3. Uspořádání VRML modelu

Cílem této kapitoly je navrhnout vhodnou strukturu VRML modelu, který vznikne konverzí z formátu DXF. Začnu jednoduchým popisem jazyka VRML; poté bude následovat vlastní návrh struktury VRML modelu (uspořádání souborů, způsob ovládání modelu, opatření pro rychlejší zobrazování atd.) a v poslední části se ve stručnosti zmíním o některých dalších vlastnostech, které by měl virtuální model v budoucnu mít (například možnost měření délek v modelu apod.).

3.1 Stručný popis formátu VRML

V této kapitole chci zhruba nastínit, jak pracuje formát VRML. Jde mi pouze o vystižení základního principu jazyka, aby mohl být další text vůbec srozumitelný; v žádném případě tedy nemůže tato kapitola nahradit práci věnovanou výhradně VRML. K některým podrobnostem jazyka se vrátím později v místech, kde to bude nutné, nicméně pro detailnější popis VRML doporučuji nahlédnout do speciální literatury, např. [6] nebo [7].

Po krátkém úvodu se nejprve budu věnovat programovým nástrojům, které jsou třeba, chceme-li pracovat s VRML. Poté nastíním základní strukturu VRML a na praktické ukázce se pokusím vysvětlit jeho princip. Další část se bude zabývat vzhledem těles ve VRML, tj. mapováním materiálů a textur na povrch těles, a na závěr uvedu několik dalších prvků VRML, důležitých pro tvorbu prostorových modelů.

3.1.1 Formát VRML obecně

VRML – **V**irtual **R**eality **M**odelling **L**anguage – je jazyk určený pro prezentaci virtuální reality v prostředí počítačových sítí. Byl vytvořen ve spolupráci několika firem zabývajících se počítačovou grafikou (iniciátorem byla firma *Silicon Graphics, Inc.*).

Poprvé byl jazyk VRML publikován v roce 1995 pod názvem VRML 1.0. Současně vznikla skupina programátorů nazvaná *VAG* (*VRML Architecture Group*), která pracovala na jeho zdokonalení; na práci se podílela i široká odborná veřejnost. Skupina VAG se posléze změnila v oficiální *VRML Consortium, Inc.* a začala spolupracovat i s mezinárodní standardizační komisí ISO.

Výsledkem práce je nová verze jazyka, nazvaná VRML 97. Tato verze je dodnes aktuální a i my se jí budeme v této práci zabývat.

VRML je **otevřený formát**. Jeho zdrojový kód je zapsán v prostých textových souborech, které mohou (ale nemusí) být kvůli úspoře dat při přenosu po síti komprimovány programem *gzip* (prohlížeče VRML v tomto případě soubor automaticky dekomprimují).

3.1.2 Nástroje pro práci s VRML

3.1.2.1 Prohlížeče VRML

Protože je jazyk VRML určen pro práci s virtuálními světy ve webovém prostředí, jsou prohlížeče koncipovány jako plug-ins webových prohlížečů. Bývají zdarma ke stažení na stránkách výrobců. Uveďme některé z nich:

Cortona Client (výrobce *Parallel Graphics, Inc.*)
CosmoPlayer (výrobce *Silicon Graphics, Inc.*)
WorldView (výrobce *Intervista Software, Inc.*)
Blaxxun Contact (výrobce *Blaxxun Technologies*)

Ovládání prohlížeče je poměrně úzce vázáno na samotný jazyk VRML (chování prohlížeče lze ve VRML souboru ovlivnit), a proto je u všech prohlížečů dosti podobné. Doporučuji nahlédnout například do návodů na stránkách výrobců.

3.1.2.2 Editory VRML

Editory jazyka VRML můžeme rozdělit podobně jako například editory HTML na dvě základní skupiny: na editory textové a grafické.

Grafické (WYSIWYG) editory umožňují tvorbu virtuálních světů přímo na obrazovce a zdrojový VRML kód vytvářejí automaticky. Uživatel nemusí být se samotným jazykem skoro vůbec obeznámen – ovšem s tou výhradou, že mu editor takto "vnutí" výslednou podobu kódu, což u rozsáhlejších modelů může být na závadu. Jako příklad grafického editoru uveďme ***Internet Space Builder*** od firmy *Parallel Graphics, Inc.*

Textové VRML editory jsou podobné jednoduchým textovým editorům (Notepad...) s tím rozdílem, že umožňují strukturované zobrazení zdrojového kódu s barevným odlišením různých jeho částí; některé jsou též schopny zobrazit mimo samotný kód i stromovou strukturu modelu (viz dále). Příkladem může být ***VRML Pad***, produkt již zmíněné firmy *Parallel Graphics, Inc.*

3.1.2.3 Další nástroje

Další aplikace pracující s VRML jsou určeny především pro redukci objemu dat, která je nutná v zájmu urychlení zobrazování virtuálního modelu. Tyto **optimalizátory** pracují různými způsoby, umožňují například automatické snížení počtu polygonů aproximujících zakřivené plochy apod.

3.1.3 Struktura VRML souboru

Základním stavebním prvkem jazyka VRML je **uzel (node)**. Uzel si můžeme představit jako jistou analogii k pojmu "objekt" v programování; je to základní jednotka, která může představovat prakticky cokoli, od graficky zobrazovaného objektu (tělesa) přes zdroje zvuku, funkce pro výpočet interpolace apod.

Název "uzel" nevznikl náhodou. Jednotlivé uzly totiž nestojí ve VRML souboru zcela samostatně, nýbrž jsou uspořádány do **stromové struktury**. Ta umožňuje, aby nastavení tzv. **rodičovského uzlu** (uzlu, který stojí v hiearchii výše) ovlivňovalo chování všech jeho **potomků**, tedy uzlů stojících níže. Zmíněné nastavení se provádí v tzv. **parametrech** uzlu. Nejnázornější bude ukázat si praktický příklad:

Řekněme, že chceme zobrazit kvádr, který bude mít velikost stěn postupně 2, 3 a 4 metry a bude umístěn na nějaké požadované pozici. Začneme popisem kvádru sama o sobě:

```
Shape {
      geometry Box {size 2 3 4}
      }
```

Uzel `Shape` slouží k popisu libovolného zobrazitelného objektu (tělesa, plochy). Jedním z jeho parametrů je parametr `geometry`, který udává tvar tělesa; hodnotou parametru je zde uzel `Box` o uvedených rozměrech, tedy kvádr. Vidíme tedy, že hodnotou parametru může být opět uzel.

Dále ovšem chceme kvádr umístit na nějakou námi požadovanou pozici a natočit jej do potřebné

polohy. To provedeme tak, že uzel Shape učiníme potomkem uzlu Transform:

```
Transform {
      translation 6.2 2.4 3.7
      rotation 0 1 0 0.7
      children Shape {
                  geometry Box {size 2 3 4}
                        }
            }
```

Vidíme, že uzel Transform (který obecně slouží k manipulaci s jinými uzly – svými potomky) má zde uvedeny parametry translation (posun do požadovaného místa) a rotation (otočení; první tři souřadnice určují osu a čtvrtá hodnota rotační úhel v radiánech).

Výhodu tohoto uspořádání, kdy uzel Shape je potomkem uzlu Transform, který mění jeho pozici, natočení apod., plně doceníme, chceme-li vytvořit objekt složený z více uzlů, např. stůl. Stůl můžeme sestavit z jednoho plochého kvádru (deska) a čtyř dlouhých tenkých kvádrů (nohy). Zápis ve VRML by mohl vypadat takto:

```
Transform {
      translation 8.5 10.0 3.5
      rotation 0 0 1 0.47
      children [
            Transform {
                  translation 1 0 0.9
                  children Shape {geometry Box {size 2 1 0.05}}
            Transform {
                  translation 2.0 1.0 0.45
                  children Shape {geometry Box {size 0.05 0.05 1.0}}
            Transform {
                  translation 2.0 0.0 0.45
                  children Shape {geometry Box {size 0.05 0.05 1.0}}
            Transform {
                  translation 0.0 1.0 0.45
                  children Shape {geometry Box {size 0.05 0.05 1.0}}
            Transform {
                  translation 0.0 0.0 0.45
                  children Shape {geometry Box {size 0.05 0.05 1.0}}
                  ]
            }
```

Vidíme, že nejvýše položený uzel `Transform` má pět potomků, které představují opět uzly `Transform`. Zápis do takovéto stromové struktury umožňuje jednu velmi důležitou věc: provedeme-li v rodičovském uzlu libovolnou transformaci (změnou parametrů `translation`, `rotation`, `scale` apod.), ovlivní tato transformace všechny potomky zároveň, aniž by jakkoli pozměnila geometrické vztahy mezi nimi. To znamená, že můžeme nejprve sestavit stůl sám o sobě, a sice pomocí správného nastavení parametrů `translation`, `rotation` apod.) v jednotlivých potomcích, a teprve potom vzniklý celek umístíme a natočíme podle potřeby (nastavením parametrů v uzlu `Transform` o úroveň výš). Kdybychom naopak chtěli zachovat umístění a natočení stolu, ale zároveň změnit jeho podobu, můžeme provést změny pouze v parametrech potomků a poloha celku zůstane stejná. (Zde vidíme, že analogie k pojmu "objekt" v objektově orientovaném programování je výstižná, protože VRML uzly splňují základní požadavky kladené na objekty: jsou hiearchicky uspořádané a potomci dědí vlastnosti rodičovských uzlů, není-li v parametrech potomků uvedeno jinak.)

Stromová struktura VRML tedy umožňuje seskupit jednotlivé uzly do většího celku, s kterým poté lze manipulovat jako s jediným uzlem. Toto seskupování se děje pomocí tzv. **skupinových uzlů**, mezi něž patří i náš uzel `Transform`, který je z nich zřejmě nejpoužívanější. V našem případě také často využijeme uzel `Group`, který nemá jinou funkci než právě seskupení uzlů (tj. neumožňuje žádné další operace se skupinou svých potomků).

3.1.4 Vzhled těles ve VRML

Zatímco v předchozí kapitole jsme se věnovali geometrickému tvaru těles ve VRML, nyní pojednáme alespoň stručně o jejich povrchu, tj. o tom, jak se ve VRML používají materiály a textury.

Povrch tělesa je definován v parametru uzlu `Shape` (viz předchozí ukázka), nazvaném `appearance`. Jak už bylo naznačeno v kapitole 2.3.4, lze ve VRML definovat povrch tělesa dvojím způsobem: jako **materiál** a jako **texturu**. Materiálem se rozumí definice barevných vlastností povrchu (vlastní barva, odražená barva...), zatímco textura je zpravidla představována obrázkovým souborem. Opakuji, že nepříliš vhodný název "materiál" zde používám proto, aby byl text v souladu s terminologií VRML.

Materiál (= barva povrchu): předveďme si zápis kvádru z předchozí ukázky, ovšem s červeným povrchem:

```
Transform {
      translation 6.2 2.4 3.7
      rotation 0 1 0 0.7

      children Shape {
                  appearance Appearance {
                        material Material {
                              diffuseColor 1.0 0.0 0.0
                                    }
                              }
                  geometry Box {size 2 3 4}
                  }
            }
```

Barva tělesa je určena parametrem `material` uzlu `Appearance`. Nastavení vzhledu povrchu je možno upravit dalšími parametry kromě uvedeného `diffuseColor`, například `ambientIntensity` (vliv celkového jasu prostoru na barvu tělesa, `specularColor` (barva tělesem odrážená) atd.

Textura: textura umožňuje dodat tělesu mnohem realističtější vzhled než pouhá barva povrchu. VRML nabízí celkem tři druhy textur: obrázek, ornament a film. Ukažme si opět stejný kvádr, ovšem s texturou z obrázkového souboru *textura.jpg*.

```
Transform {
      translation 6.2 2.4 3.7
      rotation 0 1 0 0.7

      children Shape {
                  appearance Appearance {
                        texture ImageTexture {
                              url "textura.jpg"
                                    }
                              }
                  geometry Box {size 2 3 4}
                  }
            }
```

Textury jsou definovány v parametru `texture` uzlu `Appearance`. Chceme-li jako texturu použít ornament resp. film, využijeme místo uzlu `ImageTexture` uzel `PixelTexture` resp.

MovieTexture).

Uzel Appearance má ještě jeden parametr, a sice textureTransform; ten slouží k nastavení způsobu, kterým se textura bude mapovat na těleso (natočení, měřítko). Pokud totiž tento parametr není zadán, mapuje se textura automaticky, což není vždy vhodné – například u našeho kvádru se nanese na každou jeho stěnu právě jednou a její tvar se zdeformuje podle tvaru stěny kvádru. V parametru textureTransform je možno např. nastavit, aby se obrázek nijak nedeformoval, aby se opakoval (dlaždice) apod.

3.1.5 Další důležité prvky VRML

O uzlech popisujících tvar těles pojednám podrobně později. Nyní se stručně podívejme na některé důležité doplňky VRML modelů. K většině těchto uzlů a konstrukcí se též ještě vrátím.

Pojmenované struktury: libovolné části VRML modelu (samotnému uzlu nebo celé stromové struktuře uzlů) můžeme příkazem DEF přiřadit souhrnný název. Pomocí příkazu USE můžeme takto definovaný uzel nebo strukturu znovu použít. Omezení spočívá v tom, že příkazy DEF a USE musí být použity v tomtéž VRML souboru. Pro zobecnění objektů je proto výhodnější použít níže zmíněné prototypy; s konstrukcí DEF/USE se ale přesto v modelu můžeme setkat, protože některé uzly (manipulátory ap.) vyžadují, aby objekty, s nimiž pracují, byly takto pojmenovány.

Prototypy: VRML umožňuje definovat zcela nový objekt, který se bude chovat velmi podobně jako běžný VRML uzel. Do konstrukce PROTO zapíšeme libovolný uzel nebo stromovou strukturu uzlů a definujeme, které jejich parametry bude možno měnit. Do vlastního modelu pak prototyp vložíme příkazem EXTERNPROTO, v němž uvedeme hodnoty měnitelných parametrů. Máme tak možnost vytvořit prototyp a vkládat jej do modelu v různých obměnách.

Vkládání dalších VRML souborů: do virtuálního modelu můžeme pomocí uzlu Inline vložit jiný VRML soubor. To nám umožňuje rozdělit celý model do více souborů nebo třeba propojit několik virtuálních světů umístěných na různých stanicích v síti.

Odstupňovaná podrobnost těles (LOD): zkratka LOD znamená *level of detail*; protože při pohledu z dálky nevidíme tělesa detailně, nemá smysl nutit VRML prohlížeč k výpočtům zbytečných podrobností. V uzlu LOD můžeme proto definovat tzv. náhradní reprezentaci tělesa, která je jednodušší a při pohledu z dálky těleso může nahradit. Spolu s ní též určíme, v jaké vzdálenosti od tělesa se má místo jeho původní podoby zobrazit tato náhradní reprezentace. Podrobněji viz 3.3.1.

Manipulátory: manipulátor je uzel, kterým můžeme vybavit libovolné těleso; výsledkem je, že návštěvník může toto těleso uchopit kurzorem myši a pohnout s ním.

URL odkazy: do modelu je možno vložit odkazy na libovolné URL (uniform resource locator). Děje se tak pomocí uzlu `Anchor`, který způsobí, že kliknutí myší na kterýkoli z potomků přenese návštěvníka na místo definované v parametru `url`. Tímto místem může být jiné místo v témže VRML souboru, místo v jiném VRML souboru, ale také jakýkoli soubor nebo webová adresa. Tento uzel lze tedy s výhodou použít pro teleportaci v rámci virtuálního světa nebo pro zobrazení podrobností o daném tělese třeba v textové podobě.

Zpracování dynamických událostí: VRML dokáže provádět i dynamické události. Spuštění událostí se děje pomocí čidel.

Skripty: do VRML souboru lze vkládat skripty (Java, ECMAScript), pomocí kterých je možné naprogramovat zejména složitější dynamické události, na které již výše zmíněné uzly nestačí.

3.2 Základní charakteristiky VRML modelu

Následující odstavce budou věnovány vlastnímu obsahu VRML modelu. V první části uvedu výčet všech VRML uzlů, které lze použít pro geometrické zobrazení těles; dále pojednám o způsobu dělení modelu do vrstev, o zobecňování opakujících se prvků pomocí prototypů a na závěr o ovládání celého modelu.

3.2.1 Grafické VRML uzly

Začněme stejným způsobem jako při studiu formátu DXF, tj. vyjmenujme si všechny prostředky (uzly), jak ve VRML vyjádřit nějaký geometrický tvar (viz tabulka 3.1 na následující straně). Jsou to čtyři uzly pro základní tělesa (kvádr, koule, kužel, válec), dále uzel pro obecný útvar složený z ploch, uzel pro translační nebo rotační tělesa a nakonec kvadratická b-spline plocha.

Box, Sphere, Cone, Cylinder
Základní tělesa; jejich zadání je víceméně intuitivní a nebudeme je zde podrobně vypisovat.
IndexedFaceSet
Skupina rovinných ploch o libovolném počtu vrcholů; lze z nich "slepit" jakékoli těleso o rovinných stěnách; uzel obsahuje nejprve seznam souřadnic bodů, po němž následuje seznam jednotlivých ploch, definovaných pomocí odkazů na body v tomto seznamu. Parametr `creaseAngle` umožňuje optické vyhlazení hran na styku ploch; uzel proto lze s úspěchem použít na zápis trojúhelníkových sítí, kterými aproximujeme obecně zakřivenou plochu.
Extrusion
Popisuje těleso vzniklé translací zadaného rovinného profilu podél definované trajektorie; profil je možno během translace otáčet a měnit jeho měřítko. Určitým omezením je ale fakt, že profil lze definovat pouze jako lomenou čáru složenou z úseček, tj. přejeme-li si, aby základní profil obsahoval oblouk, musíme jej aproximovat úsečkami. Opět máme možnost optického vyhlazení pomocí parametru `creaseAngle`.
NurbsSurface
B-spline plocha 2.řádu; vhodné pro definici ploch, u kterých nám nestačí aproximace trojúhelníkovou sítí. Tyto plochy jsou poměrně náročné na výpočty zobrazování, proto je rozumné jimi v modelu šetřit a kde je to jen trochu možné, použít opticky vyhlazenou lomenou plochu.

Tab. 3.1: VRML uzly vyjadřující tvar objektu

3.2.2 Rozdělení modelu do vrstev

Má-li prostorový model sloužit k podrobnějšímu studiu budovy, musí mít uživatel možnost nechat si zobrazit ty jeho části, které momentálně potřebuje. Z tohoto důvodu je třeba, aby byl model rozdělen do vrstev (podobně jako CAD výkresy).

VRML nabízí dva základní způsoby, jak provést rozvrstvení modelu:

První způsob je instinktivní: spočívá v prostém **rozdělení jednotlivých prvků modelu do různých souborů**, které se pak při vykreslení modelu mohou a nemusí zobrazit. Daný soubor zahrneme do celkového modelu pomocí uzlu `Inline`. Členění je přehledné a v mnoha ohledech praktické (např. pokud jde o rychlost zobrazování modelu), jak ještě uvidíme.

Druhý způsob je také jednoduchý: jak už bylo řečeno, struktura jazyka VRML umožňuje (ne-li

přímo vynucuje) **seskupování uzlů do větších celků** (uzel `Group`, případně `Transform`). Vytvoříme-li takové skupiny vhodně, můžeme je použít jako vrstvy, tj. manipulovat se všemi jejich členy hromadně.

Z čistě praktického hlediska shledáváme, že první způsob členění modelu je vhodnější, nebo přinejmenším pohodlnější. Uspořádání modelu do více souborů je z hlediska tvůrce modelu mnohem přehlednější a manipulace s uzlem `Inline`, kterým obsloužíme celý soubor naráz, je určitě jednodušší než postupné vyhledávání seskupených uzlů v jednom velkém VRML souboru. V případě modelu umístěného na webu přispěje použití uzlu `Inline` ke zrychlení jeho načítání, protože každý z jednotlivých VRML souborů se bude po síti přenášet teprve ve chvíli, kdy to bude nutné, zatímco při použití druhého způsobu rozvrstvení bude třeba nejprve přenést a načíst celý model.

VRML model proto bude **rozvrstven pomocí rozdělení dat do více VRML souborů**, z nichž každý bude odpovídat jedné vrstvě.

3.2.3 Prototypy – zobecnění opakujících se objektů v modelu

Dá se předpokládat, že v prostorovém modelu budovy se budou některé objekty opakovat, ať už v podobě vůbec nezměněné, nebo alespoň snadno odvoditelné z nějakého originálu. Vyskytuje-li se tentýž objekt v modelu mnohokrát, je rozumné uspořit objem dat tím, že jeho zdrojový kód v plné podobě zapíšeme pouze jednou, zatímco v ostatních místech výskytu použijeme pouhý odkaz na tento plný zápis, jinak nazývaný **prototyp**.

Prototypy se definují pomocí konstrukce **`PROTO`**, jednoho z nejmocnějších nástrojů VRML, který v podstatě umožňuje překročit hranice tohoto jazyka a vytvořit "nad ním" nový jazyk, šitý přesně na míru našim potřebám. Zjednodušeně řečeno, tento uzel odpovídá definici funkce (procedury) v různých programovacích jazycích; slouží k tomu, abychom si mohli vytvořit vlastní objekt (uzel), u něhož bude definováno, které parametry lze měnit a které ne.

Vraťme se k příkladu se stolem v kapitole 3.1.3. Stůl je tvořen jedním plochým kvádrem (deska) a čtyřmi dlouhými kvádry (nohy). Těchto pět prvků bude definováno v rámci konstrukce `PROTO` v nějakém VRML souboru. Jako měnitelné parametry můžeme nastavit např. rozměry desky (= rozměry plochého kvádru), délku nohou (= výšku tenkých kvádrů), barvu apod. Dejme tomu, že v budově je množství různě velkých, různě vysokých a různě barevných stolů. Bez použití konstrukce `PROTO` bychom museli každý z nich definovat znovu (tj. jeden plochý a čtyři dlouhé kvádry);

jestliže ovšem konstrukci PROTO použijeme, pak můžeme stůl do libovolného místa vložit příkazem EXTERNPROTO, v němž zadáme námi požadované hodnoty parametrů (rozměr desky, barvu apod.) a stůl bude vypadat přesně tak, jak požadujeme.

Konstrukce PROTO nám umožňuje umístit definice prototypů do nezávislého VRML souboru. A protože je pravděpodobné, že prototypů bude v modelu více, můžeme vytvořit **knihovnu prototypů**, tj. VRML soubor obsahující definice všech prototypů, na něž se poté budeme v místech výskytu daných objektů odkazovat.

Podrobné informace o použití konstrukce PROTO a EXTERNPROTO lze nalézt např. v [7], [11].

3.2.4 Ovládání modelu – řídící aplikace

V předchozím textu byla navržena struktura VRML modelu; nyní zbývá vyřešit podstatnou otázku, jakým způsobem se bude celý model ovládat. Zdůrazňuji, že ovládání modelu není vlastním námětem této diplomové práce; následující text proto není ničím více než zběžným návrhem, jak by model mohl být řízen. Tyto návrhy proto ani nejsou zahrnuty do návrhu konverzní aplikace v kapitole 4 (samotná konverze modelu tím nijak neutrpí).

Víme již, že máme k dispozici celou řadu VRML prohlížečů, které poskytují základní ovládací prvky potřebné k práci s modelem: procházení modelu v různých režimech, podrobné studium objektů apod. Zásadním nedostatkem dostupných VRML prohlížečů je ovšem nepřítomnost některých funkcí, které pro účely prostorového informačního systému budeme potřebovat: jde především o možnost práce s vrstvami modelu (vrstva je totiž pojem, který jsme převzali z CAD systémů; jazyk VRML sice mimoděk poskytuje možnost model rozdělit do vrstev, ale sám o sobě žádný takový pojem nezná a nepracuje s ním). Dále zde chybí **navigátor**, tj. názorný přehled o celém modelu a návštěvníkově pozici v něm. Kromě toho je pravděpodobné, že budeme chtít vybavit model i dalšími funkcemi, např. nástrojem pro měření délek. Z těchto důvodů bude praktické vytvořit vlastní řídící prvek modelu.

Nabízejí se dvě možnosti: vytvořit **externí řídící aplikaci** nebo ovládací prvek **integrovaný přímo do VRML modelu**. Výhodou externí aplikace je bezesporu fakt, že neexistuje žádné podstatné omezení jejích možností. Na druhou stranu, jazyk VRML nabízí dosti jednoduchý a efektivní způsob, jak vytvořit ovládací panel coby součást VRML modelu (lze jej totiž sestavit z prvků zachovávajících konstantní polohu a velikost a vybavených čidly dotyku, kterými se budou spouštět různé požadované dynamické události). To znamená, že máme možnost získat prostorový

informační systém včetně ovládacích prvků jako jediný soudržný celek ve formátu VRML.

Zdrojový kód ovládacího panelu vytvořeného ve VRML umístíme do souboru, který nazývám **řídícím souborem**; tento soubor se při každém otevření modelu načte jako první a zůstane načten po celou dobu práce s modelem, aby byla zajištěna neustálá přítomnost ovládacího panelu v okně s modelem.

Této vlastnosti řídícího souboru (totiž opakované načítání) je možné využít též pro manipulaci s vrstvami. Zobrazení VRML souboru (vrstvy) se děje prostřednictvím uzlu `Inline`. Uzly `Inline` pro vrstvy, které mají být zobrazeny, budou zapsány v řídícím souboru. Změní-li uživatel schéma zobrazení vrstev, vyšle tím příkaz k přepsání této části řídícího souboru, který se poté znovu načte – tímto krokem se znovu načte celý model se zobrazením vrstev obsažených v uzlech `Inline`. Současně s uzly `Inline` budou do řídícího souboru zapsány i informace o skupinách vrstev (jako komentáře, tj. bez vlivu na práci prohlížeče – budou s nimi pracovat pouze funkce pro zobrazování vrstev).

Podobným způsobem jako ovládací panel, tj. přímo v jazyce VRML jako součást řídícího souboru, bude vytvořen **navigátor**. Navigátor může mít formu malého dvourozměrného obrázku schématicky zobrazujícího půdorys budovy, na kterém se bude pohybovat výrazná značka ukazující polohu návštěvníka (poloha se určí přepočtem souřadnic návštěvníka v rámci modelu na souřadnice v rámci navigačního obrázku). Navigační obrázek vytvoří sám uživatel (například sejmutím pohledu shora na DXF model v CAD systému) a vloží jej do modelu při konverzi.

Poznámka: je možné, že rostoucí požadavky na prostorový informační systém si časem vynutí vytvoření externí řídící aplikace; způsob řízení modelu pomocí ovládacích prvků a navigátoru vytvořeného přímo ve VRML, popsaný v předchozích odstavcích, představuje “minimální konfiguraci”, která by měla dostát požadavkům běžného uživatele, ale která nemusí stačit na náročné operace při podrobném studiu virtuálního modelu.

3.3 Opatření pro rychlejší práci s VRML modelem

V kapitole 1.2.2.3 jsem uvedl několik způsobů, jak uspořit objem dat a zobrazovací výpočty a urychlit tak práci s virtuálním modelem. Zde podrobněji pojednám o třech z nich (ostatní jsou záležitostí konstrukce modelu v CAD systému, nikoli jazyka VRML).

3.3.1 Odstupňovaná podrobnost zobrazení – uzel LOD

Protože při pohledu z dálky nerozeznáváme tělesa, textury apod. do všech podrobností, je zbytečné zatěžovat počítač vykreslováním všech detailů vzdálenějších předmětů a zpomalovat tak vykreslování modelu; proto VRML umožňuje stanovit pro libovolný zobrazovaný objekt více reprezentací různého stupně podrobnosti. Například kužel lze pro pohled z určité vzdálenosti nahradit jehlanem, v ještě větší vzdálenosti trojúhelníkem apod. V praxi to vypadá tak, že zápis tělesa ve VRML souboru nestojí samostatně, ale je potomkem uzlu `LOD`; jeho sourozenci jsou právě zápisy jednodušších reprezentací tělesa. V uzlu `LOD` dále stanovíme vzdálenosti, v nichž se má přecházet mezi jednotlivými reprezentacemi. Zavádění náhradních reprezentací sice zvětšuje objem dat (prodlužuje zdrojový kód VRML), ale urychlení výpočtů při zobrazování jednodušších objektů je tak markantní, že se uzel `LOD` doporučuje používat zcela běžně. Úplný popis uzlu `LOD` viz přílohu č. 5.

Poznámka: **textury** se doporučuje použít pouze u vlastního tělesa; u jeho náhradních reprezentací je nahradíme vhodnou barvou povrchu (materiálem).

3.3.2 Načítání pouze viditelných částí modelu

Je zbytečné nutit prohlížeč, aby se zabýval objekty, které nejsou v daném pohledu momentálně vidět. Uzel `Inline` má z tohoto hlediska výhodnou vlastnost, že k reálnému načtení souboru dojde teprve ve chvíli, kdy návštěvník může jeho obsah vidět. Při prvotním načtení modelu se tedy načítají pouze ty jeho části (= vrstvy), které návštěvník vidí; zobrazení modelu je potom samozřejmě rychlejší, než kdybychom nejprve čekali na načtení všech jeho částí. Ani výpočet viditelnosti ovšem není jednoduchý, a proto lze uzel `Inline` doplnit tzv. obálkou (`bbox`), což je oblast ve tvaru neviditelného kvádru obklopujícího daný VRML soubor; výpočet viditelnosti se pak provádí pro tento jednoduchý kvádr namísto tvarově složitého souboru.

3.3.3 Omezení dohlednosti (mlha)

Pomocí uzlu `Fog` zavedeme do virtuálního modelu tzv. mlhu, která způsobí postupné mizení viditelných objektů, které při překročení zadané maximální vzdálenosti od návštěvníka (parametr

visibilityRange) zmizí úplně. Mizení objektů je realizováno jako jejich postupné překrývání barvou mlhy (parametr color).

3.4 Doplňkové vlastnosti modelu – návrhy do budoucna

V předchozím textu jsem se zabýval vlastním námětem této práce, totiž geometrickým exportem dat do formátu VRML. Aby se z takto vzniklého modelu stal prostorový informační systém, musí být následně vybaven dalšími prvky a funkcemi. V této kapitole předkládám několik návrhů.

3.4.1 Doplňující data v jiných formátech

Návštěvník virtuálního modelu sice jednotlivé objekty vidí, ale jiné informace o nich zjistit nemůže, pokud mu neumožníme zobrazit si doplňující data. Objekty v modelu, které stojí za pozornost a podrobnější studium, proto můžeme vybavit odkazy (uzel Anchor) na další soubory, ať už textové, obrazové, animační, filmové nebo zvukové. Takto lze například zobrazit stručnou historii zajímavého objektu, objasnit funkci složitého zařízení pomocí animace apod.

3.4.2 Interaktivní prvky

Interaktivními prvky se rozumí zejména možnost skokového přemístění (teleportace) na jiné místo v rámci modelu, a dále nástroje pro manipulaci s objekty modelu (např. otevírání dveří nebo – v případě barokního divadla – spuštění výměny kulis). Pro tyto účely poskytuje jazyk VRML různá čidla, manipulátory, interpolátory apod.

Manipulátory slouží k tomu, aby návštěvník mohl pohnout tělesem. Jsou celkem tři: uzel CylinderSensor převádí pohyb kurzoru na pohyb po plášti válce, uzel PlaneSensor na pohyb v rovině a uzel SphereSensor na pohyb po povrchu koule. K manipulátorům se řadí ještě uzel TouchSensor, který ovšem neumožňuje pohyb tělesa, pouze detekuje, zda nad tělesem přejel kurzor myši nebo zda nad tělesem uživatel stiskl tlačítko – používá se zejména ke spouštění **dynamických událostí**. Ty ovšem lze spustit nejen "dotykem", ale i pouhou přítomností návštěvníka na určitém místě (ProximitySensor) nebo spatřením nějakého objektu (VisibilitySensor). Ke zpracování událostí jsou třeba další uzly, zejména časovač (TimeSensor) a interpolátory (ColorInterpolator, PositionInterpolator atd.)

Pomocí těchto uzlů lze popsat různé dynamické akce, např. otevírání a zavírání dveří nebo pád tělesa na zem. Po spuštění události začne běžet časovač a interpolátor dodává v daných časových intervalech hodnoty proměnných, např. úhlu pootevření u dveří.

3.4.3 Možnost podrobného studia modelu

Pokud má virtuální model sloužit k podrobnějšímu studiu budovy, je třeba jej doplnit o některé další funkce, zejména o nástroj na měření délek či objemů v modelu.

3.5 Shrnutí – co obsahuje VRML model

Zde je stručný přehled navrhované struktury výsledného VRML modelu, jak jsem ji navrhl v této kapitole:

(a) použité grafické VRML uzly: `Box`, `Sphere`, `Cone`, `Cylinder`, `IndexedFaceSet`, `Extrusion`, `NurbsSurface`;

(b) rozdělení do vrstev: každou vrstvu modelu představuje samostatný VRML soubor; vkládání pomocí uzlu `Inline`;

(c) prototypy: definice prototypů, tj. objektů často se opakujících, jsou provedeny pomocí konstrukce `PROTO` a uspořádány v knihovnách prototypů;

(d) ovládání modelu: zřejmě pomocí ovládacích prvků vytvořených přímo ve VRML a integrovaných do modelu;

(e) urychlení práce s modelem: složitější objekty v modelu by měly mít definovány náhradní reprezentace pomocí uzlu `LOD`; jednotlivé VRML soubory jsou načítány teprve v okamžiku, kdy návštěvník může vidět jejich obsah; dohlednost je omezena použitím mlhy (uzel `Fog`);

(f) interaktivní prvky modelu – návrh do budoucna: nástroj pro měření vzdáleností v modelu; interaktivní prvky (manipulace s objekty, dynamické události, zobrazení doplňujících dat).