České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra počítačů

Diplomová práce

Živé divadlo

Leden 2005 Daniel Nytra

Anotace

Živé divadlo

Cílem této práce je ukázat směr při řešení problémů vizualizace prostorových dat. Pomáhá řešit problémy převodu nejpoužívanějšího formátů z měřické dokumentace stavebních objektů (.DGN) na jeho vizuální prezentaci (.VRML). Aplikace, která zobrazuje tyto prostorová data, je vytvořena pro zkoumání daných prvků modelu. Práce dále přehledně ukazuje optimalizaci jednotlivých formátů v rámci celého projektu.

Vše je vytvořeno a demonstrováno na barokním divadle v Českém Krumlově (památka UNESCO).

Abstract

Living theater

The aim of this work is to show the way in solutions of spatial data visualization. The work helps to solve the problems of conversion from the most used format in metrical documentation of structural objects (.DGN) to its visual presentation (.VRML). Application, which shows these spatial data, is created for the research of the model elements. The work otherwise clearly shows optimizing of each format in the horizon of the whole project.

Everything is created and demonstrated on a baroque theater in Český Krumlov - UNESCO historical sight.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 26. ledna 2005

podpis

Poděkování

Rád bych poděkoval všem, kteří přispěli k vypracování této práce podnětnými připomínkami, radami a literaturou. Byli to zejména vedoucí práce Doc.Ing. Jiří Žára, CSc. , konzultant Ing. Jindřich Hodač, Ph.D. a kolegové Radim Balík, Ing. Martin Čadík, Ing. Martin Klíma a můj dlouholetý kamarád bc. Martin Ptáček.

Dále bych chtěl poděkovat Nadaci barokního divadla zámku Český Krumlov za poskytnutí finančních prostředků na vytištění a vazbu této diplomové práce.

Daniel Nytra

Věnování

Tuto práci věnuji člověku, o kterého se vždycky můžu opřít v dobách temna.

Děkuji za všechno, maminko.

Obsah

Seznam obrázků

Seznam tabulek

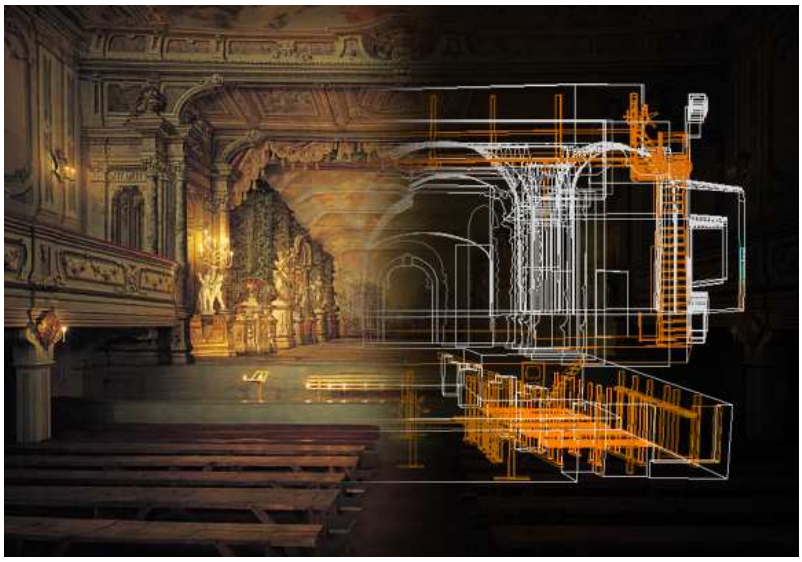
Kapitola 1

Úvod

1.1 Historie

Barokní divadlo, které se nachází na 5. nádvoří zámku v Českém Krumlově, je unikátní, kompletně dochovanou barokní divadelní scénou. Budovu divadla nechal postavit v roce 1680 kníže Jan Kristián z Eggenberku. Unikátní techniku na výměnu kulis a ostatních dekorací zhotovil vídeňský tesař Lorenz Makh, nástěnné malby, dekorace stropu, opona a kulisy byly vytvořeny rovněž vídeňskými malíři - Hans Wetschel a Leo Märkl.

Tato diplomová práce je součástí dlouhodobého projektu, který probíhá ve spolupráci Laboratoře fotogrametrie Katedry mapování a kartografie Fakulty stavební ČVUT, Správy zámku Český Krumlov a Nadace barokního divadla. V předchozích letech, bylo cílem vytvoření digitálního 3D modelu divadla (viz. Obrázek 1.1), který by měl v budoucnu sloužit k prezentaci, bádání a správě divadla. Celý projekt se stejně jako moje diplomová práce nazývá Živé divadlo, protože by výsledkem měl být vysoce dynamický prostorový model tohoto vzácného objektu. Zadavatelem tohoto projektu je Správa státního hradu a zámku v Českém Krumlově.



Obrázek 1.1: Barokního divadlo a jeho 3D model

Téma vizualizace barokního divadla v Českém Krumlově jsem si vybral v předmětu 36MUS v roce 2003. V rámci tohoto předmětu jsem převáděl prostorová data z formátu programu Microstation (.DGN) do formátu VRML. Hlavním účelem semestrální práce

bylo procházení virtuálního modelu v reálném čase s dostatečnou plynulostí zobrazování. Tento virtuální model obsahoval pouze podjevištní část barokního divadla.

Tato semestrální práce mě natolik zaujala, že jsem se jí dále zabýval, až přerostla v práci diplomovou. K již vytvořenému modelu přibyly jevištní prostory a celá práce se rozšířila i o část programovou. Ta vytvořila potřebné zázemí pro zkoumání tohoto objektu.

Chyby, které vznikaly při modelování v programu Microstation v předešlých projektech, bylo nutné eliminovat. Většina prvků modelu byla znovu vymodelována a až poté převedena na výsledný formát prostorových dat - VRML. Výsledný model je tudíž nejen bez chyb, ale je i dostatečně rychlý na zobrazování při standardních požadavcích na běžný počítač (viz. Kapitola 5.1.1).

Práce byla rozdělena na dvě etapy:

1. **vizualizace podjevištní části a jeviště**
2. **vytvoření programu** pro zkoumání daného modelu divadla

1.2 Vizualizace jevištní techniky

Vizualizace splňuje nároky (viz. Kapitola 3.1) na snadné používání 3D modelu. Myslí se tím: používání známého a osvědčeného formátu dat (DGN, VRML), rychlost načítání a procházení modelu a možnost vizualizace na jakýchkoliv počítačích (tzv. přenositelnost). Téma rychlosti zobrazování virtuálního modelu je úzce spojeno s přesností daného modelu. Jde tedy vždy o určitý kompromis mezi rychlostí a přesností scény. Všechny tyto charakteristiky jsem vzal v úvahu od samého počátku projektu.

1.3 Zkoumání objektu

V divadle je stále velké množství technického zařízení, které je zajímavé pro zkoumání. Je nutné zkoumat tyto objekty z různých pohledů, hledat dané návaznosti a možné vztahy jednotlivých částí scén. V běžné praxi jde o studování stavebních výkresů, fotografií a skic, ke kterým nemá každý přístup. Navíc je nutné mít určitou prostorovou představivost. Případné restaurování a vytvoření kopií je nákladné.

Další možností je prostorový model prohlížet v některém sofistikovaném systému, např. již zmíněném Microstation či jiném CAD systému. To ale vyžaduje znalost práce s těmito systémy.

Nároky na zkoumání dané scény tedy musí splňovat určitou uživatelskou jednoduchost. Musí být zaručena volnost prohlížení modelu, procházení a jednoduchost zkoumání jeho jednotlivých prvků.

Jeden ze základních požadavků zadavatele bylo měření vzdáleností (viz. souhrn požadavků v Kapitole 3.1). Ve scéně mohou být měřeny vzdálenosti nejen dvou zadaných bodů, ale i vzdálenosti jejich nejbližších vrcholů vybraného prvku modelu.

Pro větší komfort práce s modelem, je nutné mít určité nástroje pro kategorizaci podle nejrůznějších kritérií. Podle nich je možné potřebné prvky modelu přehledně hledat a dále s nimi pracovat. S tím jsou i svázané informace o daném prvku modelu. Ty by měly být jednoduše zpracovatelné (prohlížení, editace, opravování a ukládání).

Vytvořený program by měl být jednoduchý na instalaci (pokud je nějaká instalace potřeba), měl by být přehledný a intuitivní, jednoduše ovladatelný a spustitelný na počítačích s odlišným operačním systémem (tzv. platformová nezávislost).

Chtěl bych upozornit, že moje práce zcela nenahrazuje původní možnosti zkoumání objektů, jen ji doplňuje o možnosti další.

1.4 Podklady

Podklady byly zapůjčeny od Ing. Jindřicha Hodače, Ph.D. z Katedry mapování a kartografie Fakulty stavební ČVUT. Jednalo se o samotná prostorová data ve formátu CAD (.DGN) a dodatečné informace o jednotlivých prvcích modelu (rozpis vrstev, fotodokumentace a textové údaje).

Kapitola 2

Aktuální stav oboru a využité zdroje

V této části uvádím použité formáty dat a programy pro jejich snadnou manipulaci. Dále uvádím některé funkce vytvořeného programu s vysvětlením jejich důležitosti. Jako první je určitě na místě uvést formát dat VRML, který je stěžejní pro vizualizaci prostorových dat. Zde jsou uvedeny výhody i nevýhody použití VRML pro samotnou práci. Dále jsem popsal výhody i nevýhody použití Javy3D a jeho experimentální nadstavby Xj3D.

2.1 VRML

2.1.1 Historie

Počátky VRML sahají k firmě Silicon Graphics, která jej navrhla jako nadstavbu pro OpenInventor (aplikační knihovna známé grafické knihovny OpenGL). Dala tak základ formátu, ve kterém se zapisují tělesa a virtuální scény.

VRML mělo v posledních letech velmi dynamický vývoj. Zde jsou uvedeny nejdůležitější historické milníky:

VRML 1.0 - vznik formátu v roce 1995 firmou Silicon Graphics. Důležité bylo rozšíření využití prostorových dat o prostředí WWW.

VAG (VRML Architecture Group) - nezávislá grafická skupina programátorů a návrhářů, která vznikla v roce 1995. Tato skupina stanovila tři základní cíle pro další vývoj jazyka VRML:

1. prostředky pro popis statických světů (VRML 1.0)
2. prostředky pro popis dynamických světů
3. prostředky pro spolupráci více uživatelů ve virtuálním prostředí

Moving Worlds - pracovní název vítězné specifikace od firem Silicon Graphics a Sony. Tato specifikace se stala základem jazyka VRML 2.0.

VRML 97 - oficiální název standardu ISO s označením ISO/IEC 14772-1:1997, který byl přijat v roce 1997.

Pod názvem VRML tedy můžeme rozumět různé zkratky. V dalším textu budu pod názvem VRML mínit oficiální název standardu VRML97. Pokud je někdy v dokumentacích uvedeno VRML2.0, jedná se pouze o programátorské označení VRML97.

2.1.2 Důsledek vytvoření jazyka VRML

Na běžných počítačích znamenal vznik VRML zásadní zlom v prezentaci prostorových dat. Již se nemusely používat speciální a drahé CAD systémy či jiné náročné programy. Jazyk VRML se otevřel pro práci s prostorovými daty jakémukoliv uživateli, který přistupoval na internet.

Předpoklad pro uplatnění VRML je velký. Snadná dostupnost dat spolu s jednoduchou editací a dostupností VRML prohlížečů znamenala rozvoj VRML v nejrůznějších oblastech. Mezi ty nejdůležitější patří architektura, informační systémy, elektronické obchodování a reklama. Další předností je i to, že VRML není vázané na určitou aplikaci, ale jak dokládá i tato práce, může být používána jakýmkoliv programem.

2.1.3 Editace a prohlížení

Editace textových souborů VRML s popisem daných objektů a scén je velmi jednoduchá v každém textovém editoru. Je zde nutné zdůraznit přenositelnost tohoto formátu dat v rámci různých platforem.

Prohlížeče jsou schopné převést textový popis ze souboru VRML do obrazu virtuálního světa. Umožňují pohyb v tomto světě a dovolují případnou interakci s virtuálními prvky modelu. Nejznámější z nich jsou: CosmoPlayer (od firmy Silicon Graphics, Inc.) a WorldView (od firmy Intervista Software, Inc.).

2.1.4 Popis virtuálních objektů a důležitých metod

V této části uvádím důležité popisy virtuálních objektů spolu s metodami, na které v rámci diplomové práce dále upozorňuji:

- **Appearance** - vzhled povrchu, který přiřadí danému geometrickému objektu barevné vlastnosti a textury.
- **Group** - sdružení více uzlů do jednoho stromu.
- **IndexedFaceSet** - jde o množinu ploch, která je definovaná geometrií a vzhledem dané plochy. Parametry jsou:
  - **coord** - seznam vrcholů
  - **coordIndex** - posloupnost indexů vrcholů jednotlivých ploch (zakončeno číslem -1)
  - **solid** - všechny plochy jsou jednostranné (TRUE) či oboustranné (FALSE)
- **Inline** - vložení dalšího světa nebo objektu z vnějšího souboru do aktuálního VRML.
- **LOD (Level Of Detail)** - stupeň detailu, který je určen variací reprezentací jednoho objektu v různě určených přesnostech. Používá se v závislosti na vzdálenosti pozorovatele.
- **Transform** - společné nastavení polohy, měřítka a otočení potomků dané skupiny.
- **Viewpoint** - stanoviště ve virtuálním světě

Dále je třeba vyzdvihnout příkazy DEF a USE, které jsou důležité v rámci celé diplomové práce:

- **DEF** - pojmenování uzlu
- **USE** - vytvoření kopie

Zde je ukázka části vytvořené VRML scény pro náznak správného zápisu VRML souboru:

```
#VRML V2.0 utf8
...
DEF ID010503 Transform {
   systému
   translation -92.7853 100.137 104.2
   children [
      Transform {
         translation -3.99661 -.089721 .969963
         children [
            Shape {
               appearance DEF a Appearance {
                  material Material {
                     diffuseColor .223529 .105882 .031372
                  }
               }
               geometry IndexedFaceSet {
                  coord Coordinate {
                     point [
                        -.417 -.011 -.924, .417 .012 .924, .399 .011 -.935,
                        -.399 -.012 .935,
                     ]
                  }
                  coordIndex [
                     0 1 2 -1, 1 0 3 -1,
                  ]
                  solid FALSE
               }
            }
         ]
      }
...
}
```

Podrobné a přehledné zpracování o celém jazyce VRML obsahuje [12].

2.1.5 Optimalizace

Program Chisel (viz. Obrázek 2.1) jsem použil pro optimalizaci modelu barokního divadla. Chtěl bych podotknout, že tento program nelze použít pro X3D formát dat. Zde jsou uvedeny nejdůležitější použité funkce pro optimalizaci (více o tomto programu na [4]):

Data reduction - ukazuje procentuálně, o kolik se zmenšila velikost souboru.

Polygon count - počet polygonů ve VRML souboru

VALIDATE (Správnost) - kontrola správnosti scény. Spočítá a označí chyby (angl .error) a upozornění (angl. warnings) ve VRML souboru.

FORMAT - upraví VRML soubor do přehledné formy (odsazení řádků apod.).

CLEAN - odstraní ze souboru nepotřebné hodnoty a uzly

- Remove default fields - odstraní defaultní hodnoty VRML jazyka.
- Remove repeated value refs - odstraní opakující se hodnoty.
- Remove unused values - odstraní nepoužité hodnoty.
- Remove useless nodes - odstraní zbytečné uzly virtuálního modelu.

CONDENSE - zmenší velikost souboru

- Adjust numeric resolution - opraví číselné hodnoty u jednotlivých uzlů na zadanou přesnost.
- Create DEF/USE - zredukuje počet pomocných příkazů DEF a USE.
- Remove unused DEFs - odstraní nepoužívané příkazy DEF (nemusí platit pro viewpointy).
- Shorten DEF names - zmenší na minimální velikost označení (název) DEF příkazu.

REDUCE - urychlí procházení scény.

- Remove smallest edges - odstraní nejmenší hrany (např. procentuálně z velikosti celkové scény).
- Remove smallest triangles - stejně jako předešlý, ale pracuje se s trojúhelníky.

REORGANIZE - přebuduje VRML soubor.

- Uninline files - přidá do aktuálního souboru všechny INLINE uzly.
- Un-PROTO - soubor zbaví PROTO uzlů.
- Turn top levels DEF to Inlines - z uzlů označených DEF příkazem, které jsou nejvýše ve stromové struktuře VRML souboru, vytvoří soubory a pomocí uzlu INLINE připojí na jejich místo.

2.1.6 Nevýhody

Mezi jistá omezení, které se sebou specifikace VRML přináší, a která jsou důležitá pro tuto práci, patří:

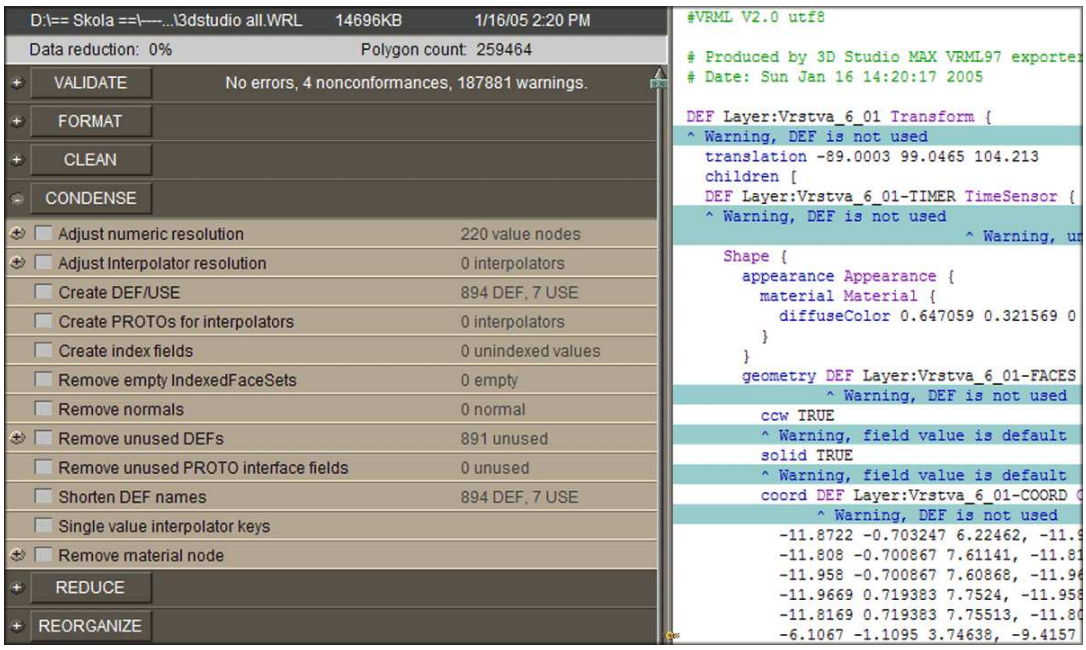
- Neposkytování možnosti používání informací o objektech (např. Meta-data v X3D).
- Jazyk VRML byl vytvořen pouze pro prohlížení, ne na editaci.

Tato omezení byla důležitá pro rozhodnutí nepostupovat standardními prostředky při vývoji programu pro zkoumání VRML modelu (viz. Kapitola 3.3.1), ale použít jiné nástroje.

2.2 3D Studio VIZ

Velkou část práce jsem věnoval zkoumáním účinků vymodelovaných těles z programu 3D Studio VIZ na exportovaná prostorová data ve formátu VRML. Nejdůležitější pravidla při modelování jsou uvedeny zde:

- Při používání instancí těles popsaných pomocí IndexedFaceSet se automaticky vytváří v exportovaném souboru VRML příkazy DEF a USE.



Obrázek 2.1: Ukázka programu Chisel

- Při používání instancí základních těles je nutné použít, po exportu do VRML souboru, optimalizační program Chisel (3D Studio VIZ nepoužije automaticky příkazy DEF a USE).
- Vytvořené kamery jsou ve VRML souboru Viewpointy, které přesně odpovídají umístění a natočení
  - Nepoužívat v názvech kamer českou diakritiku.
  - Mezery se v názvu kamer nahradí podtržítkem.
- Při vytváření základních těles musí být v parametrech tohoto tělesa povolena defaultní hodnota "Smooth" (Vyhlazení). Jestliže není, exportované základní těleso je ve VRML souboru popsáno uzlem IndexedFaceSet.
- I když se použijí stejné barvy či materiály v 3DStudiu, exporter na ně nedokáže automaticky použít příkazy DEF a USE. Optimalizační program Chisel tuto chybu jednoduše napraví (viz. Kapitola 2.1.5).
- Používají-li se při animaci v 3D Studiu VIZ změny barvy, exporter nedokáže jakkoliv tyto změny uložit do VRML souboru.
- Pokud jsou některé prvky modelu skryty, nejsou exportovány do výsledného VRML souboru.

Podrobnější popis tohoto problému je uveden v [15].

2.3 Java3D

2.3.1 Vysoký výkon, interaktivní 3D grafika

Java3D přináší důležité prvky, které ostatní programovací jazyky nemají:

- platformová nezávislost díky Javě

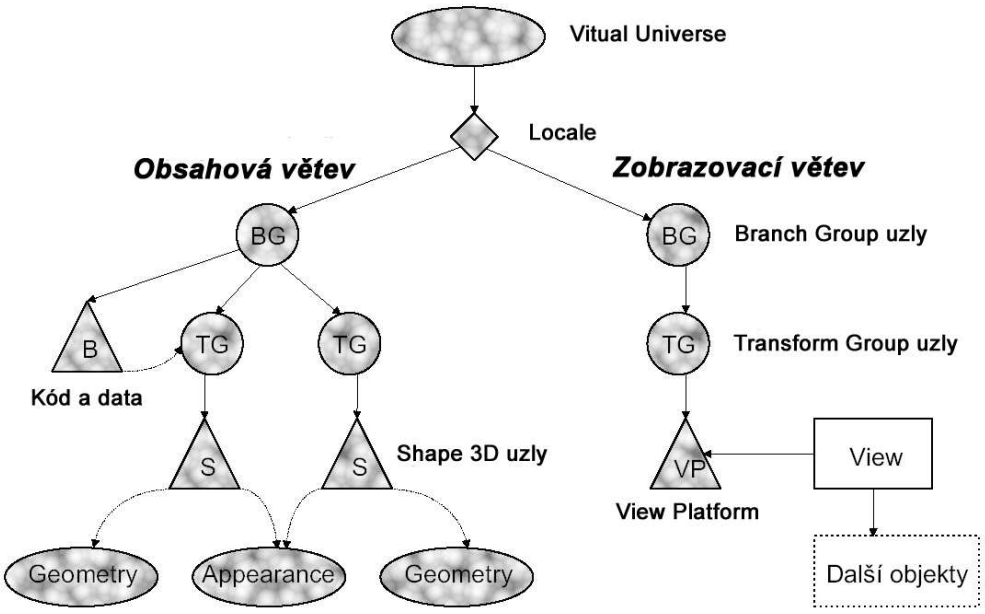
- vysoká úroveň programování
- dostupnost z internetu (applet nebo aplikace přes Java Web Start)
- možnost jednoduchého načtení VRML souboru z lokálního místa v počítači i z URL adresy na internetu

Java 3D API poskytuje interface pro jednoduchý a přitom vysoce účinný programovací model. Vytváření scén, rendrování a kontrola nad scénou jsou na vysoké úrovni. Spolu s dokonalým naplněním javovského paradigmatu : "Write once, run anywhere" (jednou napiš, spusť kdekoliv), se jedná o velmi silný nástroj.

Vše je propojeno s nízkoúrovňovým API jako je OpenGL nebo Direct3D. Java 3D je vlastně jakousi nadstavbou tohoto rozhraní pro rychlé hardwarové akcelerátory.

2.3.2 Graf scény

Java 3D API s sebou přináší jednoduché programování při řešení nejrůznějších časově náročných problémů. Mezi ně patří i traverzování grafu scény (viz. níže) a nastavování příznaků pro geometrické objekty, pro jejich možnou optimalizaci při vykreslování scény. Protože implementace Javy3D API je rozvrstvena, aby dodávala vhodný rendrovací API, může přinášet nativní grafickou akceleraci na OpenGL a Direct3D platformách. Pro využití nadstandardů zobrazování je zde možné použít prostředky pro vizualizaci a manipulaci s velkými 3D objekty. Graf scény (angl. SceneGraph) na Obrázku 2.2 je základním kamenem pro tuto práci.



Obrázek 2.2: Java3D - SceneGraph

Jde vlastně o stromovou datovou strukturu postavenou na objektové filozofii Javy. Tato datová struktura reprezentuje danou scénu. Skládá se z těchto částí:

- **Virtual Universe** - základ každé scény, který odkazuje na všechny ostatní objekty, které se ve scéně objeví. Jde vlastně o kořen stromové struktury celé scény.
- **Locale** - počátek souřadné soustavy scény.

- **Obsahová větev (angl. Content branch)** - definuje obsah scény. Zde patří geometrie, vlastnosti a umístění objektů, vlastnosti světelných zdrojů, intenzita zvuků apod.
- **Zobrazovací větev (angl. Viewing branch)** - definuje i několik výstupních zařízení, specifikuje parametry zobrazení (pozici a vlastnost kamery).
- **Branch Group uzly** - kořeny jednotlivých částí scény, které mají jen jednoho rodiče a neomezený počet potomků.
- **Transform Group uzly** - jde o transformační uzly (posun, rotace a změna měřítka), které mají jen jednoho rodiče a neomezený počet potomků.
- **Kód a Data** - uživatelem napsaný kód a jeho data pro ovlivňování scény.
- **Shape 3D uzly** - uzel který odkazuje na objekt Geometry a objekt Appearance. Může odkazovat i na více objektů Geometry, ale ty musí být stejného typu.
- **View Platform** - spravuje pozici, orientaci a změnu měřítka pozorovatele.
- **Appearance** - uchovává informace např. o materiálu povrchu, průhlednost apod.
- **Geometry** - objekt s přesně definovanou geometrií.

Graf scény lze rychle modifikovat a zobrazovat (díky tzv. behavior). Vyjmutí příslušného objektu (či více objektů) ze scény je možné provést odebráním příslušné větve z této stromové struktury.

Graf scény zde ovšem pouze určuje co má být zobrazeno. Jeho architektura umožňuje zapojení mnoha vláken (procesů), které mohou současně traverzovat stromovou strukturu kvůli různým cílům. Dále je možné u některých objektů (např. Appearance, Geometry apod.) jejich několikanásobné použití. Když jsou tedy použity ve VRML scéně příkazy DEF a USE, v grafu scény je rovněž použit odkaz na daný objekt. to výrazně šetří nároky na paměť a usnadňuje manipulaci s objekty grafu scény.

V Javě3D se používají tyto konstrukty k vytvoření scény:

Node - je abstraktní třídou každého prvku, který tvoří graf scény. Její potomci jsou *Group* a *Leaf*

Group - je abstraktní třídou prvků, které umožňují větvení grafu scény (BranchGroup, OrderedGroup, SharedGroup, Switch, TransformGroup)

Leaf - je abstraktní třídou všech listů v grafu scény. Těmito listy jsou například:

Shape3D - specifikuje tvar objektu

Sound - zvuk

Light - světelný zdroj

Behavior - uzel pro rychlé modifikace v grafu scény

Fog - mlha

NodeComponent - tato třída není součástí grafu, jelikož by porušila definici stromové struktury. Jejím účelem je několikanásobné využití svých potomků objektům Shape3D. Mezi tyto podtřídy patří například:

Geometry - geometrie

Appearance - vzhled apod.

Texture - textura

Material - materiál

Mnohem podrobnější popis podává článek [15], který je určen pro začínající programátory Javy3D.

2.3.3 Nevýhody

Java3D API je rozhraní pro práci ve 3D grafice. Má bohužel spoustu slabých míst, kvůli kterým se stává pro programátory nevyhovující:

- Jde o nadstavbu nad OpenGL či Direct3D a oproti používání např. knihoven v C++ je pomalejší. Vše je způsobeno JVM, který je již platformově závislý a který spouští tzv. bytekód.
- Samotná práce s VRML objekty není optimální (např. vyhledávání v grafu scény či nastavování nejrůznějších parametrů prvků modelu).
- Standardní program pro načítání VRML souboru do grafu scény (VRMLLoader) nepodporuje X3D formát dat.

Tyto nevýhody zatlačují Javu3D do pozadí i přes její nepochybné kvality. Další chyby, zjištěné v rámci implementace programu, jsou uvedeny podrobněji v Kapitole 5.4.

2.4 Xj3D

2.4.1 Popis

Jedná se o experimentální projekt společnosti Web3D [9], který vznikl jako nadstavba Javy3D pro VRML/X3D. Tato nadstavba může být přidána do libovolné aplikace a použita pro načítání již zmíněných souborů. Nejen, že je zde určitá nástrojová sada pro řešení VRML/X3D scén, ale je zde vytvořen i základní Xj3D Browser pro procházení VRML scény. Tyto vlastnosti jsem využil pro vlastní návrh.

Xj3D browser nemusí pracovat jen přes Javu3D, ale existuje i OpenGL Browser vytvořený právě díky rychlejšímu renderingu. Vzhledem k tomu, že musím dále s modelem pracovat a vytvořit měření vzdáleností, rozhodl jsem se pouze pro rendering - Java3D. Existují zde totiž jednoduché nástroje pro zjišťování souřadnice zadávaného bodu na ploše libovolného tělesa.

2.4.2 VRML/X3D

Xj3D velmi radikálně mění přístup ke kontrole či manipulaci objektů ve scéně. Již se na tyto objekty nenahlíží jako na nadstavbu OpenGL, ale jako na VRML scénu složenou s VRML uzlů, metod a příkazů. To značně ulehčuje programátorskou práci při vyhledávání prvků modelu. Díky Xj3D loaderu pro načtení VRML/X3D souboru (Web3DLoaderu) je zde pořád možnost použití grafu scény. Na tomto místě uvádím část kódu na vyhledávání Viewpointů:

```
Seznam_Viewpoint =
  načtená_scéna.getByPrimaryType(TypeConstants.ViewpointNodeType);

if (neexistuje žádný Viewpoint v Seznam_Viewpoint) {
```

```
  Zakaž Viewpoint Toolbar;
  Ukonči tuto funkci;
}
else {
  Povol Viewpoint Toolbar;
}

while (existuje ještě další Viewpoint v Seznam_Viewpoint) {
  node = aktuální Viewpoint;
  if (node.getPrimaryType() == TypeConstants.ProtoInstance) {
    node = jedná se o Viewpoint jako Proto instance;
    zjisti její implementaci;
  }
  desc = zjisti název Viewpointu;
  if ( (desc == null) || (desc.length() == 0)) {
    desc = "Viewpoint " + count;
  }
    Umísti Viewpoint na Viewpoint panel v prohlížeči;
  }
}
```

Záměrně jsem uvedl metodu *getByPrimaryType(int type)* přesně tak, jak se používá. Tato metoda vrací velmi rychle pole nalezených objektů, podle typu zadání. Toto názorně ukazuje, jak probíhá práce s celým Xj3D. V Javě3D se musí procházet postupně graf scény a vybírat patřičné typy uzlů.

2.4.3 Funkce pro vizualizaci měřené vzdálenosti

V této kapitole bych chtěl popsat funkci, která slouží pro vizualizaci měřené vzdálenosti. Základním tělesem pro tuto vizualizaci je válec (angl. cylinder). Zadáme-li bod A a B jako parametry, vrací nám tato funkce hodnotu *Transform3D*. Tato hodnota nastaví transformaci pro danou vizualizaci vzdáleností v podobě již zmíněného válce. Pomocné funkce jsou *calcAngle* - která počítá úhly mezi vektory a *calcSign* - která počítá znaménko výsledné rotace.

```
  private Transform3D connectCylinder(Point3d A, Point3d B) {

    // vypočítání vzdálenosti dvou bodů
    vzdálenost = A.distance(B)

    // Určení souřadnice bodu pro umístění válce
    xCoor = (A.x + B.x) / 2
    yCoor = (A.y + B.y) / 2
    zCoor = (A.z + B.z) / 2

    vektorVzdálenosti = new Vector3D(B.x-A.x,
                                     B.y-A.y,
                                     B.z-A.z)

    proj = new Vector3D(B.x-A.x,0,B.z-A.z)

    xrot = calcAngl(new Vector3D(0,1,0),vektorVzdálenosti)
```

```
    // Vypočítání úhlu pro y-ovou souřadnici
    yrot = calcAngl(new Vector3D(0,0,1),proj)

    // určení znaménka úhlu
    yrot = yrot * calcSign(new Vector3D(1,0,0),proj)

    // Inicializace 3D Transformace
    rot = new Transform3D()

    // Otáčení (rotace) kolem x-ové osy o úhel xrot
    rot.rotX(xrot)

    // Inicializace 3D Transformace
    transformace = new Transform3D()

    // Otáčení (rotace) kolem y-ové osy o úhel yrot
    transformace.rotY(yrot)

    // Násobení matic - transformace = transformace * rot
    transformace.mul(rot)

    // Posun (translace) do bodu [xCoor, yCoor, zCoor]
    transformace.setTranslation(xCoor,yCoor,zCoor)

    // Změna měřítka (Scale) v ose x o 0.1 a v ose y o vzdalenost
    transformace.setScale(0.1, vzdalenost, 0)

    return transformace
  }
```

Třída *Transform3D*, s kterou zde pracuji, je prakticky matice 4x4 reprezentující lineární transformaci bodu. Při inicializaci proměnné new Transform3D() je tato matice nastavena na:

$$Inicializace = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Metody třídy Transform3D pracují s touto maticí. Změně měřítka (Scale) v prostoru odpovídá transformační matici (resp. matici metody setScale uvedená ve funkci connectCylinder):

$$A_{Scale} = \begin{pmatrix} x & 1 & 1 & 0 \\ 1 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, A_{setScale} = \begin{pmatrix} 0.1 & 1 & 1 & 0 \\ 1 & vzdalenost & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Jestliže chceme nastavit posunutí (translaci), jsou nastaveny prvky bodu [x,y,z] v transformační matici následovně:

$$A_{Translation} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & z & 1 \end{pmatrix}$$

Stejně jednoduše probíhá otáčení kolem osy x (resp. y) o patřičný úhel α:

$$A_{RotationX} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & sin\alpha & 0 \\ 0 & -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, A_{RotationY} = \begin{pmatrix} cos\alpha & 0 & sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -sin\alpha & 0 & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Při skládání transformací se matice, reprezentující jednotlivé transformace, násobí zprava. Proto se v uvedené funkci nejdříve nastaví matice pro rotaci kolem x-ové osy, potom se vynásobí maticí reprezentující rotaci kolem y-ové osy. Dále se provede posun do určeného bodu a nakonec se změní měřítko (Scale). Podrobnější popis nejen o trojrozměrných geometrických transformacích je uveden v [20].

2.4.4 Nastavení bitů Capability

Funkce setCapabilities slouží pro nastavení bitů, které jsou posléze využity pro měření vzdáleností (viz. Kapitola 4.3.4). Tyto bity je nutné nastavit, pokud chceme určovat souřadnice bodů geometrických objektů (tzv. Shape3D), které se již nachází v aktivní části grafu scény a staly se tedy živými uzly (live nodes).

```
private void setCapabilities(Scene scene) {
  // inicializace seznamu všech Shape3D těles v grafu scény
  shape3Ds = new Vector();

  // pomocná funkce pro procházení grafu scény a naplnění
  // shape3Ds všemi tělesy Shape3D
  retrieveShape3Ds(scene.getSceneGroup());

  // Zde je příprava Appearance pro každý geometrický objekt
  scény
  RenderingAttributes ra = new RenderingAttributes();
  Appearance noVisible = new Appearance();
      // odstraní se materiály
      noVisible.setMaterial(null);
      noVisible.setRenderingAttributes(ra);
      // nastaví se tyto objekty na neviditelné
      noVisible.getRenderingAttributes().setVisible(false);

  while (seznam shape3Ds není prázdný) {

    // nastav pomocí třídy PickTool a její metody
    setCapabilities bity
    // na pozdější možnost určování hledané
    souřadnice zadaného bodu pro daný Shape3D
    PickTool.setCapabilities( Shape3D, PickTool.INTERSECT_FULL);
    // tento geometrický objekt zneviditelni
    shape3D.setAppearance(noVisible);

    Enumeration en = shape.getAllGeometries();
    while (existují nějaké geometrické objekty Shape3D) {
      // zjisti jejich geometrii
      // a podle typu geometrie každému z nich nastav příslušné
      // bity Capability
      if (jestliže se jedná o TriangleArray) {
```

```
          TriangleArray TA = (TriangleArray) geometry;
          TA.setCapability(GeometryArray.ALLOW_FORMAT_READ);
          TA.setCapability(GeometryArray.ALLOW_COORDINATE_READ);
          TA.setCapability(GeometryArray.ALLOW_COUNT_READ);
          TA.setCapability(GeometryArray.ALLOW_INTERSECT);
          TA.setCapability(GeometryArray.ALLOW_FORMAT_READ);
          TA.setCapability(GeometryArray.ALLOW_NORMAL_READ);
        }
        else if (jestliže se jedná o TriangleStripArray) {
          // nastaví bity Capability stejně jako u TriangleArray
          ...
        }
        else if (jestliže se jedná o QuadArray) {
          // nastaví bity Capability stejně jako u TriangleArray
          ...
        }
      }
    }
  }
```

Jak v této funkci můžeme vidět, prochází se všechny polygony celého grafu scény a každému z nich se změní Capabilities bity. Důležité je zde použití *RenderingAttributes*. Díky této třídě se u objektů Shape3D nastaví jejich neviditelnost a nejsou započítávány do zobrazování, ale jen do případných kolizí. Jako jedna z kolizí je interakce s myší a to je námi hledaná souřadnice bodu.

Kapitola 3

Návrh

V úvodu této kapitoly souhrnně uvádím základní požadavky na diplomovou práci. Stejně jako zadání diplomové práce, jsem návrh rozdělil na dvě části - *prostorový model ve formátu VRML* a *samotný program Browser*. Všechny části návrhu probírám odděleně, ukazuji jejich výhody i nevýhody v rámci určité části návrhu (před samotnou implementací). Na závěr shrnuji výsledek návrhu.

3.1 Základní požadavky na projekt

Zde jsou shrnuty základní požadavky na návrh celého projektu, které jsou v dalších podkapitolách upřesňovány (jak v historii návrhu, tak v konečném návrhu projektu):

- požadavky na prostorová data:
  - popis objektů pomocí VRML/X3D
  - rychlost načítání scény
  - rychlost průchodu scénou
- požadavky na program:
  - browser pro prohlížení scény. Měl by zvládat základní pohyby ve virtuálním světě (chůze, let, prohlížení, posunutí), dále musí dovolovat případnou iteraci s objekty.
  - měření vzdáleností dvou bodů. Zde je nutné podotknout, že se nemají měřit jen vzdálenosti zadaných bodů, ale i vrcholy na zadaných plochách.
  - vizualizace propojení měření vzdáleností
  - možnost provázání dat s VRML scénou
  - snadná a přehledná manipulace s objekty
  - možnost použití i jiných VRML scén
  - platformová nezávislost

3.2 Návrh řešení prostorových dat

3.2.1 VRML nebo X3D

Od počátku práce na tomto projektu bylo samozřejmostí využít soubor typu VRML. Až při dokončování diplomové práce jsem si položil otázku : "Proč nepoužívat X3D ?" Díky konzultacím v předmětu 36NUR jsem si sám zamítavě odpověděl. Důvody, proč jsem se zaměřil na formát VRML jsou tyto:

- kvalitní prohlížeče a editory VRML souborů
- 3D Studio VIZ nemá export do X3D a tudíž by byl jeden krok převodu dat na X3D, navíc.

Samotná prostorová data barokního divadla jsem tedy zpracovával ve VRML. Později, jak uvádím níže, jsem se rozhodl pro program vytvořený pomocí projektu Xj3D. Díky jeho loaderu (přesněji Web3DLoaderu), jsem dosáhl možnosti pracovat i s daty X3D a X3DV.

3.2.2 Rychlost načítání scény

Rychlost samotného načtení scény je závislá na velikosti přenášeného souboru. Dlouhou dobu jsem se zabýval tím, jakým způsobem redukovat velký objem dat na co nejmenší. Export souboru DGN do VRML programem Microstation není optimální (viz. Tabulka 5.1). Rozhodl jsem se tedy pro import do 3DStudia VIZ a zde některé prvky přemodelovat na základní tělesa. Tento známý program pro 3D modelování podporuje vrstvy, které jsou ve formátu DGN rovněž přenášené spolu s prostorovými daty. Další výhodou 3DStudia VIZ a jeho exportu do VRML je, že pokud se vytvářejí instance částí scény, je možné v optimalizaci výsledného souboru použít pomocné příkazy DEF a USE. Tím podstatně klesne velikost celého souboru.

Optimalizační program Chisel je velmi vhodným nástrojem pro zmenšení velikosti souboru VRML. Jeho přesnější popis je v Kapitole 2.1.5.

3.2.3 Rychlost průchodu scénou

Zde jsem se zaměřil na počet zobrazovaných polygonů ve virtualizaci samotného modelu. Pomocný optimalizační program Chisel má všechny potřebné nástroje na tuto optimalizaci (viz. Kapitola 2.1.5). Přemodelované geometrické objekty typu IndexedFaceSet na základní tělesa mohou urychlit zobrazování scény, díky méně náročné triangulaci. Z tohoto důvodu je do návrhu zahrnut jak program 3DStudio VIZ, tak optimalizační program Chisel.

3.3 Návrh řešení programu

3.3.1 Prohlížeč nebo-li Browser

Vývoj samotného programu prohlížeče má dlouhou historii. Můj původní návrh byl: IE5+, plug-in Cortona a javovský applet, který by komunikoval s VRML scénou pomocí EAI rozhraní. Jeho **výhody** jsou:

- rychlost procházení scény (OpenGL nebo DirectX)
- stabilita programu Cortona
- dlouhodobé používání v nejrůznějších odvětvích
- nenáročná instalace - pro spuštění je nutné nainstalovat pouze program Cortona (jako plug-in do prohlížeče IE5+) a patřičný Java applet

Po mnoha konzultacích a zkouškách implementace jsem dospěl k názoru, že takto nelze vytvořit prohlížeč, který potřebuji. Od původního návrhu jsem tedy upustil. Jeho **nevýhody**, proč jsem tak udělal, jsou uvedeny zde:

- Rozhraní EAI je funkční pouze s MSJVM, která již není od ledna roku 2001 firmou Microsoft dále vyvíjena (viz. [11]) a ani dále podporována.

- Tato varianta je nejen platformově závislá na operačním systému Windows, ale je závislá i na prohlížeči IE5+, což je v rozporu se základními požadavky (viz. Kapitola 3.1).
- Vzhledem k budoucím možnostem editace informačního XML souboru a jeho pozdější ukládání, je volba appletu nevyhovující. Applet nepodporuje ukládání souboru na disk díky jeho zvýšené bezpečnosti.
- Správa divadla (další směr vývoje projektu) by měla podporovat propojení s již fungující databází na zámku v Českém Krumlově.

Na jiný nápad návrhu implementace prohlížeče prostorových dat mě přivedl s předmětem 36NUR cvičící Ing. Martin Klíma. Základním požadavkem byla platformová nezávislost. Té lze dosáhnout jedině programovacím jazykem Java. Jestliže budu využívat experimentální projekt Xj3D (toolkit a Xj3D Browser jako nadstavba na Javu3D), jsou **výhody** této volby tyto:

- žádaná platformová nezávislost
- vysoká úroveň programování
- Xj3D Browser je již vytvořen - jednalo by se pouze o úpravu
- Xj3D Browser obsahuje všechny možnosti pohybu (podobně jako Cortona) se snadným zobrazením existujících Viewpointů
- vytvoření aplikace, kterou by bylo možné spustit přes Java Web Start
- neustále se vyvíjející projekty Xj3D, Java3D a i samotné Javy.

Jedinou **nevýhodu**, kterou jsem znal při návrhu prohlížeče, je pomalejší zobrazovací schopnosti Javy3D. To je zapříčiněné právě platformovou nezávislostí (spouštění bytekódu v JVM).

Díky výhodám uvedených výše, jsem se rozhodl prohlížeč (angl. Browser) implementovat jako aplikaci Xj3D prohlížeče s použitím nástrojů Xj3D.

3.3.2 Vzdálenost dvou bodů

Původní návrh IE5+, plug-in Cortona a applet jsem po několika týdnech implementace zavrhnul i v této části návrhu. Možnosti, jak by se dalo tento problém řešit tímto návrhem jsou teoreticky dvě:

1. Přes speciální knihovny programu Cortona - je závislé na jediném programu a je špatně dokumentovaná.
2. Vytvoření vlastní funkce v appletu, která by zjišťovala průchod paprsku scénou - neefektivní programování již fungujících metod v Javě3D.

Tato část návrhu mě jednoznačně utvrdila v použití Xj3D projektu.

3.3.3 Vizualizace vzdálenosti dvou bodů

Zde uvádím již návrh v prostředí Java3D a Xj3D. Vizualizace zadaného bodu a nejbližšího vrcholu je řešen pomocí základního tělesa - koule. Propojení dvou zadaných bodů či vrcholů jsem navrhl pomocí 3D základního tělesa - válce. Návrh počítá s možností jednoduchého odstranění či naopak zviditelnění této vizualizace.

3.3.4 Provázání dat s VRML scénou

Každá vizualizace s sebou přináší i možnost prezentace. Aby tato prezentace obsahovala velké množství informací, je nutné propojit vizualizaci s informační částí. Zde se jedná o popis jednotlivých prvků modelu (např. k čemu slouží rumpál, jak fungují posuvné kulisy apod.). Rozhodoval jsem se mezi dvěma návrhy: již kompaktní scénou ve formátu X3D (informační část by byla ukryta v META-DATA uzlech stromu scény) a mezi formátem VRML/X3D a XML informačním dokumentem. Pro VRML/X3D a XML jsem se rozhodl pro tyto výhody:

- XML informace mohou být odděleně upravovány.
- Snadné použití pro pozdější správu divadla (provázání s již fungující informační databází barokního divadla).
- VRML modely není nutné převádět na X3D a doplňovat informační data k jednotlivým prvkům modelu.

XML soubor obsahující informace by měl být do budoucna snadno editovatelný, přímo tímto programem. Proto jsem se rozhodl pro VRML/X3D s XML a jako parser XML dokumentu jsem zvolil DOM.

Samotné provázání VRML/X3D scény s XML dokumentem by mělo fungovat přes příkaz DEF ve VRML scéně s jedinečným identifikačního číslem (přesněji v Kapitole 3.3.5). Pojmenování pomocí příkazu DEF by mělo být možné v kterémkoliv uzlu modelu VRML (Group, TransformGroup nebo jakýkoliv geometrický uzel).

3.3.5 Snadné a přehledné zkoumání

Při tomto návrhu jsem vycházel z návrhu použité technologie DOM struktury XML dokumentu. Ve stromové struktuře by měli být zobrazeny všechny informace o objektech spolu s jejich hodnotami:

- **ID** - jedinečné identifikační číslo, které je provázané s VRML scénou (pojmenování příkazem DEF některý uzel VRML souboru).
- **Title** - název prvku modelu, který je zobrazen prvotně.
- **Layers** - tři vrstvy každého prvku modelu, které určují jeho polohu v divadle.
- **Description** - prezentovaná informace o daném prvku modelu.
- **SetVisible** - nastavuje viditelnost popsaného prvku v XML souboru
- **setEnable** - povoluje či zakazuje možnosti manipulace prvku modelu v prohlížeči

Přesný zápis XML dokumentu jsem uvedl i s příkladem v Dodatku B.

3.3.6 Jiné VRML scény

Návrh tohoto projektu počítá s použitím prohlížeče i s jinými VRML/X3D scénami. Musí zde být ale splněny základní podmínky (pokud nepočítám hardwarové a softwarové podmínky pro spuštění aplikace):

- Použitá prostorová data ve formátu VRML/X3D.
- Příkaz DEF použit pro jedinečnou identifikaci prvku scény a zároveň pro provázání s ID uzlem XML dokumentu.
- XML dokument s přesně stanovenou strukturou (viz. Dodatek B)

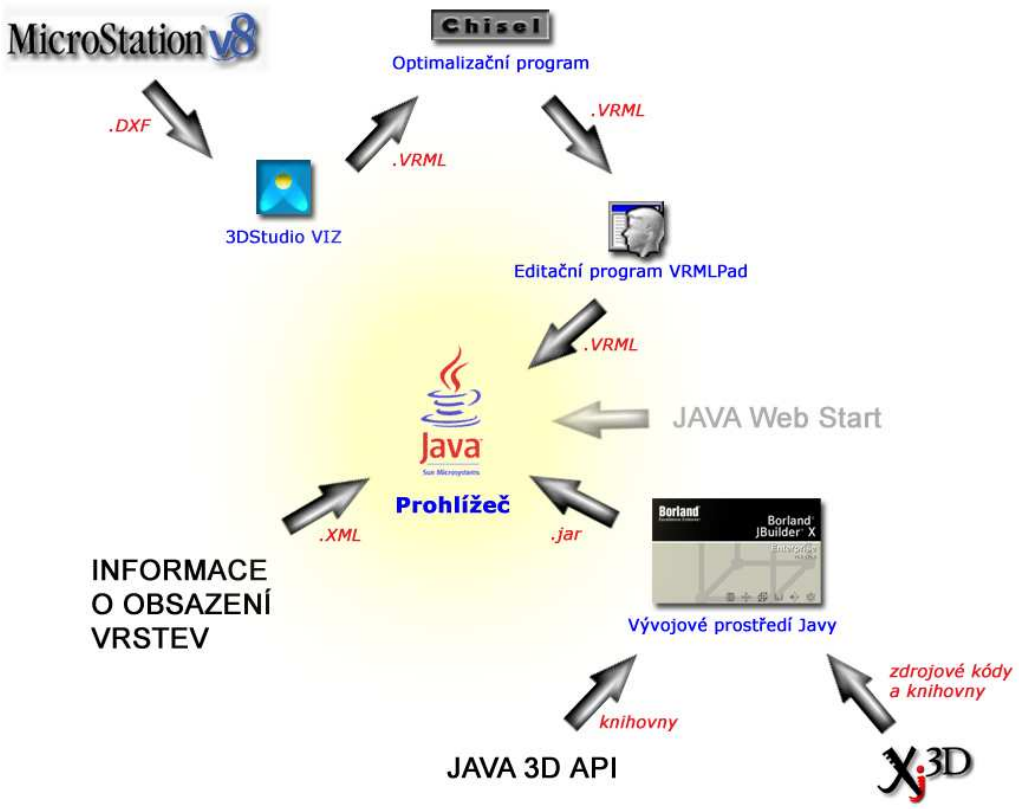
Jestliže jsou tyto podmínky splněny, je daný virtuální model spustitelný se všemi funkcemi, které jsou v návrhu uvedeny.

3.3.7 Platformová nezávislost

Tato podkapitola je uvedena jen pro úplnost. Experimentální projekt Xj3D je částečně platformově nezávislý. Jeho vývojové prostředí je sice platformově nezávislá Java, ale Java3D je již závislá (nadstavba pro OpenGL). V této době jsou již volně ke stažení příslušné Java3D JRE (více viz. [2]) pro většinu operačních systémů.

3.4 Souhrn návrhu

V návrhu počítám jak s prostorovými daty **VRML**, tak s **X3D**. Zvolil jsem programy **3D Studio VIZ**, **Chisel**, **VRMLPad** pro práci s VRML/X3D a vývojové prostředí **Borland JBuilder** (vytváření aplikace Java, Java3D a Xj3D). 3D Studio má za sebou již několik let úspěšného působení na poli modelování grafiky či její vizualizace a je velmi nepravděpodobné, že by tento program zanikl. Program Chisel lze používat po dobu třiceti dnů zdarma. VRMLPad je možné používat zdarma do určité velikosti ukládaných souborů. Na Obrázku 3.1 vidíte návrh procesu vizualizace barokního divadla a vytváření samotné aplikace Browser.



Obrázek 3.1: Vývoj projektu (červeně jsou označené typy přenášených dat) s výsledným prohlížečem, který může být použit s JAVA Web Start

Kapitola 4

Realizace

V této kapitole bych rád popsal postup realizace celého projektu. Byla zde dána váha na optimalizaci rychlosti načítání celé scény, rychlosti zobrazování a platformové nezávislosti (viz. Kapitola 3.1). Samostatné funkce jsou dostatečně popsány v JavaDoc dokumentaci na přiloženém CD-ROMu (viz. Dodatek D). Proto se zde zmíním jen o základních vazbách tříd a jejich důležitých funkcí.

4.1 Microstation a 3D Studio VIZ

Prostorový model, s kterým jsem pracoval, se skládal z podjevištní části a samotného jeviště divadla (viz. Obrázek 6.1). Pro převod jsem použil standardní import .DGN v programu 3D Studio VIZ. Tento import je natolik robustní, že i přes velký počet chyb v zadaných datech, zvládl vše velmi dobře. Chyby, které se v převáděných datech objevovaly, byly způsobeny chybným modelováním naměřených bodů v programu Microstation[1]. V 3D Studiu VIZ byly viditelné nejen 3D plochy (tzv. regiony), ale i osamocené body či obyčejné úsečky (spojnice dvou naměřených bodů). Neimportoval jsem celý model jeviště a jevištní části do 3D Studia VIZ najednou, ale vždy jen část dat pro větší přehlednost a manipulaci s jednotlivými prvky modelu. Každá část prostorových dat, uložená zvlášť v souboru typu VRML, obsahovala tedy pouze jednu vrstvu z programu Microstation. Tento soubor jsem nazval podle ID prvku modelu. Bohužel, kvůli chybám v použitém Web3DLoaderu (viz. Kapitola 5.4), jsem musel výsledný model sloučit do jediného kompaktního souboru. To mi však umožnil rychle a bez problémů program Chisel (viz. Kapitola 2.1.5).

Program Microstation, který je používán na fakultě stavební pro vizualizaci zadávaných dat, má rovněž přímý export do formátu VRML. Tento export bohužel nefunguje správně právě pro množství chyb v datech, které jsem uvedl výše. Tyto chyby jsou pak i ve výsledném formátu VRML. Zobrazení tohoto modelu programem Cortona v prohlížeči IE5+ tedy není možné.

Prvky modelu jsou importovány do 3D Studia VIZ jako hraniční reprezentace 3D objektů pomocí ploch. Např. obyčejný kvádr není importován jako základní těleso, které by bylo úspornější pro zápis dat, ale jako seznam ploch (viz. Obrázek 4.1). Tyto plochy ohraničují na první pohled stejné těleso jako základní. Jako prvotní v této práci byla redukce takto extrémně zadaných těles.

[1]Dne 20.1.2005 Ing. Radim Balík obhájil diplomovou práci na téma: "Konverze vektorových prostorových dat do formátu VRML" na Fakultě stavební. V rámci této diplomové práce odstranil veškeré chyby z modelu barokního divadla.

Obrázek 4.1: Porovnání reprezentace dvou těles (nalevo programem Microstation, napravo základní těleso z 3D Studia)

4.2 Modelování v 3D Studiu VIZ

3D Studio VIZ jsem zvolil pro jeho snadnou práci s vrstvami. Při importu, soubor DGN v sobě uchovává informaci o vrstvách a neslučuje všechny do jediné (jak by tomu bylo např. u 3D Studiu MAX).

V programu 3D Studio VIZ jsem velkou část prvků modelu přemodeloval. Dbal jsem na přesnost v rozměrech daného tělesa, umístění ve scéně i natočení v jednotlivých souřadnicích. Konzultace na toto téma probíhala na Fakultě stavební s Ing. Jindřichem Hodačem, Ph.D., v předmětech 36MUS, 36VIZ a 36NUR na Katedře počítačů a přímo na zámku v Českém Krumlově.

Hranice kompromisu mezi přesností naměřených dat, velikostí výsledného souboru VRML a rychlostí zobrazování, byla velmi ostrá. Při konzultaci jsem byl několikrát upozorněn na maximálně deseticentimetrové rozdíly oproti reálnému měření. Díky programu 3D Studio VIZ jsem však měl vše pod kontrolou.

Jak již bylo uvedeno dříve, na Obrázku 4.1 je příklad jednoho z prvku scény při přímém převodu do formátu VRML v porovnání s jeho přemodelováním ve 3D Studiu. Porovnání výsledků uvádím v Tabulce 4.1.

Dále jsem velmi často používal instance geometrických objektů ve 3D Studiu, a tak si předpřipravit scénu. Instance objektů má za následek opakování jednotlivých prvků modelu či jejich částí. Díky použití příkazů DEF a USE, se ve formátu VRML výrazně zmenší celková velikost souboru. V Tabulce 4.2 uvádím na porovnání využívání těchto funkcí.

3D těleso	Počet trojúhelníků	Velikost popisu ve VRML
Zaměřený trám v podjevištní části	104	6.000 Bytů
3D kvádr	12	548 Bytů

Tabulka 4.1: Porovnání počtu trojúhelníků a velikosti VRML souboru těles z Obrázku 4.1

3D tělesa	Velikost VRML zápisu
10x kvádr bez použití DEF a USE	4.800 Bytů
10x kvádr s použitím DEF a USE	1.200 Bytů

Tabulka 4.2: Výhody použití příkazů DEF a USE

Komplexnější porovnání uvádím v Kapitole 2.1.5, kde porovnávanám celý virtuální model barokního divadla v jednotlivých fázích jeho vývoje.

Jednotlivé prvky modelu jsem dále pojmenovával podle identifikačního čísla zadávané jako IDxxyyzz, kde:

- ID - identifikační prefix
- xx - číselné označení jednotlivých podlaží ve scéně
- yy - číselné označení určitého logického prostoru na daném podlaží
- zz - číselné označení vrstvy, která byla přiřazena při samotném geografickém zaměřování

Tyto již pojmenované skupiny byly exportovány do souboru s názvem své skupiny. Jak jsem již uvedl výše, bylo nakonec nutné ve výsledném VRML souboru nepoužívat uzly Inline. Proto je výsledná VRML scéna v jediném souboru VRML.

4.3 3D Studiu VIZ - VRML

Export na formát VRML jsem prováděl přímo standardním exporterem v 3D Studiu VIZ. Vzniklé problémy bylo nutné ošetřit a zaznamenat. Většina problémů je přímo závislá na modelování v 3D Studiu VIZ a zvolení správného postupu modelování. Vše popisuji podrobně v Kapitole 2.2.

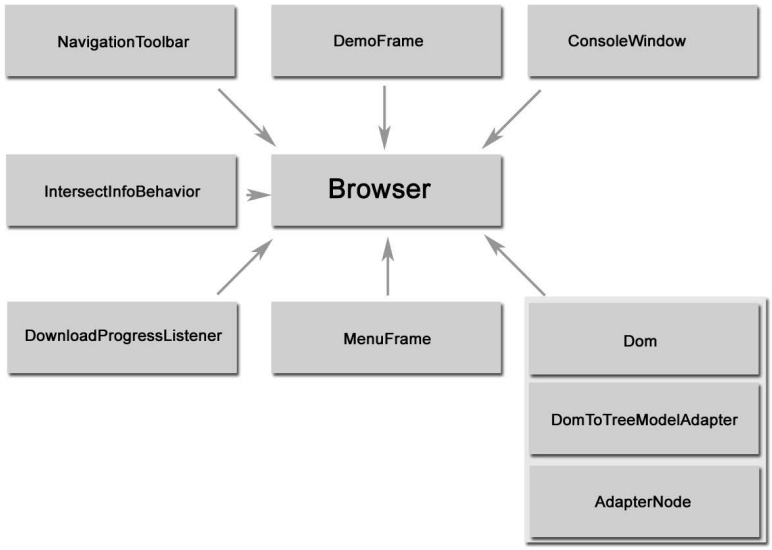
Program VRMLPad je velmi oblíben pro svoji snadnou a uživatelsky přehlednou obsluhu. Zde jsem jej používal jak pro celkovou kontrolu daného formátu dat (tzn. správnost zápisu i po optimalizaci), tak pro přepisování označených prvků modelu.

4.3.1 Struktura programu

Na Obrázku 4.2 je zobrazena struktura programu Browser.
Funkčnost zajišťují následující třídy:

NavigationToolbar.class - tato třída slouží pro práci s navigačním panelem v okně Browseru (viz. Dodatek C.3.1).

- Načítá a zobrazuje uživatelské rozhraní navigačního panelu (tzn. ikony tlačítek nebo názvy typů pohybu).



Obrázek 4.2: Struktura aplikace Browser

- Zjišťuje změnu typu pohybu a předává ji objektu Virtual Universe.
- Zajišťuje zvýraznění tlačítka aktuálního typu pohybu v navigačním panelu.

DemoFrame.class - zajišťuje základní uživatelské rozhraní okna Browser (viz. Dodatek C.3.1).

ConsoleWindow.class - tato třída zobrazuje konzoli pro výpis chyb a upozornění z programu. Je zde jen pro informaci o abnormálním chování programu.

IntersectInfoBehavior.class - třída pro měření vzdáleností. Je potomkem třídy *Behavior* a je posléze přidána do grafu scény (viz. Kapitola 2.3.2) jako další BranchGroup.

- Vizualizace bodů, nejbližších vrcholů plochy a jejich vizuální spojnice vzdáleností.
- Zjišťování souřadnic zadaného bodu, jeho nejbližšího vrcholu plochy.
- Měření vzdáleností zadávaných bodů a vrcholů ploch.
- Zviditelnění či skrytí vizualizace vrcholů a vzdáleností.

DownloadProgressListener.class - zjišťuje informace o aktuálně načítané scéně. Propojuje konzoli a informační panel (viz. výše).

MenuFrame.class - třída pro okno Menu (viz. Dodatek C.3.1 a další okna (”Distance”, ”Distance - Advanced”, ”Help”, ”About”).

- Inicializace oken ”Distance”, ”Distance - Advanced”, ”Help”, ”About”.
- Zajišťuje komunikaci všech oken s třídou Browser.class.
- Zobrazuje v okně ”Distance” potřebné informace o zadaném bodě (souřadnice, který bod se zadává, vypočítané vzdálenosti).
- Umožňuje díky oknu ”Distance - Advanced” odstranění nebo naopak vizualizaci měřených vzdáleností.
- Odstraňuje BranchGroup, vytvořenou díky třídě IntersectInfoBehavior, či ji přidává do aktivní části grafu scény. To se děje díky zavírání či otevírání okna ”Distance”.

Dom.class - třída pro provázání prvků scény s XML souborem. Zobrazuje stromovou strukturu XML dokumentu a informace o jednotlivých prvcích modelu. Dále umožňuje zviditelnění, průhlednost či odstranění těchto prvků.

DomToTreeModelAdapter.class - třída pro převod dokumentu XML na stromovou strukturu (JTree) zobrazovanou v levé části okna "Layers"

AdapterNode.class - třída využívaná DomToTreeModelAdapter.class pro zjišťování elementárních uzlů ve stromové struktuře (JTree).

Browser - hlavní třída celého programu. Jako jediná obsahuje metodu main, která inicializuje a spouští program.

- Načítá soubor modelu VRML/X3D.
- Nastavuje v grafu scény příslušné bity prvkům modelu (funkce setCapabilities) pro možnost měření vzdáleností.
- Nastavuje seznam Viewpointů.
- Zjišťuje aktuální FPS a informaci zobrazuje v informačním panelu na pozici FPS.

4.3.2 Reprezentace scény

Nejdříve je virtuální model načten do proměnné *VRMLScene* a potom přidán do *VRMLUniverse* - universe. Zde jsou jednotlivé prvky uchovávány v grafu scény jako Shape3D. Objekty Shape3D uchovávají pouze informace ohledně svojí geometrii. Jako obrazy příkazů DEF a USE VRML souboru jsou zde objekty *SharedGroup*. Tyto objekty porušují stromovou strukturu grafu scény, jelikož na jeho potomky může ukazovat více objektů. Pro měření vzdáleností je vytvořen speciální BranchGroup, který je přidán do aktivního grafu scény podle potřeby. Jestliže uživatel již nechce měřit vzdálenosti, je z této části grafu scény odstraněn.

4.3.3 Příprava prvků modelu na měření vzdáleností

Před samotným zařazením prvků modelu do scény, kdy se stávají živými (angl. live), je nutné u nich nastavit tzv. bity Capabilities. To se děje pomocí funkce *setCapabilities* uvedené v Kapitole 2.4.4. Jelikož v jejich BranchGroup je u těchto objektů nastavena jedna z hodnot *Appearance* na skrytí (angl. hide), neúčastní se samotného vykreslování daného modelu. Ovlivňují celou scénu pouze případnými kolizemi. Jedna z těchto kolizí je právě měření vzdáleností (resp. určování souřadnice zadaného bodu).

4.3.4 Vizualizace měřené vzdálenosti

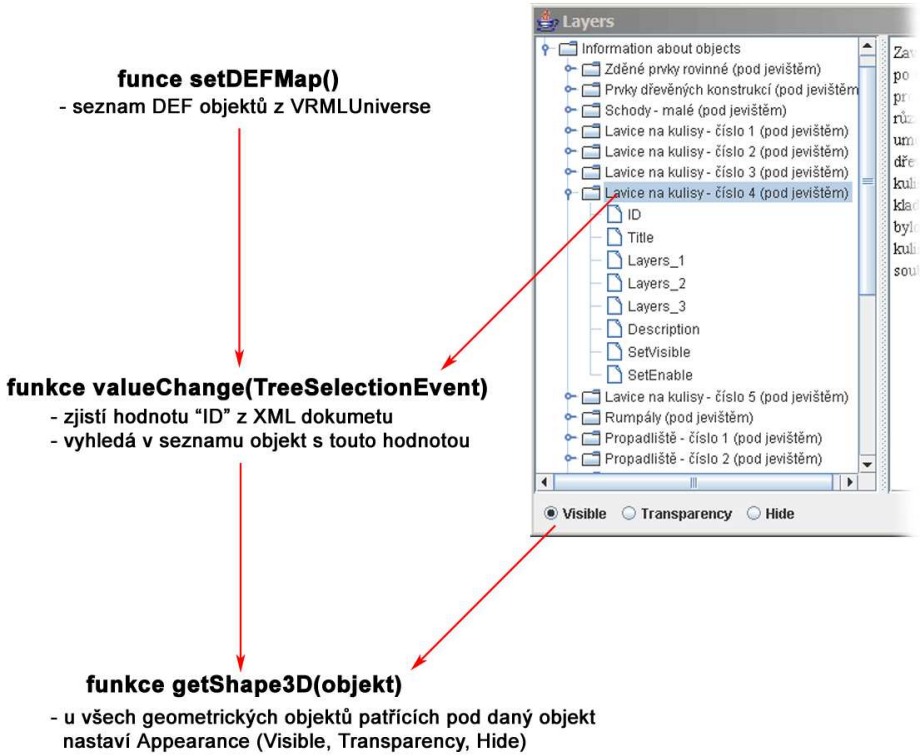
Jestliže je z třídy *IntersectInfoBehavior* vytvořen BranchGroup a přidán do objektu Locale, je rovněž nastavena zpětná reakce díky AWTEvent. Pomocí funkce processStimulus a zjištěného kritéria může začít proces zjištění souřadnic zadaného bodu. Nejdříve se určí souřadnice bodu myši na dvojrozměrné obrazovce a poté se vyšle paprsek směrem pohledu uživatele. K tomu slouží třída *pickCanvas*, která seřadí prvky scény protnuté paprskem a zjistí nejbližší. Tím dostaneme souřadnice zadaného bodu a můžeme vytvořit jeho vizualizaci. Díky třídě *PickIntersection* a její metodě *getClosestIntersection* jsme schopni najít nejbližší vrchol na ploše, která byla označena třídou pickCanvas. Vizualizaci propojení měřené vzdálenosti zobrazíme pomocí funkce na zjišťování transformace - connectCylinder (podrobněji viz. Kapitola 2.4.3).

4.3.5 Informace o objektech

Díky kvalitní dokumentaci [21] o převodu DOM struktury načteného XML souboru na stromovou strukturu (třída JTree) zobrazovanou v okně "Layers" (viz. Dodatek C.3.2), je zobrazování a práce s XML snadná. Posluchač (angl. Listener) sleduje jakékoliv kliknutí myši a podle něj zobrazuje danou stromovou strukturu v levé části okna "Layers". V pravé části tohoto okna zobrazuje požadovanou hodnotu.

4.3.6 Provázání stromové struktury XML s VRML modelem

Díky *TreeSelectionListener* je možné tzv. odchytávat označené uzly XML dokumentu v JTree formě. Jestliže takto získáme příslušnou větev XML dokumentu, dokážeme z něj i zjistit hodnotu "ID". Celé provázání mezi XML dokumentem a objektem VRMLUniverse přehledně ukazuje Obrázek 4.3.



Obrázek 4.3: Provázání XML s VRML modelem

Na závěr této kapitoly bych rád upozornil, že měření vzdáleností je možné i při zprůhlednění prvků scény (viz. Obrázek 6.2). Tyto dvě funkce aplikace Browser jsou vzájemně nezávislé.

Kapitola 5

Zhodnocení

V této kapitole ukazuji rozdíly různých druhů reprezentace VRML modelu barokního divadla. Jsou zde porovnávané VRML/X3D soubory ve všech fázích vývoje tohoto projektu. Dále uvádím výsledky testů jiných virtuálních scén. Všechny výsledky jsou uvedeny v přehledných tabulkách a scény jsou ilustrativně zobrazeny v Kapitole 6. Rovněž upozorňuji na chyby, které jsem v rámci implementace prohlížeče nalezl. Ke konci této kapitoly navrhuji další možný rozvoj projektu.

5.1 Technické vybavení

5.1.1 Použité technické vybavení

Vše bylo testované na počítači Pentium III 750MHz, paměť 384MB DDRAM, grafická karta ATI Radeon 7500, rozlišení 1280x1024 s 32bitovou hloubkou barev, operační systém Windows XP Profesional SP2. Při používání programu Cortona, byly nastaveny hodnoty standardně[1]. Při určování hodnot FPS byl pohyb nastaven na chůzi (podrobný popis jednotlivých pohybů je uveden v Dodatku C.3.1).

5.1.2 Doporučené technické vybavení

Jelikož byla tato diplomová práce rozdělena na dvě části a jelikož jsem se snažil, aby tyto části nebyly na sobě navzájem závislé, je třeba i na tomto místě doporučit technické vybavení pro každou část zvlášť:

VRML scéna barokního divadla dostatečně optimalizovaná pomocí nejrůznějších nástrojů je i přesto náročná na hardwarové vybavení. Doporučuji pro plynulost scény technické vybavení uvedené v Kapitole 5.1.1. Softwarové vybavení pro prohlížení samotné VRML scény je závislé na použitém operačním systému. U operačních systémů firmy Microsoft Windows 95/98/ME/2000/XP doporučuji IE5+ a program Cortona.

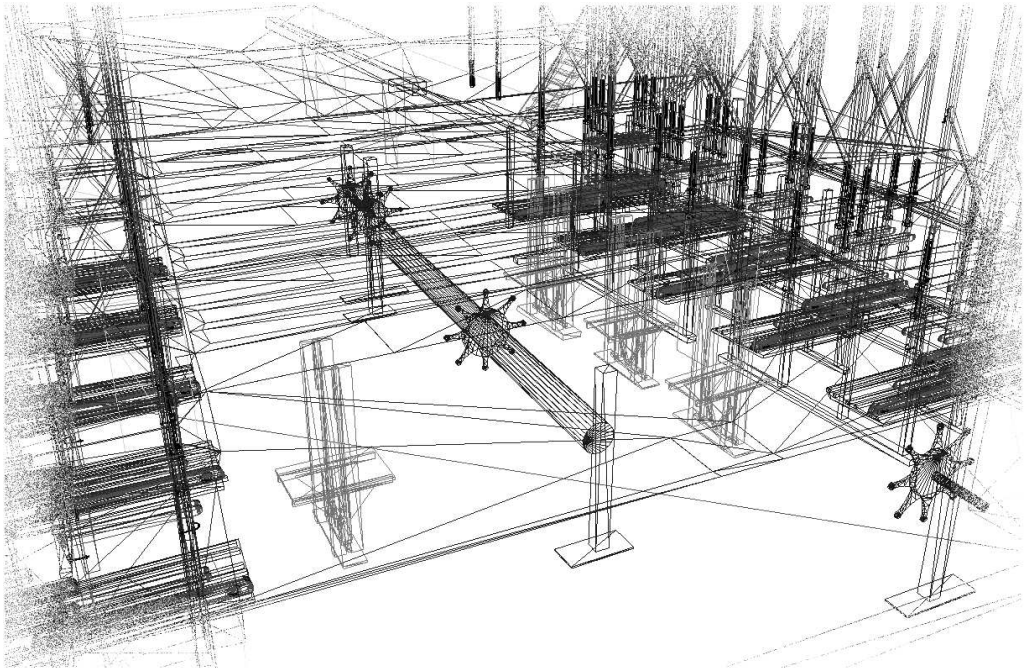
Aplikace **Browser** je náročnější na hardwarové vybavení, než je uvedené pro výše zmíněné prohlížení VRML scény. Doporučuji procesor AMD XP+ 1.5GHz, operační paměť 512MB a grafickou kartu ATI Radeon 9200 (128MB, 128bitů). Softwarové vybavení je závislé na způsobu spouštění daného programu a je podrobně a přehledně popsáno v Dodatku C.1.

[1] Renderer: "OpenGL" a v "Renderer options" bylo povoleno: "Dithers colors if needed", "Limit texture size", "Optimize textures for quality", "Use textures mip-mapping", "Intel Pentium III or higher"

5.2 Testy optimalizace

V rámci optimalizace jsem prováděl porovnání jednotlivých fází vzniku VRML scény barokního divadla (viz. Obrázek 5.1) a porovnal v Tabulce 5.1. Každá fáze vývoje znamená jinou scénu. Tyto scény jsou řazeny chronologicky:

1. Scéna - export do VRML formátu přímo z programu Microstation verze 9. Testováno na počítači uvedeném výše.
2. Scéna - export do VRML formátu z programu 3D Studio VIZ bez jakýchkoliv úprav modelu.
3. Scéna - export do VRML formátu z programu 3D Studio VIZ s modelováním celé scény (podjevištní část i jeviště). Dále byl použit program Chisel pro optimalizaci scény a vše je díky INLINE uzlům připojeno z jednotlivých souborů.
4. Scéna - stejná jako 3. Scéna, ale byly odstraněny INLINE uzly a vše bylo sloučeno do jediného souboru.
5. Scéna - jde o 4. Scénu testovanou v aplikaci Browser.
6. Scéna - jde o 4. Scénu převedenou do formátu X3D a testovanou v aplikaci Browser.



Obrázek 5.1: Drátěný model barokního divadla v Českém Krumlově

5.3 Testovací scény

Při samotné implementaci a v rámci testů jsem narazil na mnoho chyb, které jsou způsobeny nekvalitním loaderem. Pro testy na různých virtuálních scénách jsem použil jak standardní VrmlLoader z Javy3D, tak Web3DLoader z projektu Xj3D. Proto jsou i jednotlivé popisy testovaných scén rozděleny na tyto části. Pokud nebudu uvádět VRMLLoader, byl výsledek shodný s Web3DLoaderem. V Kapitole 5.4 vše shrnuji do přehledné tabulky.

Typ scény	Velikost	Počet trojúhelníků	FPS	Použitý prohlížeč	Počet DEF	Počet USE
1. Scéna	28.3 MB	Nelze prohlížet ani optimalizovat				
2. Scéna	15 MB	518.928	0.5	Cortona	894	7
3. Scéna	537 KB	10.388	5	Cortona	193	2178
4. Scéna	349 KB	10.388	5	Cortona	189	2178
5. Scéna	349 KB	10.388	4	Browser	189	2178
6. Scéna	205 KB	10.388	4	Browser	189	2178

Tabulka 5.1: Porovnání vlastností VRML souboru v různých fázích jeho vývoje

Všechny virtuální modely byly korektně načteny v programu Cortona. Vše je ilustrováno příslušným obrázkem, pokud byl model načten korektně. Pokud se model nenačetl korektně, je přiložen ilustrativní obrázek z programu Cortona. Pokud nebylo možné virtuální model vůbec zobrazit, popisuji možný důvod. Všechny tyto příklady jsou uloženy na přiloženém CD-ROM a zde jsou roztříděné podle názvu souborů:

5.3.1 Jednoduché světy

Výběr příkladů jednoduchých světů z [17]:

o-3-1a.wrl - umístění a orientace koule.

- Web3DLoader - nedokáže správně načíst Inline uzly. Pokud jsem tedy použil tento soubor bez Inline uzlů, vše probíhalo v pořádku (viz. Obrázek 6.4 a)
- VrmlLoader - dokáže načíst Inline uzly, ale nedokáže pracovat s uzly PROTO.

o-3-4.wrl - odlišné materiálové parametry použité na tři tělesa s podobnou geometrií.

- Web3DLoader - načteno v pořádku (viz. Obrázek 6.4 b)

o-3-5a.wrl, o-3-5b.wrl, o-3-5c.wrl, o-3-5d.wrl - pokrytí základních těles texturou (koule, kvádr, kužel a válec).

- Web3DLoader - obrazová textura typu .gif na základních tělesech není viditelná i když se korektně načetla. Na Obrázku 6.4 c lze tedy pozorovat pouze texturované hrany. Jestliže však použiji .png formát obrázku textury, vše funguje správně (viz. Obrázek 6.4 d). Problém nastává až u texturování válce, jak můžete vidět na Obrázku 6.4 e. Problém nanášení textur na prvky modelu barokního divadla je vidět i přímo v mém programu Browser na Obrázku 6.3.

o-3-6.wrl - použití průhledné a poloprůhledné textury.

- Web3DLoader - načteno v pořádku (viz. Obrázek 6.4 f)

o-3-7.wrl - obrazová textura obarvená v uzlu Material.

- Web3DLoader - jak vidíte na Obrázku 6.5 a, je textura .gif načtena chybně. Navíc, když je obrázek nedostupný, nezbarví se příslušné geometrické objekty (jak je tomu v programu Cortona) barvou zadanou v uzlu Material.

p-3-08.wrl - textura v uzlu PixelTexture použitá ke šrafování válce.

- Web3DLoader - chybně načtená textura (viz. Obrázek 6.5 b).
- VrmlLoader - chyba při načítání uzlu PixelTexture.

p-3-09.wrl - Hnědý stolek s využitím DEF a USE.

- Web3DLoader - vše v pořádku (viz. Obrázek 6.5 c).
- VrmlLoader - nedokázal načíst základní těleso - válec. Ten zde představuje nohy od stolu s parametrem "top FALSE". Tento parametr, který zabraňuje vykreslování horní podstavy, nemůže být s tímto loaderem použit.

p-3-11.wrl - Různé kombinace parametrů uzlu IndexedFaceSet na tři různé trojúhelníky.

- Web3DLoader - vše v pořádku (viz. Obrázek 6.5 d).

p-3-15.wrl - Obrázek nanesený na výškovou mapu.

- Web3DLoader - vše v pořádku (viz. Obrázek 6.5 e) i když se jedná o texturu obrázku typu .gif, která nefunguje na základní tělesa (viz. výše)

o-3-20.wrl - Dvě na první pohled stejné desky jsou osvětlovány jedním reflektorem.

- Web3DLoader - nenačte SpotLight světla (viz. Obrázek 6.5 f). Chyba je v zadaném parametru SpotLight - cutOffAngle, který ještě není podporván.

p-3-19.wrl - Tři tělesa postupně se ztrácející v mlze.

- Web3DLoader - vše je zobrazováno korektně (viz. Obrázek 6.6 a).

p-3-21.wrl - Pozadí ze šesti obrazů zajišťuje úplný panoramatický pohled.

- Web3DLoader - vše je zobrazováno korektně. Samozřejmě zde, jak vidíte na Obrázku 6.6 b, nebylo možné použít měření vzdáleností, jelikož zde není žádný uzel Shape3D.

p-3-23.wrl - Použití billboardů pro modely stromů a zvířat.

- Web3DLoader - vše je zobrazováno korektně. Jak vidíte na Obrázku 6.6 c, nejbližší vrcholy billboardů jsou průhledné. Korektně rovněž probíhá natáčení scény k avatarovi.
- VrmlLoader - nelze použít, protože uzel Billboard není podporován.

p-3-24.wrl - Pomocí uzlů Anchor lze návštěvníka provádět od stanoviště ke stanovišti.

- Web3DLoader - vše je zobrazováno korektně (viz. Obrázek 6.6 d).

p-3-25.wrl - Tři reprezentace téhož tělesa uzlem LOD.

- Web3DLoader - uzel LOD je zobrazován korektně, ale při měření ukazuje značné chyby. Program sice nehlásí žádnou chybu, ale samotný LOD objekt není do měření zahrnut. Při bližším zkoumání celého problému jsem došel k názoru, že jako Shape3D je brán ohraničující Bounding Box, který určuje nejvzdálenější reprezentaci LOD.
- VrmlLoader - v tomto případě se oproti Web3DLoader chová korektně (viz. Obrázek 6.6 e), odstraníme-li reprezentaci Billboard (díky zmíněné chybě tohoto loaderu - viz. p-3-23.wrl). Na tomto obrázku lze vidět, že body měření a jejich nejbližší vrcholy neleží na ploše měřeného objektu. Je to způsobeno tím, že měření vzdáleností bylo použito pro vzdálenější reprezentaci LOD. Nutno podotknout, že tento soubor s sebou přinesl další překvapení pro tento loader: nelze psát českou diakritiku v komentáři VRML souboru.

5.3.2 Složitější scény

Výběr příkladů složitějších světů z [17]:

p-4-4.wrl - použití knihovny prototypů materiálů.

- Web3DLoader - vše zobrazeno správně (viz. Obrázek 6.6 f).
- VrmlLoader - nelze použít uzly PROTO.

p-5-01.wrl - předávání událostí mezi parametry uzlů.

- Web3DLoader - reaguje v pořádku na ProximitySensor jen změnou světla. Nepřehrává zvuky formátů .mid, .mp3 ani .wav (viz. Obrázek 6.7 a).
- VrmlLoader - nedokázal model načíst, není podporován Proximity Sensor.

p-5-06.wrl - propojení zdrojů světla s detektorem dotyku.

- Web3DLoader - model načten v pořádku (viz. Obrázek 6.7 b) a uzel TouchSensor je plně funkční.
- VrmlLoader - nelze načíst model, není podporován TouchSensor.

p-5-09.wrl - barevné proměny reklamního nápisu.

- Web3DLoader - vše se načetlo korektně, jen font písma reklamního nápisu je jiný, než v programu Cortona (viz. Obrázek 6.7 c).
- VrmlLoader - nelze měřit vzdálenosti, není podporován uzel Text.

p-5-14.wrl - definice tělesa v roli průvodce s pomocí detektoru přítomnosti.

- Web3DLoader - měření vzdáleností nelze provádět díky uzlu PROTO.
- VrmlLoader - nelze načíst, není podporován uzel ProximitySensor.

p-6-02.wrl - zvýraznění vzhledu tělesa s pomocí převodního skriptu.

- Web3DLoader - Zvýraznění tělesa nefunguje korektně. Je zobrazeno varování: "Attempting to set invalid value type to SFInt32 field". Měření vzdáleností probíhá v pořádku (viz. Obrázek 6.7 d).

p-6-04.wrl - Tlačítko se dvěma stavy (červená a zelená krychle).

- Web3DLoader - Vše korektně funguje, jen měření vzdálenosti je prováděno pouze na zelené krychli a nelze tedy změřit vzdálenosti vrcholů mezi červenou a zelenou krychlí (viz. Obrázek 6.7 e).

p-6-09.wrl - Spouštění několika časovačů skriptem.

- Web3DLoader - scéna se načte korektně, ale skript hlásí varování: "Attempting to set invalid value type to SFInt32 field". Proto se ani správně nezobrazují barvy na semaforu (viz. Obrázek 6.7 f).

p-6-10.wrl - Složitější řízení animace skriptem.

- Web3DLoader - načtení scény proběhlo v pořádku (viz. Obrázek 6.8 a) , měření vzdáleností rovněž, ale tlačítka "J" a "C" pro ovlivňování chodu ručičky nefungovali správně.

5.3.3 Ostatní scény

Zde jsem se zaměřil na scény ostatní (např. formáty prostorových dat X3D, X3DV a WRZ). Uvádím pouze testování pro Web3DLoader, jelikož tyto formáty dat VrmlLoader nepodporuje:

all.x3d - X3D model barokního divadla je zobrazován v pořádku. Měření vzdáleností i práce s prvky modelu rovněž.

all.x3dv - X3D model barokního divadla ve formátu X3DV. Zobrazení modelu, měření vzdáleností i práce s prvky modelu fungují v pořádku.

all-komprimovane.wrz - komprimovaný soubor barokního divadla VRML pomocí GZIP není podporován.

magnolia.wrl - soubor VRML1.0 z [19] není podporován. Převedl jsem jej na VRML2.0 (magnolia2.wrl) pomocí programu VRML1TO2 z [18]. Pak se již zobrazoval správně (viz. Obrázek 6.8 b).

space.x3dv - tento soubor obsahuje pozadí ve formátu X3DV. Zobrazil se správně. Počet zobrazovacích barev je bohužel jen 256 (viz. Obrázek 6.8 c). Jeho protějšek ve formátu VRML - space.wrl, zobrazuje program Cortona v plné barevné škále[2].

nurbssurface.x3dv - soubor s NURBS křivkami. Pro správnou funkčnost NURBS ploch a křivek bylo nutné použít třídy NurbsCurve.class, NurbsSurface.class, NurbsRevolve.class z [8]. Zobrazení a měření vzdáleností probíhalo v pořádku (viz. Obrázek 6.8 d).

5.4 Souhrn chyb

5.4.1 Web3DLoader a VrmlLoader

V Tabulce 5.2 uvádím přehledně souhrn všech chyb loaderů, na které jsem během implementace a testování programu narazil. Tabulka je rozdělena na chyby zjištěné u načítání scény pomocí *Web3DLoader* a standardního *VrmlLoader*. Hodnoty jednotlivých buněk tabulky jsou:

- CHYBA - díky důvodu uvedeného v tabulce nebylo možné virtuální model načíst
- OK - načtení modelu proběhlo bezchybně
- ostatní nesrovnalosti vysvětlené podrobněji

Podrobně jsou tyto nedostatky a chyby popsány v Kapitole 5.3.

5.4.2 Java Web Start

Experimentální projekt Xj3D, který ve své práci využívám má stále i ve verzi M9[3] mnoho nesrovnalostí, či chyb. Zde uvádím chyby Xj3D Browseru, na které jsem upozornil přímo vývojáře tohoto projektu na [9]. Všechny tyto chyby jsou pouze u spuštění aplikace přes Java Web Start:

- Z celého seznamu viewpointů virtuální scény načte pouze první.

[2]U Obrázku 6.8 c, není zobrazena scéna z programu Cortona, jelikož jsem při vytváření a tisku této dokumentace používal pouze 256 barev.

[3]Nová verze označená M10 RC1 byla oficiálně představena 11.1.2005

Důvod	Web3DLoader	VrmlLoader
Inline uzel	CHYBA	OK
Mapování textur .GIF	špatně mapováno	špatně mapováno
Mapování textur .PNG, .JPG	OK	OK
Cylinder - vytvoření základního tělesa	OK	Nesmí být použity parametry side, bottom a top
Cylinder - mapování textur	Špatně mapováno	Špatně mapováno
PixelTexture uzel	OK	ERROR
Použití Javascript	Funguje ve většině případů	CHYBA
SpotLight	Nesmí být použit s parametrem cutOffAngle	CHYBA
Komentáře	OK	Nesmí být použita česká diakritika
Zvuk .mid, .mp3, .wav	Nepřehrává	CHYBA
PROTO uzel	OK	CHYBA
LOD uzel	Není korektně načten	OK
Obarvení textury	CHYBA	CHYBA
Billboard uzel	OK	CHYBA
ProximitySensor uzel	OK	CHYBA
TouchSensor uzel	OK	CHYBA
Text uzel	Nevyhlazené písmo fontu	CHYBA
.X3D formát souboru	OK	CHYBA
.X3DV formát souboru	OK	CHYBA
.WRZ - komprimovaný VRML	CHYBA	CHYBA
NURBS	Některé jednoduché plochy zobrazí	CHYBA
VRML1.0	CHYBA	CHYBA

Tabulka 5.2: Souhrn nesrovnalostí a chyb zjištěných při implementaci a testování programu.

- Nebyly původně zobrazeny ikony pro navigační panel (Fly, Pan, Tilt, Walk, Examine - viz. Dodatek C.1). Tato chyba se dala odstranit změnou a přidáním zdrojového souboru NavigationPanel.java k této diplomové práce.
- Nejsou zobrazeny ikony pro viewpoint panel (viz. Dodatek C.1) - "Další viewpoint", "Předchozí viewpoint" a "Základní viewpoint".

5.4.3 Ovládání programu

Zde uvádím nesrovnalosti v ovládání programu, které jsou způsobené chybně napsanými knihovnami Xj3D. Tyto chyby jsou jak v samostatně spouštěné aplikaci, tak v aplikaci spuštěné přes Java Web Start.

- Ovládání pohybu avatara přes "Examine" (přesněji popsáno v Dodatku C.3.1) automaticky předpokládá umístění zkoumaného objektu v počátku souřadnicového systému virtuálního světa. V tomto důsledku je zkoumání virtuálního světa, který leží mimo tyto souřadnice, nemožný.
- Tlačítko Xj3D Browseru "Základní viewpoint" (viz. Dodatek C.1) není funkční.

5.5 Jak je možné pokračovat

Jelikož jde o poměrně rozsáhlý projekt, uvádím na tomto místě další možné cesty:

1. **Vizualizace dat** - rozšíření VRML modelu a převod na X3D formát dat.
2. **Zkoumání prostorových dat** - přidání dalších funkcí pro uživatelsky příjemnější prostředí.
3. **Správa divadla** - propojení s databází zámku.

Vše podrobně rozvíjím v jednotlivých podkapitolách.

5.5.1 Vizualizace dat

V této diplomové práci jsem pracoval z částí modelu barokního divadla - podjevištního a jevištního prostoru. Tyto data byly bohužel neúplná z důvodu dokončování prací na samotném zaměřování celého divadla. Další části, které by měli patřit k vizualizaci celého divadla jsou: *nadjevištní část, hlediště, krov a exteriér*. Jak je zřejmé z Tabulky 5.1, je možné pro zobrazení scény použít VRML/X3D model. Nejlepší variantou je přemodelování těchto dat a jejich další optimalizace. To je nutné udělat, podle mého názoru, i pro další části modelu divadla.

Standard VRML97 je pomalu nahrazován X3D, který znamená další virtuální krok dopředu. Díky tomu, že byla tato diplomová práce napsaná za pomocí Xj3D, je možné formát X3D v aplikaci Browser použít. V době vývoje tohoto programu je ale editace prostorových dat ve formátu X3D obtížná.

5.5.2 Zkoumání prostorových dat

Načítání samotných dat pomocí technologie DOM může mít za následek vylepšení samotné *editaci XML dokumentu* (přidávání dalších prvků modelu, mazání, změny jednotlivých položek). Dále je možné rozšířit aplikaci o jednoduché *třídění prvků modelu* podle jména, vrstev apod. Poslední možností vylepšení projektu, v tomto kontextu, je nepoužívat přesně zadané položky každého prvku modelu (ID, Description, Layers, setEnable atd. - viz. Kapitola 3.3.5). Použít příslušného *XML souboru jako samotné šablony* zobrazovaných položek. V tom případě by i třídění, které jsem uvedl výše, bylo použitelné právě pro tyto položky v závislosti na použitém XML dokumentu.

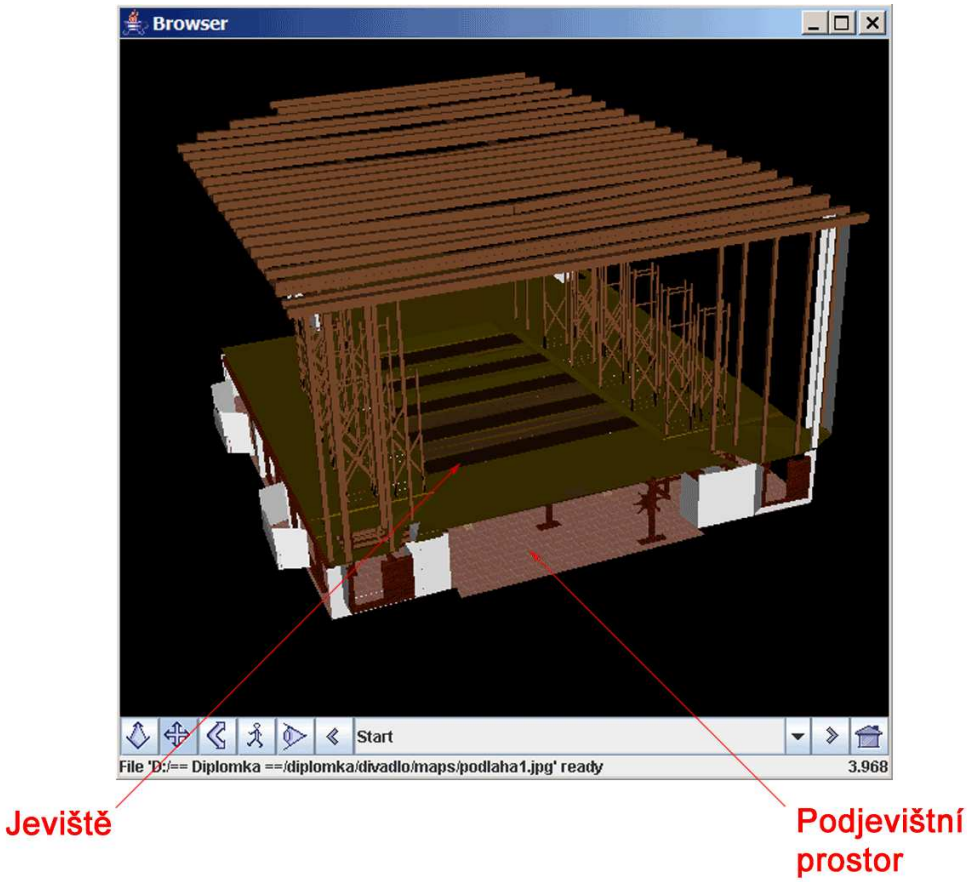
5.5.3 Správa divadla

Uložení informací v přehledném formátu dat XML je prozíravé pro pozdější možnost propojení s databází správy zámku v Českém Krumlově a pokročit k další fázi vývoje - *správa divadla*. Do XML souboru je teoreticky možné exportovat z jakékoliv databáze. Zde by šlo o tzv. off-line připojení k databázi přes XML soubor, který by se aktualizoval podle potřeb.

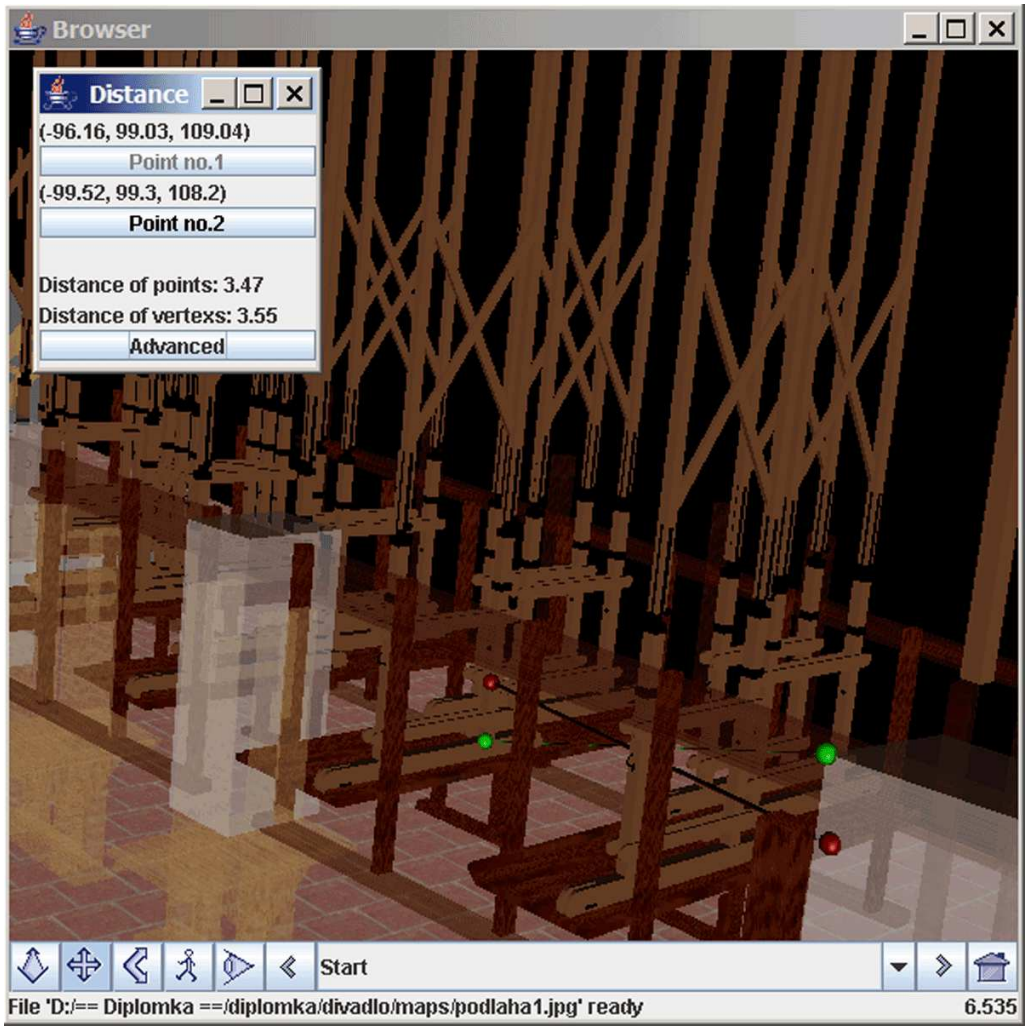
Jednalo by se o propojení s databází on-line, neexistuje lepšího prostředí propojení s databázemi, podle mého názoru, než je Java.

Kapitola 6

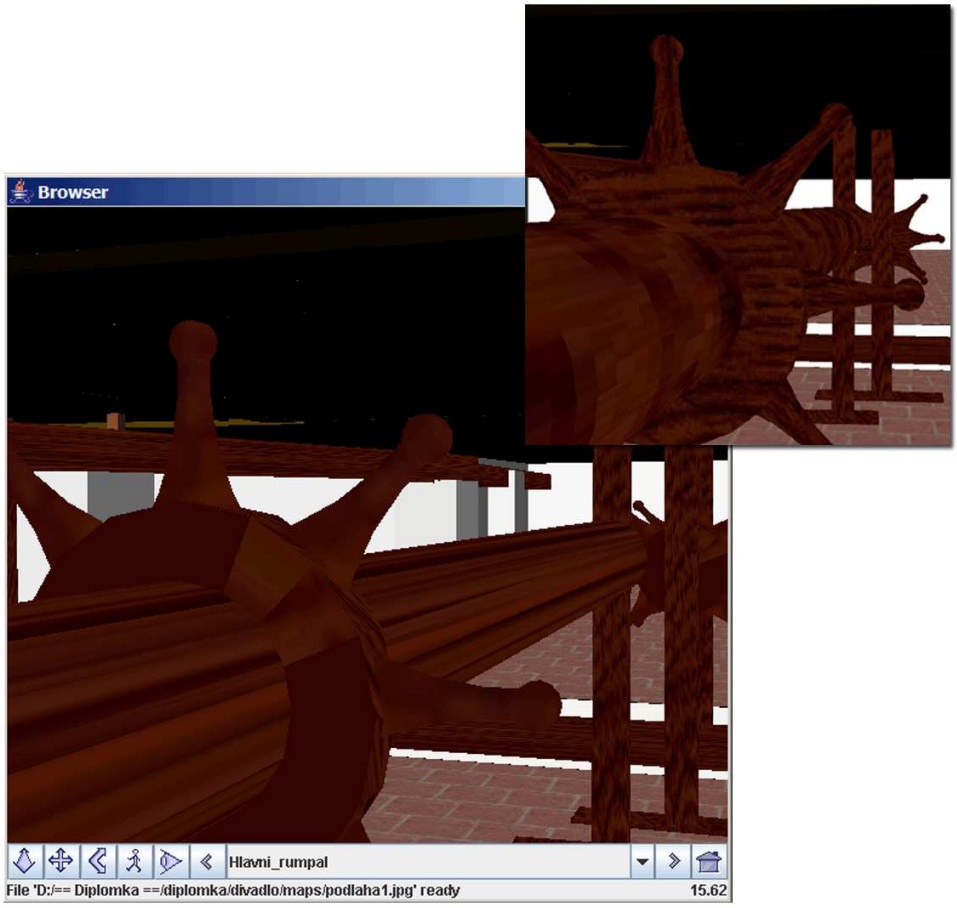
Barevná příloha



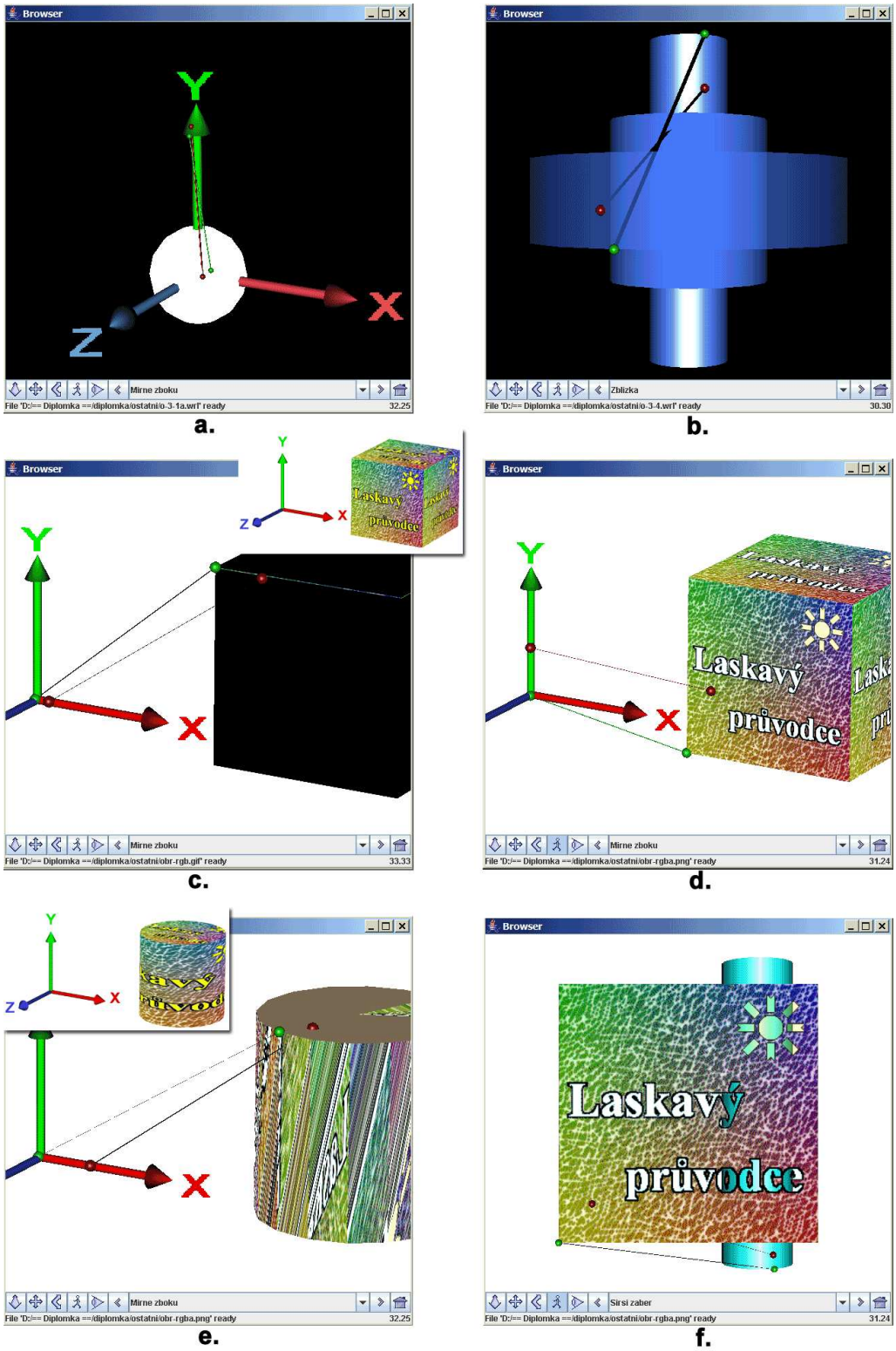
Obrázek 6.1: Virtuální model barokního divadla v Českém Krumlově - jeviště a podjevištní prostor.



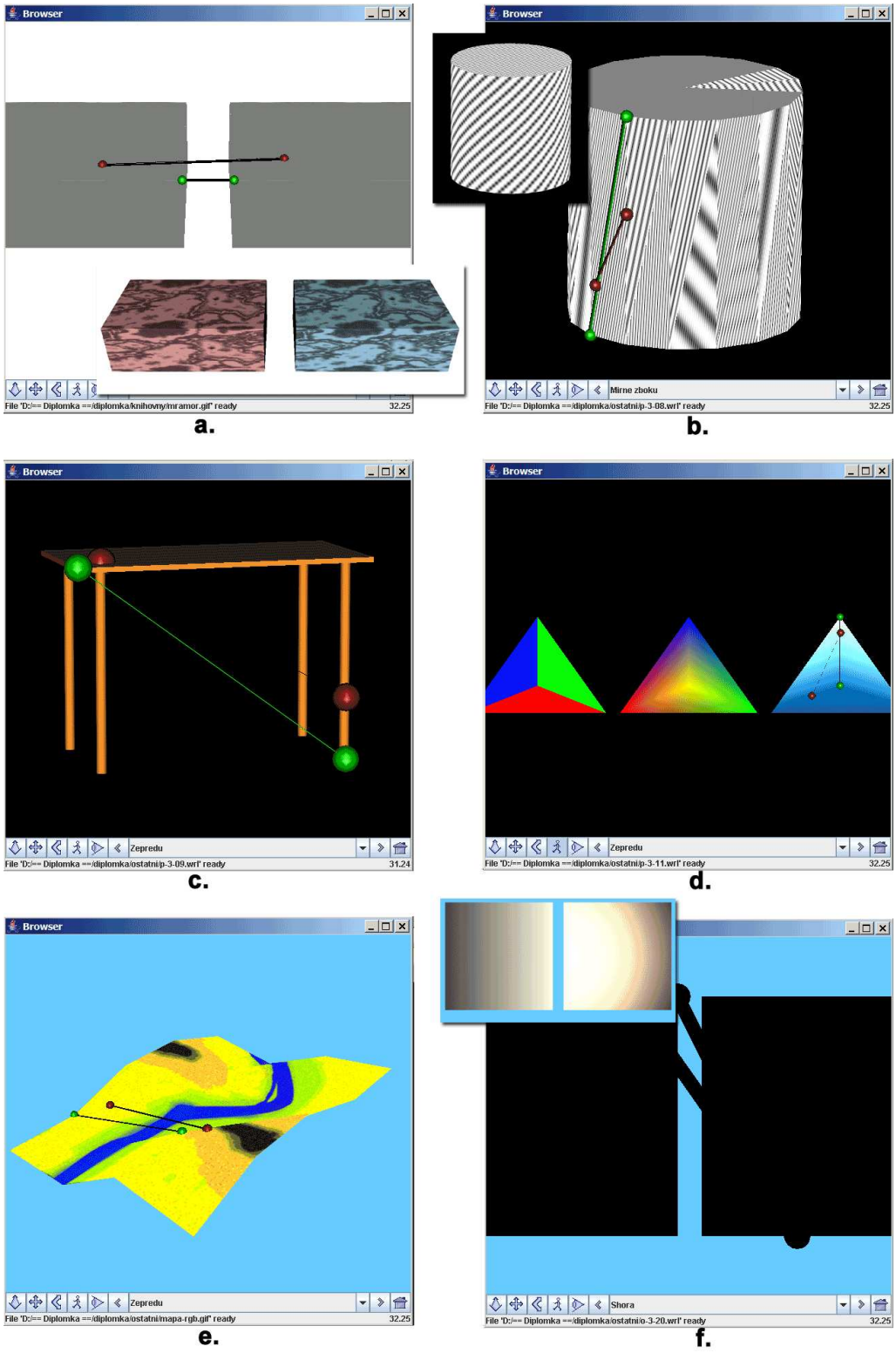
Obrázek 6.2: Měření vzdálenosti při zprůhlednění některých prvků modelu barokního divadla.



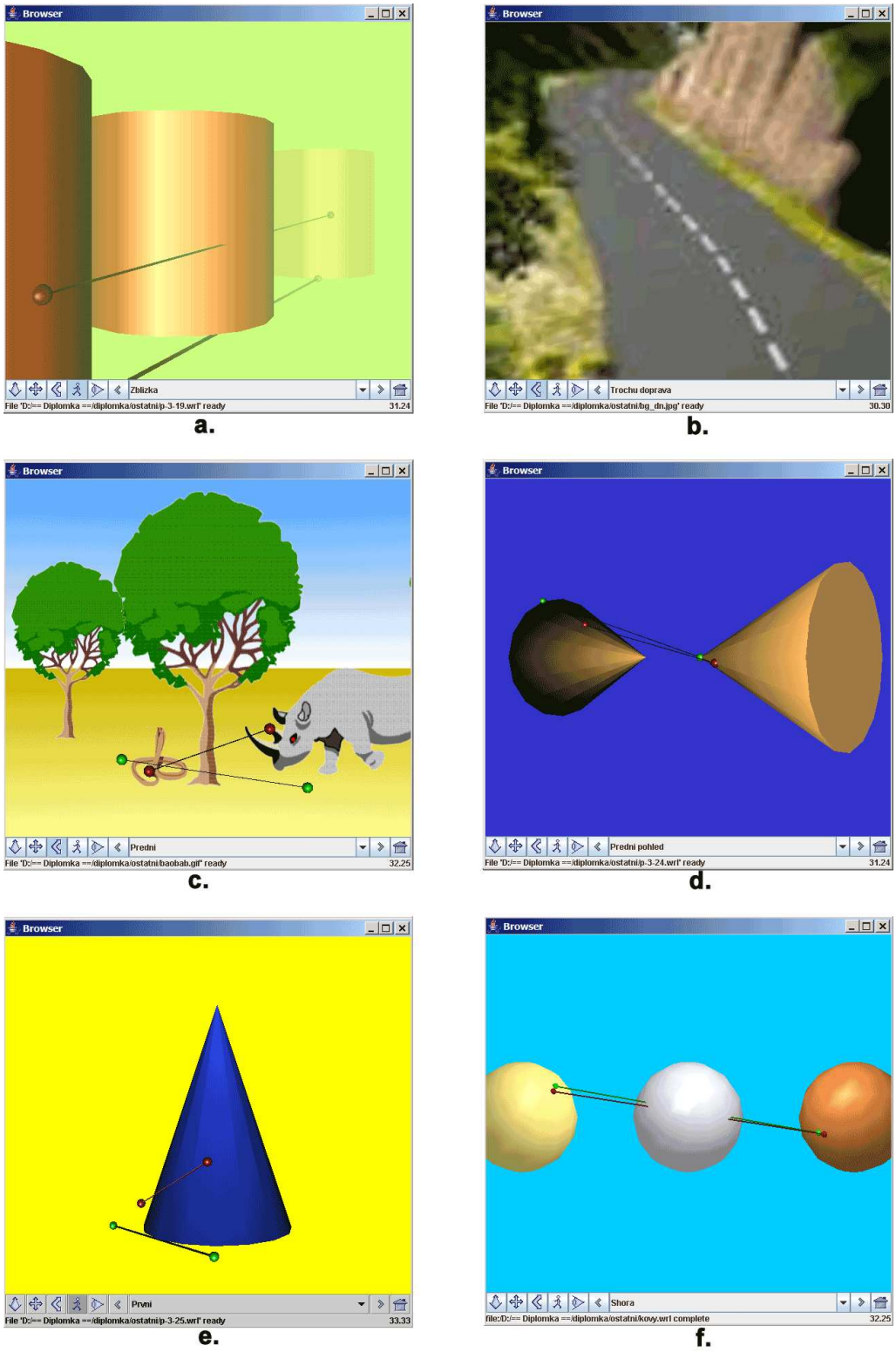
Obrázek 6.3: Ukázka špatného nanesení textury na válec (rumpál v podjevištní části modelu barokního divadla)



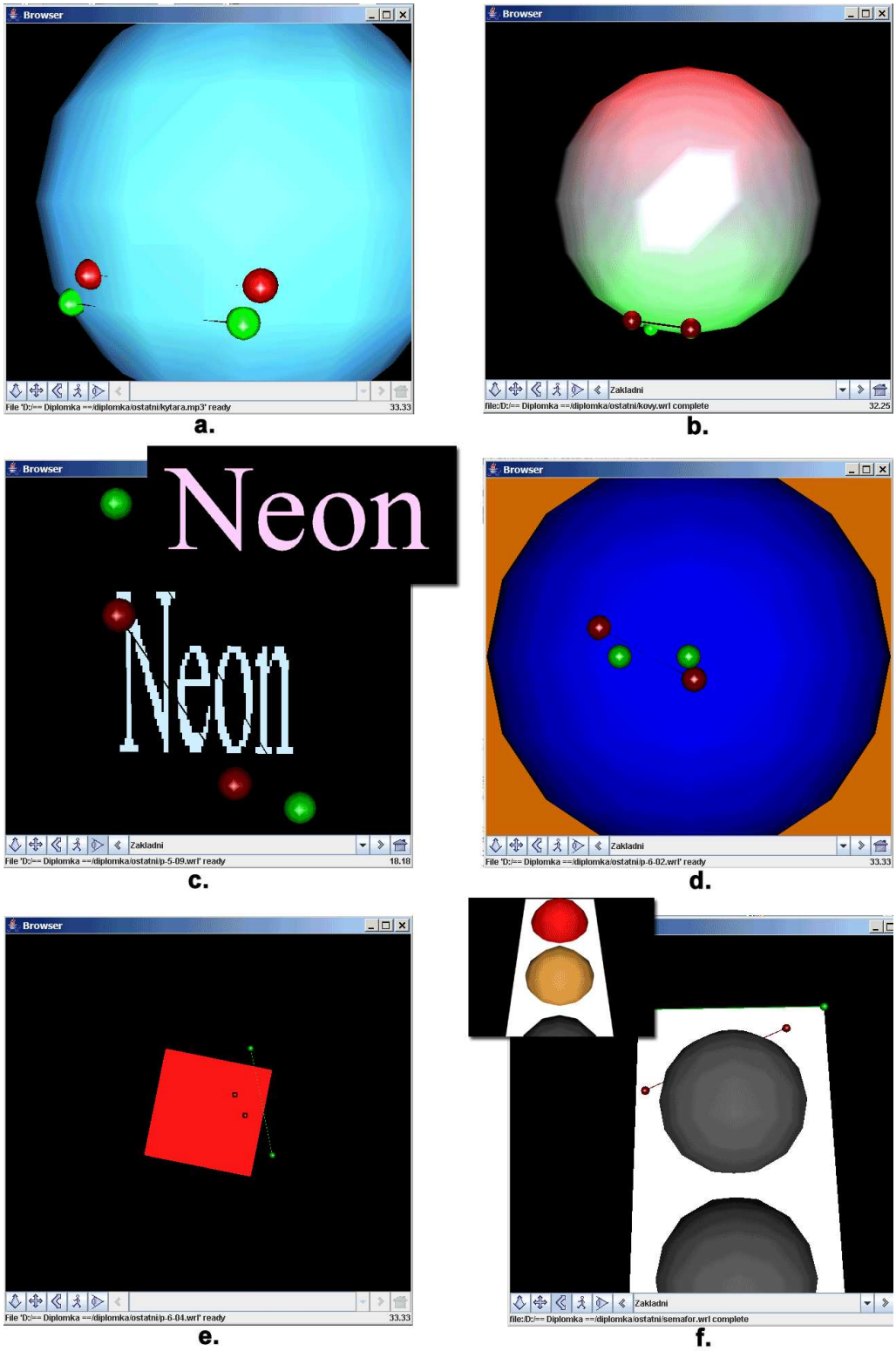
Obrázek 6.4: Testovací scény: **a.**umístění a orientace koule, **b.**odlišné materiálové parametry, **c.**pokrytí kvádru texturou .gif, **d.**pokrytí kvádru texturou .png, **e.**pokrytí válce texturou .png, **f.**průhledná textura



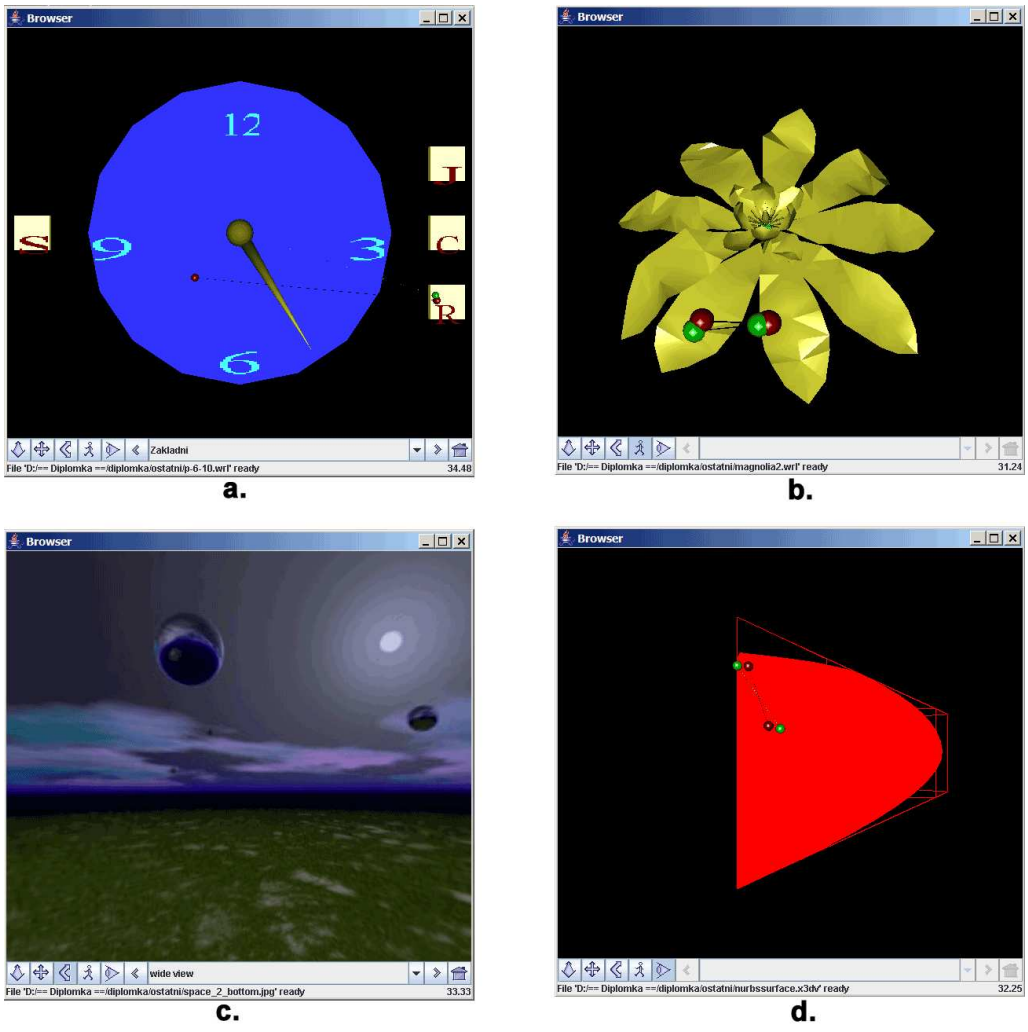
Obrázek 6.5: Testovací scény: **a.**obarvení textury, **b.**PixelTexture použit na válec, **c.**použití DEF a USE, **d.**kombinace parametrů IndexedFaceSet, **e.**výšková mapa s texturou, **f.**použití reflektoru



Obrázek 6.6: Testovací scény: **a.**použití mlhy, **b.**panorama pozadí, **c.**použití Billboardů, **d.**použití uzlu Anchor, **e.**LOD (Level Of Distance), **f.**prototypy materiálů



Obrázek 6.7: Testovací scény: **a.**detektor polohy, **b.**detektor dotyku, **c.**animace barev, **d.**použití skriptu, **e.**detektor doteku, **f.**spouštění více časovačů



Obrázek 6.8: Testovací scény: **a.**složitější animace řízená skriptem, **b.**VRML2.0 a VRML1.0, **c.**X3DV a pozadí, **d.**NURBS plocha

Kapitola 7

Závěr

Tato diplomová práce se zabývá problémem vizualizace prostorových dat. První část zkoumá vytváření optimálního modelu podjevištního a jevištního prostoru barokního divadla v Českém Krumlově. Je zde ukázána cesta, jak prostorové data převádět ze standardních CAD souborů na formát VRML/X3D. Existuje sice velké množství automatických převodů, ale celková velikost výsledného souboru a rychlost zobrazování modelu je v tomto případu nevyhovující. Proto je nutné virtuální model dále upravovat.

Problém optimalizace byl vyřešen přemodelováním v programu 3D Studio VIZ. Dále byl na exportovaný soubor VRML použit program Chisel. Tak došlo k radikálnímu zmenšení jeho velikost a počtu zobrazovaných polygonů.

V rámci druhé části byl navržen a implementován program Browser, který využívá experimentálního projektu Xj3D. Program je odladěn proti chybám a otestován na sadě testovacích scén. Díky experimentální povaze projektu Xj3D a jeho rychlému zdokonalování, by vývoj tohoto programu měl rovněž dále pokračovat. V této práci byly podrobně popsány chyby či nedostatky, které nejsou v Xj3D kompletně. Na tyto nedostatky byly vývojáři Xj3D projektu upozorněni.

V rámci prezentace daného modelu byly jednotlivé prvky scény provázány s dokumentací uloženou v XML souboru. Speciální funkce pro měření vzdáleností a práce s prvky modelu (odstraňování či zprůhledňování těchto prvků) byly vytvořeny pro možnost používání i s jinými virtuálními modely.

Zajímavým námětem na pokračování je nejen doplnit vytvořený model barokního divadla o další prostory (nadjevištní část, hlediště, krov a exteriér), ale rozšířit uživatelský komfort o další funkce. Mezi tyto funkce může patřit: třídění prvků modelu podle vrstev, editace a ukládání informačního XML dokumentu nebo provázání s již fungující databází pro správu hradu a zámku.

Literatura

[1] *Cortona VRML Client*
http://www.parallelgraphics.com/products/cortona/

[2] *Java3D API*
http://java.sun.com/products/java-media/3D/

[3] *Java Web Start Technology*
http://java.sun.com/products/javawebstart/

[4] *Chisel VRML Optimizer*
http://java.sun.com/products/javawebstart/

[5] *Microstation / Bentley*
http://www.deskware.de/webcontent/produkte/microstation/

[6] *Discreet product - 3D Studio MAX*
http://www4.discreet.com/3dsmax/

[7] *Autodesk VIZ*
http://www.autodesk.cz

[8] *Web3D Consortium - X3D Documentation*
http://www.web3d.org/x3d/

[9] *The Xj3D Project*
http://www.xj3d.org/

[10] *Java JRE a JDK*
http://java.sun.com/

[11] *MSJVM - Microsoft Java Virtual Machine*
http://www.microsoft.com/cze/windows/java/

[12] Žára, Jiří. *VRML97 - Laskavý průvodce virtuálními světy.* 1. vyd. Praha: Coputer Press, 1999. ISBN 80-7226-143-6.

[13] *ČVUT FEL - předmět 36MUS (Multimediální systémy)*
http://www.cgg.cvut.cz/ mus/

[14] *VRML97 - Laskavý průvodce virtuálními světy (ukázky)*
http://www.cgg.cvut.cz/LaskavyPruvodce/

[15] Nytra, Daniel. *36SP - Semestrální projekt.* 2003
http://nytra.aktualne.cz/36sp/

[16] Kuželka, Ondřej. *Java3D a grafika.*
http://interval.cz/clanek.asp?article=2752

[17] Žára, Jiří. *Příklady z knihy [12].*
http://www.cgg.cvut.cz/LaskavyPruvodce

[18] *VRML1TO2 - convert VRML1.0 file to VRML 2.0 file.*
http://appsrv.cse.cuhk.edu.hk/ csc5460/mirror/vrml1to2/vrml1to2E.html

[19] *3D CAFE's VIP LOUNGE.*
http://www.3dcafe.com

[20] Žára, Jiří; Beneš, Bedřich; Felkel, Petr. *Moderní počítačová grafika.* 1. vyd. Praha: Coputer Press, 1998. ISBN 80-7226-049-9.

[21] *Manipulanting Contents with DOM.*
http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/dom/

[22] Nytra, Daniel.*Nytra - Aktuálně.*
http://nytra.aktualne.cz/diplomka/

Dodatek A

Slovníček pojmů

3DStudio VIZ - software [7] pro fotorealistické vizualizace a animace architektonických i designerských modelů a scén. VIZ je využíván v architektonických animacích (oblety, průchody), studiích osvětlení/stínů, zákresech projektu do fotografie, návrzích interiérů, apod. Úzce spolupracuje s produkty firmy Autodesk (file-link), lze jej však použít i s jinými CAD aplikacemi či jako samostatný modelář.

3DStudio MAX - animační a vizualizační software [6]. Je určen pro tvorbu vizuálních efektů, animaci postav a tvorbu počítačových her. Nabízí interaktivní prostředí a rychlý rendering. Oproti programu 3DStudio VIZ neposkytuje možnost práce s vrstvami.

Applet - program v Javě, který pro svůj běh vyžaduje Java-kompatibilní prohlížeč. Nespouští se přímo (jako aplikace), nýbrž otevřením HTML dokumentu, kde je na něj umístěn odkaz pomocí speciální značky $< APPLET >$. Díky zabezpečení, nelze s tímto programem např. ukládat soubory na disk.

Aplikace - samostatný program, který ovšem vyžaduje pro svůj běh Java Platformu, nikoliv prohlížeč jako applet. Na aplikaci obecně nejsou kladena bezpečnostní omezení (může zapisovat do souboru apod.).

Appearance - objekt v Javě3D, který si udržuje určité informace o grafickém objektu. Mezi tyto informace patří: barva, šířka hran nebo materiál, ze kterého je grafický objekt vytvořen (tzn. vlastnosti povrchu vzhledem k osvětlení)

Avatar - pojmenování virtuálního dvojníka, který představuje uživatele uvnitř virtuálního světa. Okno prohlížeče je právě tím, co avatar vidí. Důležité jsou rozměry avatara, které zabraňují některé extrémně malé průchody.

Behavior - objekty zprostředkující interakci s uživatelem.

Borland JBuilder - vývojového prostředí pro vytváření aplikací či appletů v Javě.

BranchGroup - kořen jednotlivých částí scény v grafu scény (SceneGraph).

Capabilities - speciální capabilities bity u geometrických prvků Shape3D, které určují, jaké operace je možné s daným uzlem provádět, pokud se z něj stane živý uzel (angl. live node). K manipulaci slouží metody setCapability a getCapability.

Cortona - plug-in [1] pro internetové prohlížeče MSIE, Netscape a Mozilla, který umožňuje prohlížení 3D objektů na webových stránkách. U zobrazovaného 3D objektu lze měnit několik vlastností, lze s ním libovolně rotovat a měnit úhly pohledu apod.

compiled node - uzel je v optimalizovaném formátu díky použití metody compile na příslušnou větev grafu scény. Tato metoda zaručí nevratný proces, při kterém je dosaženo velmi rychlé zobrazování dané scény.

DEF - příkaz sloužící k redukci dat v datovém souboru VRML/X3D, kdy pojmenováním uzlu pomocí tohoto příkazu lze použít kopii (příkaz USE).

DGN (DesiGN) - základní formát výkresů v programu Microstation a rovněž použitý v této diplomové práci.

DXF (Drawing Interchange Format) - textový formát souborů výkresů AutoCADu. De-facto standard pro reprezentaci CAD dat v otevřeném (Autodeskem publikovaném) formátu. Textová podoba formátu DWG. DXF formát existuje i v komprimované binární podobě.

DWG (DraWinG) - binární formát souborů výkresů AutoCADu. De-facto standard pro reprezentaci CAD dat. Formát specifikovaný firmou Autodesk.

EAI (External Authoring Interface) - rozhraní komunikující mezi Java Appletem a programem Cortona.

FPS (Frame Per Second) - počet snímků za vteřinu.

Chisel - software [4] pro redukci objemu dat ve VRML.

IE5+ - internetový prohlížeč Internet Explorer verze 5.0 a vyšší

Inline - uzel pro vložení dalšího VRML/X3D světa nebo objektu z vnějšího souboru do aktuálního virtuálního světu VRML/X3D.

IndexedFaceSet - jedná se v jazyce VRML o množinu ploch, kde je zakomponována jak komplexní definice geometrie (seznam vrcholů, posloupnosti indexů vrcholů apod.), tak i popis vzhledu skupin ploch (seznam normál, barev, souřadnic textur atd.)

Java - pokročilý programovací jazyk umožňující vývoj aplikací pro více platforem (tzn. nezávislý na používaném operačním systému).

Java3D - API [2]je navrženo pro psaní 3D grafických aplikací a appletů. Přináší uživateli high-level konstrukty pro vytváření a manipulaci s 3D geometrií a pro konstrukci struktur používaných při renderování 3D geometrie.

Java JRE (Java Runtime Environment) - umožní běh javové aplikace či appletu v internetovém prohlížeči [10].

Java JDK (Java Development Kit) - slouží pro vývojáře při tvorbě appletů a aplikací [10].

Java Web Start - nadstavba pro spouštění aplikací Javy přímo z WWW stránek (viz. [3]).

JNLP - přípona datového souboru Java Web Start.

JVM (Java Virtual Machine) - virtuální stroj Javy k interpretaci přeložených tříd jazyka Java. Tento stroj tvoří rozhraní mezi platformově závislou části a Java bytekódem, který je platformově nezávislý.

Live node (živý uzel) - tento uzel je součástí aktivního grafu scény.

Loader - metoda v jazyce Java, která načítá danou virtuální scénu. V této práci jsem používal, a na testovacích modelech i porovnával, dva typy loaderu: Web3DLoader (z Xj3D projektu) a VrmlLoader (standardní Java3D).

MSJVM (Microsoft Java Virtual Machine) - JVM od firmy Microsoft, která již není podporována a na svých stránkách [11] doporučuje JVM od firmy Sun.

Microstation - software od firmy Bentley [5] umožňující vytvářet 3D modely objektů a budov. Pracuje s CADovskými formáty dat (např. DGN, DXF).

Prvek modelu - obecné pojmenování geometrických objektů seskupených do logické skupiny (např. vrstvy, dřevěné trámy apod.)

OpenGL - rozhraní a knihovna funkcí pro 2D a 3D grafické operace a manipulaci s objekty. Urychluje operace využitím hardwarové podpory.

Základní těleso - koule, kvádr, kužel a válec. Velikost zápisu těchto geometrických těles v jazyce VRML je minimální.

QuadArray - třída v Javě3D sloužící, k vytváření těles ze čtyřúhelníků.

Rendering - vytváření rastrového obrazu stínováním počítačového 3D modelu s využitím přiřazených barev a povrchových materiálů, s aplikací pozadí scény, světel, kamer, popř. atmosférických efektů.

Scene graph - datová struktura využívána při vytváření grafické aplikace napsané v prostředí Java3D. Má dvě hlavní větve - Viewing branch a Content branch. Zatímco pro jednoduché použití nemusí programátor se zobrazovací větví prakticky vůbec přijít do kontaktu, větev objektů - Content branch může být v rozsáhlejších případech obrovská. JAVA 3D umožňuje optimalizaci této struktury pomocí tzv. kompilace, která je nevratná.

Shape3D - objekt v Javě3D pro vytváření 3D objektů. Může odkazovat na instance podtříd abstraktní třídy Geometry a na objekty Appearance. Jeden objekt Shape3D může odkazovat i na několik objektů Geometry, ale ty musí být stejného typu.

TriangleArray - třída v Javě3D sloužící, k vytváření těles z trojúhelníků.

TriangleFanArray - třída pro vytváření trsu trojúhelníků

TriangleStripArray - třída v Javě3D sloužící, k vytváření těles z pásů trojúhelníků. Jde zde o efektivnější způsob z hlediska nároků na paměť než TriangleArray.

Viewpoint - stanoviště avatara určené souřadnicemi a směrem pohledu

Virtuální scéna - jde o virtuální model ve virtuální realitě, kde se veškeré děje provádějí v reálném čase a objekty zde mají trojrozměrný charakter.

Vizualizace - grafické zobrazení prostorových dat.

VRML (Virtual Reality Modeling Language) - jazyk popisu 3D modelů, scén a animací. Pod touto zkratkou se můžeme setkat s VRML1.0 a VRML97(VRML2.0).

VRML97 - ISO standart jazyka VRML s označením ISO/IEC 14772-1:1997.

VRMLPad - software od firmy Parallel Graphics pro editaci souborů typu VRML.

Wireframe rendering mode - drátový model pro znázornění 3D geometrie (v programu Cortona se zobrazí základní trojúhelníkové polygony)

WRZ - přípona komprimovaného souboru VRML (pomocí GZIP).

X3D (eXtensible 3D) - softwarová architektura společnosti Web3D [8] pro popis interaktivní 2D a 3D multimediálních aplikací. Zahrnuje jak deklarativní (objekty a relace mezi nimi), tak i procedurální elementy (spustitelný kód). X3D zavádí hierarchii objektů reprezentující 3D prostor grafických a zvukových objektu (tzv. scéna), která může být dynamicky modifikovaná pomocí různých mechanismů.

Xj3D - experimentální projekt (prohlížeč a nástrojová sada) společnosti Web3D [9], pracující s knihovnou Java3D. Je vyvíjen jako praktická ukázka implementace VRML97 a X3D.

X3DV - prostorová data X3D používající zápis VRML. Více na [8].

XML (eXtensible Markup Language) - univerzální definiční jazyk určený pro výměnu strukturovaných dat, pro určité typy informací lze definovat vlastní značky (tags). Pro zápis obsahu využívá Unicode.

Dodatek B

Formát souborů

Aplikace Browser načítá dva soubory - soubor se vstupní scénou a soubor s informacemi o daných grafických prvcích ve scéně. Scénu nelze nijak ukládat.

Soubor se vstupní scénou je standardní VRML soubor, který může obsahovat všechny uzly a příkazy podle normy VRML97.

Soubor s informacemi je XML soubor, který určuje, které informace o prvcích modelu budou vypsány v okně vrstev a které naopak nebudou. Musí obsahovat určité části, které jsou zvýrazněny tučně:

- **root** - název kořene
  - **title** - název popisu scény
  - date - datum publikace
  - author - autor informací
- **Object** - uzel, který je prvkem modelu
  - **ID** - jedinečné identifikační číslo
  - **Title** - název
  - **Layers1** - vrstva č.1
  - **Layers2** - vrstva č.2
  - **Layers3** - vrstva č.3
  - **Description** - popis prvku modelu
  - **SetVisible** - nastavení zobrazení v okně prvků modelu.
    - True - tento prvek modelu je viditelný
    - False - tento prvek modelu není viditelný
  - **SetEnable** - nastavení používání prvků modelu
    - True - tento prvek modelu bude použit (je na ni možné použít další funkce a informace o ní se zobrazují v informačním okně (viz. Dodatek C.3.2)
    - False - tento prvek modelu nebude použit

V tomto souboru se ukládají jednotlivé informace o prvcích modelu, přičemž v každém uzlu je vyjádřena jeho hodnota jako **VALUE** uzlu XML dokumentu. Pro větší přehlednost bych zde rád uvedl část XML dokument, která obsahuje pouze dva prvky modelu (zbytek je vypuštěn a nahrazen "..."):

```
<?xml version='1.0' encoding='utf-8'?>
<root
    title="Information about objects"
    date="3.1.2005"
    author="Daniel Nytra"
    >
    ...
    <Object>
        <ID>ID000109</ID>
        <Title>Schody - malé (pod jevištěm)</Title>
        <Layers_1>00</Layers_1>
        <Layers_2>01</Layers_2>
        <Layers_3>09</Layers_3>
        <Description>Není zde uveden popis</Description>
        <SetVisible>True</SetVisible>
        <SetEnable>False</SetEnable>
    </Object>
    ...
    <Object>
        <ID>ID000124</ID>
        <Title>Lavice na kulisy - číslo 4 (pod jevištěm)</Title>
        <Layers_1>00</Layers_1>
        <Layers_2>01</Layers_2>
        <Layers_3>24</Layers_3>
        <Description>Zavěšením kulis do posuvných rámů ....</Description>
        <SetVisible>True</SetVisible>
        <SetEnable>True</SetEnable>
    </Object>
    ...
</root>
```

Dodatek C

Uživatelská příručka

C.1 Instalace programu

Aplikaci lze spouštět prakticky třemi způsoby:

1. samostatná aplikace
2. aplikace spouštěná přes Java Web Start
3. aplikace spouštěná přes Java Web Start bez nutnosti Java3D

C.1.1 Samostatná aplikace

Pro spuštění samotné aplikace je nutné nainstalovat tyto dva programové balíky:

1. Java 2 Platform Standard Edition 5.0 (popř. 1.4) - JRE, která obsahuje JVM. Tu naleznete v [10][1].
2. Java 3D for Windows (OpenGL nebo DirectX Version) Runtime for the JRE, kterou naleznete v [2][2].

C.1.2 Aplikace spuštěná přes Java Web Start

Vzhledem k tomu, že Java Web Start umožňuje spouštět aplikaci kdekoliv z Internetu, je nutné pro samotné spuštění aplikace Browser nainstalovat tyto programy:

1. Java 2 Platform Standard Edition 5.0 - JRE, která obsahuje JVM a samotnou technologii Java Web Start. Tu naleznete v [10].
2. Java 3D for Windows (OpenGL nebo DirectX Version) Runtime for the JRE, kterou naleznete v [2].

Takto spustitelnou aplikaci naleznete na [22]. Tato aplikace se může samozřejmě posléze spustit i když uživatel není připojen k internetu (tzv. off-line verze).

C.1.3 Aplikace spuštěná přes Java Web Start bez Java3D

Jak již naznačuje nadpis, je možné tuto aplikaci spustit jen s instalací *Java 2 Platform Standard Edition 5.0 - JRE*, která obsahuje JVM a samotnou technologii Java Web Start. Tu naleznete v [10]. O knihovny Java3D se postará samotná Java Web Start, která s sebou přináší možnost instalace aktuální verze Java3D zcela automaticky. Takto spustitelnou aplikaci naleznete na [22].

[1] Velikost verze 1.5.0 Update 1 offline je 15MB.
[2] Velikost verze 1.3.1 je 4.1MB.

C.2 Parametry příkazového řádku

V této kapitole uvádím spouštění pouze samotné aplikace[3]. Ke spuštění v operačním systému Windows slouží dávka Browser.bat. Aplikace Browser má dva povinné parametry:

```
Browser.bat [VRML scéna] [XML informace]
```

VRML scéna - tento parametr udává jméno VRML souboru, který obsahuje vstupní virtuální scénu.

XML informace - tento parametr udává jméno XML souboru, který v sobě obsahuje informace o specifických prvcích modelu.

Oba dva tyto parametry jsou povinné.

C.3 Ovládání programu

Uživatelské rozhraní se skládá z nezávislých oken, u kterých si uživatel může měnit jejich umístění a velikost. Hlavní ovládáním programu je:

- pohyb ve scéně (viz. Kapitola C.3.1)
- zobrazení informací o prvcích modelu (viz. Kapitola C.3.2)
- nastavení parametrů vizualizace (viz. Kapitola C.3.2)
- měření vzdáleností a jeho vizualizace (viz. Kapitola C.3.3)

C.3.1 Základní dvojice

Základem je dvojice oken, menu a samotný prohlížeč. Vše je velmi názorně ukázáno na Obrázku C.1. Legendu k tomuto obrázku uvádím zde:

Menu - okno menu sloužící pro otevírání dalších oken s pomocnými funkcemi

Set layers (Vrstvy a informace o nich) - tlačítko pro otevření dalšího okna "setLayres", které podává informace o jednotlivých prvcích modelu.

Distance (Měření vzdáleností) - tlačítko pro otevření dalšího okna, které umožňuje měření vzdáleností

Help (Nápověda) - tlačítko pro otevření okna s nápovědou

About (O projektu) - tlačítko pro otevření okna, které obsahuje informace o celém projektu

Exit (Ukončení projektu) - tlačítko pro ukončení programu

Browser - okno prohlížeče s vizualizací virtuální scény a s tlačítkama pro pohyb v této scéně. Skládá se z *navigačního panelu* (Fly, Pan, Tilt, Walk, Examine), z *panelu viewpointů* (Předchozí viewpoint, Aktuální viewpoint, Další viewpoint, Základní viewpoint), z i*nformačního panelu* (Informace o objektech, FPS) a samotného *3D panelu* vizualizace:

Fly - pohyb bez působení přitažlivosti, je zapnuta detekce kolizí, aby se při pokusu o průchod objektem avatar zastavil.

[3]Spuštění programu přes Java Web Start probíhá pouhým kliknutím na příslušný odkaz souboru typu JNLP na stránkách WWW (viz. [22])

Pan - avatar se posunuje nahoru/dolů/doleva/doprava, je vypnuta přitažlivost i detekce kolizí.

Tilt - natočení avatara nahoru/dolů nebo doleva/doprava. Je vypnuta přitažlivost i detekce kolizí.

Walk - pohyb jako o "FLY" se zapnutou přitažlivostí. Jestliže se avatar pohybuje do schodů či po nerovném povrchu, přizpůsobuje se tomuto povrchu.

Examine - jde o zkoumání objektů, avatar není omezován přitažlivostí a je vypnuta detekce kolize.

Předchozí viewpoint - nastavení předchozího viewpointu jako aktuální.

Aktuální viewpoint - název aktuálního viewpointu.

Seznam viewpointů - seznam všech viewpointů.

Další viewpoint - nastavení následujícího viewpointu na aktivní.

Základní viewpoint - nastavení prvního (domovského) viewpointu na aktivní.

Informace o objektech - zde se podávají krátké informace o načítání celé scény.

FPS - zobrazení informace o Frame Per Second.

Při samotném spouštění programu se rovněž zobrazí okno **Browser Console**, která má pouze informativní charakter a není nutné jej využívat. Zobrazují se zde informace o načtených souborech a případných chybách při samotném načítání či používání virtuální scény.

Pohyb v 3D panelu se provádí pomocí myši a stisknutého tlačítka. V závislosti na zvoleném typu pohybu v navigačním panelu může uživatel chodit, létat, prohlížet si jednotlivé prvky scény či se posunovat.

C.3.2 Set layers - informace a nastavování parametrů

Toto okno slouží pro nastavování vlastností prvkům modelu. Dále jsou zde zobrazeny informace o nich. Přehledně vše ukazuje C.2 a zde uvádím k němu patřičnou legendu:

Okno prvků modelu - stromově uspořádané okno prvků modelu se všemi jeho zobrazenými objekty

Ovládací panel - nastavování vlastnosti viditelnosti v oknu "Brovser" (viz. Kapitola C.3.1) pro aktuální prvek modelu,

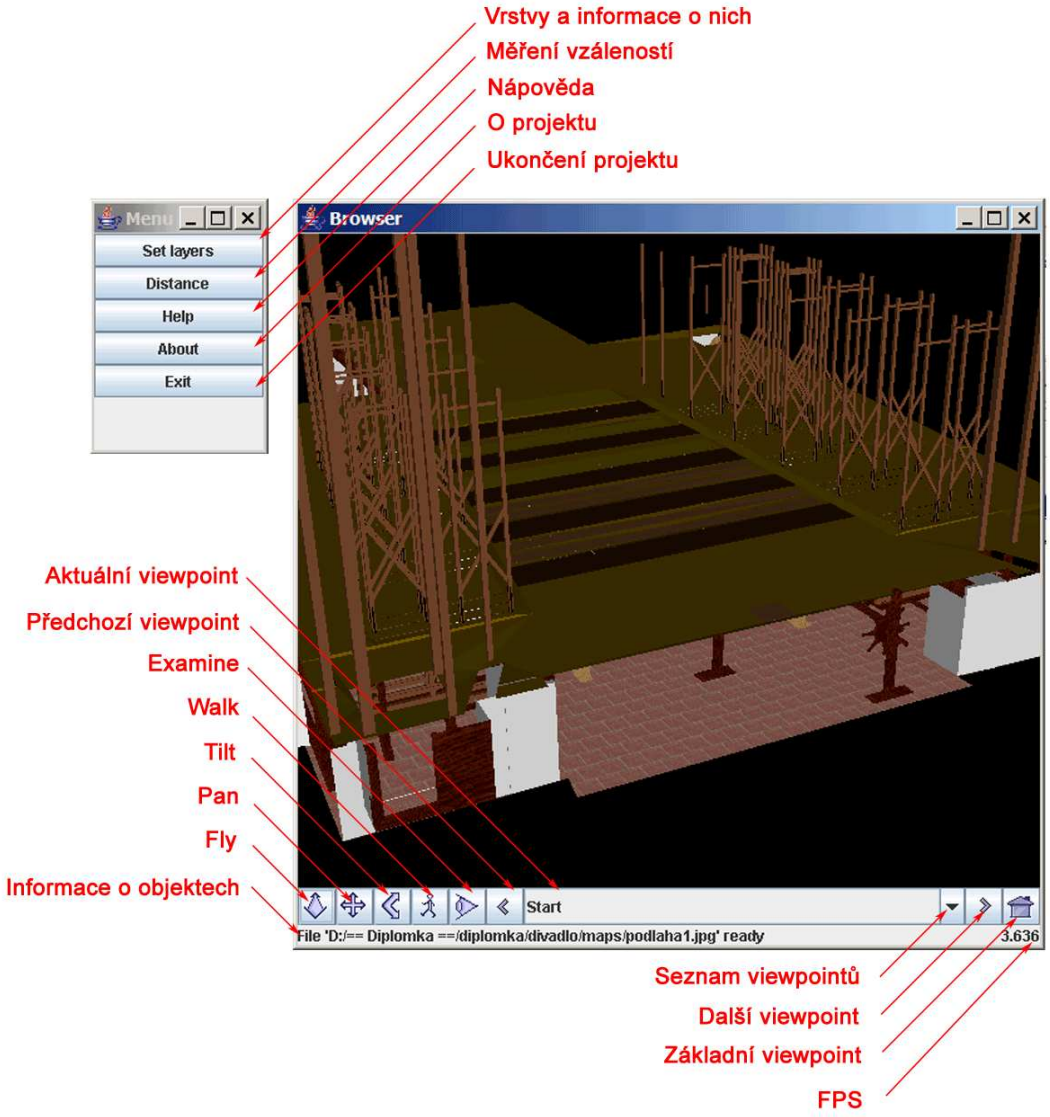
Visible - označený prvek modelu je viditelný.

Transparency - označený prvek modelu je zprůhledněn na 50

Hide - označený prvek modelu je skrytý.

Informační okno - zobrazení informací o vybraném prvku modelu

Pokud má prvek modelu v informačním souboru XML v parametru setEnable nastavenu hodnotu na FALSE (viz. Dodatek B), je pro ni nefunkční ovládací panel.



Obrázek C.1: Základní dvojice oken - "Menu" a "Browser"

C.3.3 Distance - měření vzdáleností

Toto okno zobrazuje měřené hodnoty vzdáleností dvou bodů ve scéně. Je zde možnost upravovat některé další podrobnější parametry díky tlačítka "Advanced". Měření je dostupné i uživatelům, kteří vlastní pouze jednotlačítkovou myš. Na Obrázku C.3 je vše přehledně popsáno a zde uvádím příslušnou legendu:

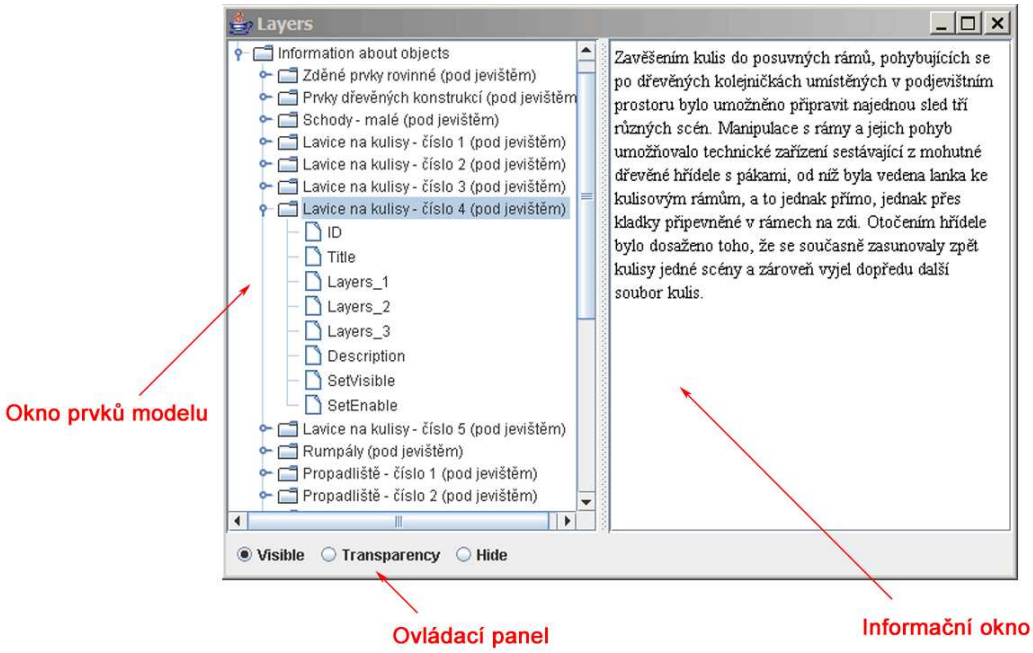
Souřadnice bodu 1 - zde se zobrazí souřadnice (x,y,z) bodu 1 po zadání tohoto bodu

Souřadnice bodu 2 - zde se zobrazí souřadnice (x,y,z) bodu 2 po zadání tohoto bodu

Distance od points (Vzdálenost bodů) - zde je zobrazena vypočítaná vzdálenost dvou zadaných bodů (až po zadání obou). Po každé změně jednoho či druhého bodu za nový se tato hodnota aktualizuje

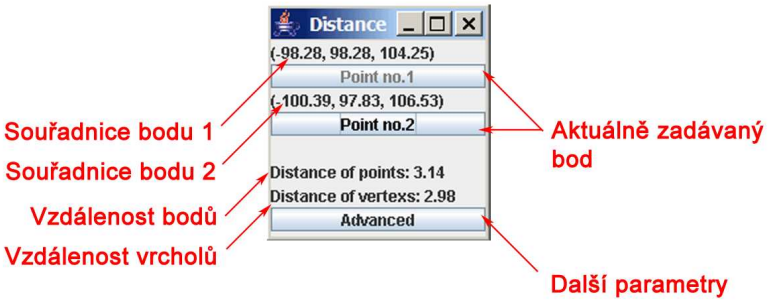
Distance of vertexs (Vzdálenost vrcholů) - stejné jako u vzdálenosti bodů, jen, že se zobrazuje vzdálenost nejbližších vrcholů zadávaného bodu

Point no.1/2 (Aktuálně zadávaný bod) - tlačítka pro změnu zadávaného bodu. Aktivní bod je vizualizován tmavější barvou písma na jeho příslušném tlačítku



Obrázek C.2: Okno "Layers"

Advanced (Další parametry) - tlačítko pro otevření okna s podrobnějšími parametry pro nastavení měření vzdáleností (viz. C.5)

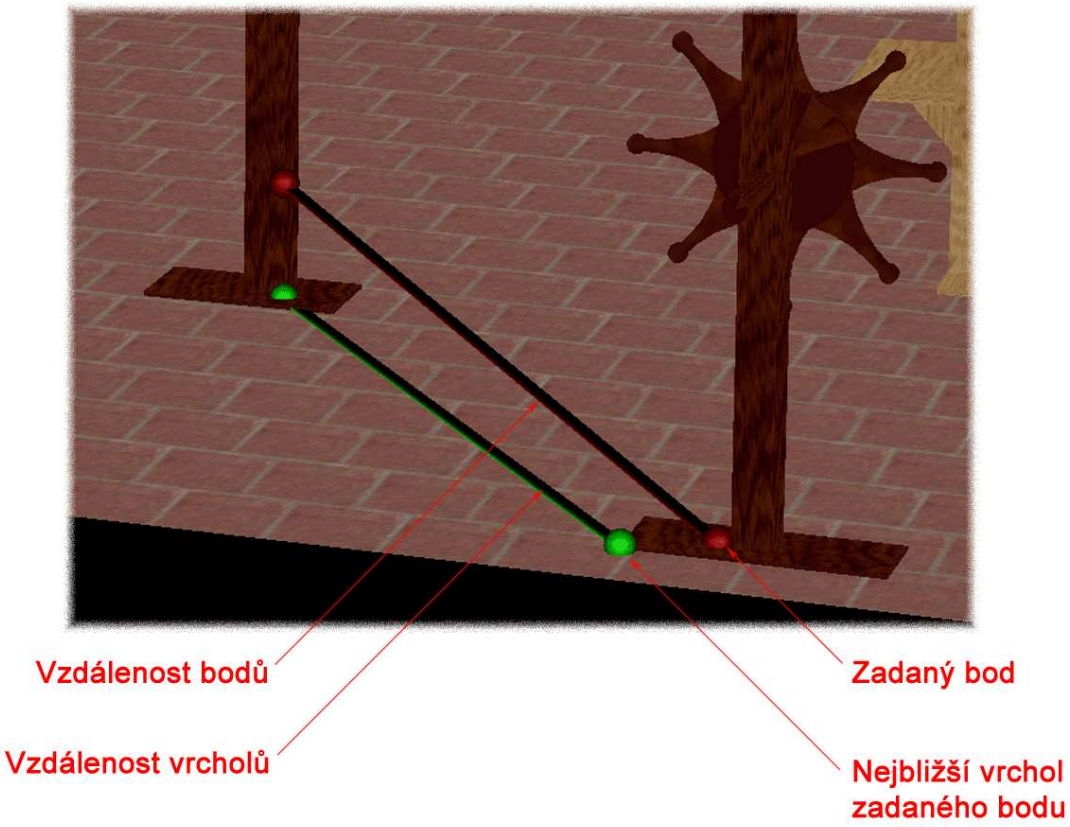


Obrázek C.3: Okno "Distance"

Postup zadávání bodů:

1. Kliknutím myši do scény zadáte první bod
   - Vizualizací zadaného bodu je červená koule (viz. Obrázek C.4), která označuje místo kliknutí na povrchu geometrického objektu a zelená koule určující nejbližší vrchol tohoto objektu
   - V okně "Distance" se zobrazí souřadnice bodu 1
2. V okně "Distance" změníte aktuální zadávaný bod (aktuálně zadávaný bod je označen na tlačítku tmavěji)
3. Kliknutím myši do scény zadáte druhý bod

- Vizualizace proběhne stejně jako u zadávání prvního bodu
- V okně "Distance" se zobrazí souřadnice bodu 2
- Vypočítají a zobrazí se v okně "Distance" se vzdálenosti mezi zadanými body a mezi zadanými nejbližšími vrcholy
- Zobrazí se vizualizace měřených vzdáleností (viz. Obrázek C.4) v závislosti na nastavených parametrech (viz. Kapitola C.3.3)



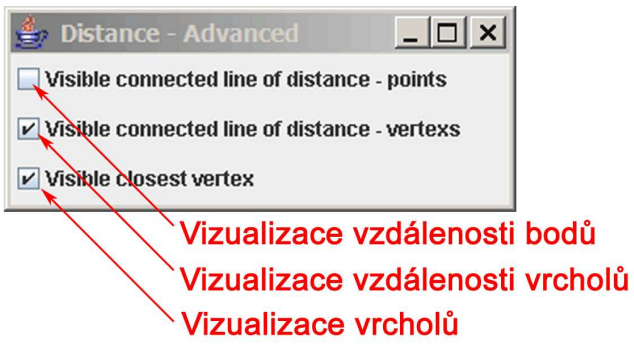
Obrázek C.4: Vizualizace měření vzdáleností v okně prohlížeče

Okno "Advanced" nebo-li "Další parametry" slouží pro nastavování parametrů vizualizace měření vzdáleností. Jeho ovládání je jednoduché a je popsáno na Obrázku C.5. Zde uvádím příslušnou legendu:

Visible connected line of distance - points (Vizualizace vzdálenosti bodů) - zobrazí/skryje vizualizaci vzdálenosti mezi zadanými body (viz. Obrázek C.4)

Visible connected line of distance - vertexs (Vizualizace vzdálenosti vrcholů) - zobrazí/skryje vizualizaci vzdáleností mezi nejbližšími vrcholy zadaných bodů (viz. Obrázek C.4)

Visible closest vertex (Vizualizace vrcholů) - zobrazí/skryje vizualizaci nejbližších vrcholů zadaných bodů (viz. Obrázek C.4)



Obrázek C.5: Okno "Advanced"

C.3.4 Help, About, Exit

Tato okna slouží zcela intuitivně:

Help (Nápověda) - otevření okna s krátkou nápovědou

About (O projektu) - otevření okna s informací o celém projektu

Exit (Ukončení projektu) - ukončení celého programu (program je možné rovněž ukončit standardním uzavřením okna Prohlížeče či okna "Menu" v rámci daného systému)

Dodatek D

Obsah přiloženého CD-ROM

Tato příloha obsahuje základní popis přiloženého CD-ROM. Podrobnější informace jako je seznam souborů nebo pokyny pro spuštění programu, jsou uvedeny v souborech radme.txt a index.html v kořenovém adresáři přiloženého CD-ROM. Vlastní CD-ROM obsahuje:

- Abstract diplomové práce v češtině a angličtině ve formátu HTML.
- Kompletní text diplomové práce.
- Program Browser ve formě tzv. univerzálního bytekódu.
- Program Browser ve formě zdrojového kódu.
- Programátorskou dokumentaci systému.
- Instalační soubory prostředí Java - JRE a Java3D
- Virtuální model barokního divadla v Českém Krumlově ve formátech VRML a X3D.
- Testovací scény.