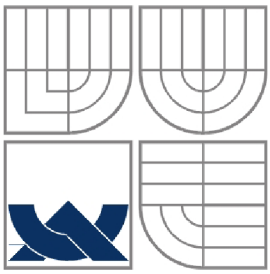
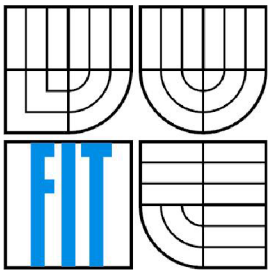


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

UNIVERZÁLNY GRAFICKÝ EDITOR IMPORT/EXPORT

UNIVERSAL GRAPHIC EDITOR
IMPORT/EXPORT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

TIBOR KÁDEK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ALEŠ SMRČKA

BRNO 2007

Zadání bakalářské práce

Řešitel: **Kádek Tibor**

Obor: Informační technologie

Téma: **Univerzální grafický editor – import/export**

Kategorie: Počítačová grafika

Pokyny:

1. Seznamte se s projektem Univerzální grafický editor (UGE) a jeho V/V rozhraním. Nastudujte některé vektorové a rastrové grafickými formáty (např. SVG, EPS, PDF, PNG).
2. Analyzujte problematiku vykreslování a exportu instance grafu v UGE. Vyhodnoťte efektivní řešení exportu grafu do volitelných grafických formátů.
3. Navrhněte rozhraní pro zásuvné moduly sloužící pro import a export grafů. Navrhněte import grafu z formátu UGML (UGE Graph Markup Language), navrhněte zásuvný modul pro export do vektorových i rastrových formátů.
4. Implementujte vámi navržené zásuvné moduly pro import a export grafu do několika vybraných formátů. Implementujte řádkový převodník formátu UGML do vybraného grafického formátu.
5. Zhodnoťte dosažené výsledky a navrhněte možná pokračování projektu.

Vedoucí: **Smrčka Aleš, Ing.**, UITS FIT VUT

Datum zadání: 1. listopadu 2006

Datum odevzdání: 15. května 2007

Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně

Abstrakt

Práca sa zaoberá problematikou tvorby grafov v systéme UGE (Univerzálneho grafického editora) a možnosťami, ktoré ponúka pre programátorov zásuvných modulov. Cieľom práce bolo navrhnutie formátu UGML (UGE Graph Markup Language), implementácia zásuvného modulu pre import a export grafu systému UGE, implementácia exportu do vektorového formátu SVG a niekoľkých rastrových formátov. Práca rozširuje možnosti jednotlivých častí systému, hlavne aplikačného rozhrania a prácu s grafickou reprezentáciou grafu.

Kľúčové slová

Univerzálny grafický editor, UGE, libuge, zásuvný modul, graf, XML, UGML, SVG, libxml, GUI, grafické užívateľské rozhranie, GTK+, aplikačné rozhranie, API, PNG, DOM, SAX, import, export

Abstract

This thesis deals with creating graphs in application UGE (Universal graphic editor) and facilities it offers for plugins programmers. Goal of the thesis is to design UGML (UGE Graph Markup Language) data format, implement plugin for importing/exporting graph, implement SVG graphic format export and export into several bitmap formats. Thesis extends particular parts of application, especially API (application program interface) and manipulating graphical representation of objects.

Keywords

Universal graphic editor, UGE, libuge, plugin, graph, XML, UGML, SVG, libxml, GUI, graphical user interface, GTK+, application programming interface, API, PNG, DOM, SAX, import, export

Citácia

KÁDEK, Tibor.: *Univerzálny grafický editor – Import/Export*. Bakalářská práce, Brno, FIT VUT v Brně, 2007

Univerzálny grafický editor - import/export

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Aleša Smrčku.
Ďalšie informácie mi poskytli Petr Golich, Miroslav Jadrný, Jaroslav Košulič, Ladislav Varga – členovia týmu pracujúcich na projekte.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Tibor Kádek
11.05.2007

Poďakovanie

Rád by som špeciálne poďakoval vedúcemu práce Ing. Alešovi Smrčkovi, za jeho cenné rady a pedagogický prístup vysokej kvality.

Obsah

1 Úvod

V dnešnej dobe prudko rastie množstvo informácií, ktoré je dôležité získať v čo najkratšom čase. Vzniká potreba informácie efektívne šíriť a úspešne prezentovať s dôrazom na správnu interpretáciu. Jednou z vhodných foriem vyjadrovacieho prostriedku prezentácie je diagram – graf. Diagram je vo všeobecnosti zjednodušeným štruktúrovaným vizuálnym znázornením údajov a vzťahov častí systému. Vzhľadom k rastúcej zložitosti systémov rastú aj nároky kladené na tvorbu a spracovávanie diagramov. Uspokojiť tieto požiadavky sa snaží projekt Univerzálneho grafického editoru, ktorým sa táto práca ďalej zaoberá.

1.1 Stručný opis kapitol práce

Kapitola 2.
Opisuje problematiku grafov a predstaví motiváciu vzniku a možnosti projektu Univerzálneho grafického editora (UGE). Opíše stavbu systému, štruktúru grafu, komunikáciu so zásuvnými modulmi a aplikačné rozhranie projektu (API).

Kapitola 3.
Zaoberá sa návrhom formátu UGML (UGE Graph Markup Language) pre systém UGE.

Kapitola 4.
Podrobne predstaví rastrový formát PNG (Portable Network Graphics), vektorový formát SVG (Scalable Vector Graphics) a jeho využitie v projekte.

Kapitola 5.
Zaoberá sa prípravami a implementáciou modulu Import/Export a opisuje prístup k spracovaniu XML dokumentu pomocou DOM (Document Object Model) a SAX (Simple API for XML)

Kapitola 6.
Záverečná kapitola zhodnotí dosiahnuté výsledky, prínos práce a predloží námety na ďalší vývoj projektu.

2 Univerzálny grafický editor (UGE)

Projekt univerzálneho grafického editoru (UGE) si kladie za cieľ poskytnúť prostriedok na kreslenie grafov a umožniť prevádzať nad nimi ďalšie operácie.

> ***Definícia:*** Graf G je usporiadaná dvojica G = (V, H), kde V je neprázdna konečná množina vrcholov (uzlov) grafu a H je množina dvojíc z množiny V nazývaných hrany grafu. [3]

Rôzne schémy a diagramy vyskytujúce sa v praxi, sú v podstate len špeciálnym prípadom grafu. To znamená, že obsahujú uzly, ktoré niečo vyjadrujú (napríklad formou grafickej reprezentácie) a hrany, ktorými sú tieto uzly prepojené a popisujú medzi nimi určitý vzťah.

Takýmito diagramami s vlastnosťami grafu môžu byť napríklad rôzne druhy stavových diagramov, diagramy tried (UML diagramy všeobecne), entitno-relačné diagramy (ERD), konečné automaty, Petriho siete, relačné schémy databáz, topologické schémy sietí a mnohé iné schémy aj mimo obor informatiky (schéma organizačnej štruktúry a pod.).

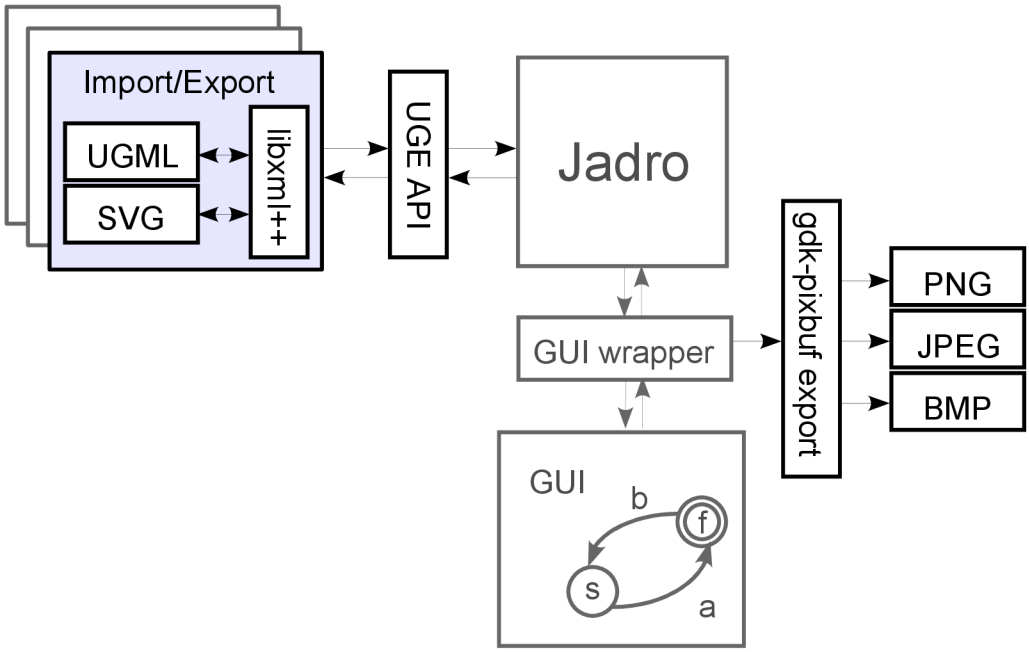
Existuje síce veľké množstvo aplikácií, ktoré sú schopné pracovať práve s vyššie menovanými prípadmi grafov, ale ich použitie je väčšinou obmedzené buď len na vytvorenie vizuálnej reprezentácie, alebo v prípade, že dokážu nad vykreslenou schémou prevádzať simulácie, sú úzko zamerané len na konkrétnu problematiku.

Motiváciou ku vzniku projektu UGE bolo vytvoriť systém, ktorý pracuje so všeobecným grafom a ponúka možnosť popísať jeho špecifické chovanie a vizuálnu podobu prvkov pre dané potreby. Koncový užívateľ by potom nemusel na tvorbu rôznych typov diagramov použiť viaceré programy, ale postačil by mu jeden *univerzálny grafický editor*.

2.1 Myšlienka systému UGE

Program je postavený na myšlienke vytvoriť prívetivé grafické prostredie pre koncových užívateľov, v ktorom je možné vytvárať, upravovať a prevádzať rôzne operácie nad všeobecným grafom a umožniť výsledné dielo uložiť, znovu načítať a ďalej spracovávať. Zároveň poskytuje prostriedky pre programátorov zásuvných modulov, ktorý majú pomocou aplikačného rozhrania možnosť programovateľne ovplyvňovať chovanie grafu, sprístupniť rôzne simulácie nad grafom a reagovať na úkony vykonané užívateľom. Pri tvorbe bol dôraz kladený na jednoduchú rozšíriteľnosť, prenositeľnosť a univerzálnosť riešenia.

2.2 Stavba systému UGE a modul Import/Export

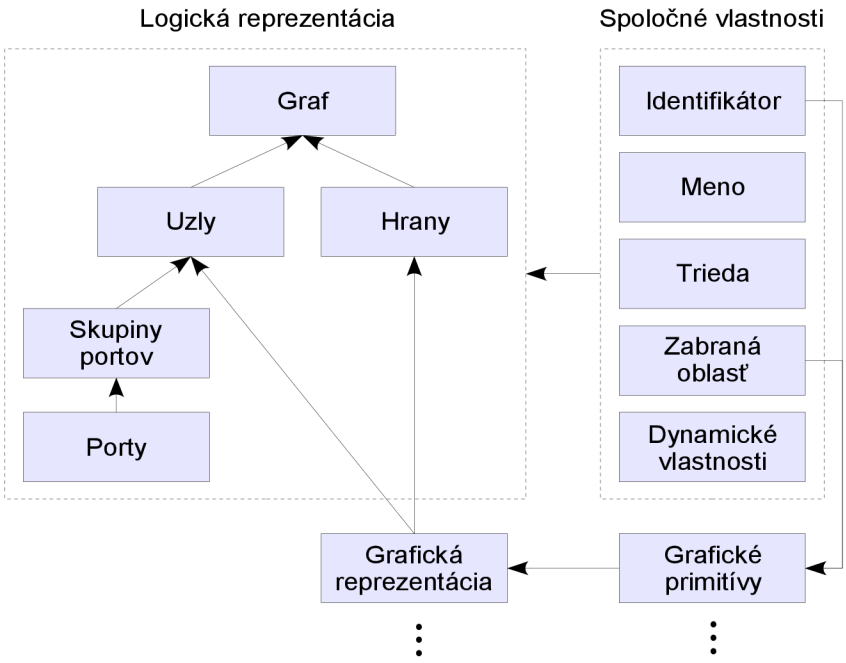


Obrázok 2.1: Stavba systému UGE s modulom Import/Export

Na obrázku 2.1 je zobrazená stavba programu, ktorá je rozdelená do štyroch hlavných blokov. Základnú časť tvorí *jadro* napísané v jazyku C++, ktoré zabezpečuje komunikáciu medzi jednotlivými blokmi a poskytuje štruktúry pre uchovanie dát v pamäti a následne prácu s inštanciami grafových a grafických tried. S jadrom komunikuje *grafické užívateľské rozhranie (GUI)* vytvorené pomocou grafického toolkitu GTK+ (GIMP Toolkit) cez *GUI wrapper*. Tento obal bol pridaný kvôli možnosti v budúcnosti jednoducho zmeniť knižnicu na vytváranie grafických užívateľských rozhraní. Na prácu s kresliacou plochu je používaná grafická knižnica GDK (GIMP Drawing Kit) obalujúca grafický zobrazovací protokol pre rastrové zobrazovacie zariadenia. GUI zasiela jadru požiadavky vyvolané užívateľom, ktoré môže odchytiť tvorca zásuvného modulu formou signálnych správ a ďalej spracovať. Na signály generované jadrom je možné sa napojiť pomocou registrácie spätných volaní na funkcie, ktorých realizácia je popísaná v kapitole o komunikácií systému s modulom. Ďalším blokom je aplikačné rozhranie (API), ktoré poskytuje prívetivé funkcie na komunikáciu zo systémom a zjednodušuje tak vývoj zásuvných modulov pre systém UGE.

2.3 Štruktúra grafu v UGE

Graf sa skladá s uzlov, portov, skupinou portov a hrán. Jeho logická reprezentácia je nasledovná:



Obrázok 2.2: Štruktúra grafu v UGE

Uzol má svoju grafickú reprezentáciu vyjadrenú súborom základných grafických primitív akými sú úsečka, obdĺžnik, kruh, elipsa, krivka. Všetky nastaviteľné vlastnosti grafických primitív sú podrobne opísané v kapitole 4.1.3. Existujú dva typy uzlov – jednoduchý a zložený. Jednoduchý slúži na priame napojenie hrán na jeden prípojný bod umiestnený v strede vykreslenej oblasti. Takýto uzol má zväčša jednoduchú grafickú reprezentáciu napr. podobu stavu v stavových automatoch. Pri zloženom uzle je napájanie hrán realizované cez prípojné body – porty, ktoré uzol obsahuje. Použitá grafická reprezentácia býva už komplexnejšia napr. schéma hardvérového komponentu multiplexora.

Port má v rámci uzlu zadanú relatívnu pozíciu a typ orientácií napojených hrán (z, do, ...). Porty sú zoskupované do skupiny portov zo špecifickými vlastnosťami, ktoré sú na porty aplikované. Vlastnosťou môže byť napr. svetová strana z ktorej sa hrana napája (východ, západ, sever, juh a ich kombinácie), typ reprezentácie hrany napájajúci sa na port a rozostup medzi portami v prípade dynamických portov.

Hrana má udaný uzol alebo port z ktorého vychádza a vchádza a typ (priama, ortogonálna, krivka), ktorý sa použije pri jej grafickej reprezentácie.

Elementy grafu majú svoje jednoznačné číselné identifikátory a unikátne názvy (grafické primitíva majú len číselný identifikátor). Ďalším identifikátorom je parameter triedy, vďaka ktorému je možné identifikovať, ktorému modulu dané uzly prislúchajú. Modul si môže podľa ľubovôle pridávať ďalšie vlastnosti základných dátových typov. Systém medzi nimi prevedie automatickú konverziu, ktorú môže programátor pri volaniach využiť. Všetky vlastnosti elementov grafu s ktorými môže zásuvný modul pracovať, sú podrobne opísané pri návrhu natívneho formátu na ukladanie grafov vytvorených v UGE v kapitole 3 a pri možnostiach aplikačného rozhrania.

S detailným opisom interných štruktúr, prostriedkov jadra a grafického užívateľského rozhrania sa môžete oboznámiť v práci [4]. Kvôli rozšíreniam možností API bolo do interných štruktúr jadra zasahované, ale ostal zachovaný jednotný prístup popísaný v práci. Rovnako aj pri užívateľskom rozhraní kvôli rozširovaniu možností pri vykresľovaní a tvorbe súborových dialógov pre import a export. Niektoré zmeny sú popísané v kapitole 5, ktorá opisuje implementáciu modulu import/export.

2.4 Komunikácia s modulmi

Zásuvné moduly sú implementované formou dynamickej knižnice. Zavedenie knižnice do pamäte je realizované volaním funkcie *dlopen*, ktorej sa predá jej názov a vráti popisovač (handler). Ten je spolu s menom vopred známej knižničnej funkcie predaný funkcii *dlsym*, ktorá vracia adresu symbolu, vďaka ktorému môže byť následne funkcia volaná. [5].

Pri štarte aplikácie sa spustia interné inicializácie programu, vytvorí sa inštancia grafu, potom tabuľka spätných volaní a načítajú sa dostupné dynamické knižnice (zásuvné moduly – pluginy). Jadro očakáva pri každom module dve povinné funkcie a volá ich. Volaním funkcie *handshake* sa získajú základné informácie o moduloch akými sú názov, verzia a stručný popis. Následne sa zavolá inicializačná funkcia *plugin_init,* v ktorej má plugin možnosť zaregistrovať si spätné volania funkcií na signály. Signály generuje jadro pri rôznych akciách vyvolané užívateľom alebo internými procesmi jadra. Prehľadáva sa tabuľka spätných volaní a volajú sa funkcie napojené na daný signál. Prvou možnosťou, vďaka ktorej je možné sa napojiť na jeden zo signálov, je funkcia *signal_connect*. Ďalšou, určenou na pridanie položiek do grafického užívateľského rozhrania je funkcia *uge_add_menu_item*. Signály sú vymenovaného typu a ich zoznam je dostupný v súbore *common.h*.

2.5 Aplikačné rozhranie programu (API)

Aplikačné rozhranie programu poskytuje programátorom zásuvných modulov funkcie, pomocou ktorých je možné vykonávať rôzne operácie nad grafom. Je možné získavať a nastavovať vlastnosti jednotlivých grafových a grafických objektov, pridávať, odoberať a upravovať položky grafického užívateľského rozhrania, zobrazovať užívateľom rôzne druhy dialógových okien, registrovať a odstraňovať spätné volania funkcií.

Aby bola univerzálnosť projektu pojatá v maximálnej miere, prototypy funkcií aplikačného rozhrania sú deklarované v jazyku C a tým je zaistená možnosť programovania zásuvných modulov v jazykoch podporujúce volania takto deklarovaných funkcií.

Jedným s hlavných zameraním práce bolo opraviť a rozšíriť možnosti aplikačného rozhrania projektu a sprístupniť v plnej miere možnosti práce s grafom zásuvným modulom. Voľbou bolo preto implementovať ukladanie a načítanie natívneho formátu programu (UGML) práve cez aplikačné rozhranie a tým zároveň demonštrovať jeho možnosti pre ďalšie zásuvné moduly.

2.5.1 Funkcie aplikačného rozhrania

Pomocou funkcií aplikačného rozhrania môže modul ovplyvňovať behaviorálne procesy jadra.
Ich funkcionálna kategorizácia je nasledovná:

1. Práca s modulom
   - získanie základných informácií o module
   - inicializácia modulu
   - ukončovanie (finalizácia) modulu
2. Obsluha signálov generovaných jadrom.
   - registrácia spätných volaní funkcií
   - registrácia spätných volaní pre položky grafického užívateľského rozhrania
   - odstránenie spätných volaní
3. Práca s grafickým užívateľským rozhraním
   - pridávanie a mazanie položiek hlavného menu
   - nastavenie kontextového menu pre grafové elementy
   - zobrazenie informačných a chybových dialógov
   - volanie dialógových okien pre nastavenie parametrov elementov grafu
   - volanie a nastavenie súborových dialógov

4. Práca s grafom
    - vytváranie a mazanie elementov grafu (uzol, port, skupina portov, hrana)
    - získanie všetkých identifikátorov uzlov, portov, skupiny portov, hrán
    - získanie všetkých identifikátorov skupiny portov patriacich zadanému uzlu
    - získanie všetkých identifikátorov portov patriacich zadanej skupine portov
    - získanie všetkých identifikátorov hrán pripojených k zadanému uzlu
    - získanie zdrojového a cieľového uzlu pre zadanú hranu
    - získanie rôznych vlastností oblasti, ktoré elementy zaberajú
    - nastavenie a získanie typu jednotlivých elementov
    - nastavenie a získanie vlastností patriacich ku konkrétnemu typu elementu
    - vytváranie nových vlastností základných dátových typov, zadanému elementu
    - nastavenie a získavanie pridaných vlastností elementov
5. Práca s grafickou reprezentáciou elementov
    - pridávanie a mazanie grafických primitív
    - nastavenie špecifických vlastností grafických primitív
    - nastavenie spoločných vlastností grafických primitív (farba výplne a obrysu, hrúbka a typ vykresľovaných čiar, spôsob a tvar ich zakončenia)
    - vytváranie a úprava textových elementov
    - export grafickej reprezentácie do rôznych rastrových formátov

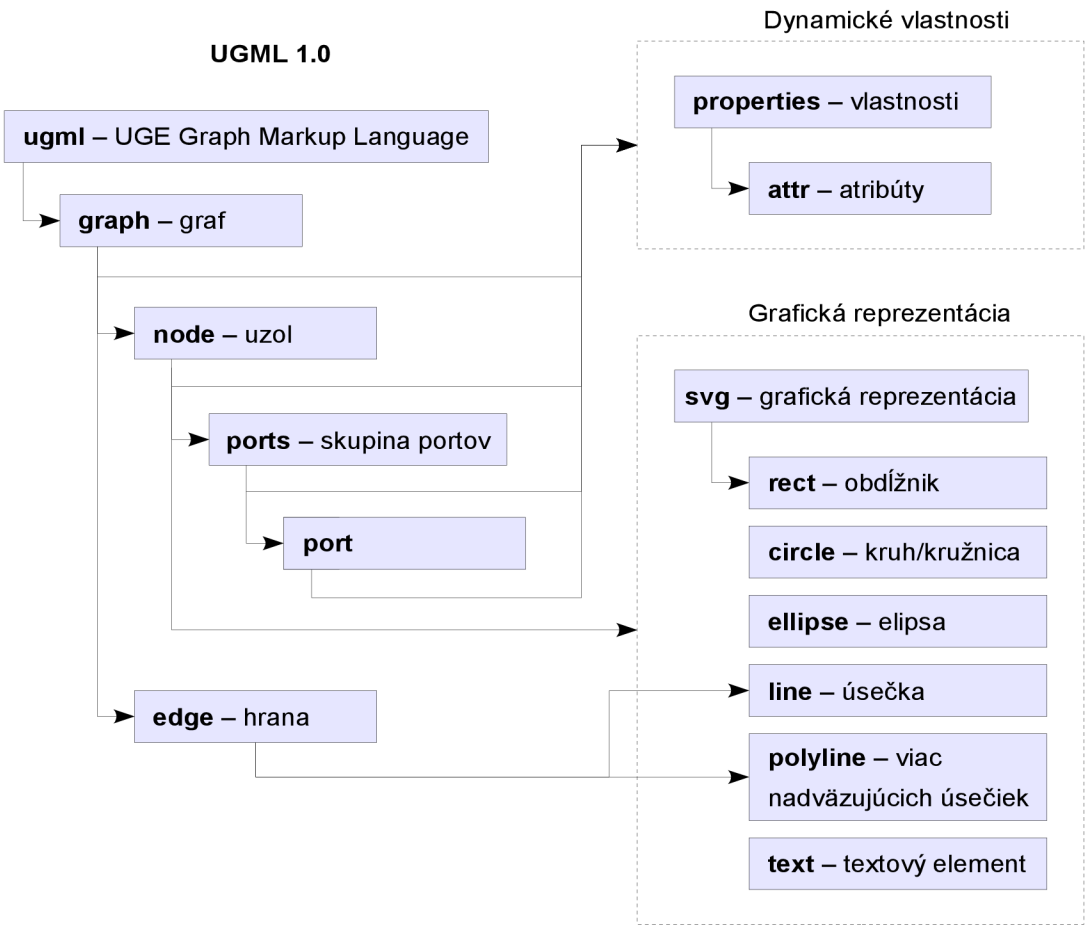
Je vidieť, že aplikačné rozhranie projektu ponúka množstvo funkcií a umožňuje tým rýchly vývoj zásuvných modulov na ktorých je využiteľnosť systému závislá. Funkcie sú deklarované v hlavičkovom súbore uge.h a po jeho vložení, spolu s vymenovanými typmi dostupných v súbore common.h, ich môže modul volať. Ich úplný zoznam a spôsob použitia je popísaný v programovej dokumentácii projektu.

3 Formát UGML

UGML – UGE Graph Markup Language bol navrhnutý pre potreby aplikácie UGE. Je založený na značkovacom jazyku XML (eXtensible Markup Language), ktorý slúži na štrukturalizáciu dát a je v dnešnej dobe už de facto štandardným formátom na ukladanie a výmenu informácií. UGML popisuje logickú štruktúru grafu a prístupné vlastnosti jednotlivých elementov. Na popis grafickej reprezentácie uzlov bola použitá syntax formátu SVG (Scalable Vector Graphics) a jeho použitie je popísané v kapitole 4.1.3.

3.1 Štruktúra UGML

Nasledujúce stromové zobrazenie reprezentuje logickú štruktúru elementov s názvom značky.



Obrázok 3.1: Štruktúra grafu v UGML

3.1.1 Vlastnosti elementov grafu

spoločné atribúty elementov

class – trieda pre identifikáciu elementu modulom vlastníka

name – unikátne meno elementu

ugml – koreňový element

atribúty:

version – verzia formátu kvôli prípadným zmenám v budúcnosti

xmlns – identifikátor menného priestoru vo forme URI. (XML Namespaces) [6]

graph – reprezentácia grafu v UGE

atribúty:

name – meno grafu

node – uzol grafu

atribúty:

type – typ uzlu

single – jednoduchý uzol s priamym napojením hrán

multiple – zložený uzol s napojením hrán cez porty, ktoré uzol obsahuje

etype – typ orientácie možných napojení hrán

none – neorientované hrany

source – hrana vychádza z uzlu

target – hrana vchádza do uzlu

multi in – viac hrán vchádza do uzlu

multi out – viac hrán vychádza z uzlu

multi inout – viac hrán vchádza/vychádza do/z uzlu

x, y – hodnoty ľavej hornej súradnice oblasti uzlu na osi x, y kresliacej plochy

ďalšie spoločné atribúty

príklad:

```
<node type="single" x="50" y="50" class="state" name="meno1">
    <svg ...
</node>
```

ports – uzol grafu

atribúty:

type – typ hrán napájajúcich sa na port

direct – priame hrany

polyline – hrany skladajúce sa z viacerých čiar

orthogonal – hrany sú ortogonálneho charakteru napr. pri HW komponentu

bezier – hrany sú bezierove krivky

direction – smer napojovania hrán

n, s, e, w, ne, nw, se, sw – skratky svetových strán a ich kombinácií z angličtiny (north – sever atď.)

attr – vlastnosť portov

static – statické

dynamic – dynamicky generované

distance – vzdialenosť medzi portami v prípade dynamicky generovaných portov

ďalšie spoločné atribúty

príklad:

```
<ports type="orthogonal" attr="static" direction="w">
    <port ...
</ports>
```

port – miesto kde sa hrana pripája v prípade zloženého uzlu

atribúty:

type – typ orientácie možných napojení hrán

none – neorientované hrany

source – hrana vychádza z portu

target – hrana vchádza do portu

multi in – viac hrán vchádza do portu

multi out – viac hrán vychádza z portu

multi inout – viac hrán vchádza/vychádza do/z portu

x, y – hodnoty pozície v rámci oblasti uzlu na osi x, y

ďalšie spoločné atribúty

príklad:

```
<port name="in1" type="target" x="0" y="20" />
```

edge – hrana grafu

atribúty:

type – typ hrany

direct – priame hrany

polyline – hrany skladajúce sa z viacerých čiar

orthogonal – hrany sú ortogonálneho charakteru napr. pri HW komponentu

bezier – hrany sú bezierove krivky

source – uzol alebo port z ktorého hrana vychádza

target – uzol alebo port do ktorého hrana vchádza

ďalšie spoločné atribúty

properties – ďalšie vlastnosti základných dátových typov pridané modulmi

attr – atribút definovaný modulom

atribút

name – meno atribútu

príklad:

```
<properties>
    <attr name="DATAWIDTH">16</attr>
    <attr name="comment">Komentar</attr>
</properties>
```

svg – grafická reprezentácia uzlov

atribúty:

width – šírka vykresľovanej oblasti

height – výška vykresľovanej oblasti

xmlns – identifikátor menného priestoru vo forme URI. (XML Namespaces)

4 Vektorové a rastrové grafické formáty

Pri definovaní obrazových informácií môžeme pristupovať formou vektorovej alebo rastrovej grafiky. Vektorová popisuje obrazovú informáciu pomocou základných geometrických primitív (napr. bod, priamka, vektor, krivka, mnohouholník), ktoré sa dajú vyjadriť matematickými rovnicami. Rastrová grafika popisuje jednotlivé usporiadané obrazové body (pixle) v pomyselnej mriežke, kde každý bod má určenú svoju presnú polohu, farbu a prípadne priehľadnosť (alfa kanál). V ďalších podkapitolách si predstavíme jednotlivé formáty a ich spôsob ukladania obrazových informácií.

4.1 Scalable Vector Graphics (SVG)

SVG je značkovací jazyk a formát súboru, ktorý popisuje dvojrozmernú statickú alebo animovanú vektorovú grafiku pomocou značkovacieho jazyka XML. SVG je otvorený štandard, ktorý vytvorilo konzorcium W3C zodpovedné za štandardy ako HTML a XHTML. [7]

Jazyk SVG umožňuje definovať tri typy grafických objektov:

- vektorové tvary – kružnica, elipsa, obdĺžnik, úsečka, krivka atď.
- rastrové obrázky
- textové objekty

Objekty môžu byť združované a rôzne transformované s možnosťou aplikovať viaceré transformácie naraz. Tie je možné zapísať buď pomocou transformačnej matice, alebo dostupnými funkciami na posun, rotáciu, skosenie a zmenu merítka. Je možná štylizácia objektov tzn. dynamické formátovanie pomocou kaskádových štýlov (CSS) alebo XSLT transformáciami. Objekt môže byť použitý ako tzv. maska priehľadnosti na iný objekt či skupinu a zlúčiť sa s vykresleným pozadím. Na objekty môže byť aplikované orezávanie pomocou hraničiacej krivky a rôzne ďalšie filtrové efekty, ktoré sa aplikujú až pri vykresľovaní. Je možné definovať objekty ako šablóny na ktoré sa potom dá v dokumente ďalej odkazovať. Základným farebným priestorom pre SVG je sRGB, ale pomocou CSS je možné vložiť ICC profily pre rôzne typy zariadení. Na dynamičnosť a interaktivitu SVG obrázkov je možné použiť skriptovanie cez aplikačné rozhranie DOM (Document Object Model), ktoré umožňuje animácie objektov založených na štandarde ECMAScript (JavaScript) alebo SMIL (Synchronized Multimedia Integration Language). Často sa na redukciu objemu dát SVG dokumentov (textové súbory) využíva kompresia algoritmom GZIP a vytvorené súbory sú označované ako SVGZ.

Vzhľadom na tak veľké možnosti ktoré formát SVG ponúka, nie sú ešte všetky prehliadače a vektorové editory schopné všetky spomenuté vlastnosti aplikovať.

4.1.1 História SVG

SVG vyvinula W3C SVG Working Group od roku 1998 po tom, ako Macromedia a Microsoft predstavili Vector Markup Language (VML), kým Adobe Systems a Sun Microsystems predložili konkurenčný formát známy ako PGML. Pracovnej skupine predsedal Chris Lilley z W3C.

- SVG 1.0 sa stalo Odporúčaním W3C 4. septembra 2001. [8]
- SVG 1.1 sa stalo Odporúčaním W3C 14. januára 2003. [9] SVG špecifikácia bola rozložená do modulov a tým vznikla možnosť previazania aj s inými štandardmi založených na XML. (XHTML, MathML, XForms a pod.)
- SVG Tiny a SVG Basic (Mobilné SVG profily) sa stali Odporúčaním W3C 14. januára 2003. Sú opísané ako profily SVG 1.1.
- SVG Tiny 1.2 sa stalo Kandidátnym Odporúčaním W3C 10. augusta 2006. [10] SVG Full 1.2 je v súčasnosti Pracovným náčrtom W3C. SVG 1.2 Mobile bol zámerne uvoľnený ako profil a neskôr prepracovaný na úplnú špecifikáciu vrátane všetkých potrebných súčastí SVG 1.1 a SVG 1.2. Podobne prepracovaný náčrt SVG Full 1.2 zatiaľ nebol uvoľnený.

4.1.2 Podpora SVG

Podpora špecifikácie SVG v prehliadačoch a aplikáciach väčšinou nieje úplná. V niektorých prehliadačoch ako Internet Explorer je potrebný na prehliadanie SVG obrázkov externý plugin napr. Adobe SVG Viewer. Čiastočná natívna podpora je v internetových prehliadačoch založených na vykresľovacom jadre Gecko od verzie 1.8 (Firefox, Camino, Epiphany), v prehliadači Opera od verzie 8 a v prehliadačoch založených na KHTML s KSVG (Konqueror). Vektorové editory podporujúce formát SVG sú napr. Inkscape, Adobe Illustrator, CorelDRAW, Xara Xtreme atď. Medzi knižnice na prácu s formátom patrí Batik SVG Toolkit, Cairo, librsvg a ďalšie.

Podpora formátu v UGE v rámci UGML je taktiež len čiastočná a poskytuje možnosť pracovať so základnými grafickými vektorovými tvarmi, ich spoločnými vlastnosťami a textom.

4.1.3 Formát SVG v UGML

Pre grafickú reprezentáciu uzlov vo formáte UGML bol vybraný formát SVG. Jeden z dôvodov nevytvárať vlastný formát, ale použiť aspoň časť už existujúceho, bol umožniť tvorcom zásuvných modulov „nakresliť si“ uzly vo svojom obľúbenom vektorovom editore a po rýchlej úprave ich použiť pre import z UGML.

4.1.3.1 Grafické primitíva

V tejto kapitole si predstavíme základné grafické primitíva, ktoré je možné použiť na popis grafickej reprezentácie uzlov v UGML.

Elementy

rect – obdĺžnik

atribúty:

x, y – hodnoty pozície ľavého horného rohu obdĺžnika na osi x, y

width – šírka obdĺžnika

height – výška obdĺžnika

ďalšie spoločné atribúty

príklad:

```
<rect x="20" y="20" width="200" height="100" fill="red" />
```

circle – kruh alebo kružnica v prípade že je nastavená vlastnosť výplne (*fill*) na *none,* alebo nieje uvedená

atribúty:

cx, cy – hodnoty pozície stredu kružnice na osi x, y

r – polomer kružnice

ďalšie spoločné atribúty

príklad:

```
<circle cx="26" cy="26" r="25" fill="blue" />
```

ellipse – elipsa

atribúty:

cx, cy – hodnoty pozície stredu elipsy na osi x, y

rx, ry – veľkosť polomeru na x-ovej a y-ovej osi

ďalšie spoločné atribúty

príklad:

```
<ellipse cx="10" cy="10" rx="90" ry="70"
         fill="none" stroke="red" stroke-width="1" />
```

line – úsečka

atribúty:

x1, y1 – počiatočný bod úsečky na osi x, y

x1, y1 – koncový bod úsečky na osi x, y

ďalšie spoločné atribúty

príklad:

```
<line x1="10" y1="10" x2="50" y2="50"
      stroke="black" stroke-width="3" />
```

polyline – viac nadväzujúcich úsečiek

atribúty:

points – zoznam súradníc na osi x, y vo forme *x1,y1 x2,y2 x3,y3 ...*

príklad:

```
<polyline points="10,190 20,190 20,150 50,150 50,190 80,190"
          stroke="green" stroke-width="2" />
```

text – textový element

atribúty:

x, y – hodnoty pozície základnej čiary prvého písmena na osi x, y

font-size – veľkosť textu v bodoch

ďalšie spoločné atribúty

príklad:

```
<text x="20" y="35" font-size="20" fill="black">Text</text>
```

Spoločné atribúty

fill – farba výplne

hodnota: *farba*

implicitná hodnota: *none* – bez výplne

stroke – farba obrysu

hodnota: *farba*

implicitná hodnota: *black* – čierna

stroke-width – šírka vykresľovanej čiary

hodnota: *číslo*

implicitná hodnota: *1*

stroke-dasharray – určuje charakter prerušení a medzier prerušovanej čiary

hodnoty: *none*

dasharray – pole čísiel striedavo udávajúcich dĺžku čiary a medzery

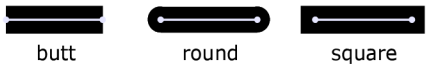
implicitná hodnota: *none*

stroke-linecap – určuje tvar zakončenia otvorenej cesty

hodnoty: *butt*

round

square



implicitná hodnota: *butt*

stroke-linejoin – určuje tvar použitý pri ostrých hranách

hodnoty: *miter*

round

bevel

implicitná hodnota: *miter*

príklad:

```
... fill="none" stroke="black" stroke-width="1" ...
```

Farba

- kľúčové slová farieb
- hodnota RGB funkcie
- hexadecimálna hodnota

kľúčové slová základných farieb a ich reprezentácia vo forme *rgb* funkcie a hexa kódu

none – priehľadná

white – biela	rgb(255,255,255)	#FFFFFF
black – čierna	rgb(0,0,0)	#000000
blue – modrá	rgb(0,0,255)	#0000FF
green – zelená	rgb(0,255,0)	#00FF00
yellow – žltá	rgb(255,255,0)	#FFFF00
red – červená	rgb(255,0,0)	#FF0000

4.2 Portable Network Graphics (PNG)

PNG je grafický formát určený pre bezstratovú kompresiu rastrovej grafiky. Ponúka podporu pre indexované farby až do 48-bitovej farebnej hĺbky z palety farebného modelu RGB (aditívny spôsob miešania farieb). Bežne je používaná 24-bitová verzia TrueColor. V režime odtieňoch sivej – Graycolor podporuje 1, 2, 4, 8 alebo 16 bitov na pixel. Pri 64-bitovej farebnej hĺbke pre RGBA farebný model je pridaná podpora 8-bitovej priehľadnosti (tzv. alfa kanál), čo znamená, že obrázok môže byť v rôznych častiach rôzne priehľadný a tak je možné jemné vyhladenie obrysov textu obrázka a precízneho priehľadného tieňovania (umožňuje ešte aj jemnejšiu 16-bitovú verziu alfa kanálu). Ďalej ponúka možnosť verného zobrazovania pomocou gama korekcie, ktorá podporuje automatické prispôsobenie jasu a kontrastu, správu farieb ICC profilmi, možnosť prekladania, detekcia poškodenia súboru (CRC kód), presné časové razítko poslednej zmeny, ďalej rozmer pixelu z ktorého je možné zistiť pomer strán obrázka, nastavenie farby pozadia, korekcie bielej farby, ktorá môže obsahovať textové metainformácie (ISO 8859-1, UTF-8) atď. Formát je preto tvorený niekoľkými špecifickými časťami (chunks), ktoré sú nositeľmi informácie pre tieto vlastnosti [11]. Na kompresiu používa algoritmus Deflate, ktorý je kombináciou algoritmu LZ77 a Huffmanovho kódovania a pred samotnou kompresiou je možné dáta vhodnou voľbou filtra prefiltrovať a zvýšiť tým kompresný pomer.

4.2.1 História PNG

Formát PNG bol vyvinutý v roku 1995 ako zdokonalenie a náhrada formátu GIF, ktorý bol patentovo chránený bezstratovým kompresným algoritmom LZW84, ale dnes už platnosť patentu vypršala. Ďalším cieľom bolo prekonanie limitácie na 256 farieb pri formáte GIF. Naopak nevýhodou formátu oproti GIF je, že neumožňuje jednoduché animácie, pretože v dobe keď sa formát vytváral, táto vlastnosť nebola pri GIF využívaná a tak bolo rozhodnuté nezahrňovať do formátu podporu pre animácie. Existujú síce dva návrhy APNG (Animated Portable Network Graphics) a MNG (Multiple-image Network Graphics), ktoré rozširujú formát o podporu animácií, ale zatiaľ sa nepresadili.

- Prvá verzia sa stala Odporúčaním W3C 1. októbra 1996. [12]
- Druha verzia sa stala Navrhovaným Odporúčaním W3C 20. mája 2003 a medzinárodným štandardom ISO/IEC 15948:2002 (E). [13]
- Posledná verzia sa stala Odporúčaním W3C 10. novembra 2003 a je tiež medzinárodným štandardom ISO/IEC 15948:2003 (E) [14]

5 Modul Import/Export

Kapitola sa zaoberá procesom príprav a implementáciou zásuvného modulu na import a export grafu Univerzálneho grafického editora. Výstupom projektu je dynamická knižnica pracujúca s formátom UGML, ktorý bude využívať systém UGE ako svoj natívny formát pre ukladanie a načítanie svojich dátových štruktúr. Ďalším výstupom je modul pre export do vektorového formátu SVG a rastrových formátov PNG, JPEG, BMP. Posledným výstupom je riadkový prevodník formátu UGML do SVG.

5.1 Prípravy pred implementáciou

Pred samotnou implementáciou modulu pre import a export grafu bolo potrebné navrhnúť formát na ukladanie dát, ktorý by svojou štruktúrou zahrnul všetky aspekty univerzálneho grafu vytvoreného v systéme UGE. Kedže dostupné existujúce formáty rôznych grafických editorov nespĺňali požiadavky, bol navrhnutý nový formát UGML (UGE Graph Markup Language), ktorého špecifikácia je popísaná v kapitole 3.

5.1.1 Rozšírenie možností aplikačného rozhrania

Dôležitou súčasťou príprav bolo vytvorenie funkcií aplikačného rozhrania, ktoré by pokrývali všetky potreby modulu a ďalších zásuvných modulov vytváraných v budúcnosti. Aplikačné rozhranie síce obsahovalo niektoré základné funkcie pre prácu s logickou reprezentáciou grafu, ale vzhľadom na nedostatok času pri testovaní prvej alfa verzie nepracovali všetky korektne a bolo ich potrebné upravovať.

Novými implementačnými rozšíreniami logickej reprezentácie boli funkcie na prácu zo skupinami portov, portami a hranami. Boli pridané funkcie na vytváranie, mazanie, nastavovanie a získavanie špecifických vlastností jednotlivých elementov grafu. Vytvorili sa funkcie cyklických priechodov na získavanie identifikátorov portov a skupiny portov podla ich vlastníka a predošlého identifikátoru. Ďalej boli pridané funkcie na získanie hodnôt užívateľsky vytvorených vlastností základných dátových typov, funkcie na zistenie počtu pridaných vlastností podľa vlastníka a funkcie na získavanie identifikátorov. Identifikátory sú potrebné na ďalšie spracovanie vlastností objektov.

Bolo pridané rozšírenie pre prácu s grafickou reprezentáciou grafu, akou je vytváranie a mazanie základných grafických primitív (úsečka, kruh, elipsa, obdĺžnik), nastavovanie a získavanie špecifických ako aj spoločných vlastností akými sú farba výplne a obrysu, hrúbka a typ vykresľovaných čiar, spôsob ich zakončenia a tvar zakončenia pri ostrých hranách.

Pre takúto činnosť museli byť implementované taktiež iteratívne prístupy funkcií, ktorých návratovou hodnotou je identifikátor grafickej primitívy, vďaka ktorému by bolo možné s vytvoreným objektom ďalej pracovať. Ďalej bolo potrebné rozoznať typ grafického objektu, aby bolo možné nastavovať jeho špecifické vlastnosti. Kedže v prvej verzii projektu nemali grafické elementy svoj identifikátor a nevracali svoj typ, musela byť táto funkcionalita do jadra pridaná. O identifikátor sa stará zdedená trieda vo forme unikátneho generátora celočíselnej hodnoty v rámci grafu. Druh primitívy je hodnotou vymenovaného typu, ktorá sa nastavuje v konštruktory pri vytváraní grafického objektu.

5.1.2 Rozšírenie možností grafického užívateľského rozhrania

Ďalšie rozšírenia možností projektu sa týkali grafického užívateľského rozhrania, konkrétne vytvorenie súborových dialógových okien. Vytvorenie rozhrania pre napojenie takýchto dialógov na signály vyvolanými ovládacími prvkami nástrojovej lišty. Možnosť ich volania cez aplikačné rozhranie a nastavovane vlastností akými sú filtrovanie výpisu podľa nastavených prípon a nastavenie nových názvov formátu dát ďalšími zásuvnými modulmi.

Užívateľ má prístup ku trom typom súborových dialógov. Jeden je prispôsobený na rýchle uloženie natívneho formátu a zobrazí len vstupné pole na zadanie názvu súboru a vyberateľnú položku naposledy použitých a obľúbených adresárov vo forme roletového menu (select box). Je tu samozrejme aj možnosť rozbalenia na spôsob klasického stromového prehľadávania adresárov. Ďalší typ dialógu je pripravený na ukladanie do rôznych formátov a pridáva možnosť zvoliť si formát súboru do ktorého sa bude ukladať. Prednastavené sú rôzne rastrové formáty ako PNG, JPEG, TIFF, ICO, alebo BMP a vektorový formát SVG, ktorého štandardná podpora bola do projektu pridaná. Posledným typom je dialóg na otváranie súborov s možnosťou filtrovania zobrazovaných súborov. Implicitne je zapnuté filtrovanie podľa prípony natívneho formátu aplikácie .ugml a .xml, ale hodnoty filtra je možne dodatočne nastaviť.

5.2 Export rastrových formátov

Na export rastrových formátov bolo použité rozšírenie grafickej knižnice GDK (GIMP Drawing Kit), ktorá pracuje medzi grafickým zobrazovacím protokolom (nad knižnicou Xlib) a prostriedkami grafického toolkitu GTK+ (GIMP Toolkit). Plocha na vykresľovanie aplikácie pracuje s rastrovou reprezentáciou – *pixmapou*, ktorá umožňuje prácu s farbami a má vyššiu bitovú hĺbku ako klasická *bitová mapa* (udáva len či sa ma daný pixel vykresliť). Do pixmapy sa vykreslujú grafické primitívy pomocou základných grafických výstupných funkcií.

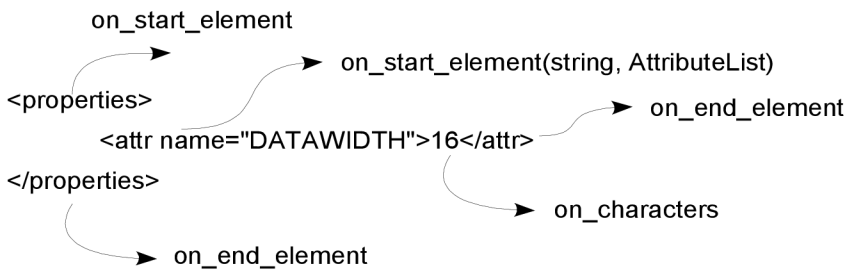
Export do vybraných formátov je realizovaný implementáciou funkcií, ktoré využívajú knižnicu gdk-pixbuf. Pomocou nej je možné načítať obrazové dáta z vykreslenej plochy a ďalej spracovať. Využíva sa funkcia *gdk_pixbuf_get_from_drawable*, ktorá skopíruje obrazové dáta z vykreslenej pixmapy typu *GdkDrawable* a konvertuje ich na RGB(A) reprezentáciu typu *GdkPixbuf*. Takto pripravené dáta potom môžu byť jednoducho uložené do niektorého z podporovaných rastrových formátov [15].

5.3 Implementácia modulu Import/Export

Na implementáciu modulu bol použitý jazyk C++ a na spracovanie formátu UGML bola použitá knižnica libxml++ [16]. Knižnica umožňuje efektívnu prácu s formátom XML v jazyku C++ a je postavená na veľmi rozšírenej knižnici libxml napísanej v jazyku C, originálne vyvinutej pre projekt GNOME. Knižnica libxml++ je prístupná pod licenciou LGPL verzie 2, ktorá umožňuje prípadné spojovanie aj s neslobodným (proprietárnym) softvérom.

5.3.1 DOM a SAX

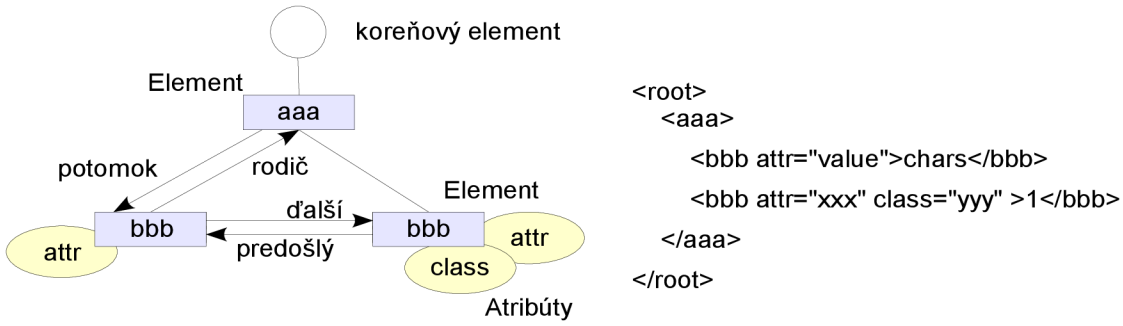
Na manipuláciu s dátami XML dokumentov je možné použiť rôzne prístupy. Najznámejšie aplikačné rozhrania na manipuláciu s XML formátom sú typu SAX (Simple API for XML) a DOM (Document Object Model).



Obrázok 5.1: SAX – postupné volanie udalostí

SAX umožňuje sériový prístup k XML tzv. udalosťami riadené spracovanie. Dokument sa rozdelí na jednotlivé časti podľa počiatočnej a koncovej značky, obsahu elementu a pod. Užívateľ si zaregistruje spätné volania funkcií na jednotlivé udalosti a tie sa pri prechádzaní dokumentom pri nájdení elementu postupne volajú. Ďalšie spracovanie je potom už v réžii programátora. Výhodou takéhoto sekvenčného spracovania, keď sa postupne prečíta celý dokument, je rýchlosť a nízka pamäťová náročnosť. Nevýhodou jednosmerného spracovania je nemožnosť vrátiť sa k dátam

z predošlých stavov bez opätovného prečítania dokumentu. Stavy a rozpracované dáta si musí programátor ukladať do vlastných stavových premenných a dátových štruktúr. Je omnoho pohodlnejšie pristupovať k ľubovoľnej časti XML dokumentu, čo je možné vďaka prístupu DOM.

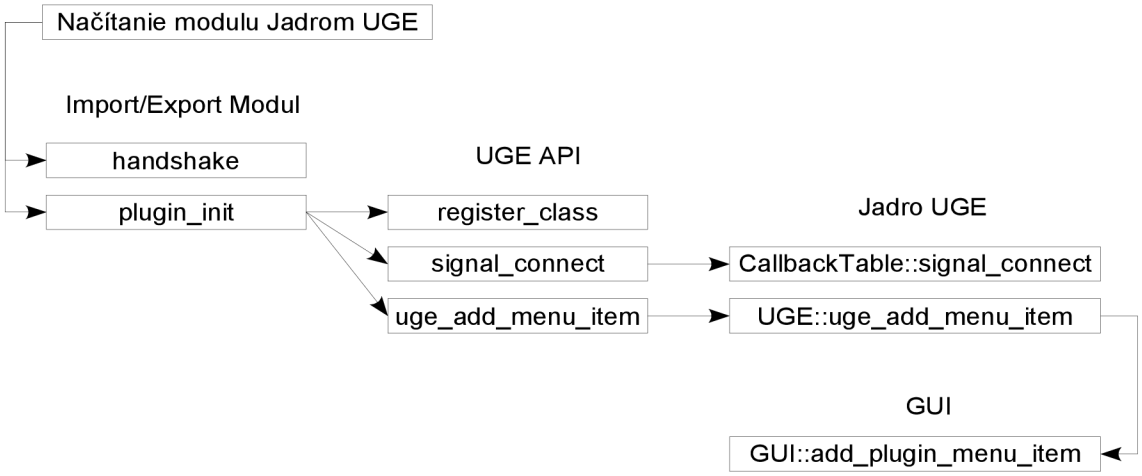


Obrázok 5.2: DOM – stromová hierarchia objektov

DOM umožňuje pristupovať k dokumentu v podobe stromovej reprezentácie. XML dokument sa celý uloží do pamäte ako stromová hierarchia objektov zastupujúca jednotlivé prvky, ktorými sú elementy, atribúty a textové dáta. Strom je potom možné ľubovoľne prechádzať a s objektami priamo manipulovať. Nevýhodou je vyššia pamäťová náročnosť a nižšia rýchlosť.

5.3.2 Inicializácia modulu

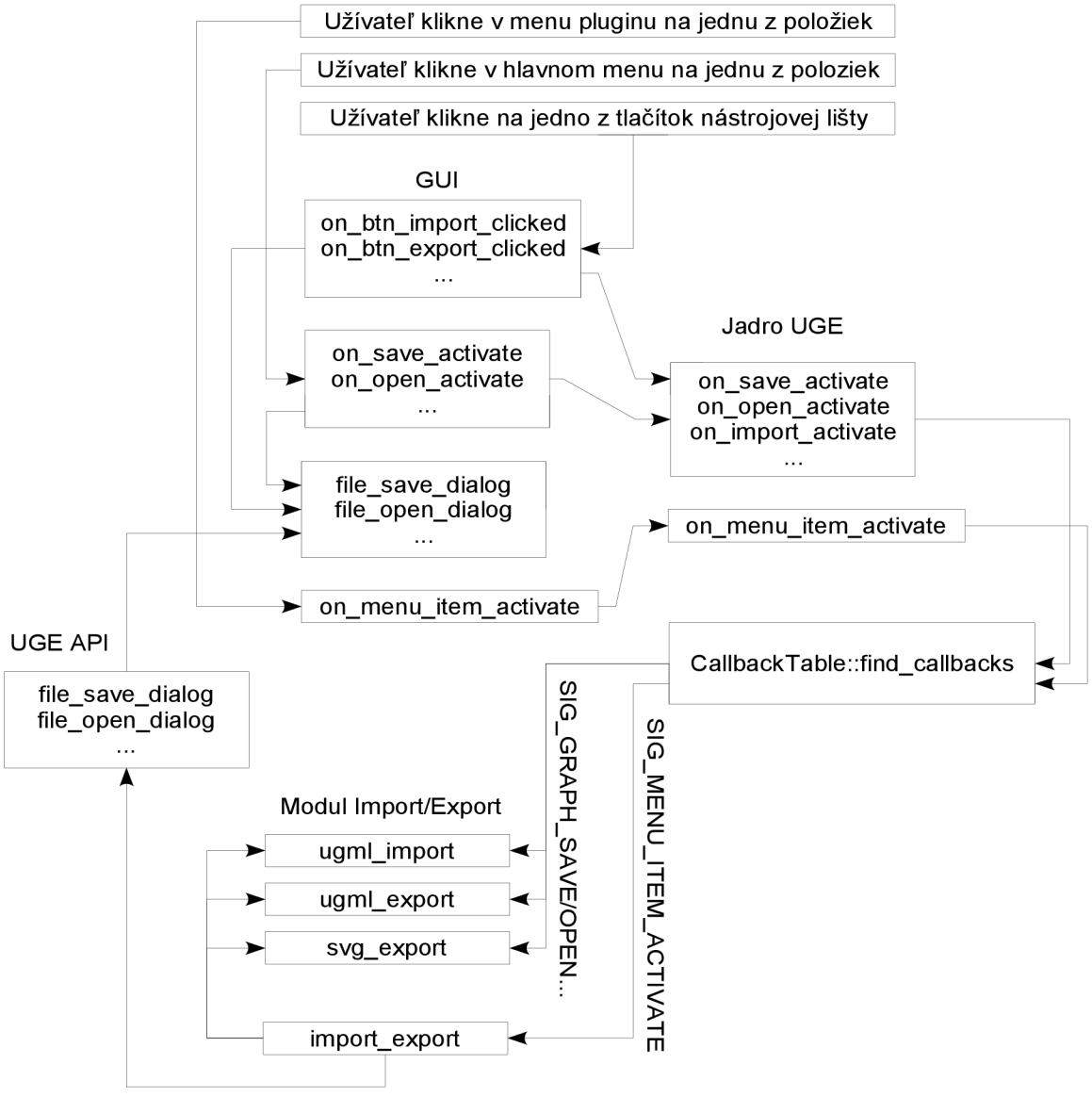
Pri načítaní modulu jadro volá funkciu *handshake*, vďaka ktorej sa získajú základné informácie o module (názov, verzia a popis). Následne zavolá inicializačnú funkciu *plugin_init,* v ktorej si pomocou funkcie *signal_connect* modul zaregistruje spätné volania funkcií pre signály otvoriť, uložiť, uložiť ako, import, export a pridajú sa obdobné položky aj do menu modulu (*uge_add_menu_item*). Nasledujúci obrázok schematicky znázorňuje hlavné volania funkcií pri inicializácii modulu.



Obrázok 5.3: Schéma priebehu inicializácie modulu

5.3.3 Spracovanie udalostí vyvolané užívateľom

Pri akciách užívateľa sa postupne volajú funkcie udalostí grafického užívateľského rozhrania. Ak sú to udalosti pre prácu s ukladaním a načítaním grafu, zobrazia sa príslušné súborové dialógové formuláre, vďaka ktorým má užívateľ možnosť zadať názov súboru na spracovanie, alebo ho vyberá s prístupného zoznamu súborov. Ak užívateľ potvrdí výber, volajú sa udalosti jadra, ktoré prehľadávajú tabuľku spätných volaní a postupne volajú napojené funkcie pre daný signál. Priebeh tejto komunikácie je zobrazený na obrázku 5.4.

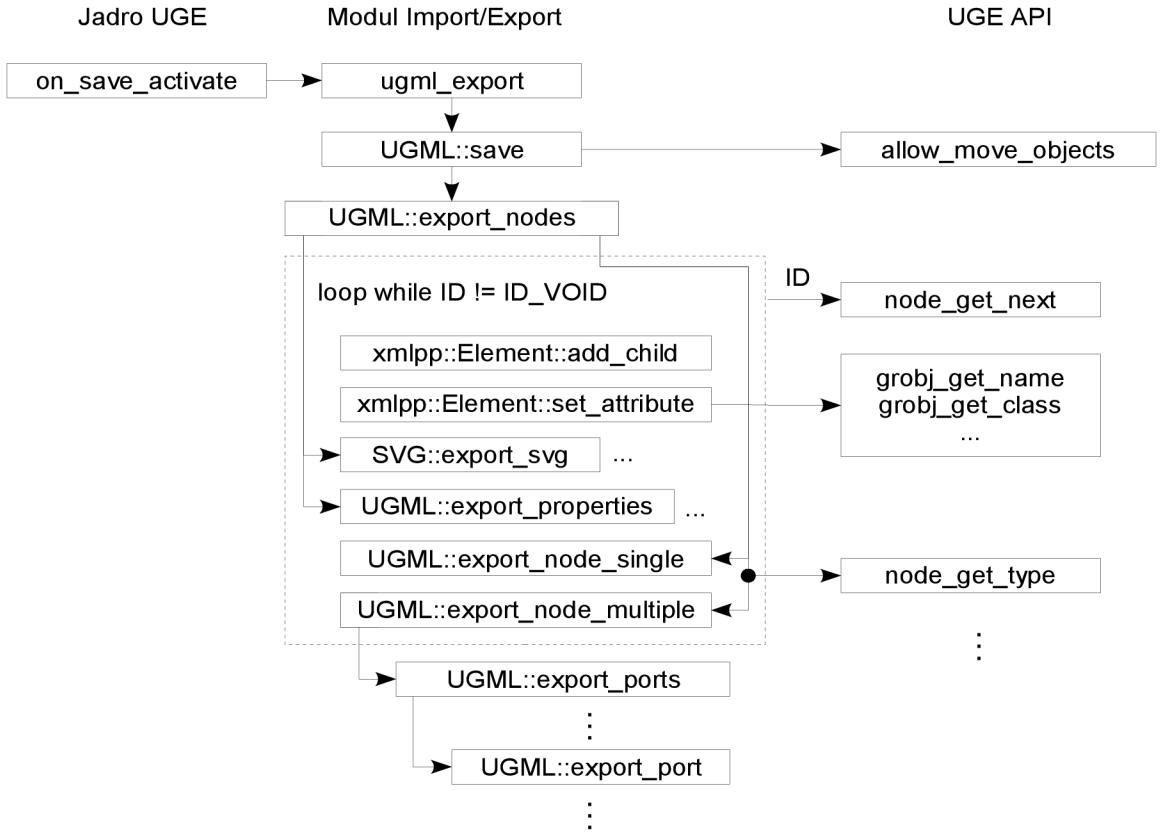


Obrázok 5.4: Schéma funkčných volaní pri udalostiach vyvolané užívateľom

5.3.4 Import/Export grafu

Pri aktivácii signálu SIG_GRAPH_SAVE užívateľom jadro zavolá funkciu pre export grafu, ktorá je na daný signál napojená. Volaná funkcia zablokuje možnosť hýbať s objektami počas ukladania a vytvorí koreňový element dokumentu. Vytvorí sa potomok (graph), ktorý je predaný ďalším funkciám na spracovanie uzlov a hrán. Postupným získavaním identifikátorov objektov grafu pomocou cyklenia sa pridávajú do stromu XML dokumentu ďalší potomci (node, edge). Volaním funkcií aplikačného rozhrania UGE sa pomocou identifikátora získavajú špecifické a spoločné vlastnosti objektov, ktoré sa nastavia atribútom XML elementov a volajú sa funkcie na spracovanie dynamických vlastnosti a grafickej reprezentácie objektu. V prípade zloženého uzlu sa zavolá funkcia na spracovanie skupiny portov a portov samotných. Volaným funkciám sa vždy predá aktuálne spracovávaný XML element a identifikátor objektu. Obrázok 5.5 znázorňuje začiatok takéhoto spracovania.

Import grafu pracuje obdobným spôsobom len obrátene. Do pamäte sa načíta obsah dokumentu, skontroluje sa verzia formátu a jeho správnosť a ak sa neodchytí výnimka hlásiaca chybu, rekurzívne sa prechádza stromom a postupne sa cez UGE API volajú funkcie na vytváranie objektov a nastavovanie ich vlastností. Nakoniec sa zavolá funkcia na prekreslenie kresliacej plochy.



Obrázok 5.5: Schéma funkčných volaní pri udalostiach vyvolané užívateľom

6 Záver

Predmetom práce bolo navrhnúť formát UGML (UGE Graph Markup Language), ktorý bude využívať systém UGE ako svoj natívny formát pre ukladanie a načítanie svojich dátových štruktúr. Implementovať export grafu do vektorového formátu SVG a niekoľkých rastrových formátov (PNG, JPEG, BMP...). Výstupom je aj riadkový prevodník grafickej časti formátu UGML do formátu SVG. Prínosom práce bolo rozšírenie možnosti jadra, užívateľského a aplikačného rozhrania projektu, ktoré v dostatočnej miere pokryjú požiadavky programátorov pri tvorbe zásuvných modulov. Dôraz bol kladený hlavne na manipuláciu s grafickou reprezentáciou grafu.

Možnosť podieľať sa na projekte Univerzálneho grafického editora priniesla veľa skúseností z oblasti vývoja väčších tímových projektov. Veľkým prínosom bolo zoznámenie sa zo spôsobom komunikácie zo zásuvnými modulmi a s princípom fungovania dynamických knižníc. Ponúknuť má čo aj premyslený objektovo orientovaný návrh, celková modularita a univerzalita systému, samotná realizácia komunikácie pomocou signálov a systém spätných volaní. Zavedenie systému CVS pre správu verzií zdrojových súborov projektu bolo taktiež prínosom.

Pri implementácii sa objavovali nové a nové nápady, ako ďalej rozširovať a zdokonalovať celý systém. Jeho koncepcia je tak premyslená, že je veľmi jednoduché rozširovať jeho možnosti a to nie len pomocou zásuvných modulov s jednoduchým využitím UGE API, ale aj jednotlivých účelne oddelených častí systému. Do úvahy pripadá napríklad zmena používanej grafickej knižnice na vektorovo orientovanú knižnicu Cairo, ktorá podporuje rozmanité výstupné zariadenia a jej aplikačné rozhranie poskytuje širokú škálu operácií. Sú nimi rôzne transformácie, skladanie priehľadných obrázkov, antialiasing textu a export do formátov PostScript, PDF a SVG sú samozrejmosťou. Systém by bol možný potom jednoducho portovať aj na iné platformy zmenou GUI a vďaka UGE API aj port do rôznych programovacích jazykov. Do užívateľského rozhrania by bolo možné doplniť formuláre na nastavovanie implicitných hodnôt napr. rôznych grafických vlastností primitív a zároveň umožniť modulom dynamicky pridávať ďalšie. Implementácia časovej osi pre animácie a mnoho ďalšieho.

Literatúra

[1] SKONNARD, A., GUDGIN, M.: *XML – pohotová referenční příručka : referenční příručka programátora ke XML, XPath, XSLT, XML Schema, SOAP a dalším*. 1. vyd. Praha : Grada, 2006. 342 s. ISBN 80-247-0972-4.

[2] STEPHEN, Prata: *Mistrovství v C++*. 2. aktualiz. vyd. Brno : Computer Press, 2004. 1006 s. ISBN 80-251-0098-7.

[3] prispievatelia Wikipédie.: *Graf (matematika) : Wikipédia, Slobodná encyklopédia*. [online]. Január 2007. [cit. 2007-05-02]. Dostupné na internete: <http://sk.wikipedia.org/w/index.php?title=Graf_%28matematika%29&oldid=760318>

[4] JADRNÝ, Miroslav: *Univerzální grafický editor*. Brno : FIT VUT v Brně, 2006. Bakalárska práca.

[5] Hewlett-Packard Development Company, L.P.: *HP-UX Linker and Libraries User's Guide : Shared Library Management Routines*. [online]. November 1997. [cit. 2007-05-05]. Dostupné na internete: <http://docs.hp.com/en/B2355-90655/ch06s06.html>

[6] BRAY, E. et. al.: *Namespaces in XML 1.0 (Second Edition) : W3C Recommendation 16 August 2006*. [online]. August 2006. [cit. 2007-05-03]. Dostupné na internete: <http://www.w3.org/TR/2006/REC-xml-names-20060816/>

[7] prispievatelia Wikipédie.: *Scalable Vector Graphics : Wikipédia, Slobodná encyklopédia*. [online]. Apríl 2007. [cit. 2007-04-27]. Dostupné na internete: <http://sk.wikipedia.org/w/index.php?title=Scalable_Vector_Graphics&direction=prev&oldid=933631>

[8] Bowler, J. et al.: *Scalable Vector Graphics (SVG) 1.0 Specification : W3C Recommendation 04 September 2001*. [online]. September 2001. [cit. 2007-04-29]. Dostupné na internete: <http://www.w3.org/TR/2001/REC-SVG-20010904/>

[9] Andersson, O. et al.: *Scalable Vector Graphics (SVG) 1.1 Specification : W3C Recommendation 14 January 2003*. [online]. Január 2003. [cit. 2007-04-28]. Dostupné na internete: <http://www.w3.org/TR/2003/REC-SVG11-20030114/>

[10] Andersson, O. et al.: *Scalable Vector Graphics (SVG) Tiny 1.2 Specification : W3C Candidate Recommendation 10 August 2006*. [online]. August 2006. [cit. 2007-04-29]. Dostupné na internete: <http://www.w3.org/TR/2006/CR-SVGMobile12-20060810/>

[11] Wikipedia contributors.: *PNG : Wikipedia, The Free Encyclopedia*. [online]. Máj 2007. [cit. 2007-05-08]. Dostupné na internete: <http://en.wikipedia.org/w/index.php?title=PNG&oldid=128731134>

[12] ADLER, M. et al.: *PNG (Portable Network Graphics) Specification Version 1.0 : W3C Recommendation 01-October-1996*. [online]. Október 1996. [cit. 2007-06-08]. Dostupné na internete: <http://www.w3.org/TR/REC-png-961001>

[13] ADLER, M. et al.: *Portable Network Graphics (PNG) Specification (Second Edition)... : W3C Proposed Recommendation 1 October 1996, revised 20 May 2003*. [online]. Máj 2003. [cit. 2007-05-08]. Dostupné na internete: <http://www.w3.org/TR/2003/PR-PNG-20030520/>

[14] ADLER, M. et al.: *Portable Network Graphics (PNG) Specification (Second Edition)... : W3C Recommendation 10 November 2003*. [online]. November 2003. [cit. 2007-05-08]. Dostupné na internete: <http://www.w3.org/TR/2003/REC-PNG-20031110/>

[15] QUINTERO, Mena: *The gdk-pixbuf Library*. [online]. Február 2007. [cit. 2007-05-08]. Dostupné na internete: <http://developer.gnome.org/doc/API/2.0/gdk-pixbuf/>

[16] CUMMING, M., DE VIENNE, C.: *libxml++*. [online]. Marec 2006. [cit. 2007-05-09]. Dostupné na internete: <http://libxmlplusplus.sourceforge.net/>