Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Dynamické modelování v jazyce VRML

Jaroslav Žďánský

Vedoucí bakalářské práce:
Ing. Jiří Vaněk, Ph.D.

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Dynamické modelování v jazyce VRML" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 20. 4. 2009 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Jiřímu Vaňkovi, Ph.D. za odborné vedení, cenné rady, trpělivost a ochotu při vedení mé bakalářské práce.

Dynamické modelování v jazyce VRML

Dynamic modelling in the VRML

Souhrn

Cílem bakalářské práce je ukázat možnosti dynamického modelování pomocí jazyka VRML a zároveň je demonstrovat na konkrétním příkladu modelu domu. Přiblížit problematiku vytváření trojrozměrného světa pomocí prostého textu s orientací na webovou prezentaci od vysvětlení problematiky, přes vytvoření 3D modelu, až po představení dynamických prvků zajišťujících interaktivitu.

Výsledkem bude plně interaktivní model skutečného domu, kterým bude pozorovatel moci procházet, otevírat dveře, rozsvěcet světla, uskutečnit průlet, případně odejmutím střechy si prohlednout interiér domu.

Summary

The main goal of this thesis is to show possibilities of dynamic modelling in the VRML language and also illustrate it on a concrete example of house. To introduce this problem of creating a three-dimensional world using plain text with orientation on a web presentation from the basis, through creating 3D model, to introduce dynamic elements obtaining interaction.

The result will be fully interactive model of the real house, where can an explorer walk through, open door, turn on lights, realize a tour around the model, possibly see interior of the house by removing the roof.

Klíčová slova: Virtuální realita, modelování, 3D, VRML, scéna, interaktivita

Keywords: Virtual reality, modelling, 3D, VRML, scene, interaction

OBSAH:

1. ÚVOD

Virtuální realita se velmi rychle dostává z oblasti vědecko-fantastických filmů na obrazovky počítačů nejen vědeckých pracovníků, techniků, ale díky svým rozsáhlým možnostem dnes i běžných uživatelů. Po počátečním překvapení z nového pojmu virtuální realita se většina z nás již nepozastavuje nad nesmyslným spojením dvou slov s opačným významem (virtual = fiktivní, neskutečný; real = skutečný, pravý) a správně si vysvětlujeme virtuální realitu jako prostředí, které umožňuje práci v trojrozměrném prostoru, vymodelováním v paměti počítače. (10)

Základem virtuální reality jsou postupy, které se nacházející po řadu let v oboru počítačové grafiky. Jde zejména o tvorbu prostorových modelů a scén, manipulaci s nimi, pohyb v trojrozměrném prostoru a zobrazování v reálném čase. Tyto metody bývají umocněny použitím speciálních periferií, které zajišťují obrazovou, zvukovou a hmatovou interakci. (10)

1.1. HISTORIE

Vývoj jazyka VRML začal počátkem roku 1994 na první World Wide Web (WWW) konferenci v Ženevě, věnované virtuální realitě spojené s internetovou prezentací. Účastníci této konference se shodli, že je potřeba vytvořit určité nástroje, které by umožňovaly pracovat s 3D objekty zobrazitelnými na webových stránkách. Tyto stránky by neobsahovaly pouze dvojrozměrné informace (text, obrázky, tabulky, …) popsané jazykem HTML, ale také trojrozměrné (např. modely reálného světa) popsané nějakým novým jazykem. Takovým, který by uživateli umožňoval se volně pohybovat mezi objekty nebo se přenášet do jiných částí nebo dokonce do úplně jiných virtuálních světů formou odkazů. Výsledkem jednání byla jasná představa o podobě jazyka, který dostal pracovní název Virtual Reality Markup Language. Skupina účastníků se dohodla, že bude po skončení konference dále pracovat na přesné specifikaci jazyka. (9)

1.1.1. VRML 1.0

Zanedlouho začala diskuse o vývoji první verze VRML. Ohlas na uspořádanou konferenci byl obrovský. Počet členů konference se za týden rozrostl na jeden tisíc. Většina účastníku se poté shodla, že není potřeba budovat VRML úplně od prvopočátku, že by mohli navázat na již existující technologie. Vítězem z několika kandidátů byl zvolen formát Open Inventor ASCII od firmy Silicon Graphics, Inc. (9)

Formát Open Inventor plně popisuje 3D scény pomocí polygonálních objektů, světel, materiálů a vlastností objektů. Glavin Bell a Rikk Carey předělali formát Open Inventoru na VRML za pomocí námětů z uskutečněné konference. Firma Silicon Graphics usoudila, že tento nový formát má budoucnost na trhu a začala vytvářet vývojové prostředky pro VRML. Slovo „Markup“ bylo později změněno na „Modeling“, aby lépe vyjadřovalo grafickou podstatu VRML. (9)

V roce 1995 se z řad vedoucích expertů VRML konference vytvořilo seskupení VRML Architecture Group (VAG). Skupina VAG si stanovili cíl neustat v úsilí o vybudování veřejně známého, plně interaktivního standardu, který by popisoval trojrozměrné virtuální světy. Členy skupiny tvořily osobnosti jako Gavin Bell, Brian Blau, Rikk Carey, Jon Marbry, Tony Parisi a Mark Pesce a další. Po několika schůzkách se skupina rozhodla zrevidovat svůj dosavadní přístup k VRML 1.0 a začala pracovat na dokumentu, který dostal pracovní název Moving Worlds později VRML 2.0. (10)

1.1.2. VRML 97

Na mezinárodní konferenci Siggraph 96 byla 4. srpna roku 1996 publikována první oficiální specifikace VRML 2.0. Poslední a nejnovější verze ji nahradila v dubnu 1997. Tato verze byla také podána k ISO (International Standardization Organization) jako DIS (Draft International Standard). (9)

VRML 97 je název standardu ISO/VRML 14772-1:1997, který je až na několik změn, upravenou specifikaci a rozdílů ve funkčnosti v podstatě identický s VRML. V důsledku těchto změn byl nakonec název změněn. (9)

ISO standard VRML 97 vznikl z potřeby umístit do statické scény nejen dokonalejší statické objekty, ale především začlenit do ní objekty pohyblivé. Jednalo se o změnu

pozice objektu, ale také schopnost reagovat na určité podněty od uživatele i jiných objektů, schopnost pohybu nebo dokonce schopnost změnit svou barvu a tvar. (9)

1.1.3. X3D

Ve specifikaci jazyka VRML 97 existují některé části, které nebyly vhodně řešeny. Například nemožnost zápisu plošných tvarů, podpora NURBS (Non-Uniform Rational B-Splines/Surfaces) bez nutnosti použití rozšiřujících modulů apod. (7)

Z tohoto důvodu byl představen nový formát určený pro popis plošných i prostorových scén, který byl nazván zkratkou X3D neboli celým jménem Extensible 3D. Slovo „Extensible“ použité v plném názvu tohoto formátu prozrazuje, že se jedná o formát založený na dnes velmi populárním jazyku XML (Extensible Markup Language). X3D je v současnosti alternativou pro jazyk VRML. (7)

1.2. MOŽNOSTI VRML A JEHO VYUŽITÍ

Obecně lze říci, že využití virtuální reality je vhodné tam, kde je potřeba uživateli představit nějaký reálný model nebo mu umožnit vyzkoušet si, jak by se sám v tomto reálném modelu choval nebo naopak, jak by na interaktivní reakce reagoval samotný virtuální objekt nebo svět.

Výhodou VRML je jeho orientace na internet, na webové stránky. Každý, kdo je na internet připojen, může okamžitě vstupovat do virtuálních prostorů, zkoumat prostorová data, případně si je ukládat na svůj disk.

Důležité je, že obsáhlý 3D model nemusí být zpracováván na straně serveru a uživateli distribuován formou hotového výstupu, což by bylo velice náročné na přenos dat, ale díky tomu, že dochází k přenosu pouze textového souboru, který takřka celý 3D model definuje, mohou tohoto využívat i uživatelé s pomalejším připojením k internetu.

Proč se model přenáší „takřka“ celý v textovém souboru? Protože je potřeba myslet na textury, které jsou stahovány zvlášť, zde záleží pouze na tvůrci, do jaké míry tyto textury zoptimalizuje, aby nedocházelo k pomalému zobrazování celé scény.

1.2.1. PŘÍKLADY VYUŽITÍ

Virtuální modelování s sebou přináší velké množství využití a výhod, některé z nich budou zmíněny.

Postupy, instruktáže

Pokud si například koupíte složený nábytek v obchodě, tak je možné využít 3D modelů všech částí a pomocí skriptů vytvořit postup ke složení celého kusu nábytku.

Architektura

Jazyk VRML umožňuje vytvoření virtuálního domu, jakožto simulace návštěvy nemovitostí před koupí nebo pronájmem. Zákazník si může celý dům včetně interiéru prohlédnout, aniž by musel někam chodit.

Informační systémy

Široké využití se zde nabízí pro virtuální prohlídku jakýchkoliv prostor nebo měst. Lze takto zprostředkovat návštěvu muzea, výstavy, plánovaných architektonických celků, měst, atd.

Vizualizace dat

Pomocí VRML je možné vytvářet tabulky a prostorové grafy s možností dynamického chování v čase. Např. je možné přehledně zobrazit populační vývoj na mapě státu pomocí vystouplých tvarů jednotlivých okresů.

1.2.2. STRUČNÁ CHARAKTERISTIKA JAZYKA VRML:

- Virtuální světy tvořené prostorovými objekty jsou kombinovány s multimediálními prvky, jakými jsou obraz, video, zvuk.
- Při tvorbě virtuálních světů lze využívat prvky zapsané jednak lokálně v souborech tak kdekoliv v síti Internet. Stejně tak lze mezi různými světy plynule přecházet, podobně jako přecházíme mezi stránkami WWW.
- Animace, interakce a manipulace s virtuálními objekty je zajištěna jednotným a přehledným způsobem. Stejné prostředky se používají pro popis statických i dynamických světů. Statické světy lze snadno rozšířit na dynamické a obráceně.

- Součástí jazyka jsou definice způsobů pohybu uživatele (chůze, let, zkoumání objektů), podpora automatické navigace ve virtuálním prostředí, popis reakce na chování uživatele.
- Popis virtuálních světů je ukládán pouze v textovém, tedy snadno čitelném tvaru. Velikost souborů je pak možno výrazně snížit kompresí pomocí programu gzip, aniž by bylo nutné se starat o jejich zpětné dekódování.

2. CÍL PRÁCE A METODIKA

Cílem bakalářské práce je představit modelovací jazyk VRML nejen obecně, ale převážně se zaměřením na oživení virtuálního světa. Základní vlastnosti VRML zde budou zmíněny, aby tato práce byla přínosem nejen pro tvůrce se zkušenostmi s 3D modelováním, ale také pro ty, kteří se tímto jazykem dosud nesetkali. V práci budou vysvětleny metody, pomocí kterých lze zabezpečit interaktivitu uživatele s virtuálními objekty, a zároveň vše bude prezentováno na praktických příkladech umístěných do zhotoveného rodinného domu.

Bakalářská práce bude orientována převážně na dynamické modelování, tj. tvorbu virtuálního světa s využitím prvků jazyka VRML, které zajišťují interaktivitu uživatele s vyhotovenou scénou, díky čemuž je uživatel do takovéhoto modelu mnohem více vtažen.

Koncept bude zaměřen také na zpětnou dekompozici, aby bylo možné kdykoliv model rozložit do jednotlivých částí, se kterými bude následující práce rychlá a přehledná. K tomuto bude mimo jiné využito tzv. prototypů, kterým bude věnována jedna z kapitol.

Ačkoliv lze do jazyka VRML importovat řadu objektů vytvořených pomocí jiných nástrojů, nebudou zde použity. Model bude vytvořen výhradně v jazyce VRML s využitím pouze textového editoru respektive s využitím editoru VrmlPad od společnosti Parallel Graphics. Importované objekty, které lze následně využít, mohou být modely vytvořené v programech CAD, případně jiných modelovacích nástrojích, jakožto také části kódu, rozšiřující stávající vlastnosti uzlů, napsané v jazycích Java a JavaScript (ECMAScript).

Důvodem absence importovaných objektů je zaměření práce na možnosti samotného jazyka VRML v oblasti dynamického modelování a jejich demonstrace na praktickém příkladu.

Takovýto model by bylo možné zpracovat například také v jazyce X3D. Ale protože je syntaxe tohoto formátu VRML velice podobná, autor se rozhodl model zpracovat v jazyce VRML, který je mu bližší z důvodu předchozích zkušeností.

2.1. VRML A JEHO POUŽITÍ PŘI MODELOVÁNÍ BUDOV A PODOBNÝCH OBJEKTŮ

Jedno z odvětví, ve kterém lze virtuální realitu uplatnit, je stavění budov. Možnost projít se v domě, který teprve bude postaven, je nesporná výhoda. Tímto je možné například zjistit, že schody do druhého patra jsou příliš strmé nebo že daná místnost má málo oken. Zjištění podobných nesrovnalostí a závad může pomoci k vytvoření skutečně vyhovujícího návrhu.

Budovy ve VRML se vytvářejí stejně jako ostatní objekty – skládáním ze základních těles. Toto zdánlivě prosté konstatování s sebou ve skutečnosti přináší řadu větších či menších praktických problémů. Stěny jsou tvořeny z kvádrů, dveře pomocí kvádru, animace a touch sensoru atd.

Již bylo zmíněno, že budovy se stavějí z kvádrů, ale představa budovy, sestávající z desetitisíců drobných cihliček je mylná. Teoreticky samozřejmě nic nebrání v tom, aby byla budova postavena tímto způsobem – ale pokud pomineme časovou náročnost takového projektu, zbývá tu ještě omezující faktor současného hardwaru. Pokud přihlédneme k množství objektů a výpočetních operací, které s nimi musí daný prohlížeč provádět, mohou nastat problémy, například: špatné načtení scény, textur nebo se dokonce scéna nemusí načíst vůbec.

Daleko vhodnější je sestavování budov z “panelů”, což v podstatě odpovídá představě “velké cihličky”. Rozdíl spočívá ve velikosti. Pro počítač je totiž velký rozdíl, zda zobrazí deset tisíc “cihliček” nebo sto “panelů”.

Nabízí se tu další metoda, opět velmi podobná, sestavování celých stěn přímo z jednoho kusu. Autor ji považuje za nejvhodnější ze stavebnicových principů, ale nevýhoda je zjevná: nelze opakovaně vkládat jedno těleso a postavit z něj celou budovu. Každý kvádr zde má jiné rozměry. Z tohoto důvodu práce trvá déle. Celkový výsledek je sice rychlejší, ale na silnějších počítačových sestavách rozdíl není příliš markantní. Tato metoda je však vhodná, pokud píšeme program bez editoru. Zde platí pravidlo “méně znamená více”, neboť každé další těleso navíc znamená dalších několik řádek, a při větších budovách je rozdíl velice citelný. Jakákoliv ruční metoda je však zjevně pomalejší než vytváření v dobrém editoru.

2.2. TVORBA 3D MODELU POMOCÍ TEXTU

Textový soubor obsahující popis virtuálního světa je možné psát v libovolném ASCII editoru. Postačí dát tomuto souboru příslušnou příponu. Jednodušší a pohodlnější je práce s některým VRML editorem, který pro tvorbu scény poskytne přiměřenou podporu a nápovědu. Editory obvykle při psaní nabízejí seznamy tzv. uzlů scény, jejich atributů (parametrů), zobrazují strukturu scény, usnadňují volbu barev a podobně. Jedním z takových editorů je VrmlPad od firmy Parallel Graphics. Tento program autor zvolil proto, že zvyšuje přehlednost a orientaci v kódu. Editor lze přirovnat k HTML editoru Home Site, který odlišuje jednotlivé značky (tagy) barvami. Obrázek č. 1 zobrazuje rozlišení syntaxe.

```
#VRML V2.0 utf8

   WorldInfo {title "Jaroslav Zdansky - Pohyb terasy" }

PROTO TerasaMOV
      [ field SFColor    barva     .5 .5 1
        field SFVec3f    posunuti  0 0 0
        field SFRotation natoceni  0 1 0 0
        field SFVec3f    meritko   1 1 1]
{
 Transform {
    translation IS posunuti
    rotation    IS natoceni
    scale       IS meritko
    children [


Group {

    EXTERNPROTO  DvereObyc []  "terasa.vrl#Terasa"

    children [

    DEF SENZOR TouchSensor {}
    DEF CAS1    TimeSensor { cycleInterval 2 }
    DEF ZAVES1 PositionInterpolator {
            key [0, 1]
            keyValue [0 0 0, -15 8 -15 ]
        }


    DEF DVERE Transform  { children DvereObyc { }}
DEF hodnoty Script {
    eventIn SFFloat  set_poloha
    eventOut SFFloat  poloha_changed
    eventIn SFTime   set_zmacknuto
    field SFBool  stav FALSE
      url "javascript:
    function set_zmacknuto (h,t)       // zapamatovani si stavu zapnuto-vypnuto
      {
         stav =!stav}

    function set_poloha (h,t)         // zmena polohy
    {  if(stav) poloha_changed=h
        else poloha_changed=1-h
            }

    "}


]}


]


ROUTE SENZOR.touchTime TO hodnoty.set_zmacknuto
ROUTE SENZOR.touchTime       TO CAS1.startTime
ROUTE CAS1.fraction_changed  TO hodnoty.set_poloha
ROUTE hodnoty.poloha_changed TO ZAVES1.set_fraction
ROUTE ZAVES1.value_changed   TO DVERE.translation


}
}
```

Obrázek č. 1 – Zobrazení kódu pomocí programu VrmlPad

3. MODELOVÁNÍ POMOCÍ JAZYKA VRML

Tato kapitola obsahuje základní informace o prohlížení scény, principu orientace, struktury jazyka VRML a několik základních prvků vhodných pro vytváření statického modelu.

3.1. ZPŮSOB PROHLÍŽENÍ SCÉNY

Virtuální svět, či virtuální scéna bývá uložena jako textový soubor s příponou „.wrl“. Podobně jako pro zobrazení HTML dokumentu je k prohlížení „.wrl“ souboru zapotřebí program – prohlížeč neboli klient.

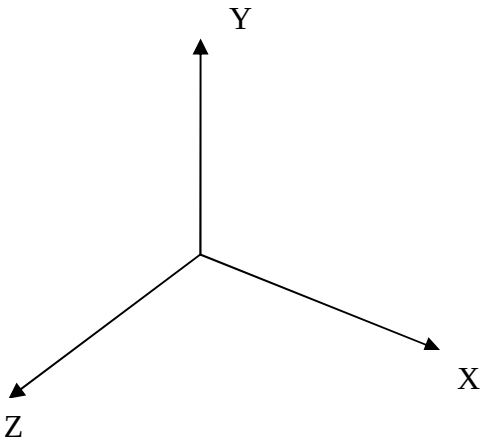
Jedná se o program, který je schopen převést textový popis ze souboru VRML do obrazu virtuálního světa. Navíc umožňuje pohyb v tomto světě a případnou interakci s virtuálními objekty.

Může to být některý z prohlížečů (klientů) dodávaných spolu s webovými prohlížeči, nebo jiný volně šiřitelný prohlížeč, který se instaluje jako zásuvný modul (plug-in) do webového prohlížeče. Přičemž mezi nejčastěji podporované internetové prohlížeče patří: Internet Explorer, Netscape a Mozilla. Mezi nejrozšířenější klienty patří Cortona VRML Client od firmy Parallel Graphics, či klienti Blaxxun Contact, CosmoPlayer a mnohé jiné.

Vzhled a ovládání těchto klientů je různý. Většina z nich mívá v dolní části ovládací panel, s jehož pomocí se lze pohybovat po virtuálním prostoru. Přepínáním ovládacích prvků je možné zvolit, zda aktivita kurzoru bude převáděna na pohyb v prostoru nebo na manipulaci s předměty.

3.2. ZÁKLADNÍ ORIENTACE V PROSTORU

Systém prostorových souřadnic je ve VRML orientován podle obrázku č. 3. Směr vzhůru je totožný s kladným směrem osy „y“. Země či podlaha, po které se pohybujeme, je rovnoběžná či přímo totožná s rovinou „xz“. Kladná poloosa „x“ míří doprava, kladná poloosa „z“ směřuje k nám. Základním pohledem je směr záporné osy „z“. (10)



Obrázek č. 2 – Základní orientace v prostoru

3.2.1. AVATAR

Jelikož se jedná o virtuální realitu, nestačí pouze zmínit možnosti prohlížení objektů za pomocí rotace, posunu, apod., ale především možnosti procházení virtuální scény a poznávání vymodelovaného světa. K tomu, aby bylo možné toto zprostředkovat, vznikl tzv. dvojník neboli osoba, kterou je možné za pomocí klávesnice a myši ovládat.

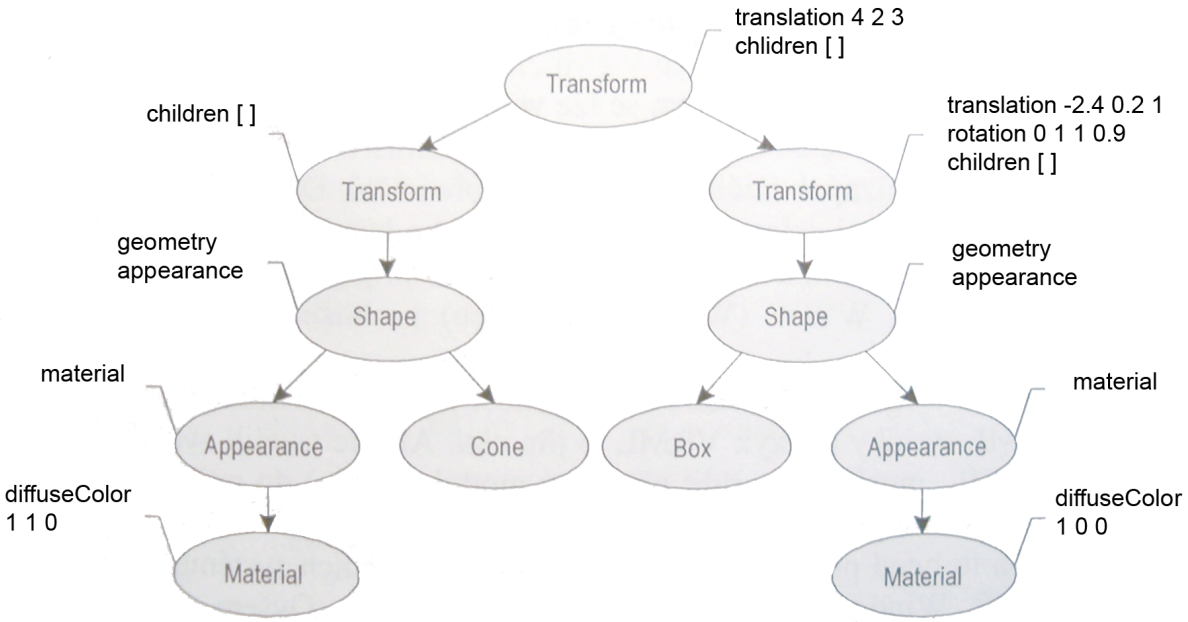
Tato osoba, která představuje nás samotné uvnitř virtuálního světa, se nazývá avatar[1]. Okno prohlížeče je právě tím, co avatar vidí. Avatarovi lze definovat rozměry šířky, výšky očí a maximální výšky překročitelné překážky, které mu brání procházet malými průchody. V interaktivně navržených světech lze detekovat, v jakém místě stojí, co vše vidí, zda naráží na překážku nebo zda vstupuje do hlídané zóny. Avatarovu polohu a vlastnosti je možné určit pomocí uzlu *NavigationInfo*.

3.3. STRUKTURA SCÉNY

Standard VRML 97 definuje množinu objektů pro vytváření trojrozměrných scén. Tyto objekty jsou definovány ve zvláštních strukturách – uzlech (*nodes*), které jsou uspořádány do hierarchické struktury zvané strom (*tree*). Nejjednodušší strom tvoří jediný uzel. Tento uzel se nazývá kořen (*root*). Každý uzel může být propojen

[1] Jméno avatar pochází z hinduistické mytologie, kde označuje dočasnou tělesnou schránku, do které se vtěluje Bůh při své návštěvě Země.

s několika dalšími uzly, tzv. potomky *(children)*. Naopak z pohledu potomků je předchozí uzel jejich rodič *(parent)*. Strom by ovšem nebyl stromem, pokud by nastala situace, kdy se rodič stane potomkem svého potomka. Tato situace bývá označována, že ve stromu vznikl cyklus. Celou scénu pak tvoří graf scény *(scene graph)*, což je les stromů (graf složený z alespoň jednoho stromu. Každý strom v lese stromů je samostatný, na ostatních stromech nezávislý subjekt. Uzly v jednotlivých stromech spolu však mohou komunikovat prostřednictvím událostí *(ROUTE… TO…)* (9)



Obrázek č. 3 – Struktura scény (9)

3.4. ZÁKLADNÍ ÚTVARY

Tento jazyk má pouze čtyři základní geometrická tělesa – kouli, kvádr, kužel a válec. Všechna jsou umístěna tak, aby jejich těžiště bylo v počátku souřadné soustavy. Výjimkou je kužel, jehož střed je situován v počátku souřadnic tak, že v počátku souřadnic leží střed jeho osy.

3.4.1. INICIÁLNÍ HODNOTY ZÁKLADNÍCH TĚLES:

Kvádr

```
Box {  size          2     2     2}
```

Koule

```
Sphere { radius      1}
```

Kužel

```
Cone {bottomRadius   1
      height         2}
```

Válec

```
Cylinder {   radius  1
             height  2}
```

3.4.2. UZLY PRO DEFINICI SLOŽITĚJŠÍCH TĚLES

Další skupinou geometrických těles je skupina uzlů pro vytváření přírodních útvarů nebo složitějších těles. Mezi jejich výhody patří především schopnost dobře popsat obecná tělesa nebo měnit svůj tvar.

ElevationGrid

Výšková mapa (vhodné pro vytváření např. hor).

Extrusion (Vytahování)

Tažené těleso vytvořené z několika průřezů.

IndexedFaceSet

Obecné těleso složené s plošek.

IndexedLineSet

Obecné těleso složené z úseček

PointSet

Obecné těleso složené z bodů.

3.5. Použití DEF a USE

Bylo by časově náročné zapisovat nebo kopírovat stále stejné části kódu, pokud by měly být použity vícekrát nebo z nich složen rozsáhlejší model. Tento postup by byl z lidského hlediska velice náročný, z počítačového dokonce zbytečný.

Existuje možnost, jakým způsobem jednou zapsanou informaci opakovaně využít. Jakmile je uzel již definován pomocí příkazu *DEF* (viz. Obrázek č. 4), je možné v další části souboru vytvořit jeho kopii příkazem *USE* (viz Obrázek č. 5). Vždy je pochopitelně nutné uvést jméno definovaného uzlu. Nejčastěji tohoto autor využíval při definici materiálů, které obsahovaly více parametrů. V opačném případě by docházelo ke zbytečnému prodlužování kódu, zhoršování čitelnosti a orientace v kódu. Na obrázcích je definován materiál a textura objektu. V prvním řádku „DEF PodlahaLozniceBila…“ je příkaz *DEF* použit pouze pro snadnější orientaci v kódu, nikoliv pro následné využití objektu příkazem *USE* – VrmlPad dokáže zobrazit jejich seznam, což podstatně ulehčuje vyhledávání a zpětnou editaci objektů.

```
DEF PodlahaLozniceBila  Transform {
#podlaha loznice
    children [
Transform{
translation 0 -0.15 -0.05
    children Shape {
        geometry Box {size 4.25 0.1 4.05}
        appearance DEF omitka Appearance {
                material Material {
                    diffuseColor 0.5 0.5 1}
                    texture ImageTexture {
                        url "textures/omitka.jpg"
                        repeatS FALSE
                        repeatT FALSE }
                        }
            }

    }
]
}
```

Obrázek č. 4 – Definice objektu s využitím příkazu DEF

Na následujícím obrázku je vidět zkrácený zápis s využitím již definovaného materiálu „omitka“.

```
DEF PodlahaKoupelnaBila
#podlaha koupelna
Transform{
translation -5 -.15 -0.6
    children Shape {
        geometry Box {size 3.05 0.1 2.9}
        appearance USE omitka}

    }
```

Obrázek č. 5 – Použití definovaného objektu s využitím příkazu USE

3.6. PROTOTYPY

Příkazy *DEF* a *USE* však slouží pouze k definování základních uzlů. Pokud je potřeba této funkce využít u komplexnějšího modelu pro opakované použití ve virtuálním světě, nabízí se velice užitečná funkce *PROTO*.

Prototypy umožňují definovat nové, vlastní uzly a stromové struktury, vytvářet knihovny specializovaných objektů. Pomocí konstrukce *PROTO* můžeme definovat, co nám v jazyce VRML chybí, přičemž výsledek bude vypadat stejně, jako kterýkoliv standardní uzel VRML. Prototyp představuje vzor uzlu, který můžeme vkládat do stromové struktury VRML a modifikovat nastavení jeho parametrů. Prototypy umožňují mnohem více možností, než nabízí použití příkazů *DEF* a *USE*.

```
PROTO OknoKoupelna-Spajz
      [ field SFColor    barva     .6 .5 .1
        field SFVec3f    posunuti  0 0 0
        field SFRotation natoceni  0 1 0 0
        field SFVec3f    meritko   1 1 1]
{
 Transform {
    translation IS posunuti
    rotation    IS natoceni
    scale       IS meritko
    children [

        Transform   {translation 0 0 -.02
          children  Shape {
              geometry Box {size .6 1.4 .01}
              appearance Appearance {
                material Material {
                    diffuseColor IS barva transparency 0}
                    texture ImageTexture {
                        url "textures/oknokuchynout.gif"
                        repeatS FALSE
                        repeatT FALSE }
                            }
                        }
                    }

     Transform  {translation 0 0 .02
          children  Shape {
              geometry Box {size .6 1.4 .01}
              appearance Appearance {
                material Material {
                    diffuseColor IS barva transparency 0}
                    texture ImageTexture {
                        url "textures/oknokuchynin.gif"
                        repeatS FALSE
                        repeatT FALSE }
                            }
                        }
                    }

        Transform{
            translation 0 0 0
                children Shape {
                    geometry Box {size .6 1.4 .01}
                    appearance Appearance {
                    material Material {
                    diffuseColor 0.9 0.9 1 transparency .494}
                            }
                        }
                    }

  ]
}
}
```

Obrázek č. 6 – Kód definice prototypu

```
EXTERNPROTO oknokoupelna-spajz
      [ field SFColor    barva
        field SFVec3f    posunuti
        field SFRotation natoceni
        field SFVec3f    meritko]
      "oknaP.wrl#OknoKoupelna-Spajz"

Group {
  children [ oknokoupelna-spajz { posunuti -5 1.65 -2}          #koupelna
             oknokoupelna-spajz { posunuti -2.9 1.65 -2}           #spajz

  ]
}
```

Obrázek č. 7 – Kód použití definovaného prototypu

Obrázek č. 7 je rozdělen do dvou částí. V první jsou popsány parametry okna a rámu, ve druhé je toto okno opakovaně vkládáno do virtuálního světa jako běžný uzel se jménem *oknokoupelna-spajz*. Stejně jako u standardních uzlů, také zde má každý z jeho parametrů předdefinovanou hodnotu, takže při vkládání oken do virtuálního světa postačí zapisovat pouze parametry, jejichž hodnoty se liší od základních. Mezi nejdůležitější parametry, které je možné u prototypů měnit, patří: barva *(color),* natočení *(rotation)* a měřítko *(scale).*

3.7. GRAFICKÉ EDITORY VRML

Tyto nástroje, editory, nebo také buildery jsou vhodným pomocníkem při vytváření větších objektů ve VRML. Zpočátku je doporučeno psát VRML zdrojové kódy ručně v textovém editoru, aby se případný zájemce o programování ve VRML naučil programovat, osvojil si základní dovednosti a získal zkušenosti, které se nedají získat z příruček a manuálů. Při větších projektech se builder stává nezbytností, je proto vhodné zvolit odpovídající nástroj.

Když autor s modelováním v jazyce VRML začínal, neměl k dispozici žádnou literaturu, a tudíž musel jednotlivé modely vytvářet pomocí builderu. Následně procházel vygenerované kódy, aby zjistil, na jakém principu jazyk VRML funguje a jaký význam mají jednotlivé parametry a uzly.

Jelikož při vytváření modelu v bakalářské práci nebyl žádný z těchto builderů použit, tak je o nich uvedeno jen několik informací.

3.7.1. PŘEHLED GRAFICKÝCH EDITORŮ

RenderSoft VRML editor

Jméno: RenderSoft VRML editor

Autor: RenderSoft software

Popis programu:

Ačkoliv RenderSoft VRML editor obsahuje mnoho příjemných drobností, které ulehčují práci, autor jej nepovažuje pro vytváření VRML za použitelný kvůli přílišným nedostatkům při manipulaci se scénou a tělesem. Chybí podpora VRML97.

Internet space builder

Jméno: Internet space builder (ISB)

Autor: Parallelgraphics software

Popis programu:

Internet space builder je builder širokého využití, který má širokou podporu pro vytváření budov, jejich blízkého okolí a interiéru. Bohužel jeho VRML kód není příliš čitelný a je v něm spousta zbytečně deklarovaných parametrů, které kód prodlužují. Jednotlivé předměty větší složitosti se v něm modelují složitě.

Spazz3D

Jméno: Spazz3D VRML 97 Authoring Tool.

Autor: Virtock technologies, Inc.

Popis programu:

Spazz3D je nástroj, který byl vytvořen pro usnadnění práce ve VRML jako takovém, podporuje práci se senzory, avatary, animacemi atd. Je určen pro vytváření malého množství velmi složitých předmětů. Není vhodný pro tvoření velmi rozměrných objektů (např. budov). Je ale potřeba si zvyknout na zmatené funkce a nepříliš uživatelsky přátelského prostředí.

3.8. HARDWAROVÉ NÁROKY

Každá grafická aplikace je většinou velmi náročná na hardwarové konfiguraci počítače. Renderování scény je spojeno s vysokým vytížením procesoru, grafické karty a značnými nároky na kapacitu operační paměti. I u méně složitých modelů je velice důležité, aby grafická karta hardwarově podporovala renderování 3D grafiky, a to Direct3D nebo OpenGL. Při zobrazení v softwarovém režimu dochází ke značnému trhání a zpomalení celé scény. K bezproblémovému zobrazení složitější scény je zapotřebí procesor alespoň o kmitočtu kolem 500 MHz a zmiňovaná 3D akcelerace Direct3D nebo OpenGL.

4. DYNAMICKÉ MODELOVÁNÍ VE VRML

Dosud bylo zmíněno pouze modelování statických objektů bez jakékoliv interakce. Aby bylo možné existující virtuální svět oživit, je důležité zmínit další prvky, pomocí nichž bude možné model obohatit o interaktivní vlastnosti a funkčně posunout možnosti již hotového světa na vyšší úroveň.

K tomuto je možné využít dynamických uzlů, které reagují na chování uživatele (avatara), a vysílají informace v podobě tzv. událostí (events) a mechanismus předávání událostí mezi jednotlivými uzly – konstrukce *ROUTE… TO…*.

Velice významným pomocníkem při vytváření dynamického modelu jsou prototypy, zmíněné v jedné z předchozích kapitol, které dokáží zpřehlednit vytvářený kód a tím usnadní autorovi orientaci. (9)

4.1. UDÁLOSTI

Událost je základním prostředkem, který umožňuje rozhýbání statickým scén neboli světů. Můžeme si ji představit jako datový záznam, který je předáván mezi uzly v okamžiku, kdy z nějakého důvodu dojde ke změně hodnoty parametru uzlu. Uzel, v němž událost na základě takového podnětu vznikne, je označován za vysílatele události. Podobně uzel, k němuž byla data o události dopravena k dalšímu zpracování, událost přijal a přijatá data uložil do svého parametru.

4.2. PARAMETRY DATOVÝCH TYPŮ

Parametry jsou kromě svého datového typu dále charakterizovány svým vztahem k událostem. Existují tři základní třídy parametrů:

field

Statická veličina s iniciální hodnotou. Lze ji změnit pouze zapsáním nové hodnoty v souboru VRML, nikoliv však dynamicky.

eventIn

Parametr schopný přijmout událost daného datového typu. Hodnota parametru je změněna přijetím události.

eventOut

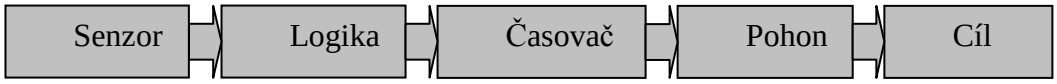
Parametr schopný vyslat událost v okamžiku, kdy dojde ke změně jeho hodnoty. Tato změna je nejčastěji vyvolána chováním avatara nebo je důsledkem zpracování jiných událostí.

U mnoha uzlů je žádoucí, aby parametr měl nejen svoji iniciální hodnotu, ale aby současně dokázal přijmout událost, která jeho hodnotu změní a aby tuto změnu vyslal v podobě události dalším uzlům. Proto byla zavedena ještě čtvrtá, rozšiřující třída parametrů, nazvaná *exosedField*. Každý parametr této třídy v sobě zahrnuje schopnosti všech tří základních tříd.

exposedField

Parametr s iniciální hodnotou, který je schopný přijmout třídy měnící jeho hodnotu a také po změně své hodnoty vysílat události.

Pro větší přehlednost je na obrázku č. 8 uvedeno logické schéma dynamické akce:



Obrázek č. 8 – Schéma dynamické akce (10)

4.3. ČASOVAČ

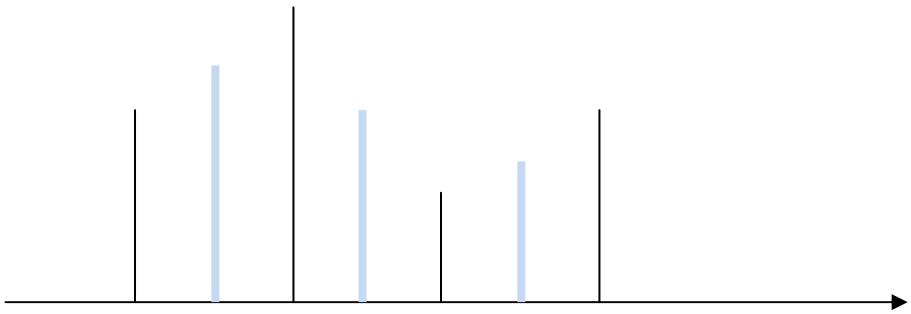
Významnou roli v každé interaktivní akci hraje čas. Jen málo dějů ve skutečném světě probíhá naráz, skokovou změnou. Většina dynamických jevů má naopak zřetelný časový rozměr – v určitém čase je děj zahájen, z počátečního klidového stavu je objekt uveden do pohybu, který po určitou dobu trvá, a nakonec dochází k ukončení děje přechodem do klidového stavu. Časování takovýchto akcí zajišťuje uzel *TimeSensor*.

Na tento *TimeSensor* je možné pohlížet jako na zařízení, které je schopno průběžně vysílat události – časové impulsy. Frekvence vysílání těchto impulsů nelze ovlivnit, protože závisí na výkonu počítače. Z toho také vyplývá, že není nutné přesné určení doby trvání konkrétní události, ale spíše je důležité, zda nějaký děj předchází jinému ději, zda určitá akce následuje po jiné, či zda je několik dějů zahájeno ve stejný okamžik. (10)

Příklad s použitým časovačem bude představen v následující kapitole Interpolátory, kde budou zmíněny další důležité uzly využité v událostech v modelu domu.

4.4. INTERPOLÁTORY

Interpolátory[2] jsou uzly sloužící v jazyce VRML k pohonu objektů. Každý takovýto uzel obsahuje klíčové hodnoty, ze kterých prohlížeč vychází. Například při změně polohy, a dopočítává mezilehlé body (souřadnice). Na následujícím schématu (Obrázek č. 9) je přiblížen princip lineární interpolace.



Obrázek č. 9 – Interpolace

[2] Pojmem interpolace bývá označován proces vyhledávání mezilehlých hodnot na základě předem daných konstant, které se nazývají klíčové hodnoty. Každá klíčová hodnota se vztahuje k nějakému času.

Vodorovná osa představuje čas a černé úsečky klíčové hodnoty, úsečky modré barvy jsou následně dopočítány. Vypočítávaná hodnota leží ve schématu na spojnici klíčových hodnot, tedy na přímce. Proto tato interpolace bývá nazývána lineární.

Lineární interpolace umožňuje na základě malého množství klíčových hodnot generovat velké množství hodnot nových. To je velmi příhodné pro animace. Pro pohyb tělesem po prostorové dráze, postačuje definovat zvolit interpolátor pro výpočty prostorových souřadnic a zadat mu ty polohy tělesa, v nichž se dráha výrazněji zakřivuje. Interpolátor poté dopočítá souřadnice všech ostatních poloh. (10)

4.4.1. PŘEHLED INTERPOLÁTORŮ

ScalarInterpolator

„Přepočet času na číslo s desetinou tečkou, univerzální interpolátor pro řízení parametrů v mnoha uzlech.“ (10)

PositionInterpolator

„*Výpočet polohy jednoho bodu v prostoru, vhodné pro určení trajektorie pohybu objektů.”* (10)

ColorInterpolator

„Výpočet hodnoty jedné barvy, typicky v materiálovém uzlu nebo ve světelném zdroji.“ (10)

OrientationInterpolator

„Určení osy a úhlu natočení, nejčastěji pro uzel Transform*, ale třeba i pro stanoviště.“* (10)

NormalInterpolator

„Generování množin normálových vektorů, které se použijí v uzlu Normal*.“* (10)

4.5. DETEKTORY

Virtuální světy se stávají v důsledku jejich reakcí na avatarovu aktivitu. Detektory dotyku představují čidlo, po jehož aktivaci mohou být zahájeny předdefinované akce. Ve všech těchto případech hraje avatar aktivní roli, jeho ruka uchopí manipulátor nebo se dotkne objektu a teprve poté je dynamická akce zahájena.

Ve virtuálním modelu je možné spouštět dynamické akce, aniž by avatar musel kamkoliv ukázat kurzorem nebo kliknout. To je výhodné zejména tehdy, pokud je potřeba akce spouštět automaticky, bez přičinění uživatele, který by ani nemusel postřehnout, že se ve scéně vyskytoval nějaký aktivační bod.

K tomuto slouží uzly, které zajistí zahájení dynamické akce při detekci avatarovi polohy a jeho chování.

4.5.1. PŘEHLED DETEKTORŮ

Collision (detekce nárazu)

Zjišťuje, zda avatarova postava narazila do určitého objektu. (10)

ProximitySensor (detektor přítomnosti)

Sleduje avatarovu polohu a orientaci v prostoru omezeným pomyslným kvádrem. (10)

VisibilitySensor (detektor viditelnosti)

Určuje, zda se daná oblast obklopená pomyslným kvádrem dostala do avatarova zorného prostoru a zda je alespoň část této oblasti viditelná. (10)

TouchSensor (detektor dotyku)

Sleduje aktivitu avatarovi ruky, zda nad daným objektem pouze přejela nebo přímo na tento objekt avatar kliknul. (10)

„Aby mohl být objekt senzorem, který bude reagovat na myš, musí být ve zdrojovém souboru jeho tělo společně s tělem daného senzoru v poli children daného skupinového uzlu.“ (9)

Na následujícím obrázku (Obrázek č. 10) je vysvětleno použití výše popsaného prvku *ProximitySensor*.

```
################Muj Pokoj################
PROTO SvetloMujPokoj
      [ field SFColor    barva     .6 .5 .1
        field SFVec3f    posunuti  0 0 0
        field SFRotation natoceni  0 1 0 0
        field SFVec3f    meritko   1 1 1]
{
 Transform {
    translation IS posunuti
    rotation    IS natoceni
    scale       IS meritko
    children [

Group {
  children [
     Transform  {translation 0 0 0
          children  Shape {
              geometry Sphere {radius .17}
              appearance Appearance {
                  material Material {emissiveColor .7 .7 .7} }
              }
          }

    DEF SVETLO02 PointLight { on FALSE location 0 -.5 0 radius 4 }

    Transform {translation 0 -1 0
    children
    DEF AKCE02 ProximitySensor {size 6 3 4}
    }
  ]
}

]
ROUTE AKCE02.isActive TO SVETLO02.on
}
}
```

Obrázek č. 10 – Kód využití uzlu ProximitySensor

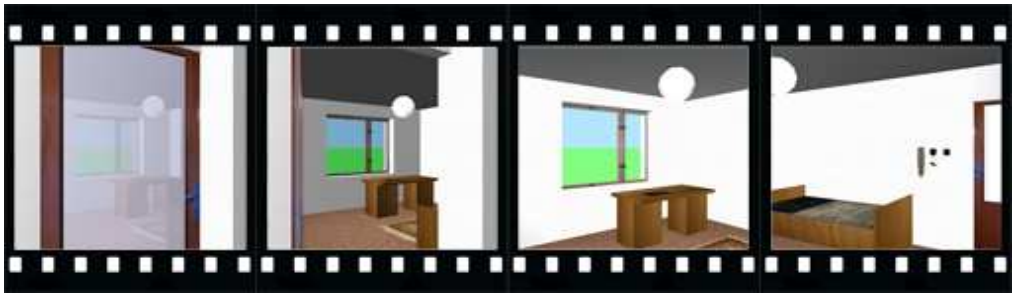
Do místnosti „můj pokoj“ je vložen dynamický uzel, který detekuje přítomnost avatara. Tento objekt „osvětlení místností“ je řešen pomocí zmiňovaných prototypů. Jakmile avatar vstoupí do místnosti, dynamický uzel *ProximitySensor* vyšle událost dalšímu uzlu – zdroji světla *SpotLight*. Původně vypnutý zdroj světla se rozzáří, protože

jeho parametr *on* se nastaví na hodnotu *True*. Jakmile avatar místnost opustí, uzel *ProximitySensor* vyšle událost, která způsobí vypnutí světla.

Na obrázku č. 10 je uveden již zmiňovaný příkaz *ROUTE… TO…*, který slouží k propojení vysílacích a přijímacích uzlů respektive jejich parametrů. Právě ty parametry, které jsou takto propojeny, si mohou navzájem předávat události. Uzly, mezi jejichž parametry se události předávají, musejí být nejprve pojmenovány příkazem *DEF*. Do příkazu *ROUTE* se následně zapisují dvojice, které jsou tvořené individuálním jménem uzlu a jeho parametru, spojené tečkou. První dvojice (hned za slovem *ROUTE*) představuje místo, odkud je událost vysílána, druhá dvojice (za slovem *TO*) označuje místo přijetí události.

Uzel *ProximitySensor*, nazvaný *AKCE02* vysílá událost při vstupu avatara do určité oblasti a událost při jeho odchodu.

Na příkladu tohoto pokoje je také patrné, že události obsahují data různých typů. Pro zapnutí a vypnutí světla je třeba přenášet údaje *True* a *False*, tedy typu *SFBool*. Uzel, který přijímá událost, proto smí zapisovat dodávané údaje pouze do těch parametrů, jejichž datový typ je shodný s datovým typem konkrétní události.



Obrázek č. 11 – Rozsvícení světelného zdroje

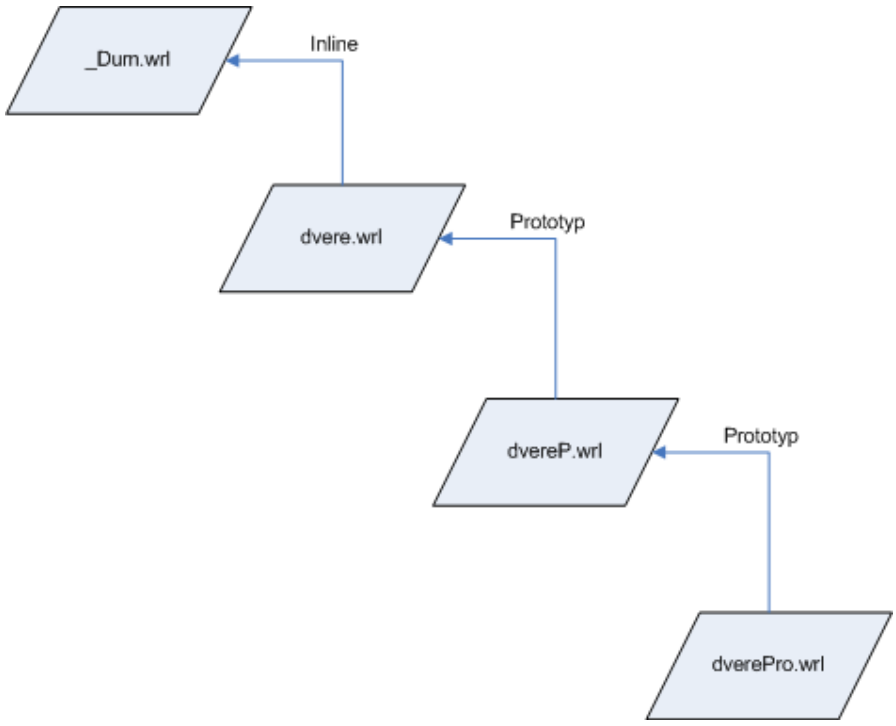
„Časově závislé uzly generují událost isActive (je aktivní) s hodnotou True pokaždé, když se uzel stane aktivním. Událost nese hodnotu False, jestliže se uzel stane neaktivním (konec cyklu). Tyto dva okamžiky jsou jediné, kdy se událost vysílá. Bylo mylné se domnívat, že se událost vysílá při každém kroku, kdy je uzel aktivní.“ (9)

4.6. MANIPULACE S OBJEKTY

S objekty lze manipulovat více způsoby, posunutím po rovině, otáčením, libovolným přesouváním v prostoru, ale také akcí vyvolat pohyb po předdefinované trajektorii. Takový pohyb je možné několikrát opakovat nebo jej ukončit na místě počátku, např. u otevírání a automatického zavírání dveří.

4.6.1. POHYB PŘEDMĚTU

V této části je plně využito možností prototypů, jakožto schránek umožňujících pojmout více objektů a následně s každým jednotlivým objektem manipulovat nebo jej modifikovat. V modelu jsou dveře koncipovány tak, že jejich základ, soubor *dverePro.wrl*, obsahuje definici veškerých dveří umístěných v modelu. Tento soubor objektů je vnořen do souboru *dvereP.wrl*, ve kterém jsou za pomocí uzlů *TouchSensor*, *TimeSensor* a *OrientationInterpolator* oživeny. Soubor *dvere.wrl* následně obsahuje definici umístění veškerých dveří v domě a je vložen do hlavní scény pomocí uzlu *inline*. Tento postup je naznačen v následujícím schématu.



Obrázek č. 12 – Schéma principu využití prototypů

„Praktický význam uzlu inline je v tom, že při vytváření světa lze vycházet z hotových objektů a z nich postupně vytvořit svět nový. Vkládané objekty mohou být umístěny na vzdáleném serveru a nemusí zabírat místo na pevném disku.“ (9)

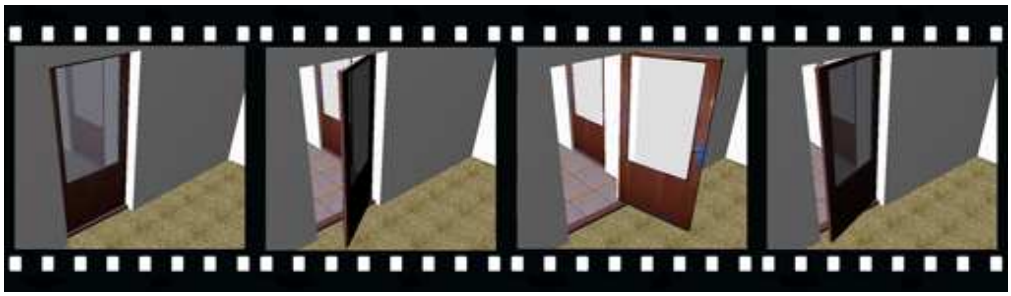
Kód s použitím prototypu byl již uveden, zbývá tedy uvést část kódu, díky kterému se dveře po kliknutí otevřou a během pěti sekund následně samy zavřou.

```
PROTO Dvere01
      [ field SFColor    barva     .5 .5 1
        field SFVec3f    posunuti  0 0 0
        field SFRotation natoceni  0 1 0 0
        field SFVec3f    meritko   1 1 1]
{
 Transform {
    translation IS posunuti
    rotation    IS natoceni
    scale       IS meritko
    children [


Group {

    EXTERNPROTO  DvereObyc []  "dverePro.wrl#DvereObyc"

    children [


    DEF SENZOR TouchSensor {}
    DEF CAS1    TimeSensor { cycleInterval 5 }

    DEF ZAVES1  OrientationInterpolator {
                 key [0, 0.2, 0.3, 0.7, 0.9, 1]
                 keyValue [0 1 0 0,    0 1 0 -1.6, 0 1 0 -1.8,
                 0 1 0 -1.8, 0 1 0 -0.2, 0 1 0 0]
                     }

    DEF DVERE Transform  { children DvereObyc { }}


]}

]
ROUTE SENZOR.touchTime        TO CAS1.startTime
ROUTE CAS1.fraction_changed   TO ZAVES1.set_fraction
ROUTE ZAVES1.value_changed    TO DVERE.rotation
```

Obrázek č. 13 – Oživení objektu dveří

Senzor *TouchSensor* detekuje kliknutí avatara na objekt, následně aktivuje uzel *TimeSensor*, který spustí dle parametrů uzlu *OrientationInterpolator* akci otevření a zavření dveří ve scéně.

Na následujícím obrázku (Obrázek č. 14) je tato akce znázorněna.



Obrázek č. 14 – Otevíraní a zavíraní dveří

Obdobným způsobem je řešeno také otevírání branky a brány u vjezdu do zahrady.

4.6.2. POSUN PŘEDMĚTU

Manipulaci s objekty, například jejich posunování, je možná s využitím uzlu *PlaneSensor*, který převádí pohyb kurzoru na posun po povrchu pomyslné roviny kolmé na osu z.

Umožňuje návštěvníkovi virtuálního světa změnit polohu vybraného objektu nebo celé skupiny objektů. Avatar s jeho pomocí dokáže svojí virtuální rukou uchopit objekt a změnit jeho vlastnosti popsatelné transformacemi v prostoru. V okamžiku, kdy je na obrazovce kurzor přesunut nad objekt, který je pod vlivem manipulátoru, lze aktivováním kurzoru (stisknutím tlačítka myši) s objektem manipulovat – měnit jeho polohu. Na příkladu bude prezentován pouze uzel *PlaneSensor*, avšak využitím uzlů *SphereSensor* nebo *CylinderSensor* je možné měnit také orientaci či měřítko objektu.

Příklad použití uzlu *PlaneSensor* neboli rovinného manipulátoru je demonstrován na popelnici před domem.

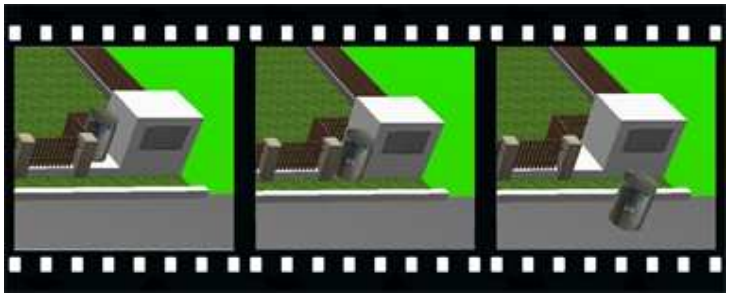
```
PROTO PopelniceMOV
      [ field SFColor    barva     .5 .5 1
        field SFVec3f    posunuti  0 0 0
        field SFRotation natoceni  0 0 0 0
        field SFVec3f    meritko   1 1 1]
{
 Transform {
    translation IS posunuti
    rotation    IS natoceni
    scale       IS meritko
    children [


Group {

    EXTERNPROTO  Popelnice []  "popelnice.wrl#Popelnice"

    children [

    DEF POSUNUTI PlaneSensor {maxPosition 0 0 minPosition 0 -2}
    Transform {
     rotation 1 0 0 -1.57
   translation 0 0 -2
  children


    DEF PopelniceAction Transform  {
        rotation 1 0 0 1.57
        children Popelnice {}

    }

}

]
}


]


ROUTE POSUNUTI.translation_changed TO PopelniceAction.translation
}
}
```

Obrázek č. 15 – Změna polohy předmětu

Ve stromové struktuře jsou nad sebou dva uzly *Transform* a pod nimi je umístěn uzel *Popelnice*. Nejbližší rodič uzlu *Popelnice* tento objekt dočasně otočí do stejné roviny, ve které generuje *PlaneSensor* souřadnice, tedy do roviny xy. Na tento transformační uzel jsou směrovány hodnoty z manipulátoru. Nadřazený transformační uzel provede otočení objektu zpět do původní roviny. Výsledná kombinace transformací zajistí požadovaný směr posunu popelnice. Aby se popelnice pohybovala pouze ve

vymezeném prostoru, je potřeba definovat parametry rovinného manipulátoru *maxPosition* a *minPosition*.

Na následujícím obrázku (Obrázek č. 16) je tento posun zobrazen.



Obrázek č. 16 – Manuální posun objektu

5. PRAKTICKÉ ZPRACOVÁNÍ 3D MODELU VIRTUÁLNÍHO SVĚTA

V této kapitole je představen samotný vymodelovaný dům a pro lepší představu je vždy srovnán se skutečným domem, podle kterého celý projekt vznikl. Vždy je uvedena fotografie skutečného domu a pod ní zobrazen model ze stejné pozice. Na obrázku č. 17 a 18 je dům zobrazen zepředu zleva. Na obrázcích č. 19 a 20 zepředu zprava a na zbylých dvou porovnáních je na dům nahlíženo zepředu na vchod a zezadu na terasu.

Při vytváření modelu domu autor vycházel ze stavebních plánů. Tyto plány nebyly při výstavbě přesně dodrženy, takže byl vytvořen „kompromisní“ model, využívající z části plánů a z části skutečných parametrů rodinného domu.

3D scéna obsahuje několik úhlů pohledu, které byly využity pro vytvoření snímků, a také obsahuje dvě předdefinované cesty prezentující model. První cesta začíná u hlavního vchodu do domu, chůzí avatara obchází dům z levé strany dozadu, postupně se pohled zvedá výše a umožňuje pohled okny do interiéru pokojů. Druhá cesta nabízí formou obletu kolem celého domu pohled ze všech stran z výšky 10 metrů. Nechybí také pohled s názvem Avatar, který je připraven pro přemístění pozorovatele na silnici před dům, odkud může vyrazit na prohlídku chůzí a využívat šipek na klávesnici pro navigaci a kliknutí myší pro manipulaci s předměty nebo jejich aktivaci.

5.1. ÚHEL POHLEDU FOTO 1

Obrázek č. 17 – Foto 1 – Realita

Obrázek č. 18 – Foto 1 – Virtuální scéna

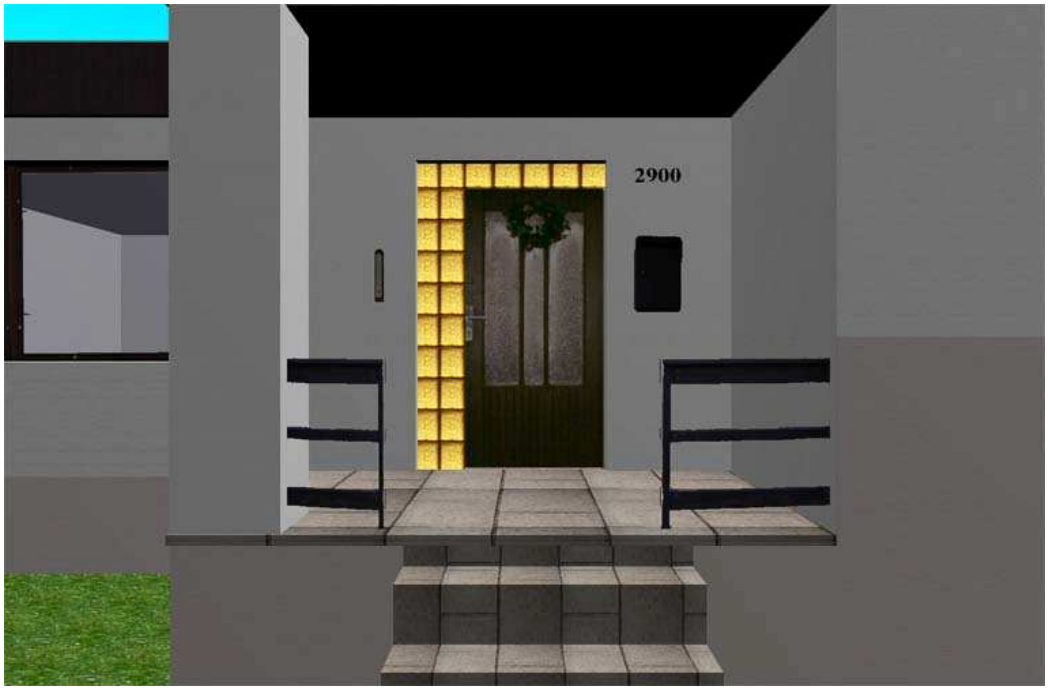
5.2. ÚHEL POHLEDU FOTO 2

Obrázek č. 19 – Foto 2 – Realita

Obrázek č. 20 – Foto 2 – Virtuální scéna

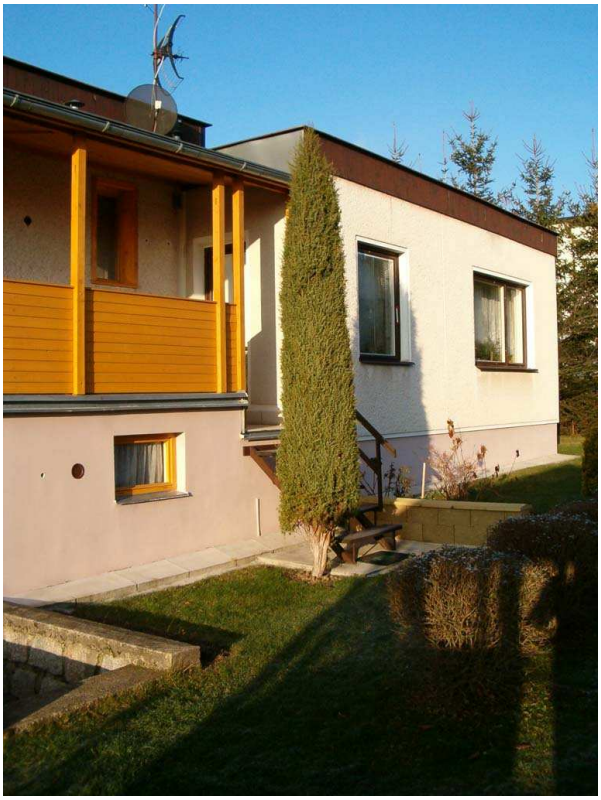
5.3. ÚHEL POHLEDU FOTO 3

Obrázek č. 21 – Foto 3 – Realita



Obrázek č. 22 – Foto 3 – Virtuální scéna

5.4. ÚHEL POHLEDU FOTO 4



Obrázek č. 23 – Foto 4 – Realita

Obrázek č. 24 – Foto 4 – Virtuální scéna

6. ZÁVĚR

Bakalářská práce demonstruje využití jazyka VRML pro tvorbu 3D scény za pomocí pouze textového editoru. Čímž zprostředkovává pohled na samotný kód reprezentující vymodelovaný objekt.

Využitelnost jazyka VRML je velice široká, ale při tvorbě rozsáhlých scén nebo modelů může docházet ke snižování přehlednosti kódu. Pokud se nedodržují jisté zásady, například využívání prototypů, funkcí DEF a USE, případně Inline, atd., které umožňují vložení jednotlivých částí nebo celých objektů do scény, stává se kód velice nepřehledným, což výrazně snižuje možnosti dodatečných úprav, případně aktualizací.

Přínosy možností 3D modelování vidí autor především v umožnění zobrazit objekty ve skutečné velikosti a skutečně vypadající včetně různých interaktivních vlastností, které mohou být pro daný objekt typické. Je možné si například prohlédnout část města, která teprve má být vystavěna, realizovat virtuální prohlídku interiérů, více či méně rozsáhlých projektů a to vše orientované na webovou prezentaci s minimálními nároky na datovou propustnost.

Díky tomu, že je celý trojrozměrný svět definován pomocí textu, jsou nároky na konektivitu mizivé. Využití větších textur, zvukových souborů nebo videí by ovšem mělo na rychlost stažení celé scény výrazný vliv.

Nespornou výhodou tohoto způsobu vytváření trojrozměrného světa je možnost pomocí textu definovat tělesa takřka jakýchkoliv rozměrů a tvarů doplněných o přednastavené chování. Nicméně nevýhodu představuje značná časová náročnost a rozsáhlost celého kódu. Tento model autor vytvářel zhruba tři měsíce a délka celého kódu dosahuje 10 426 řádků.

7. SEZNAM LITERATURY

1. 3D VRML Tutorial. *Lighthouse 3D: A Resource for 3D Programmers on OpenGL, VRML, and Shockwave 3D.* [Online] Lighthouse. [Citace: 15. 11 2008.] http://www.lighthouse3d.com/vrml/tutorial/.

2. Features - Internet Space Builder - Web 3D Products. *Internet Space Builder - Web 3D Products.* [Online] Paralle Graphics. [Citace: 20. 3 2009.] http://www.parallelgraphics.com/products/isb/features/.

3. Gergelitsová, šárka. *VRML v příkladech.* Praha : BEN - technická literatura, 2004. str. 224. ISBN 80-7300-138-1.

4. RenderSoft VRML Editor - General Features. *RenderSoft VRML Editor.* [Online] RenderSoft. [Citace: 23. 3 2009.] http://pachome2.pacific.net.sg/~jupboo/.

5. Spazz3D VRML 97 Editor - Features. *Spazz3D VRML 97 Editor.* [Online] Spazz3D. [Citace: 15. 3 2009.] http://www.spazz3d.com/.

6. The Virtual Reality Modeling Language Specification. *VRML 2.0 Specification.* [Online] Graphcomp. [Citace: 15. 11 2008.] http://graphcomp.com/info/specs/sgi/vrml/spec/.

7. Tišnovský, Pavel. XML + 3D = X3D. *Root.cz.* [Online] [Citace: 18. 2 2009.] http://www.root.cz/clanky/xml-3d-x3d/.

8. VRML. [Online] Wikipedia. [Citace: 23. 11 2008.] http://en.wikipedia.org/wiki/Vrml.

9. Zrzavý, Jakub**.** *VRML Tvorba dokonalých www stránek.* Praha : Grada Publishing spol. s r.o., 1999. str. 204. ISBN 80-7169-643-9.

10. Žára, Jiří. *VRML 97 Laskavý průvodce virtuálními světy.* Praha : Computer Press, 1999. str. 238. ISBN 80-7226-143-6.

11. Žára, Jiří. Příklady z knihy. *VRML 97 - Laskavý průvodce virtuálními světy.* [Online] 1999. [Citace: 9. 12 2008.] http://www.cgg.cvut.cz/LaskavyPruvodce/.

8. SEZNAM OBRÁZKŮ

9. SEZNAM PŘÍLOH

Příloha č. 1 – 3D model domu na CD

10. REJSTŘÍK