintilla/include/). Vo vzorových aplikáciách budeme chcieť zobrazovať zdrojové kódy vytvorené v jazyku C++, preto z tohto súboru potrebujeme zistiť čo najviac informácií o tomto jazyku. Prvým údajom ktorý zistíme je vnútorný identifikátor jazyka C++. Ten môžeme nájsť pod definíciou `SCLEX_CPP` a jeho hodnotou je číslo 3:

```
19 #define SCLEX_PYTHON 2
20 #define SCLEX_CPP 3
21 #define SCLEX_HTML 4
```

Podobným spôsobom môžeme nájsť aj identifikátory ostatných programovacích jazykov.

Následne potrebujeme získať informácie o jednotlivých syntaktických štýloch dostupných v jazyku C++ (kľúčové slová, komentáre, preprocesor, čísla, reťazce atď). Tieto informácie sa taktiež nachádzajú v súbore SciLexer.h:

```
111 #define SCE_C_DEFAULT 0 // normálny text
112 #define SCE_C_COMMENT 1 // komentár
113 #define SCE_C_COMMENTLINE 2 // riadkový komentár
114 #define SCE_C_COMMENTDOC 3 // dokumentácia
115 #define SCE_C_NUMBER 4 // číslo
116 #define SCE_C_WORD 5 // kľúčové slová
117 #define SCE_C_STRING 6 // reťazec
118 #define SCE_C_CHARACTER 7 // znak
119 #define SCE_C_UUID 8
120 #define SCE_C_PREPROCESSOR 9 // preprocesor
121 #define SCE_C_OPERATOR 10 // operátory ( + - * / …)
122 #define SCE_C_IDENTIFIER 11 // identifikátor
123 #define SCE_C_STRINGEOL 12 // viacriadkový reťazec
124 #define SCE_C_VERBATIM 13
125 #define SCE_C_REGEX 14 // regulárny výrez
126 #define SCE_C_COMMENTLINEDOC 15 // riadková dokumentácia
127 #define SCE_C_WORD2 16 // dodatočné kľúčové slová
128 #define SCE_C_COMMENTDOCKEYWORD 17 // kľúčové slovo v komentári
129 #define SCE_C_COMMENTDOCKEYWORDERROR 18 // ks v riadkovom komentári
130 #define SCE_C_GLOBALCLASS 19
```

Keď už máme všetky potrebné informácie, môžeme začať vytvárať konfiguračný XML súbor. Ten obsahuje párovú značku `<styleLexer>`, vo vnútri ktorej sa nachádzajú elementy popisujúce jednotlivé štýly daného programovacieho jazyka určené značkou `<style />`.

Tab. 3. Parametre elementu styleLexer

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
defaultBackColor	Farba	#FFFFFF	Prednastavená farba pozadia textu
defaultTextColor	Farba	#000000	Prednastavená farba textu
lexerID	Číslo	Žiadna	Vnútorný identifikátor jazyka
lexerName	Text	Žiadna	Názov zvýrazňovača (nepovinný parameter)

Tab. 4. Parametre elementu style

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
backColor	Farba	Hodnota defaultBackColor	Farba pozadia textu
bold	Logická	Nie	Hrubé písmo
italic	Logická	Nie	Kurzíva
keywords	Text	Žiadna	Kľúčové slová oddelené medzerou
textColor	Farba	Hodnota defaultTextColor	Farba textu
underline	Logická	Nie	Podčiarknuté písmo

Obsah súboru lexerCPP.xml:

```
 1 <?xml version="1.0" encoding="utf-8"?>
 2 <styleLexer lexerName="C++" lexerID="3" defaultTextColor="#000000"
   defaultBackColor="#FFFFFF">
 3   <style /> <!-- Štýl pre ID = 0: SCE_C_DEFAULT -->
 4   <style textColor="#A0A0A0" /> <!-- ID = 1: SCE_C_COMMENT -->
 5   <style textColor="#A0A0A0" /> <!-- ID = 2 -->
 6   <style textColor="#8080FF" bold="1" /> <!-- ... -->
 7   <style textColor="#0000FF" />
 8   <style bold="1" keywords="asm auto bool break case catch char
     class const const_cast continue default delete do double
     dynamic_cast else enum explicit export extern false float for
     friend goto if inline int long mutable namespace new operator
     private protected public register reinterpret_cast return short
     signed sizeof static static_cast struct switch template this throw
     true try typedef typeid typename union unsigned using virtual void
     volatile wchar_t while" />
 9   <style textColor="#FF0000" />
10   <style textColor="#FF0000" />
11   <style />
12   <style textColor="#00A000" />
13   <style />
14   <style />
15   <style textColor="#FF0000" />
16   <style />
17   <style />
18   <style textColor="#8080FF" bold="1" />
19 </styleLexer>
```

Rodičovský element nám umožňuje definovať prednastavenú farbu textu a jeho pozadia. Jednotlivé detské elementy u ktorých chceme použiť túto prednastavenú farbu tak môžu zostať úplne prázdne, je však potrebné ich uviesť kvôli správnemu prideleniu identifikátora. Nemusíme tak urobiť len v prípade, ak sa tieto prázdne elementy nachádzajú na konci zoznamu.

```
343 #if wxUSE_DATEPICKCTRL_GENERIC
344     menuDate->AppendCheckItem(Calendar_DatePicker_Generic,
345                               _T("Use &generic version of the control")
346 #endif // wxUSE_DATEPICKCTRL_GENERIC
347     menuDate->AppendSeparator();
348     menuDate->Append(Calendar_DatePicker_AskDate, _T("&Choose date...\t
349 #endif // wxUSE_DATEPICKCTRL
350
351     // now append the freshly created menu to the menu bar...
352     wxMenuBar *menuBar = new wxMenuBar;
353     menuBar->Append(menuFile, _T("&File"));
354     menuBar->Append(menuCal, _T("&Calendar"));
355 #if wxUSE_DATEPICKCTRL
356     menuBar->Append(menuDate, _T("&Date picker"));
357 #endif // wxUSE_DATEPICKCTRL
358
359     menuBar->Check(Calendar_Cal_Monday, true);
360     menuBar->Check(Calendar_Cal_Holidays, true);
361     menuBar->Check(Calendar_Cal_Month, true);
362     menuBar->Check(Calendar_Cal_Year, true);
363
364 #if wxUSE_DATEPICKCTRL
365     menuBar->Check(Calendar_DatePicker_ShowCentury, true);
366 #endif // wxUSE_DATEPICKCTRL
367
368     // ... and attach this menu bar to the frame
369     SetMenuBar(menuBar);
370
371     m_panel = new MyPanel(this);
372
373 #if wxUSE_STATUSBAR
374     // create a status bar just for fun (by default with 1 pane only)
375     CreateStatusBar(2);
376     SetStatusText( T("Welcome to wxWidgets!"));
```

Obr. 26. Ukážka zvýrazneného zdrojového kódu

Vráťme sa späť k hlavnému demo súboru. Na to aby aplikácia mohla použiť zvýrazňovače zdrojového kódu, je potrebné pridať ich definície do zoznamu zvýrazňovačov zdrojového kódu. Urobíme tak rovnakým spôsobom akým sme definovali zoznam obrázkov – medzi párovú značku `<lexerList>` vložíme jednotlivé značky `<lexer />`, z ktorých každá predstavuje jeden zvýrazňovač zdrojového kódu. Tieto zvýrazňovače budú následne dostupné pod číselnými indexami podobne ako je tomu pri zozname obrázkov.

```
12   <lexerList>
13     <lexer file="skinFiles/lexerCPP.xml" />
14   </lexerList>
```

5.1.3 Štýly ovládacích prvkov

Keďže v aplikácii sa nachádza viacero rozdielnych ovládacích prvkov, sú definície ich vzhľadu rozdelené podľa názvov XML elementov. Každý z týchto elementov obsahuje atribút name ktorý ho jednoznačne spája s ovládacím prvkom v aplikácii. Tieto elementy sa nachádzajú vo vnúti párovej značky `<windowStyles>`.

Element background

Tento element určuje štýl ovládacieho prvku `ownerDrawnPanel`. Pomocou jeho parametrov môžeme určiť štýl pozadia a okrajov.

Tab. 5. Parametre elementu background

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
borderColor	Farba	#000000	Farba okrajov
borders	Text	Žiadna	Zobrazované okraje - kombinácia písmen T, R, B, L
colorA	Farba	#D0D0D0	Prvá farba pozadia
colorB	Farba	Žiadna	Druhá farba pozadia
colorC	Farba	Žiadna	Tretia farba pozadia
colorD	Farba	Žiadna	Štvrtá farba pozadia

Štýl pozadia prvku je možné určiť pomocou štyroch farieb. V prípade ak je definovaná len jedna farba, bude mať ovládací prvok pozadie tejto farby. Ak sú definované farby dve, ovládací prvok bude mať pozadie tvorené farebným prechodom z `colorA` do `colorB`. V prípade troch farieb to bude prechod `colorA --> colorB -> colorC` a v prípade farieb štyroch prechod `colorA --> colorB -> colorC -> colorD`.

Taktiež môžeme určiť ktoré okraje budú zobrazené. Urobíme tak ľubovoľnou kombináciou písmen `T`, `R`, `B`, `L` v paremetri `borders`. Písmeno `T` značí vrchný okraj (Top), písmeno `R` okraj pravý (Right), písmeno `B` potom okraj spodný (Bottom) a na záver písmeno `L` okraj ľavý (Left). Na poradí ani veľkosti písmen nezáleží. Napríklad kombinácia `RlT` určí, že bude zobrazený pravý, ľavý a vrchný okraj.

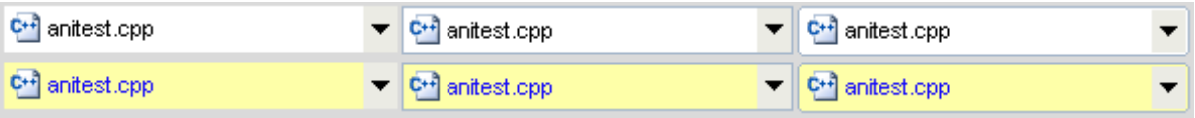
Obr. 27. Ukážka rôznych štýlov pozadia

Element control

Tento element umožňuje určiť štýl okrajov ovládacích prvkov (žiadne / klasické / zaoblené), farbu pozadia a textu.

Tab. 6. Parametre elementu control

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
backColor	Farba	#FFFFFF	Farba pozadia
borderColor	Farba	#000000	Farba okrajov
hasBorder	Logická	Nie	Má okraj?
rounded	Logická	Nie	Je okraj zaoblený?
textColor	Farba	#000000	Farba textu



Obr. 28. Ukážka rôznych štýlov okrajov

Element static

Element popisujúci vzhľad ovládacieho prvku `ownerDrawnStatic` slúžiaceho na zobrazenie textu. U tohto elementu môžeme určiť štýl pozadia rovnakým spôsobom ako pri elemente `background`. Naviac môžeme určiť parametre textu. Konkrétne jeho farbu, veľkosť, zarovnanie a hrúbku.

Tab. 7. Parametre elementu static

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
bold	Logická	Áno	Hrubé písmo
fontSize	Číslo	9	Veľkosť textu
height	Číslo	20	Výška ovládacieho prvku
textAlign	Text	left	Zarovnanie textu
textColor	Farba	#000000	Farba textu
title	Text	Žiadna	Zobrazovaný text
* Pozn. tabuľka ďalej obsahuje rovnaké parametre ako element background			

Parameter zarovnanie textu môže nadobúdať hodnoty `left` (zarovnanie doľava), `right` (zarovnanie doprava) alebo hodnotu `center` (zarovnanie na stred). Hodnotu `title` obsahujú viaceré elementy v XML súbore. Môžeme tak vytvoriť demo súbory s rovnakým vzhľadom a obsahom, no vo viacerých jazykových mutáciách. Urobíme tak vytvorením kópie demo súboru a preložení týchto textov.

Obr. 29. Ukážka rôznych štýlov ovládacieho prvku ownerDrawnStatic

Element button

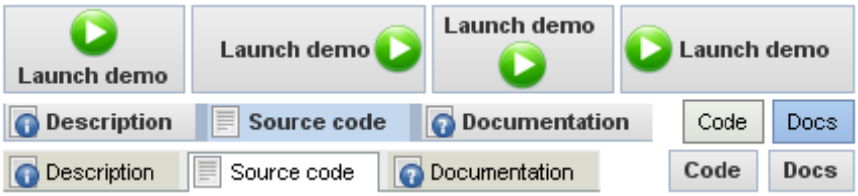
Pomocou tohto elementu môžeme popísať vzhľad tlačítka. Na rozdiel od predchádzajúcich elementov obsahuje tento element ďalšie detské elementy určujúce jeho vzhľad v rôznych stavoch. Element `stateNormal` určuje vzhľad v tlačítka v normálnom stave, tj. keď sa s tlačítkom nič nedeje. Element `stateHover` popisuje vzhľad keď sa nad povrchom tlačítka nachádza kurzor myši a element `stateSelected` určuje vzhľad keď je tlačítko vybrané – užívateľ naň klikol myšou.

Tab. 8. Parametre elementu button

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
fontSize	Číslo	9	Veľkosť textu
icon	Cesta	Žiadna	Cesta k súboru s obrázkom
iconPosition	Text	left	Zarovnanie obrázku
textAlign	Text	left	Zarovnanie textu (left, right, center) – len ak tlačítko nemá obrázok
title	Text	Žiadna	Zobrazovaný text

Tab. 9. Parametre elementov stateNormal, stateHover a stateSelected

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
bold	Logická	Nie / Nie / Áno	Hrubé písmo
textColor	Farba	#000000	Farba textu
* Pozn. tabuľka ďalej obsahuje rovnaké parametre ako element background			



Obr. 30. Ukážka rôznych štýlov ovládacieho prvku ownerDrawnButton

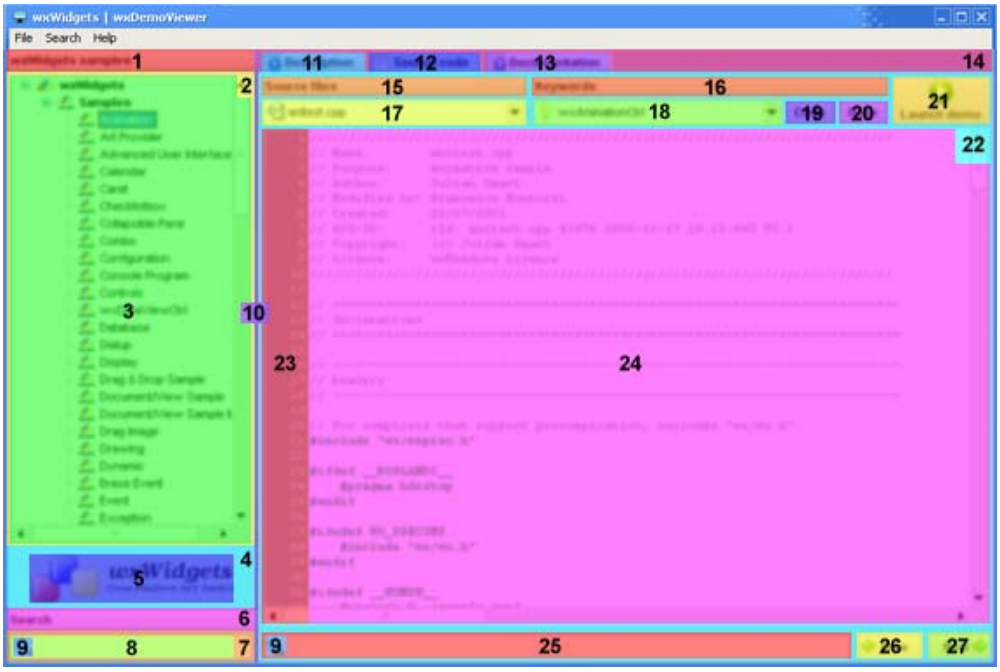
Element other

Tento element popisuje vzhľad ovládacích prvkov, ktorých parametre nie je možné určiť pomocou predchádzajúcich elementov. Medzi tieto patrí napríklad farba textu a pozadia vybraného textu v prehliadači zdrojového kódu, alebo cesta k obrázku s logom. Z tohto dôvodu nemajú tieto elementy predpísanú štruktúru parametrov. Tieto parametre sú tak určené až na základe názvu prvku ku ktorému patria.

Tab. 10. Parametre elementu other rozdelené podľa názvov ovládacích prvkov

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
logoFile – obrázok s logom			
file	Cesta	Žiadna	Cesta k súboru s obrázkom
sourceCodeViewerMore – prehliadač zdrojového kódu			
lineNumberBackColor	Farba	#DADADA	Farba pozadia čísiel riadkov
lineNumberTextColor	Farba	#000000	Farba textu čísiel riadkov
selectionTextColor	Farba	#000000	Farba vybratého textu
selectionBackColor	Farba	#DADADA	Farba pozadia vybratého
fontSize	Číslo	10	Veľkosť textu
searchIcon – ikona zobrazená pri vyhľadávacích poliach			
file	Cesta	Žiadna	Cesta k súboru s obrázkom
splitterBackColor – predeľovač hlavného okna			
backColor	Farba	#DADADA	Farba pozadia
descriptionColors – stránka s popisom vzorovej aplikácie			
backColor	Farba	#FFFFFF	Farba pozadia
textColor	Farba	#000000	Farba textu
docToolbarBackColor – panel nástrojov v prehliadači dokumentácie			
backColor	Farba	#DADADA	Farba pozadia

Ako už bolo spomenuté, jednotlivé elementy sú z prvkami v aplikácii prepojené pomocou parametra `name`. Informácie o tom ktorý identifikátor v tomto parametri spája element s ovládacím prvkom môžeme zistiť pomocou nasledujúceho obrázku a tabuľky:



Obr. 31. Rozdelenie ovládacích prvkov na strane so zdrojovým kódom

Tab. 11. Rozdelenie elementov podľa príslušných ovládacích prvkov

ID	Element	Parameter name	ID	Element	Parameter name
1	static	mainTitle	15	static	sourceFilesTitle
2	background	mainTree	16	static	keywordsTitle
3	control	mainTree	17	control	sourceFiles
4	background	mainLogo	18	control	sourceKeywords
5	other	logoFile	19	button	keywordsSearchCode
6	static	mainSearchTitle	20	button	keywordsSearchDocs
7	background	mainSearch	21	button	launchButton
8	control	mainSearch	22	background	sourceTab
9	other	searchIcon	23	other	sourceCodeViewerMore
10	other	splitterBackColor	24	control	sourceCodeViewer
11	button	tabDescription	25	control	sourceSearch
12	button	tabSource	26	button	sourceSearchPrev
13	button	tabDoc	27	button	sourceSearchNext
14	background	tabsPanel			
* Pozn. v tabuľke sa nenachádzajú identifikátory descriptionColors a docToolbarBackColos					

Na základe týchto identifikátorov a vlastností elementov môžeme do nášho demo súboru medzi párovú značku `<windowStyles>` vložiť príslušné elementy. V nasledujúcom kóde môžete vidieť ich skrátený výpis:

```
16 <windowStyles>
17   <background name="mainTree" colorA="#c3d7ef" />
18   ...
23   <control name="mainTree" backColor="#FFFFFF" borderColor="#a6b7cb"
     textColor="#000000" hasBorder="true" rounded="true" />
24   ...
30   <static name="mainTitle" colorA="#f0f0f0" colorB="#d9d9d9"
     title="wxWidgets samples" textColor="#333333"
     borderColor="#8d9cad" borders="TB" fontSize="8" textAlign="left"
     height="20" bold="true" />
31   ...
35   <button name="tabDescription" title="Description"
     icon="skinFiles/info.png" fontSize="8" textAlign="left"
     iconPosition="left" >
36     <stateNormal colorA="#efefef" colorB="#d9d9d9"
       textColor="#333333" borderColor="#a6b7cb" borders="TRLB"
       bold="true" />
37     <stateHover colorA="#f8f8f8" colorB="#ececec"
       textColor="#333333" borderColor="#a6b7cb" borders="TRLB"
       bold="true" />
38     <stateSelected colorA="#fefefe" textColor="#333333"
       borderColor="#a6b7cb" borders="LTR" bold="true" />
39   </button>
40   ...
83   <other name="logoFile" file="skinFiles/wxWidgetsLogo.png" />
84   ...
90 </windowStyles>
```

5.2 Definícia obsahu aplikácie

5.2.1 Dokumentácia

Aplikácia umožňuje vo vnútornom prehliadači dokumentácie zobraziť dokumentáciu vytvorenú vo formáte Microsoft HTML Help Workshop (prípony súborov .hhp, .hhk a .hhc). K vytvoreniu dokumentácie môžeme použiť nástroj Doxygen (www.doxygen.org), ktorý z komentovaných zdrojových súborov vygeneruje dokumentáciu v rôznych formátoch. Medzi podporované výstupné formáty tohto nástroja patrí napríklad LaTeX, RTF, PDF a aplikáciou wxDemoViewer požadovaný MS HTML Help Workshop. Keďže vygenerovaná dokumentácia obsahuje veľké množstvo malých súborov (až niekoľko tisíc),

je vhodné celú dokumentáciu optimalizovať pre rýchlejšie načítanie. Veľké množstvo súborov minimalizujeme zabalením vygenerovaných súborov do ZIP archívu. Podmienkou je, aby v koreňovej zložke tohto archívu bol prítomný súbor index.hhp. Pre urýchlenie načítania dokumentácie môžeme tiež vytvoriť tzv. kešovací súbor. Ten vytvoríme pomocou nástroja hhp2cached ktorý je súčasťou distribúcie wxWidgets. Vznikne tak súbor index.hhh.cached, ktorý priložíme k pôvodnému súboru index.hhp.

Na to aby aplikácia vedela ktorý súbor dokumentácie má použiť je potrebné do konfiguračného demo súboru pridať element popisujúci cestu k jeho umiestneniu. Urobíme tak vložením značky `<documentation />`, ktorá má jediný parameter `file`:

```
92 <documentation file="docFiles/wxWidgets.zip" />
```

alebo

```
92 <documentation file="docFiles/index.hhp " />
```

Prvý prípad určuje cestu k dokumentácii v archíve, druhý prípad potom k indexovému súboru dokumentácie. V prípade ak element s dokumentáciou neuvedieme, nebudú sa v aplikácii nachádzať funkcie s ňou spojené – napríklad prehliadač dokumentácie, tlačítko pre vyhľadávanie kľúčového slova v dokumentácii a pod.

5.2.2 Vzorové aplikácie

Hlavným cieľom aplikácie je prezentovanie jej obsahu. Týmto obsahom budú pravdepodobne ukážky rôznych vzorových aplikácií daného programovacieho jazyka alebo knižnice. Keďže zdrojové kódy týchto aplikácií môžu byť vytvorené pomocou rôznych programovacích jazykov ktorých výsledkom bude napríklad spustiteľný binárny súbor alebo skript. Prezentačná aplikácia tak musí návrhárovi dema umožniť zahrnúť všetky tieto aspekty do konfiguračného súboru.

Konfiguračný demo súbor umožňuje pomocou niekoľkých elementov určiť všetky vlastnosti prezentovaného obsahu. Jedným z týchto elementov je element `<demos>`. Pomocou jeho parametrov môžeme napríklad určiť príponu spustiteľného súboru na rôznych platformách, prípadne aplikáciu ktorou bude súbor otvorený. Môžeme tak napríklad ošetriť situáciu kedy na operačnom systéme Linux spustiteľná aplikácia nemá žiadnu príponu, na OS Windows má príponu exe a na OS Mac príponu app. Tiež môže nastať situácia, kedy budeme potrebovať spustiť skript pomocou jeho interpretera, no tento bude mať na rôznych operačných systémoch iný názov – parametre elementu demos nám

tak umožnia ošetriť aj túto situáciu. Parameter `title` potom určuje hlavný názov prezentovaného obsahu ktorý bude zobrazený v záhlaví okna aplikácie ako aj v koreni stromovej štruktúry.

Tab. 12. Parametre elementu demos

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
extWin	Text	Žiadna	Prípona spustiteľného súboru na OS Windows
extUnix	Text	Žiadna	Prípona na OS Linux
extMac	Text	Žiadna	Prípona na Mac OS
extOther	Text	Žiadna	Prípona na ostatných OS
openWithWin	Text	Žiadna	Aplikácia ktorou bude súbor otvorený na OS Windows
openWithUnix	Text	Žiadna	Aplikácia OS Linux
openWithMac	Text	Žiadna	Aplikácia na Mac OS
openWithOther	Text	Žiadna	Aplikácia na ostatných OS
title	Text	Žiadna	Názov prezentovaného obsahu
description	Text	Žiadna	Popis prezentovaného obsahu
imageIndex	Číslo	0	Index obrázka v zozname obrázkov

Ďalším elementom popisujúcim vzorové aplikácie je element `<demoGroup>`, pomocou ktorého môžeme vytvárať skupiny aplikácií v stromovej štruktúre. Jeho dva parametre určujú zobrazovaný text a index do zoznamu obrázkov. Vzorové aplikácie potom vkladáme medzi jeho párovú značku.

Tab. 13. Parametre elementu demoGroup

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
imageIndex	Číslo	0	Index obrázka v zozname obrázkov
title	Text	Žiadna	Názov skupiny vzorových aplikácií

Jednotlivé vzorové aplikácie sú predstavované elementom `<demo>`, ktorého atribútmi sú názov a popis vzorovej aplikácie a index do zoznamu obrázkov. Medzi jeho párovú značku vkladáme ďalšie elementy určujúce napríklad zoznam zdrojových súborov `<sourceFiles>` a kľúčových slov `<keywords>`. Tiež sa tu nachádza element

`<executable />` určujúci cestu k spustiteľnému súboru. Tu si treba dávať pozor na zapísanie cesty k súboru bez prípony – tá je už definovaná v elemente `<demos>`.

Tab. 14. Parametre elementu demo

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
description	Text	Žiadna	Popis vzorovej aplikácie
imageIndex	Číslo	0	Index obrázka v zozname obrázkov
title	Text	Žiadna	Názov vzorovej aplikácie

Tab. 15. Parametre elementu sourceFile

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
file	Cesta	Žiadna	Cesta k zdrojovému súboru
imageIndex	Číslo	0	Index obrázka v zozname obrázkov
lexer	Číslo	0	Index zvýrazňovača zdrojového kódu

Tab. 16. Parametre elementu keyword

Názov parametra	Typ hodnoty	Prednastavená hodnota	Popis
imageIndex	Číslo	0	Index obrázka v zozname obrázkov
title	Text	Žiadna	Kľúčové slovo

Príklad definície jednej vzorovej aplikácie môžete vidieť v nasledujúcom výpise:

```
 93 <demo title="Animationn" description="This sample shows how you can
    use wxAnimationCtrl.&lt;br&gt;&lt;br&gt;&lt;img
    src=&quot;demoFiles/animate/animate.jpg&quot;&gt;">
 94   <sourceFiles>
 95     <sourceFile file="demoFiles/animate/anitest.cpp" imageIndex="1"
         lexer="0" />
 96     <sourceFile file="demoFiles/animate/anitest.h" imageIndex="2"
         lexer="0" />
 97   </sourceFiles>
 98   <keywords>
 99     <keyword title="wxAnimationCtrl" imageIndex="3" />
100     <keyword title="wxAnimation" imageIndex="3" />
101   </keywords>
102   <executable file="demoFiles/animate/animate" />
103 </demo>
```

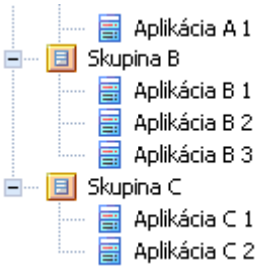
V popise vzorovej aplikácie si môžeme všimnúť časť HTML kódu so zakódovanými entitami ktorý spôsobí vloženie obrázka na koniec textu.

V prípade že v definícii vzorovej aplikácie neuvedieme časť so zdrojovými súbormi, nebude pre túto aplikáciu dostupná záložka s prehliadačom zdrojového kódu – záložka s popisom a dokumentáciou však zostane. Podobne ak neuvedieme definíciu kľúčových slov, nebude funkcionalita s nimi spojená pre túto aplikáciu dostupná.

Stromovú štruktúru vzorových aplikácií určíme vkladaním elementov `<demo>` medzi značky `<demoGroup>`:

```
104 <demo title="Aplikácia A 1"> ... </demo>
105
106 <demogroup title="Skupina B">
107   <demo title="Aplikácia B 1"> ... </demo>
108   <demo title="Aplikácia B 2"> ... </demo>
109   <demo title="Aplikácia B 3"> ... </demo>
110 </ demogroup>
111
112 <demogroup title="Skupina C">
113   <demo title="Aplikácia C 1"> ... </demo>
114   <demo title="Aplikácia C 2"> ... </demo>
115 </ demogroup>
```

Na základe uvedeného kódu tak vznikne nasledujúca štruktúra:



Obr. 32. Príklad stromovej štruktúry

Na základe všetkých vyššie uvedených informácií boli vytvorené demonštračné súbory prezentujúce vlastnosti knižnice wxWidgets. Tieto obsahujú definície všetkých vzorových aplikácií distribuovaných spoločne s knižnicou a dokumentáciu vygenerovanú nástrojom Doxygen z najnovšej verzie zdrojových súborov wxWidgets (2.9.0).

ZÁVER

Teoretická časť práce sa zameriava na popis knižnice wxWidgets, popisuje stručnú históriu jej vývoja a vyzdvihuje jej výhody oproti iným podobne zameraným knižniciam. Venuje sa tiež praktickému popisu vytvárania jednoduchej vzorovej aplikácie a v závere zhodnocuje niektoré z dostupných vývojových nástrojov pre túto knižnicu.

V praktickej časti sa práca venuje tvorbe demonštračnej a dokumentačnej aplikácie pre knižnicu wxWidgets ktorej vytvorenie bolo hlavným cieľom tejto práce. Keďže cieľom aplikácie bude prezentácia jej obsahu, bolo dôležité navrhnúť užívateľské rozhranie ktoré by určitým spôsobom upútalo užívateľa. Bolo tak nutné navrhnúť ovládacie prvky ktorých vzhľad môže určiť návrhár samotného obsahu. Pre ukladanie informácií o vzhľade a obsahu bolo taktiež nutné navrhnúť vhodnú štruktúru a formát konfiguračných súborov. Vznikla tak aplikácia wxDemoViewer ktorej popisu sa venuje väčšia časť praktickej časti – nachádza sa tu popis jej štruktúry z programátorského hľadiska a stručná užívateľská príručka. Aplikácia sa snaží o čo najväčšiu univerzálnosť a môže tak byť použitá na prezentáciu mnohých ďalších programovacích jazykov a knižníc. Záver praktickej časti potom na praktickom príklade popisuje celý postup tvorby jednoduchého konfiguračného súboru.

Aplikácia bola tiež úspešne pridaná k projektu wxCode (http://www.wxcode.com), ktorý je verejným zdrojom prídavkov a komponent pre knižnicu wxWidgets. Aplikácia je tak dostupná každému kto sa chce o knižnici dozvedieť viac. Prípadne môže slúžiť vývojárom ktorý sa ešte len rozhodujú ktorú z dostupných knižníc si vybrať a možno tak rozhodne práve v prospech wxWidgets.

ZÁVER V ANGLIČTINE

The theoretical part of the work describes the wxWidgets software library, brief history of its evolution and its advantages over other similar libraries. It also deals with practical creation of a simple application and finally presents some of available development tools designed for this library.

The practical part of the work is devoted to the creation of demonstration and documentation application for the wxWidgets library which was the main objective of this work. Because the application will be used for presentation of its content, it was important to design the user interface which would in some way drew a user. It was necessary to propose controls whose appearance may be determined by a very content. Also an appropriate structure and format of the configuration files storing information on the design and the content was proposed and used during the development process. A description of inner structure of the application and user guide takes the major part of the practical section of the work. The application seeks to maximize the versatility and can be used for the presentation of many other programming languages and libraries. End of the practical part describes the process of making a simple configuration file.

wxDemoViewer has been also successfully submitted as an wxWidgets addition to the wxCode project (http://www.wxcode.com), which is an official public repository for wxWidgets addons and components. The application is available for free to anyone who wants to learn more about the wxWidgets library. Alternatively, it may be used with developers which isn’t yet decide which of the available libraries to use and can help to choose in favor of wxWidgets.

ZOZNAM POUŽITEJ LITERATÚRY

[1] BLIŽŇÁK, Michal. Systémové programování. 1. vyd. Zlín: UTB ve Zlíně, 2005. 202 s. ISBN 80-7318-364-1

[2] HARMS, Daryl. Začínáme programovat v jazyce Python. 1. vyd. Praha: Computer Press, 2003. 456 s. ISBN 80-7226-799-X

[3] LOGAN, Syd. Cross-Platform Development in C++. Addison Wesley Professional, 2007. 547 s. ISBN 032124642X.

[4] LUTZ, Mark. Naučte se Python. 1. vyd. Praha: Grada Publishing, 2003. 339 s. ISBN 80-247-0367-X

[5] MASTERS, Jon. Linux profesionálně: programování aplikací. 1.vyd. Brno: Zoner Press, 2008. 539 s. ISBN 978-80-86815-71-8

[6] SMART, Julian, HOCK, Kevin. Cross-Platform GUI Programming with wxWidgets. Prentice Hall, 2006. ISBN 0-13-147381-6

[7] Dokumentácia knižnice wxWidgets na WWW [online]. [cit. 18. 5. 2009]. Dostupný z WWW: (http://docs.wxwidgets.org/stable/)

[8] Webové stránky knižnice Qt [online]. [cit. 18. 5. 2009]. Dostupný z WWW: (http://www.qtsoftware.com/products)

[9] Webové stránky knižnice wxPython [online]. [cit. 18. 5. 2009]. Dostupný z WWW: (http://www.wxpython.org/)

[10] wxWiki [online]. [cit. 5.18.2009]. Dostupný z WWW: http://wiki.wxwidgets.org

ZOZNAM POUŽITÝCH SYMBOLOV A ZKRATIEK

API **A**pplication **P**rogramming **I**nterface – aplikačné rozhranie

GUI **G**raphical **U**ser **I**nterface – užívateľské rozhranie

CVS **C**oncurrent **V**ersions **S**ystem – systém na správu verzií

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

PRÍLOHA P I: VÝPIS ZDROJOVÉHO KÓDU VZOROVEJ APLIKÁCIE

```
#include <wx/wx.h>

class HelloWorldApp : public wxApp
{
    public:
        virtual bool OnInit();
};

class HelloWorldFrame: public wxFrame
{
    public:
        // konštruktor
        HelloWorldFrame(const wxString& title);

        // obsluha udalostí
        void OnAbout(wxCommandEvent& event);
        void OnExit(wxCommandEvent& event);

    private:
        // deklarácia tabuľky udalostí
        DECLARE_EVENT_TABLE()
};


bool HelloWorldApp::OnInit()
{
    // vytvoríme hlavné okno aplikácie
    HelloWorldFrame* frame = new HelloWorldFrame(wxT("Hello World"));

    // zobrazíme toto okno
    frame->Show();

    return true;
}

IMPLEMENT_APP(HelloWorldApp);

// tabuľka udalostí pre triedu HelloWorldFrame
BEGIN_EVENT_TABLE(HelloWorldFrame, wxFrame)
    EVT_MENU(wxID_ABOUT, HelloWorldFrame::OnAbout)
    EVT_MENU(wxID_EXIT,  HelloWorldFrame::OnExit)
END_EVENT_TABLE()


HelloWorldFrame::HelloWorldFrame(const wxString& title) :
wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(640, 480))
{
    // menu súbor
    wxMenu *fileMenu = new wxMenu;
    fileMenu->Append(wxID_EXIT, wxT("&Ukončiť program\tAlt-X"),
                                wxT("Ukončí program"));

    // menu pomoc
    wxMenu *helpMenu = new wxMenu;
    helpMenu->Append(wxID_ABOUT, wxT("&O Programe...\tF1"),
```

```
                    wxT("Zobrazí informácie o programe"));

    // pripojenie menu do hlavného menu
    wxMenuBar *menuBar = new wxMenuBar();
    menuBar->Append(fileMenu, wxT("&Súbor"));
    menuBar->Append(helpMenu, wxT("&Pomoc"));

    // priradenie hlavného menu oknu aplikácie
    SetMenuBar(menuBar);

    // vytvorenie statusbaru a nastavenie úvodného textu
    CreateStatusBar(1);
    SetStatusText(wxT("Víta vás aplikácia Hello World!"));

    // statický text zobrazovaný v okne aplikácie
    wxStaticText *text = new wxStaticText(this, wxID_ANY, wxT("Hello
    World"));

    // sizer ktorý tento text vycentruje
    wxBoxSizer *sizer = new wxBoxSizer(wxVERTICAL);

    // umiestnenie prvku wxStaticText medzi dve pružné medzery
    sizer->AddStretchSpacer(); // pružná medzera
    sizer->Add(text, 0, wxALIGN_CENTER);
    sizer->AddStretchSpacer(); // pružná medzera

    // priradenie sizeru oknu aplikácie
    SetSizer(sizer);
}

void HelloWorldFrame::OnAbout(wxCommandEvent& event)
{
    // zobrazenie správy
    wxMessageBox(wxT("Víta vás aplikácia Hello World!"),
                 wxT("O Programe"), wxICON_INFORMATION);
}

void HelloWorldFrame::OnExit(wxCommandEvent& event)
{
    // ukončenie aplikácie
    Destroy();
}
```