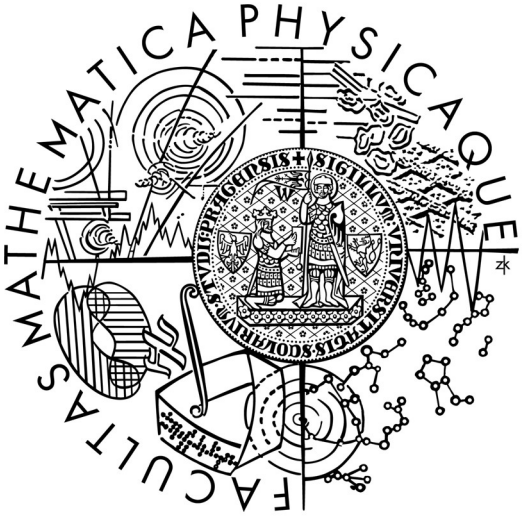
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Michal Halaša

Visualizátor struktury webu

Katedra softwarového inženýrství

Vedoucí diplomové práce:
RNDr. Leo Galamboš, Ph.D

Studijní program:
Informatika

Studijní plán:
Softwarové systémy

Poďakovanie

Chcel by som sa poďakovať RNDr. Leovi Galambošovi, Ph.D. za pomoc pri písaní tejto diplomovej práce.

Čestné prohlášení

Prohlašuji, že jsem svou diplomovou práci napsal(a) samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze, dne 17. dubna 2009

Michal Halaša

vlastnoruční podpis

Obsah

Abstrakt

Název práce: Visualizátor struktury webu

Autor: Michal Halaša

Katedra: Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Leo Galamboš, Ph.D.

e-mail vedoucího: galambos@cythres.cz

Abstrakt: Webový robot generuje strukturu webu včetně popisků (anchor texty) příslušejících jednotlivým odkazům. Cílem práce je navrhnout a implementovat metodu visualizace tohoto orientovaného grafu. Implementace bude realizována v prostředí Java.

Implementace musí být schopna dynamicky měnit zobrazení v závislosti na zvolených kritériích - zobrazování pouze určitých odkazů nebo webových stránek. Zároveň musí být možné shlukovat skupiny odkazů nebo webových stránek pro zjednodušení zobrazované struktury.

Klíčová slova: webový robot, orientovaný graf, hyperbolická vizualizácia, java 3d

Title: Visualizer of the Web Structure

Author: Michal Halaša

Department: Department of Sofware Engineering

Supervisor: RNDr. Leo Galamboš, Ph.D.

Supervisor's e-mail address: galambos@cythres.cz

Abstract: Web crawler generates web structure including the notes (anchor texts) which belong to individual links.Thesis target is to analyze and implement method for visualisation of this oriented graph structure. Implementation language platform is Java.

Implementation must be able to denamically change the view according to defined criteria – showing only certain links or web pages. The grouping of links or web pages must be also available.

Keywords: web crawler, oriented graph, hyperbolic visualization, java 3d

1 Motivácia

Webový vyhľadávací robot Egothor[1] pri prechádzaní štruktúry webových stránok generuje indexové súbory, v ktorých je popísaná štruktúra odkazov (previazaní) medzi jednotlivými prehľadávanými stránkami. Cieľom tejto diplomovej práce je vizualizácia týchto dát pomocou grafovej reprezentácie na obrazovke počítača tak, aby bola možná dynamická práca a manipulácia zobrazovanej množiny dát. Jedná sa pritom o dáta s miliónovými počtami uzlov a hrán medzi nimi a súčasné zobrazovacie grafové algoritmy sú optimalizované pre stromy s maximálne 100.000 uzlami. Pri väčších počtoch prvkov grafu dochádza k dramatickému poklesu prehľadnosti a čitateľnosti informácii v zobrazovaných grafových štruktúrach.

Takáto vizualizácia pomôže pochopiť a analyzovať spôsoby previazania stránok medzi sebou. Tieto údaje môžu byť vysoko použiteľné pre dizajnérov a architektov webových riešení pri analýze dostupnosti a prístupnosti jednotlivých častí webu. Taktiež dokážeme zobraziť vstupné a výstupné odkazy webových stránok do iných domén, čo pomôže pri SEO[2] optimalizácii.

Cieľom je jednoduché a intuitívne zobrazenie v užívateľsky príjemnom prostredí desktopovej aplikácie na platforme Java bez neštandardných externých závislostí.

V prvej kapitole si popíšeme algoritmus H3, z ktorého sme vychádzali, potom v stručnosti predstavíme existujúci program H3 Viewer a jeho fungovanie, a nakoniec popíšeme, ako sme postupovali pri implementácii našej verzie algoritmu H3. Na záver si zhrnieme výsledky a vymenujeme otvorené problémy, ktoré by sa dali vyriešiť v budúcnosti.

1 http://www.egothor.org
2 Search Engine Optimization

2 H3 Algorithm

Ako základ pre implementáciu našeho algoritmu poslúžil algoritmus H3 publikovaný v roku 2000 v doktorskej práci Tamary Munzner s názvom *Interactive Visualization of Large Graphs and Networks* na Stanfordskej univerzite v Kalifornii [1].

Algoritmu H3 bol priamo navrhnutý na zobrazovanie štruktúry webových stránok a odkazov medzi nimi. Pre zobrazenie používa 3D hyperbolický priestor, ktorý je premietaný do euklidovskej gule v troj rozmernom priestore, ktorá je premietaná na dvoj rozmerný monitor počítača. Jeho výhodou je starostlivo vybraná štruktúra kostry grafu a možnosť zmeny analyzovaného miesta v grafe pomocou techniky Focus+Context View – veľké okolie je viditeľné okolo práve analyzovaného miesta v grafe. Tým je dosiahnuté to, že aj pri veľkých grafových štruktúrach je v každom mieste grafu rovnaká informačná hustota.

Je uspôsobený na vizualizáciu špeciálneho typu grafov - kvazi-hierarchických grafov. Jedná sa o grafy, pre ktoré je pri stavbe kostry jednoducho definovateľný preferovaný predok uzlu zo všetkých odkazov do uzlu. Z pohľadu teórie grafov ide o graf s minimálnou kostrou, kde váhou na hranách je informácia o polohe uzla (adrese stránky). Jedná sa o orientované stromy, ktoré ale nemusia byť nutne acyklické. Výhodou však je, ak existuje čo najmenšie množstvo spätných hrán.

Vizualizácia dát previazania webových stránok pomocou grafu je obzvlášť vhodná, keďže priamo štruktúra DNS záznamov má stromovú štruktúru, a aj väčšina stránok má ako základnú organizačnú štruktúru strom. Tiež je veľmi dôležité, že zobrazovanie informácii pomocou stromového zobrazenia je pre ľudí prirodzené, intuitívne a veľmi jednoduché na pochopenie a ovládanie.

Hyperbolický priestor bol zvolený pre zobrazovanie preto, lebo poskytuje projekciu nekonečného priestoru do konečného vykresľovacieho priestoru a vďaka tejto vlastnosti, môže mať každý uzol pridelený dostatočne veľký zobrazovací priestor tak, aby nedochádzalo ku kolíziám.

Prehusteniu, a tým pádom strate prehľadnosti, pri projekcii veľkých a hustých grafov sa predchádza pomocou možnosti prechádzania po grafe – približovaniu a odďaľovaniu jednotlivých uzlov a taktiež orezávaniu grafu podľa potreby.

V euklidovskom priestore môžu byť stromy vykreslené iba v prípade, že smerom k listom sa priestor pridelený jednotlivým uzlom exponenciálne zmenšuje, keďže počet uzlov sa vo vyváženom grafe exponenciálne zvyšuje. Potom v prípade analyzovania dát pri koreni stromu sa úplne stráca schopnosť simultálnej analýzy dát pri listoch z dôvodu exponenciálne odlišnej mierky.

Narozdiel od toho, môže v hyperbolickom priestore zaberať každý uzol približne rovnaký priestor, a preto každá časť grafu nesie rovnakú informačnú hodnotu.

Pre podrobný popis zvoleného postupu zobrazovania a informácie o matematickom modele použitom pri premietaní hyperbolického priestoru do Euklidovského 3D priestoru, viď kapitolu 3.2 v práci Tamary Munzner [1].

3 WebView

Našim cieľom bolo prepísanie pôvodnej implementácie algoritmu H3 v jazyku C++ do programovacieho jazyka Java na platforme JavaSE s použitím knižnice Java3D ako zobrazovacej technológie 3D grafiky. Vstupom (zobrazovanými dátami) boli indexové súbory vygenerované webovým robotom Egothor.

Egothor je výkonný open-source textový vyhľadávací stroj, napísaný výlučne v Jave s podporou 64bitov, výkonným indexovacím algoritmom, inteligentným rozpoznávaním indexovaných dát a univerzálnym značkovacím algoritmom.

3.1 Použité technológie

Platforma Java bola zvolená z dôvodu svojej multiplatformnosti a hlavne preto, že celý webový robot Egothor je naprogramovaný v Jave. Konkrétne bola zvolená desktopová verzia Javy – JavaSE – a to vo verzii 6, a preto je potrebné pred spustením aplikácie WebView nainštalovať na počítač JRE[3] tejto verzie. Veľkou výhodou tejto verzie Javy je prepracovaná podpora pre Swing (konkrétne Swing Application Framework) a celková integrácia s operačným systémom.

Z použitia Javy ako platformy vychádza aj použitie Java3D ako knižnice pri vykresľovanie akcelerovanej 3D grafiky. Táto knižnica poskytuje high level prístup ku grafickému procesoru a odtieňuje programátora od programovania konkrétnych grafických primitív, čo urýchľuje a zjednodušuje prácu s 3D grafikou a nevnucuje koncovému užívateľovi použitie konkrétnej implementačnej knižnice 3D rozhrania – podporované sú, okrem iných, aj najpoužívanejšie technológie na prácu s 3D grafikou - OpenGL a DirectX.

Nevýhodou použitia Javy3D bolo to, že jej základná komponenta pre vykresľovanie – *Canvas3D* – je tzv. heavyweight[4] komponenta, a preto jej súčasné použitie s lightweight komponentami Swingu môže pôsobiť problémy, a preto sme umiestnili vykresľovanie 3D grafiky do samostatného okna (*JFrame*). V novej verzii Java3D 1.5.2 už existuje aj lightweight komponenta *JCanvas3D*, ale prepis s použitím tejto komponenty nie je triviálnou operáciou, takže zostáva ako plán do budúcnosti.

Ďalšou použitou technológiou ORM[5] mapovanie pomocou JPA[6] do databáze. Pomocou technológie JPA sa odtieňuje nutnosť používania SQL dotazov pri manipulácii s objektami v DB a taktiež nie je potrebné vytvárať pred použitím schému pomocou DDL príkazov.

3 Java Runtime Environment (http://www.java.com/en/download/index.jsp)
4 Viac o heavyweight a lightweight komponentách na http://java.sun.com/products/jfc/tsc/articles/mixing/
5 Object-Relational Mapping (http://en.wikipedia.org/wiki/Object-relational_mapping)
6 Java Persistence API (http://en.wikipedia.org/wiki/Java_Persistence_API)

Ako databáza je použitá Apache Derby[7] z dôvodu jednoduchosti použitia, nízkej spotreby pamäti RAM a možnosti použitia v embedded móde (nie je potrebná inštalácia serveru, databázový engine je spustený v JVM klientského procesu).

Samostatná implementácia prebiehala na platforme Netbeans[8], a to konkrétne vo verzii 6.5+. Dôvodom pre výber tejto platformy bolo znova jej použitie pri vývoji aplikácie Egothor a hlavne jej excelentná podpora frameworku Swing a desktopovej Javy všeobecne.

3.2 Implementácia H3 algoritmu

Ako inšpirácia pri našej implementácii H3 algoritmu poslúžil program H3Viewer[9], napísaný v jazyku *C++* s použitím knižníc *OpenGL/Mesa* a *STL*. a predchádzajúce práce Tamary Munzner na algoritme H3, ktorých implementácie sú voľne prístupné – WebOOGL[10] a Geomview[11].

Implementácia algoritmu H3 existovala aj v komerčnom produkte Site Manager firmy SGI, ale tento produkt už dlho nie je ani podporovaný ani vyvíjaný. Program zobrazoval štruktúru konkrétneho webu a bol určený pre dizajnérov webových stránok.

Naša implementácia je v programe Java na platforme JavaSE s pooužitím technológie Java3D na vykresľovanie 3D grafiky, čím sa zabezpečí plná prenositeľnosť medzi rôznymi architektúrami.

3.3 H3 Viewer

Zdrojové kódy programu H3 Viewer[12] poskytli dobrý náhľad do dátovej štruktúry použitej pre H3 algoritmus a hlavne z nich vidieť algoritmy použité na stavbu a vykreslenie kvazi-hierarchického stromu v hyperbolickom priestore. Analýza týchto zdrojových kódov bola na začiatku našej práce hlavným zdrojom informácii a inšpirácie. Prepisom C++ kódu do Javy sme získali základ dátovej štruktúry a hlavne netriviálne matematické manipulácie s dátami potrebné pri výpočte a projekcii súradníc prvkov grafu medzi hyperbolickým a Euklidovským priestorom.

7 http://db.apache.org/derby/
8 http://www.netbeans.org
9 http://graphics.stanford.edu/papers/h3draw/
10 http://www.geom.uiuc.edu/software/weboogl/
11 http://www.geomview.org/docs/
12 http://graphics.stanford.edu/~munzner/h3/

Vstupom pre H3 Viewer je textový súbor, v ktorom každý riadok reprezentuje jeden vrchol grafu a má formát:

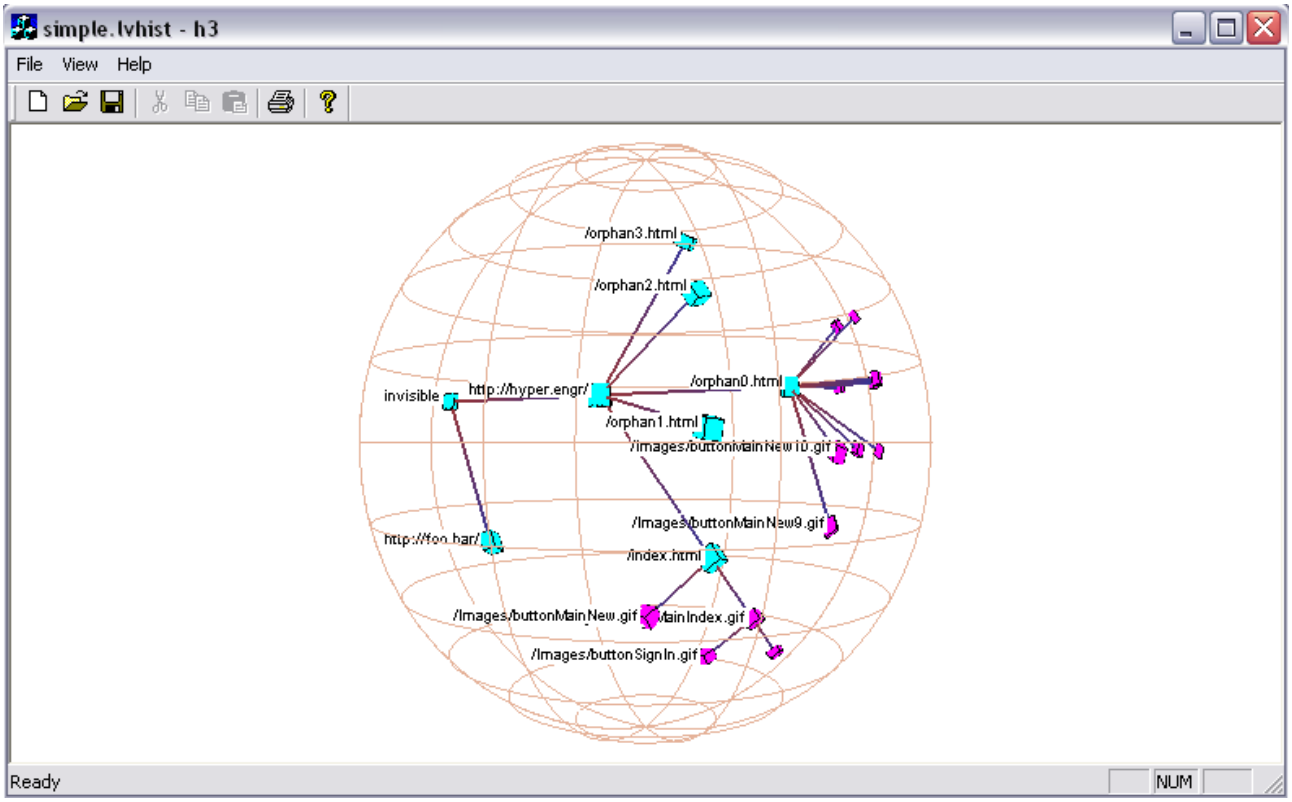
```
depth identifier 1 group1 [group2 ... groupN]
```

Príklad:

```
0 http://hyper/ 1 html main
1 http://hyper/index.html html main
1 http://hyper/logo.gif image main
1 http://hyper/old.html html orphan
```

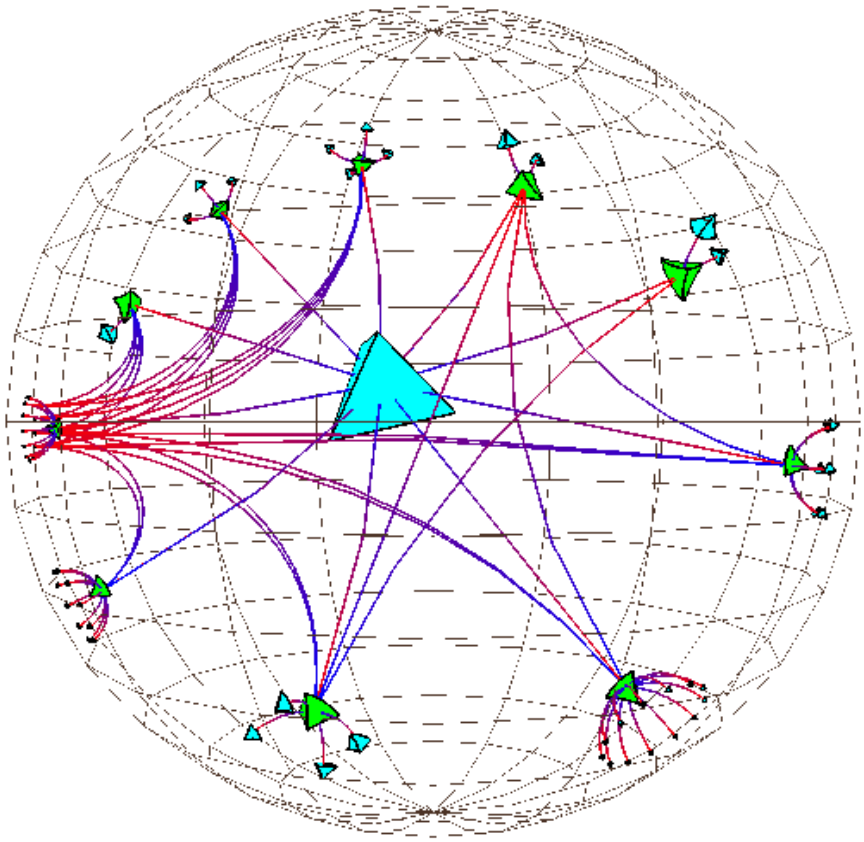
Výstupom je vizuálna reprezentácia grafu v hyperbolickom priestore premietnutom do gule v Euklidovskom priestore. Pri jednotlivých uzloch sú zobrazené URL adresy uzlov.

Zobrazenie jednoduchého grafu s cca 25 uzlami je na Obrázku č.1.



Obrázok 1: H3 Viewer s jednoduchým grafom

Predchodcom programu H3 Viewer bol projekt Webviz[13], pomocou ktorého bol analyzovaný princíp zobrazovania dát v hyperbolickom priestore a možnosti premietania grafu z euklidovského do hyperbolického priestoru (Obrázok č.2).



Obrázok 2: Webviz – conformal model projekcia

3.4 Dátové štruktúry

Základom dátovej štruktúry zobrazovaného grafu je myšlienka oddelenia zobrazovaných informácii a tzv. dodatočných dát, ktoré nie sú priamo nutné k vykresleniu grafu (popisné informácie uzlov a hrán). Tým sa dosiahne to, že v pamäti (grafickej karty aj RAM) budú uložené len momentálne potrebné dáta a všetky ostatné informácie sa budú dohľadávať len pri určitej akcii užívateľa. Vzhľadom na vysokú náročnosť na CPU pri predspracúvaní zdrojových dát a na použitie pamäti RAM pri ich vykresľovaní je každá úspora v množstve spracovávaných dát dôležitá.

Preto pre reprezentáciu vrcholov, hrán grafu a ich vlastností (ako napríklad farba, viditeľnosť, súradnice v Euklidovskom aj hyperbolickom priestore) boli použité polia s presne definovaným

13 http://graphics.stanford.edu/papers/webviz/

poradím prvkov v nich. Kladie to určitú mieru zodpovednosti pri manipulácii s takto uloženými dátami, ale na druhej strane to prináša úsporu použitej pamäte oproti sofistikovanejším objektovým modelom reprezentácie dát.

V najväčšej miere bol tento prístup použitý v triede *webview.graph.Graph.*

3.5 Vstupné dáta

Výstup webového robota Egothor je rozdelený do 4 typov súborov:

1. *pgw*.lnk* – popis štruktúry odkazov

```
1221 2: 3125 5554
4663 0:
1783 7: 0 5555 1000 5556 5557 5558 0
```

2. *pgw*.red* – popis štruktúry automatických presmerovaní (http redirect)

```
19 121
78 606
5 831
665 1109
```

3. *pgw*.anc* – texty odkazov na stránkach

```
19 121
78 606
5 831
665 1109
```

4. *doc.aux* – názvy a adresy stránok

```
1 http://www.cuni.cz:80/
2 http://www.cuni.cz:80/UK-21.html
3 http://www.cuni.cz:80/?view=text
4 http://www.cuni.cz:80/UKENG-1.html
```

Podrobnejší popis sa nachádza na stránkach[14] projektu Egothor.

3.6 Štruktúra zdrojových kódov

Zdrojové kódy programu sú priložené na CD v podadresári *SOURCE* ako Netbeans projekt.

Obsah jednotlivých podadresárov adresára *src/* podľa abecedy:

- *META-INF/*
  súbor *persistence.xml* s nadefinovaným ORM mapovaním do embedded databáze Apache Derby[15], ktorá sa používa na ukladanie dodatočných informácii potrebných pri vykresľovaní grafu, tj. url stránok, texty a url odkazov zo stránok
- *webview/*
  okná a dialógy použité v programe
- *webview/base*
  JPA[16] triedy mapované do DB a obslužné triedy na načítanie grafovej štruktúry z DB
- *webview/generate/*
  testovacie triedy na generovanie grafových štruktúr
- *webview/graph/*
  triedy reprezentujúce vykresľovaný graf, základom je trieda *webview.graph.Graph*
- *webview/h3/*
  matematická, vykresľovacia a transformačná trieda pre hyperbolický priestor
- *webview/navigation/*
  navigačné triedy na obsluhu podnetov z polohovacieho zariadenia

14 https://www.egothor.org/~galambos/twiki/bin/view/Egothor/RobotFiles
15 http://db.apache.org/derby/
16 http://en.wikipedia.org/wiki/Java_Persistence_API

- *webview/render/*
  triedy vykresľujúce graf (upravené Java3D vykresľovacie plátno Canvas3D, kresba os XYZ a trieda obsahujúca nastaviteľné parametre)
- *webview/resources/*
  doplnkové nastavenia pre Swing Application Framework

3.7 Spustenie a ovládanie programu

Doporučené parametre pri spustení:

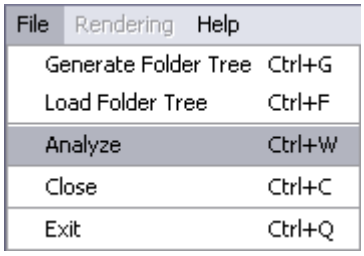
```
$> java -Xms256m -Xmx1536m -jar WebView.jar
```

Prvý parameter nastavuje počiatočnú veľkosť alokovanej pamäte pre JVM a druhý parameter nastavuje maximálnu veľkosť. Hodnota cca 1536MB je maximálna hodnota pre 32-bitovú JVM. V prípade, že sa na počítači nenachádzajú aspoň 2GB operačnej pamäti, tak môže byť táto hodnota znížená, ale treba počítať s degradovaním výkonu.

K ideálnemu ovládaniu programu je vhodná kombinácia polohovacieho zariadenia (myš) a klávesnice.

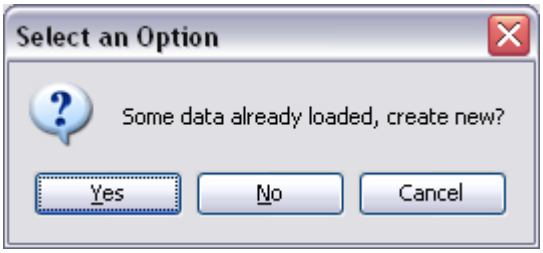
V prípade, že nemáme k dispozícii vstupné dáta z robota Egothor, tak pre prezentáciu schopností H3 Algorithmu v programe WebView sú určené prvé dve položky menu *File* (Obrázok č.3). Pomocou *Generate Folder Tree* sa vygeneruje strom z adresárovej štruktúry lokálneho disku zo zvoleného adresára a jeho podadresárov. Takto vygenerovaný strom sa potom zobrazí pomocou položky *Load Folder Tree*. V tomto prípade ale nie je možné zobrazovať informácie k vybraným uzlom, keďže nie sú dostupné. K dispozícii je plne funkčná navigácie po strome.

Štandardný postup spustenia začína vo vstupnej metóde *main* triedy *webview.WebViewApp*, ktorá zobrazí okno *webview.MainView*. V ňom pomocou menu vyvoláme analýzu externých vstupných dát z programu Egothor pomocou položky menu *File → Analyze*.



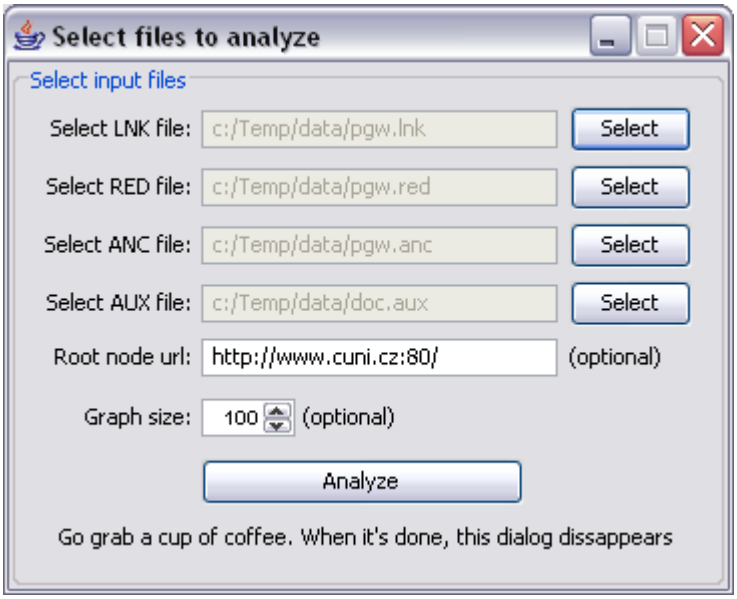
Obrázok 3: Menu File

V prípade, že už boli predtým zanalyzované, tak sa ukáže potvrdzovací formulár na premazanie dát (Obrázok č.4).



Obrázok 4: Potvrdenie premazania vstupných dát

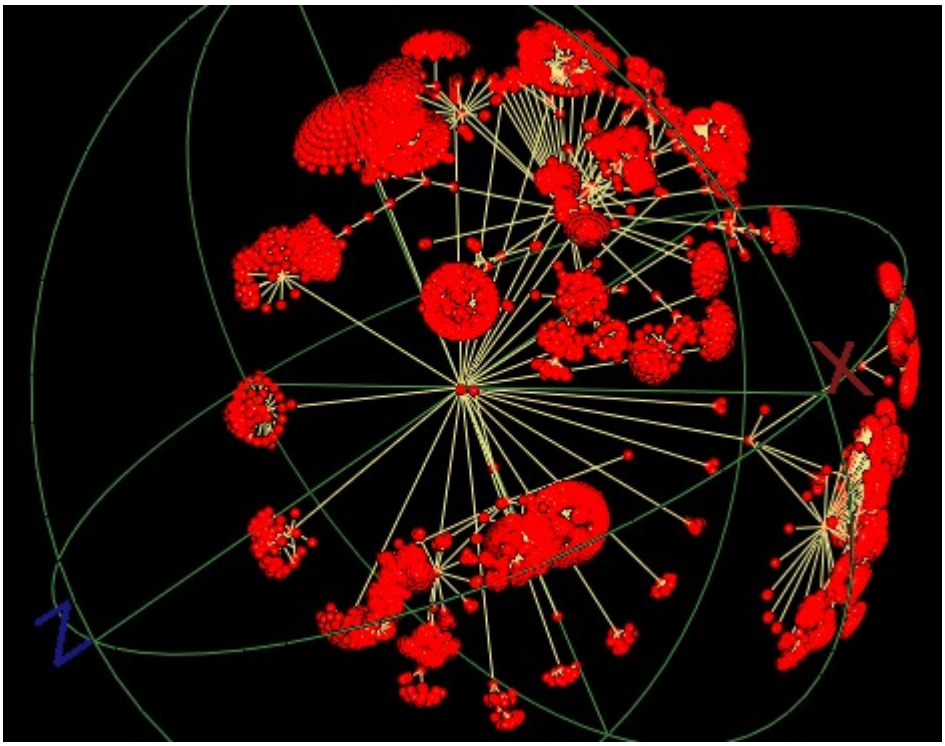
Po potvrdení sa zobrazí dialógové okno pre zadanie vstupných parametrov a vstupných súborov (Obrázok č.5).



Obrázok 5: Zadanie vstupných parametrov

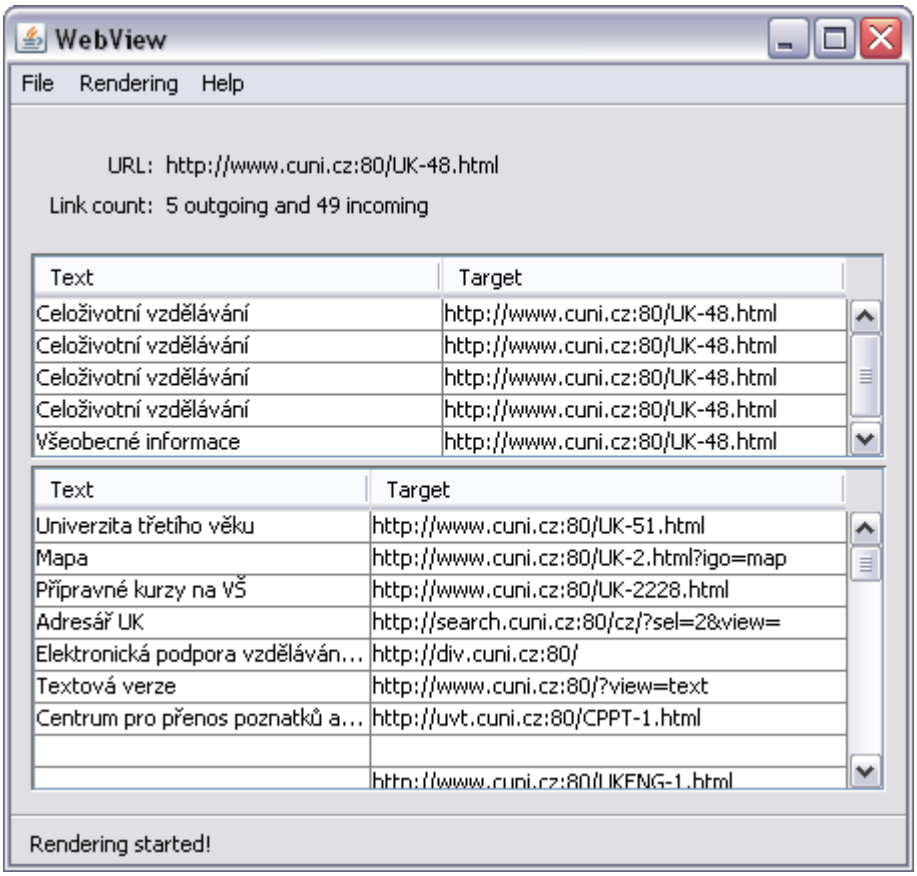
Po potvrdení tlačítkom *Analyze* sa spustí vstupná analýza a transformácia dát do dátovej štruktúry potrebnej na vykreslenie grafu a ich následné uloženie do DB. Táto operácia je časovo veľmi náročná v priamej úmere s veľkosťou vstupných dát od niekoľkých sekúnd po niekoľko hodín, preto je pre testovanie vhodné zadávať parameter *Graph size* dostatočne malý. Ak je špecifikovaný ako nula, tak sa spracujú všetky dostupné dáta.

Po skončení analýzi dát sa zobrazí samotný graf (Obrázok č.6).



Obrázok 6: Načítaný graf

V hlavnom okne (Obrázok č.7) sa zobrazujú informácie k vybranému uzlu (pomocou kliknutia myšou) a to jeho adresa a počet a popis jednotlivých odkazov z uzlu reprezentujúceho stránku (vstupných aj výstupných).



Obrázok 7: Informácie o vybranom uzle reprezentujúcom stránku

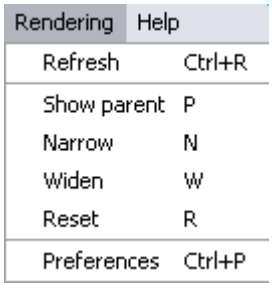
Navigácia grafu pomocou myši:

- *left-click* – výber uzla
- *right-click* – presunutie vybraného uzla do stredu vykresľovaného priestoru
- *drag-and-drop* – otáčanie zobrazovaného priestoru
- pohyb kolieskom myši – približovanie a odďaľovanie

Navigácia pomocou klávesnice:

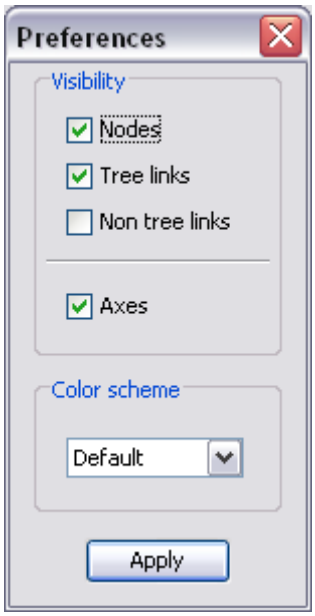
- N – orezanie grafu na podgraf vybraného uzla
- W – následne po orezaní grafu pomocou W nastane posun po ceste smerom do koreňa
- P – presun do predka po ceste do koreňa

Navigácie po grafe je tiež možná pomocou výberu položiek v menu *Rendering* (Obrázok č.8).



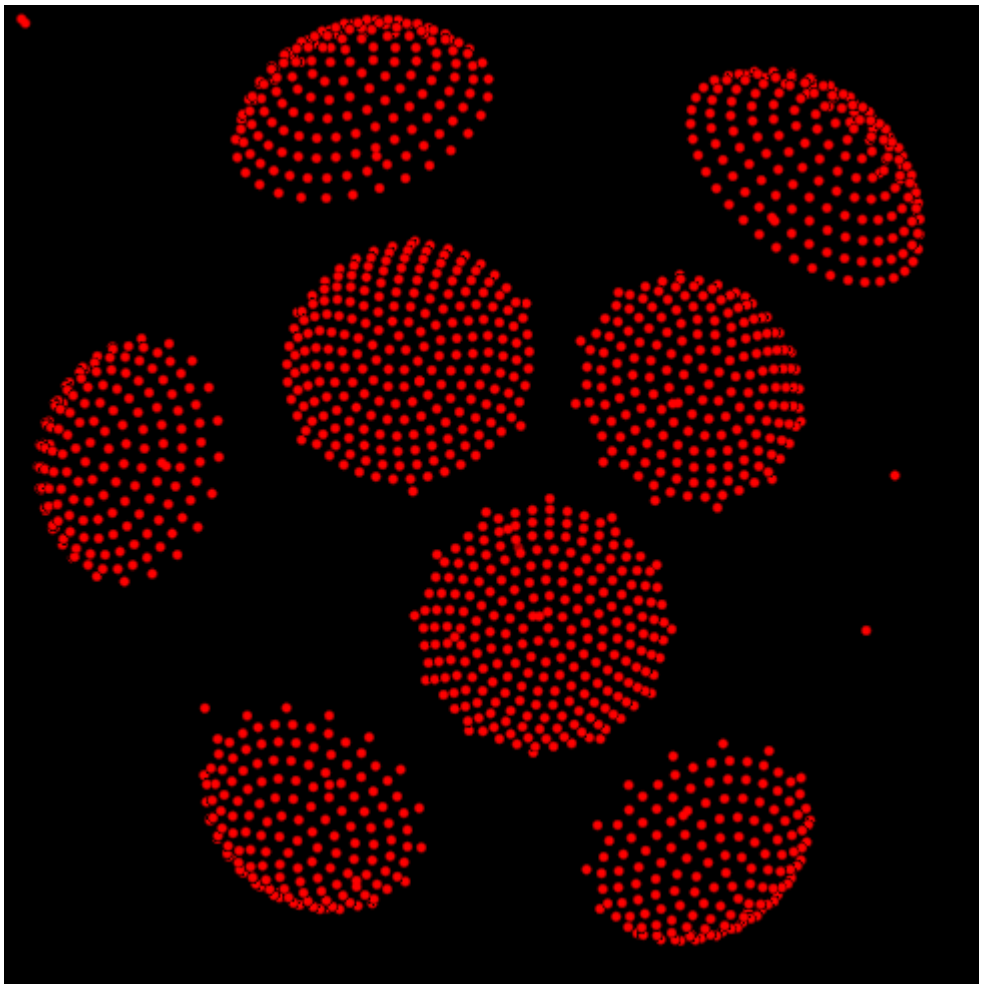
Obrázok 8: Menu Rendering

Cez podpoložku *Preferences* sa vyvolajú predvoľby nastavenia grafu (Obrázok č.9).



Obrázok 9: Nastavenia

Napríklad pomocou vypnutia (stromových aj nestromových) odkazov a os XYZ sa zobrazia len vrcholy grafu (Obrázok č.10).



Obrázok 10: Zobrazenie len vrcholov grafu

3.8 Dôležité triedy

- *webview.GlFrame extends JFrame*
  okno s Java3D obsahom
- *webview.base.Analyze*
  načíta vstupné súbory podľa zadaných parametrov v metóde a získanú dátovú štruktúry uloží do embedded databáze
- *webview.Graph*
  trieda reprezentujúca vykreslený graf, jeho vrcholy a hrany a ich vlastnosti
- *webview.navigation.EventHandler implements KeyListener, MouseListener, MouseMotionListener, MouseWheelListener*
  obsluha vstupov (klávesnica a myš) od užívateľa
- *webview.render.MainRenderLoop implements Runnable*
  navigácia a manipulácia s grafom
  implementuje interface *Runnable* aby nedochádzalo k oneskoreniu alebo oneskorenému vykonávaniu príkazov od užívateľa
  dôležité metódy:
  - *render()* nastavuje, ktoré uzly a hrany sa majú vykresliť
  - *pickNode()* na výber vrcholu
  - *rotateDisplay()* na rotáciu zobrazovanej gule
  - *translate()* použitá pri presune vrcholu do počiatku
- *webview.render.VertexRenderer*
  stará sa o vykresľovanie uzlov a hrán grafu zo štruktúry triedy *Graph*
- *webview.render.ViewParameters*
  zoznam všetkých empirických konštánt a vizuálnych reprezentácii

Všetky triedy v zdrojových kódoch programu obsahujú stručný popis funkcionality.

4 Zhrnutie

Podarilo sa nám vytvoriť program na vizualizáciu dát generovaných pomocou webového robota Egothor. Program je schopný zobrazovať stromové grafy s niekoľkými miliónmi objektov v reálnom čase. Ich predspracovanie zo surového vstupového stavu do stavu použiteľného pre WebView je síce značný, ale jednorázový, takže akceptovateľný.

Škálovateľnosť programu je možná pomocou zväčšovania rozsahu pamäti RAM, keďže na 32-bitovom operačnom systéme s 32-bitovým JRE[17], ktorá má obmedzenie 2GB per JVM[18] zvláda vykresliť do 50 miliónov objektov grafu, takže napríklad cca 10M uzlov a 35M hrán medzi nimi. V 64-bitovej Jave by sme sa mohli ľahko dostať k niekoľko násobným číslam. Vykresľovanie takto veľkých grafov je stále relatívne interaktívne.

Do budúcnosti sú možné tieto vylepšenia:

- postupné vykresľovanie grafu, ktoré by umožnilo zobrazovanie grafov, ktoré sa nezmestia do pamäte RAM
- rozšírené možnosti inteligentného zhlukovania uzlov a odkazov (pomocou dodatočnej informácie, napríklad URL)
- vylepšenie vykresľovacieho algoritmu aby lepšie pokrýval celý zobraziteľný priestor

4.1 Problémy

Najväčším problémom bolo obmedzenie veľkosti alokovanej RAM v 32-bitových operačných systémoch, ktoré nedovoľovalo prácu s veľkými grafmi s desiatkami miliónov vrcholov.

Ďalším, už spomínaným, problémom bola určitá nedokonalosť knižnice Java3D, keďže firma SUN upustila od vývoja tejto knižnice z dôvodu zamerania sa na iné zobrazovacie technológie – primárne JavaFX. To sa prejavovalo hlavne v nedokonalej spolupráci vykresľovacej komponenty Camvas3D s komponentami z knižnice Swing.

Pri vykresľovaní veľkých a hustých grafov (milióny prvkov) dochádzalo niekedy k pádu ovládača grafickej karty z dôvodu chyby v implementácii Java3D.

```
[Java3D] Warning: Fail to lock Vertex Buffer - D3DERR_DRIVERINTERNALERROR
```

17 Java Runtime Environment -kolekcia behového prostredia a utilít
18 Java Virtual Machine – behové prostredie Javy

Prílohy

K diplomovej práci je priložené CD so zdrojovými kódami programu, s testovacími dátami a touto diplomovou prácou vo formáte PDF.

Bibliography

[1] Interactive Visualization of large Graphs and Networks: Tamara Munzner, http://graphics.stanford.edu/papers/munzner_thesis/all.onscreen.pdf

[2] Soumen Chakrabarti: Mining the Web: Discovering Knowledge from Hypertext Data. Amsterdam: Morgan Kaufmann, 2003.