TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

BAKALÁŘSKÁ PRÁCE

Virtuální model Ústavu informačních technologií a elektroniky

Liberec 2008 Lukáš Marek

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Virtuální model Ústavu informačních technologií a elektroniky

Lukáš Marek

Studijní program:	B2612 – Elektrotechnika a informatika
Studijní obor:	2612R001 – Elektronické informační a řídicí systémy
Pracoviště:	Ústav informačních technologií a elektroniky
	Fakulta mechatroniky a mezioborových inženýrských studií
	Technická univerzita v Liberci
	Studentská 2, 461 17, Liberec 1
Školitel:	Ing. Jindra Drábková, Ph.D.

Volná stránka pro zadání

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/200 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem Technické Univerzity v Liberci, která má právo požadovat ode mne přiměřený příspěvek na úhradu nákladů vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury pod vedením školitele.

V Liberci 16.5.2008 ……...

Lukáš Marek

Poděkování

Především bych chtěl poděkovat Ing. Jindře Drábkové, Ph.D. za odborné vedení, cenné rady a poskytnuté informace, které napomohly k vzniku této práce.

Abstrakt

Virtuální model Ústavu informačních technologií a elektroniky

Lukáš Marek

Cílem bakalářské práce je seznámit se s prostředím VRML, s možnostmi jeho editace a prohlížení, naučit se pracovat se základními prvky VRML a pomocí těchto znalostí vytvořit virtuální model Ústavu informačních technologií a elektroniky. Model obsahuje informace o rozmístění členů a interaktivní průchody k nim.

Klíčová slova: VRML, VrmlPad, Cosmo Worlds, virtuální model

Abstract

Virtual model of Institute of Information Technology and Electronics

Lukáš Marek

The main topic of bachelor work is to acquaint with VRML environment, with possibilities of editing and browsing, to learn to work with basic elements VRML and by the help of this knowledge to create virtual model of Institute of Information Technology and Electronics in VRML. Model contains information about placement of members and interactive transit to them.

Keywords: VRML, VrmlPad, Cosmo Worlds, virtual model

Obsah

Seznam obrázků

1 Úvod

Cílem této bakalářské práce je seznámit se s prostředím VRML, s možnostmi jeho editace různými editory, naučit se vytvářet základní stavební prvky VRML a prohlížet je v různých prohlížečích. Pomocí těchto znalostí vytvořit z plánku prostoru Ústavu informačních technologií a elektroniky (ITE) virtuální model. Do tohoto modelu doplnit informace o rozmístění členů ústavu. K takto vytvořenému modelu navrhnout různé pohledy na model a vytvořit interaktivní průchody místnostmi ústavu.

1.1 Co je to VRML?

VRML je zkratka Virtual Reality Modeling Language, česky modelovací jazyk virtuální reality. Jedná se o rozšířený grafický formát a jazyk pro vytváření 3D modelů a virtuální reality. Tomuto rozšíření vděčí třem věcem, které VRML charakterizují:

1. Jednoduchosti, pro vytvoření 3D prostoru využívá jednoduchých objektů jako je kvádr, koule, linka, výškové pole a po použití barev, textur dokonce i obrázků a videí lze vytvořit na svém počítači virtuální realitu podobnou té skutečné.
2. Orientaci na internet, vše, co ve VRML vytvoříme, se ukládá v textové podobě, a tak je implementace VRML do internetu jednoduchá.
3. Normě ISO, jazyk VRML se vyskytoval v různých verzích pak ale byl upraven za spolupráce několika firem, odborníků a odborné veřejnosti do podoby mezinárodní formy ISO.

Odpověď na otázku: „Co je to VRML?“ není jednouchá, bude proto lepší říci, k čemu se používá. Používá se k vytváření virtuálních modelů a virtuální reality na počítači. Na internetových stránkách [3] se uvádí, že „VRML je 3D analogický k HTML. To znamená, že VRML motivuje skutečností, že některé informace se nejlépe sdělují ve 3D jako například hry, inženýrství, architektura a jiné další obory vyžadující interakce a zkoumání objektů“.

1.2 Historie VRML

První záznamy o jazyku VRML pocházejí z 80. let minulého století, kdy ve firmě Silicon Graphic, Inc navrhli programátoři knihovnu pro práci s prostorovými objekty, nazvanou Invertor. Tato knihovna byla počátkem 90. let upravena a vylepšena, vznikla tak nová knihovna pro práci s prostorovými objekty OpenGL a k ní aplikační knihovna OpenInvertor. Právě tato knihovna se stala základním kamenem VRML.

Firma Silicon Graphics na konci roku 1995 definuje formát VRML 1.0 pro popis statických světů. Vzniká skupina návrhářů a programátorů nazvaná VAG (VRML Architecture Group) zabývající se vývojem jazyka VRML 1.0, jako další vývoj si určují vytvořit:

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

Zanedlouho vzniká formát VRML 2.0, který splňuje tyto podmínky a v roce 1997 spolupracuje WAG s mezinárodní standardizační organizací ISO na vzniku VRML v podobě mezinárodní normy. Ta nese název DIS 14772-1 a v dubnu 1997 získává název VRML 97.

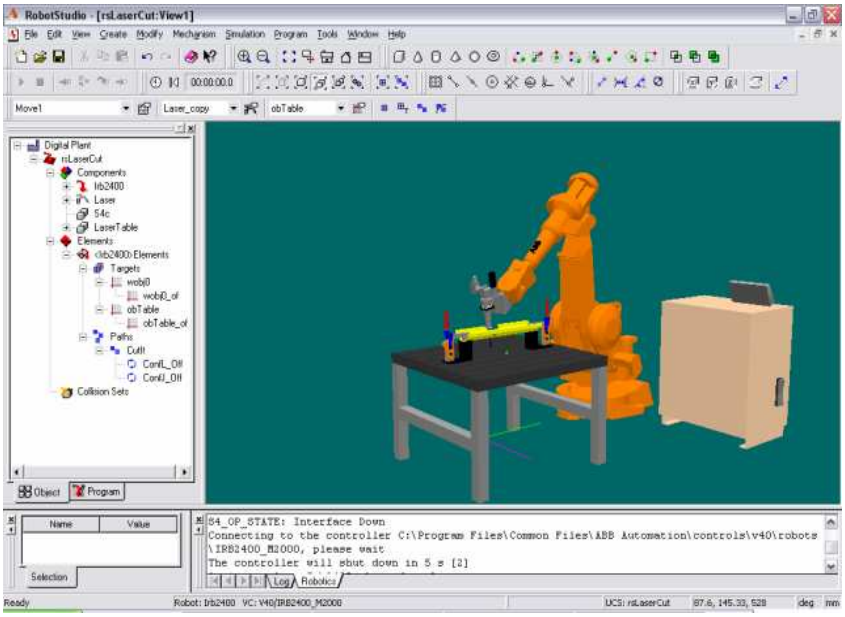
VRML vzniklo rychle a v několika různých formátech. VRML bylo veřejně přístupné na internetu a řada firem si ho upravila a vyvinula pro své potřeby. Vzniklo tak množství prohlížečů a konvertorů do VRML. Tento různorodý vývoj skončil po přijetí ISO normy. V současné době se můžeme setkat s názvy VRML 97, VRML 2.0 a VRML 1.0. VRML 1.0 stará dosluhující verze, VRML 2.0 je pouze programátorské označení VRML 97, jenž se obecně nazývá jednoduše VRML.

1.3 Využití VRML

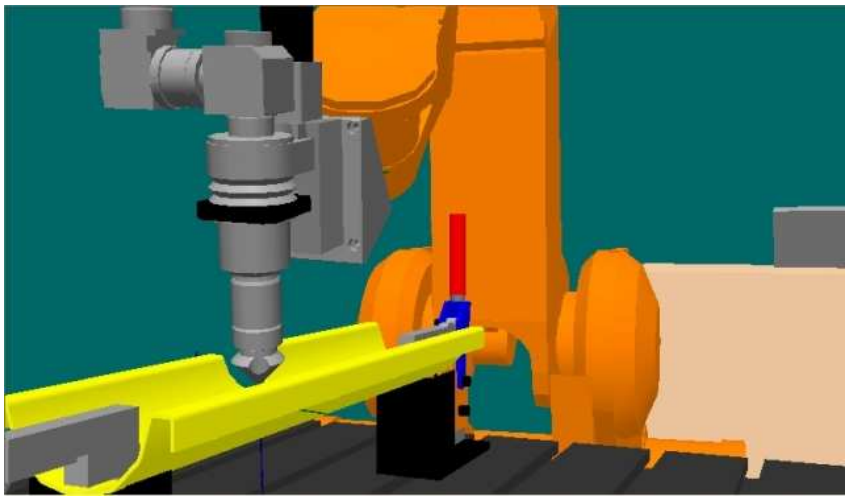
Před vznikem a přijetím VRML bylo vytváření virtuálních modelů výsadou pouze několika drahých programů, například CAD. Ale v těchto programech neprobíhalo zobrazování virtuální reality ve skutečném čase, program nejdříve realitu vygeneroval, a pak ji jako obrázek zobrazil, což je nevhodné pro prezentaci na internetu a větší rozšířenost programů, protože tyto programy byly velice drahé. Vznik VRML přinesl zobrazování virtuální reality v reálném čase a díky své jednoduchosti vložení do www stránek se může každý s přístupem na internet pohybovat ve virtuálním světě a stačí mu k tomu, když bude mít ve svém prohlížeči nainstalovaný VRML prohlížeč.

Využití VRML na internetu se chopily hlavně realitní kanceláře nabízející prohlédnout si budovu zvenku i zevnitř. Umožňují klientovi podívat se na to, jak bude vše vypadat dříve než bude pozdě něco změnit. Další možností je prezentace výrobků. Pomocí 3D modelu si můžete prohlédnout, jak vypadá ze všech stran a dokonce i jak bude vypadat u vás doma. Pro turisty jsou vytvořeny virtuální prohlídky zajímavých míst, historických oblastí, muzei a výstav. Existují i programy pro studenty, například

RobotStudio obrázek 1.1 a 1.2, ve kterém si vložíte virtuálního robota daného druhu, ten lze potom naprogramovat jako skutečného robota a sledovat, jak se hýbe. Škola tak nemusí vlastnit drahé roboty, stačí tento program. Dokonce se dají najít i hry napsané ve VRML.



Obrázek 1.1: RobotStudio 1



Obrázek 1.2: RobotStudio 2

Další příklady projektů s využitím VRML jsou:

Motor: www.tecnolution.com/html/modelli_3d_pro.html

Ljublijana: www.logon.si/nmlj/nmlj-us.html

Přistání na Marsu: www.mars.sgi.com/worlds/4th_planet/models/mp_latest_vlo.html

Výstava: www.mimentx.com/demo-expo-2008/

Prohlídka domu: www.reint.cz/virtualni-prohlidka.php

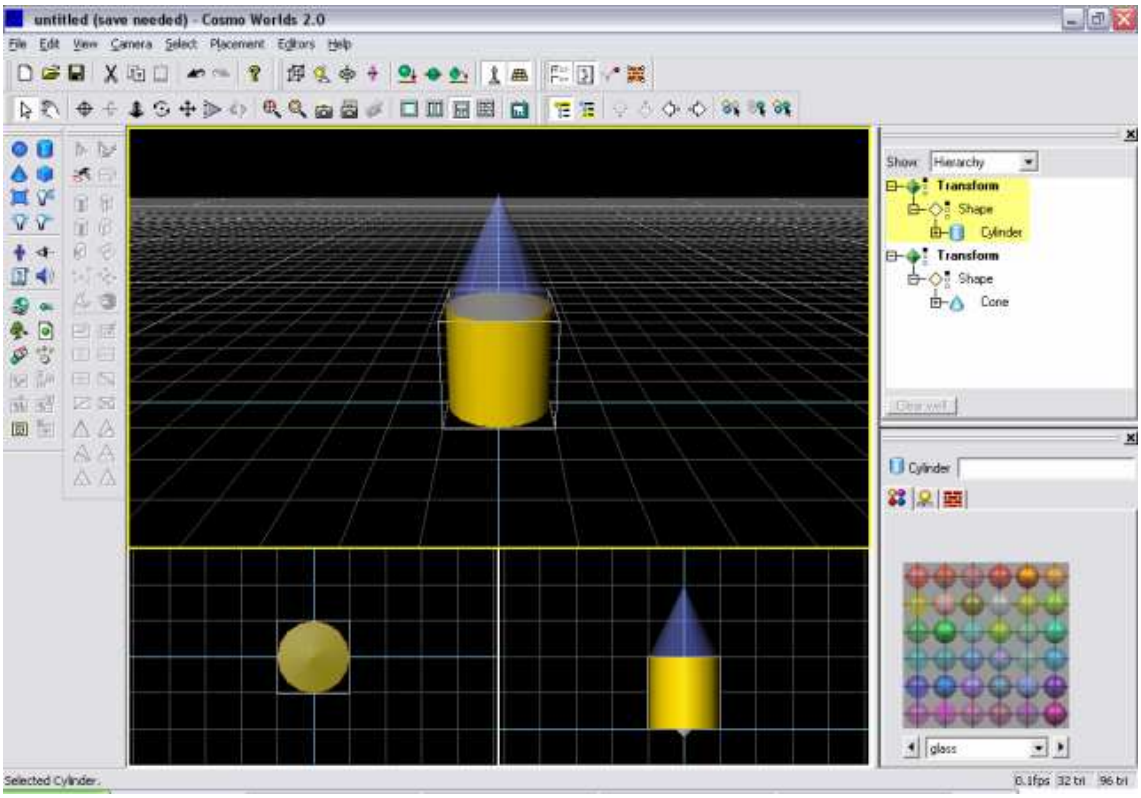
Hry: vrmlworlds.com/games/

2 Editory a prohlížeče

2.1 Editory

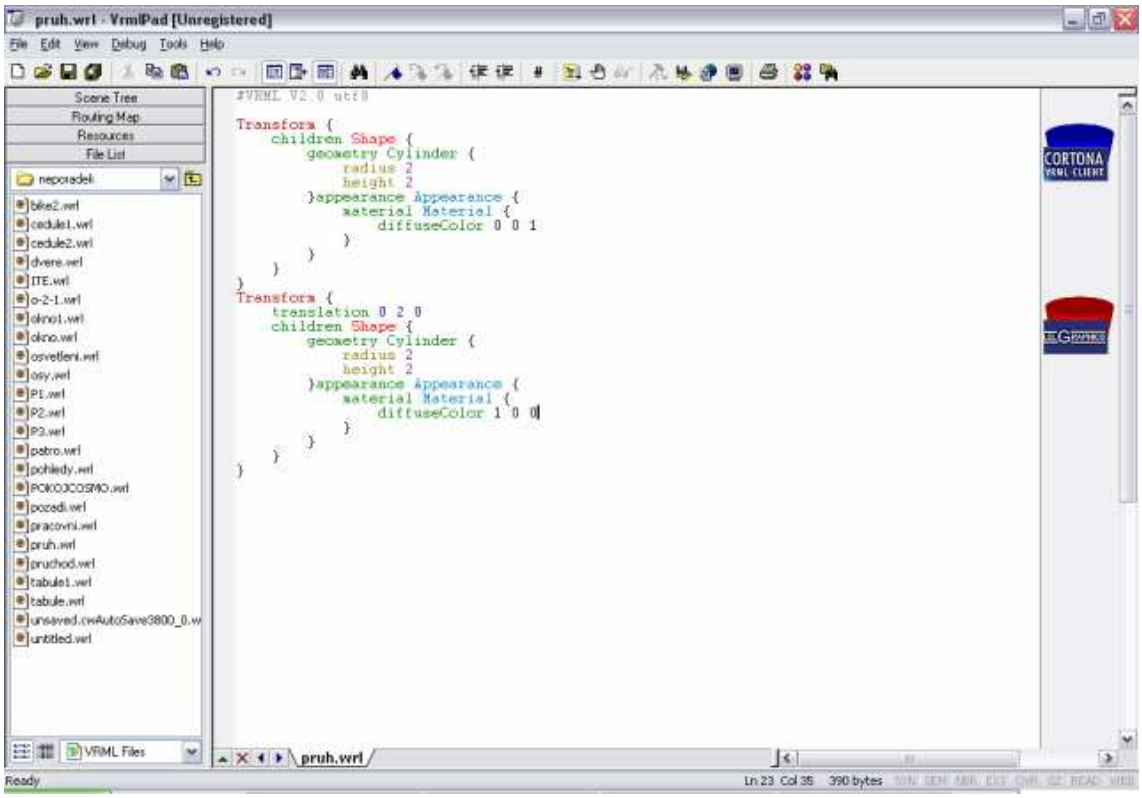
Pro editaci VRML souborů se můžete rozhodnout ze dvou různých způsobů možností editace, a to psaní pouze textově nebo přímo ve 3D prostředí. Každý má své výhody a nevýhody.

Jeden ze zástupců editorů pracující přímo s objekty ve 3D je editor Cosmo Worlds od firmy Silicon Graphics (obrázek 2.1). V Cosmo Worlds slouží pro editaci objektů panel nástrojů se základními tělesy, hlavní okno rozdělené do tří pohledů ROOM, TOP a FRONT, poslední okno zobrazuje aktuální stromovou strukturu souboru a panel s výběrem barvy nebo textury.



Obrázek 2.1: Prohlížeč Cosmo Worlds 2.0

Zástupce textových editorů VrmlPad je o dost jednodušší (obrázek 2.2). Hlavní okno pro psaní textu a dvě postranní okna. V pravém okně se ve verzi 2.0 zobrazují miniatury vytvořených těles. Kromě jiného obsahuje dvě užitečná tlačítka na spuštění souboru v prohlížeči Cortona Control a EditMaterial. Při psaní kódu prohlížeč nabízí výpis dostupných uzlů a parametrů, důležitá je zkratka CTRL + mezerník – ta doplní závorky a iniciální hodnoty aktuálního parametru.



Obrázek 2.2: VrmlPad 2.0

Zkoušel jsem pracovat v obou dvou editorech. Jako lepší pro vytváření složitějších světů mi přišel VrmlPad. Na první pohled jednoduchý, ale vyžaduje od uživatele znalost VRML 97. Výhoda je, že zadáváte přesné hodnoty parametrů a uzlů. Ze začátku se to může zdát pracné a zdlouhavé, ale po chvíli si práci s VrmlPadem osvojíte. CTRL + mezerník a tlačítko s výběrem barvy ušetří spoustu práce. To, že není vidět vzájemná poloha těles, řeší rychlé tlačítko spuštění aktuálního souboru v prohlížeči Cortona Control, který je implantován do VrmlPadu.

Hlavní výhodou Cosmo Worlds je jednoduchost vložení základních těles, jejich rotace a přesun pomocí myši je jednoduchý, ale přesunout dva objekty vůči sobě o nějaký přesný rozměr už je zdlouhavější. Je nutné se „proklikat“ ve stromové struktuře na správný uzel a ručně dopsat hodnoty. Pro vytváření složitějších uzlů je nutné důkladnější seznámení s programem.

Výhody a nevýhody

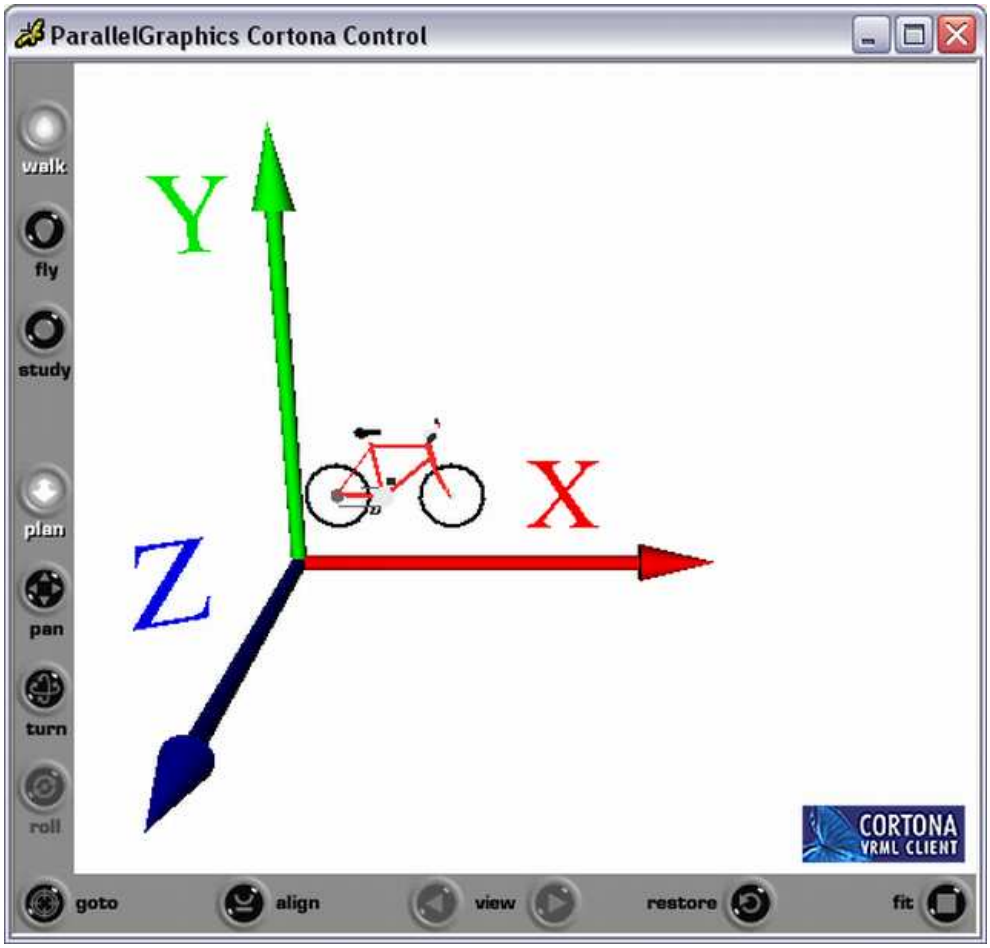
Cosmo Worlds		VrmlPad 2.0	
+	viditelná vzájemná poloha těles	+	psaní přímo kódu
+	snadné nanášení barev a textur	+	rychlé přivyknutí
+	tvorba animací	−	nutnost znalosti VRML 97
+	malá znalost VRML 97	−	pracnost modelování
−	malá přesnost umístění		
−	hledání funkcí a uzlů		
−	při změně pohledů ztráta orientace		

2.2 Prohlížeče

Prohlížeče nám umožní prohlédnout si vytvořený soubor. Existují různé druhy prohlížečů, ale základ mají stejný, a to je základní způsob pohybu ve virtuálním světě. Většina prohlížečů funguje i jako rozšiřující plugin pro internetové prohlížeče, které narazí-li na soubor VRML rovnou ho spustí v prohlížeči. Jako příklad uvedu dostupný prohlížeč ParalellGraphics Cortona Control (viz obrázek 2.3).

- První tři tlačítka nalevo (`WALK`, `FLY`, `STUDY`) slouží k přepínání způsobu pohybu, aktivní jsou jen ta, která jsou autorem souboru definována.
- Další čtyři tlačítka určují na jaký pohyb bude pohyb myši či směrových klávesnic převeden. `PLAN` slouží k pohybu dopředu, dozadu a otáčení do stran, oproti tomu `PAN` pohyb do stran jakoby bokem.
- `TURN` - slouží k otáčení nahoru, dolů a do stran, `ROLL` umožní otočit se do vývrtky hlavou dolů.
- `GOTO` - Pomocí tohoto tlačítka se můžeme přiblížit námi vybranému místu kliknutím požadovaným směrem.
- `ALIGN` - narovná Avatarova záda tzn. pohled se nasměruje tak, aby byl rovnoběžný s rovinou *z*.
- `VIEW` - umožňuje vybírat a přepínat mezi definovanými pohledy.

- `RESTORE` – vrátí pohled do posledního navštíveného definovaného pohledu.
- `FIT` – posune pohled na virtuální svět tak, aby největší zobrazovaný rozměr souhlasil s šířkou nebo výškou okna prohlížeče.

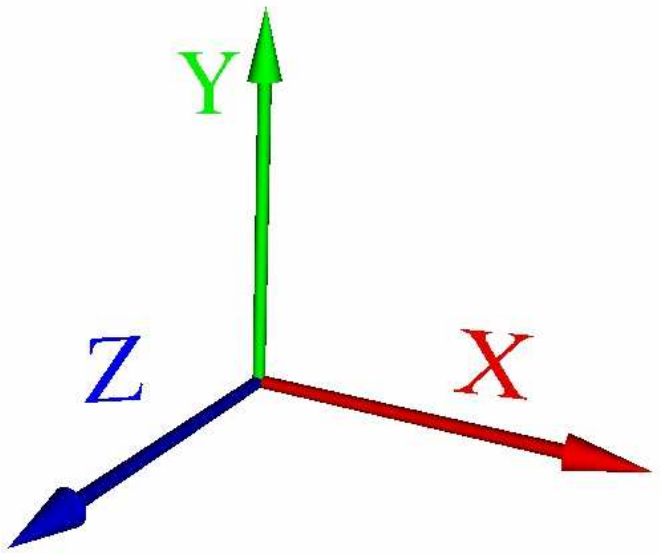


Obrázek 2.3: Cortona VRML prohlížeč

3 VRML 97

3.1 Souřadnicový systém

První, co si musí uživatel VRML osvojit, je orientace os (obrázek 3.1). Při spuštění prohlížeče VRML je základní pohled situován tak, že se prohlíží virtuální svět osou *z* v záporném směru, doprava je kladný směr osy *x* a nahoru kladny směr osy *y*.



Obrázek 3.1: Orientace os

3.2 Struktura souboru VRML

Soubor v jazyce VRML 97 se skládá z uzlů tvořící hierarchickou (stromovou) strukturu. V parametrech uzlů se mohou objevovat jiné uzly a každý tento uzel má následující vlastnosti:

- může být pojmenován,
- obsahuje parametry, parametrem může být i uzel, potom vzniká tzv. vztah rodič-potomek, který je základem pro tvorbu virtuální scény,
- do uzlu lze vložit odkaz na už existující uzel,
- uzly si mohou předávat data,
- všechny identifikátory se rozlišují podle malých a velkých písmen,
- poznámky v programu se označují znakem # .

Soubor taky obsahuje hlavničku, ve které jsou informace pro prohlížeč, o jaký soubor se jedná. Další informace může doplnit do souboru uživatel pomocí uzlu `WorldInfo`. Soubor popisující virtuální svět ve VRML se ukládá s příponou *.wrl

3.3 WorldInfo (Globální informace)

Jedná se o informační uzel, umisťuje se na začátek souboru a slouží pro dokumentární účely.

```
WorldInfo {
     title "Virtuální svět"      # název virtuálního světa
     info ["autor: Lukáš Marek"
,"datum: 30.3.2008"]   # posloupnost jakýchkoliv znaků
v uvozovkách}
```

3.4 ViewPoint (Stanoviště)

Uzel `ViewPoint` slouží k nadefinování různých pozorovacích míst po virtuální scéně. Lze nastavit parametry pozice a natočení pohledu , ten se nastaví tak, že nejprve zadáme osu okolo které se má otočit a úhel v radiánech. `ViewPoint` může soubor obsahovat i více a přepínat lze mezi nimi buď plynule nebo skokem.

```
Viewpoint {
      position 0 1.8 5  # pozice ze které pozorujeme virtuální
svět
      orientation 1 0 0 -0.1  # změna směru pohledu
      description "pohled"    # pojmenování pozice
      fieldOfView 0.785398    # zorný úhel
      jump TRUE               # povolení plynulého přechodu na
stanoviště,FALSE = přechod skokem }
```

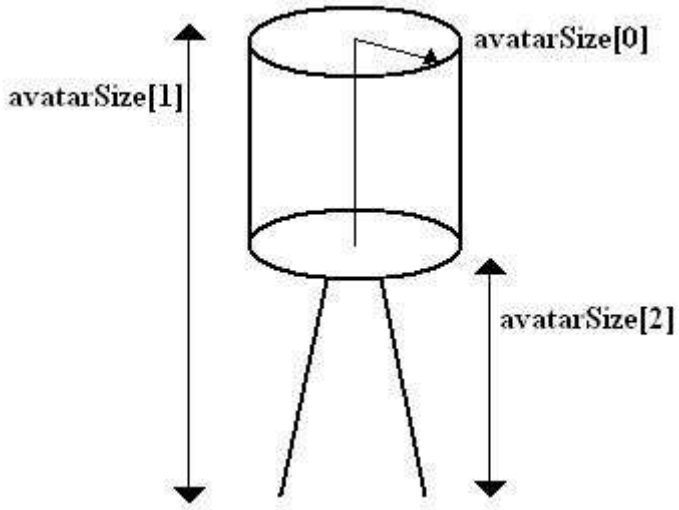
3.5 Avatar

Avatar je dvojník uživatele ve virtuálním světě. Uživatel vidí v prohlížeči to, co vidí Avatar, pokud se tedy chceme podívat doprava musíme otočit Avatara doprava. Některé prohlížeče obsahují tlačítka, které vrátí Avatara do poslední navštívené definované pozice nebo skloní pohled tak, aby byl v rovině *x*, *y*, tzn. narovnají Avatarova záda. Důležité ale jsou Avatarovy vlastnosti definované uzlem `NavigationInfo`.

3.5.1 NavigationInfo

Uzel `NavigationInfo` obsahuje vlastnosti Avatara, které jsou předdefinované tak, aby odpovídaly člověku.

```
NavigationInfo {
     avatarSize [0.25, 1.6, 0.75]   # rozměry Avatata viz
obrázek 3.2
     headlight TRUE  # Avatarova svítilna zapnuta, FALSE =
vypnuta
     visibilityLimit 0    # dohled, 0 = nekonečno
     speed 1              # rychlost přesunu
     type ["WALK", "ANY"] # nastavení způsobu pohybu }
```



Obrázek 3.2: Rozměry Avatara

`avatarSize[0]` je vzdálenost jak blízko se může Avatar přiblížit k objektu, tzn. v uvedeném příkladě neprojde škvírou užší než 0,5 .

`avatarSize[1]` je výška Avatara tzn. neprojde dveřmi nižšími než 1,6 .

`avatarSize[2]` je výška, na kterou Avatar dokáže vystoupit tzn. nevyjde schod vyšší než 0,75 .

Posledním parametrem navigačního uzlu je seznam povolených možností jak Avatara řídit, parametr `TYPE`. Existují tři základní způsoby pohybu Avatara doplněné dvěma dalšími možnostmi. Uvedené způsoby uvádí J. Žára v knize [2]:

- WALK – chůze po podložce, Avatar se pohybuje podle zvlnění podložky a překonává překážky o velikosti `avatarSize[2].` Působí na něj gravitační zrychlení v záporném směru poloosy Y. Je zapnuta detekce kolizí, Avatar neprojde zdí.

- FLY – let, stejné vlastnosti jako walk pouze je vyřazena přitažlivost a tak muže Avatar létat.
- EXAMINE – způsob, při kterém je Avatar nejméně omezován běžnými fyzikálními zákony. Je nejlepší ke zkoumání objektů ze všech stran a úhlů. Rychlost pohybu je nezávislá na parametru SPEED. Je vypnuta detekce kolizí.
- ANY – nejedná se o způsob ovládání, ale o doporučení prohlížeči, že může uživateli povolit přepínání mezi typy. Implicitní nastavení je WALK a ANY, není-li hodnota ANY zadána, lze volit jen mezi vypsanými typy.
- NONE – prohlížeč schová ovládací prvky a současně přestane převádět pohyb myši na pohyb Avatara. Tento způsob se používá pouze, když virtuální svět obsahuje aktivní tělesa, schopná po doteku přesunout Avatara na další stanoviště.

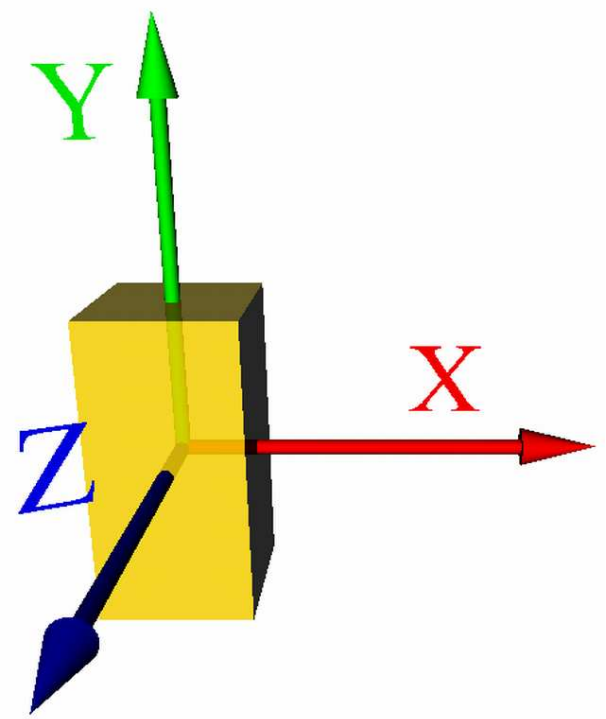
3.6 Základní geometrická tělesa

Základním stavebním kamenem tvoření virtuálního světa jsou ve VRML čtyři jednoduchá tělesa, `Box`, `Cone`, `Cylinder`, `Sphere`, česky: kvádr, kužel, válec a koule. Bez použití posunutí je střed všech těles v počátku, mimo kužel, kde je počátek v místech odpovídající středu válce. Více je možné vidět v příkladech.

3.6.1 Box (kvádr)

Kvádr má pouze jeden parametr a `size`, do kterého se zapíší požadované rozměry *x*, *y*, *z* . Poloha kvádru se počítá od jeho středu jak je vidět na obrázku 3.3.

```
Transform {
      children Shape {
            geometry Box {
                  size 2 4 2
}}}
```

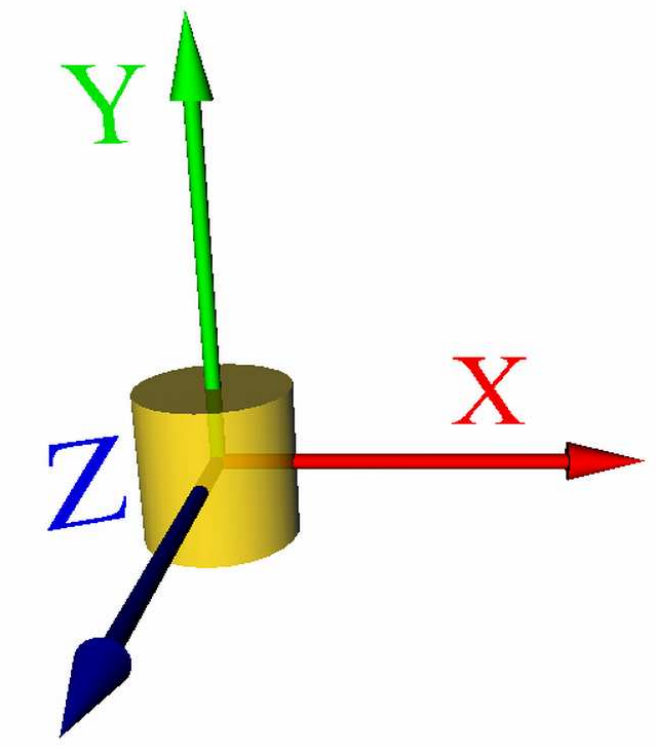


Obrázek 3.3: Box (kvádr)

3.6.2 Cylinder (válec)

Válec má parametry `height` (výška) a `radius` (poloměr podstavy). Poloha válce se počítá jako u kvádru od jeho středu. U válce lze nastavit viditelnost jednotlivých ploch pomocí parametrů `top, bottom, side`. Tyto parametry slouží pro nastavení viditelnosti pláště `side`, a podstav `top` a `bottom`. Nastavením viditelnosti pouze pláště nevznikne trubka, protože vnitřní strana těles není vykreslována, z důvodu snížení nároků na vykreslení virtuální reality.

```
Transform {
      children Shape {
            geometry Cylinder {
                  height 2          # výška
                  radius 1          # poloměr podstavy
                  top TRUE          # viditelnost horní podstavy
                  bottom TRUE       # viditelnost spodní podstavy
                  side  TRUE        # plášť válce
}}}
```

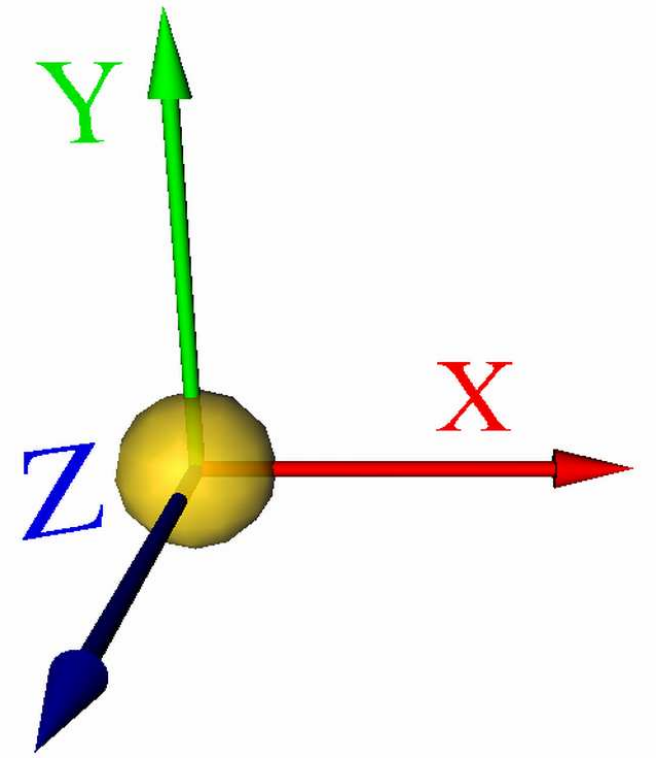


Obrázek 3.4: Cylinder (válec)

3.6.3 Sphere (koule)

Uzel koule je nejjednodušší uzel s parametrem `radius` (poloměr).

```
Transform {
     children Shape {
          geometry Sphere {
            radius 1
 }}}
```

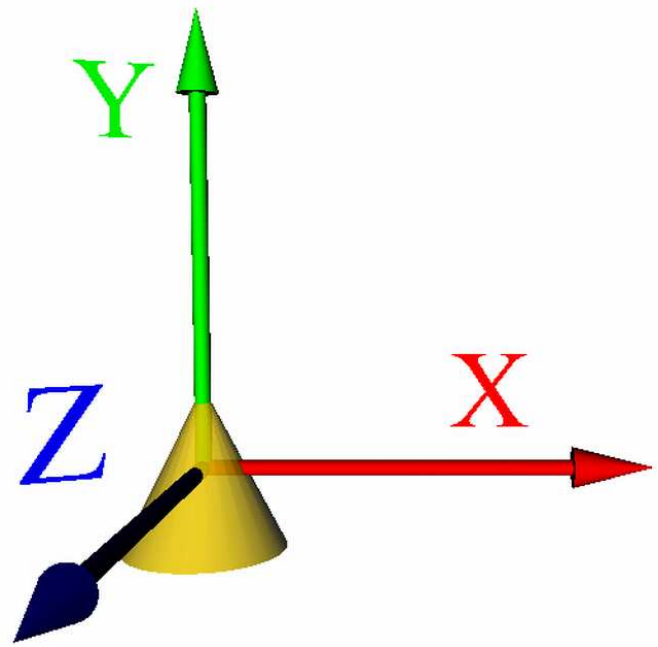


Obrázek 3.5: Sphere (koule)

3.6.4 Cone (kužel)

Kužel je co do počtu parametrů nejsložitější základní těleso. Parametr `height` značí jak má být vysoký a `bottomRadius` poloměr podstavy, navíc jsou tu uzly `bottom` a `side`. Je-li `bottom` nastaven na FALSE z kuželu není vidět podstava a kužel je zespodu neviditelny, je vidět pouze vnější plášť. Parametr `side` zase skryje plášť kuželu a je vidět pouze podstava a to jen zespodu.

```
Transform {
      children Shape {
            geometry Cone {
                  height 2          # výška kužele
                  bottomRadius 1    # průměr podstavy
                  bottom TRUE       # viditelnost podstavy
                  side TRUE         # viditelnost pláště
}}}
```



Obrázek 3.6: Cone (kužel)

3.7 Transformace

Abychom mohli výše uvedená tělesa umístit do virtuálního světa, je třeba naučit se pracovat se dvěma uzly. S jejich pomocí se vytváří stromové struktury a shrnují objekty a jejich vlastnosti do jednoho celku.

3.7.1 Transform

Tento uzel je tzv. rodičovský, který definuje souřadnicový systém pro své potomky zapsané do parametru children. Uzel `Transform` obsahuje tři parametry `scale`, `translation`, `rotation` a jeden uzel (potomek) Shape.

3.7.2 Shape

Uzel `Shape` určuje geometrické tvary a vlastnosti povrchu jednotlivých objektů. Obsahuje jeden parametr `geometry` určující tvar a parametr `appearance` určující povrchové vlastnosti.

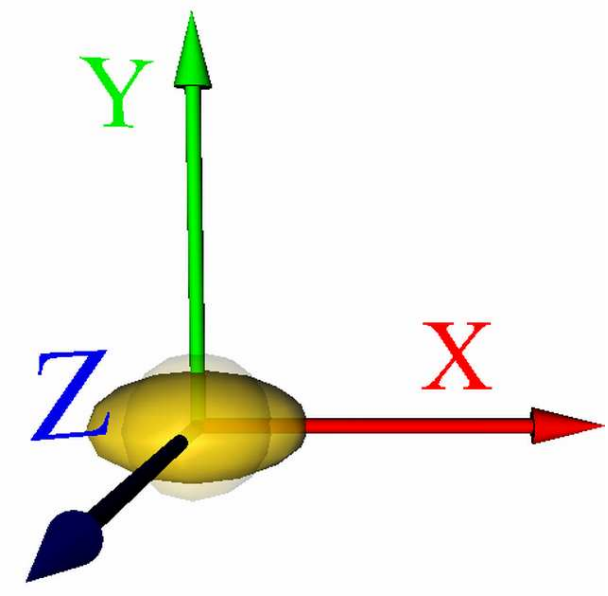
3.8 Měřítko, posun a otočení

Po vložení objektu do virtuálního je možné těleso někam posunout a otočit, k tomu použijeme parametry uzlu `Transform` a to `translation` (posunutí), `rotation` (otočení) a `scale` (měřítko).

3.8.1 Scale (měřítko)

Pomocí `scale` můžeme měnit měřítko podél každé z os *x*, *y*, *z*. Je tak možné vytvořit z koule např. elipsoid (obrázek 3.7).

```
Transform {
     scale 1.5 0.75 1   # změní rozměr koule o poloměru jedna na
     x = 1,5  y  = 0,75 a z = 1
     children Shape {
          geometry Sphere {
                radius 1
     }}}
```

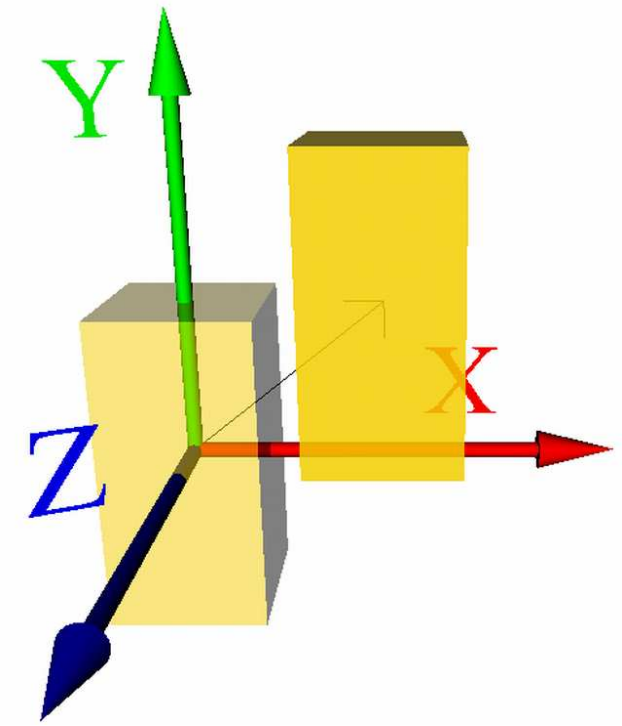


Obrázek 3.7: Scale (měřítko)

3.8.2 Translation (posun)

Parametr `translation` slouží k posunu objektu po osách.

```
Transform {
     translation     2.5 2 0   # posun o x 2,5 a y o 2
     children Shape {
          geometry Box {
               size 2 4 2 }}}
```

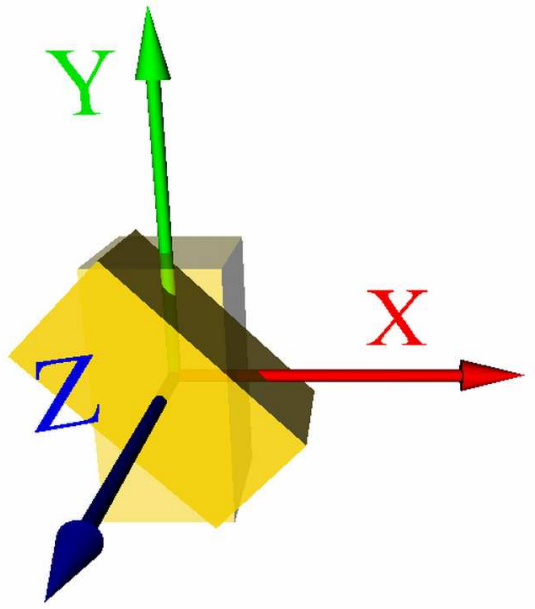


Obrázek 3.8: Translation (posun)

3.8.3 Rotation (osa a otočení)

Parametr `rotation` slouží k otočení objektu podél zadané osy o daný úhel, který se zadá v radiánech. Chceme-li otočit těleso podél osy Z o 45° napíšeme:

```
Transform {
      rotation  0  0 1 0.785
      children Shape {
            geometry Box {size 2 4 2}}
```



Obrázek 3.9: Rotation (otočení)

3.9 Povrch těles

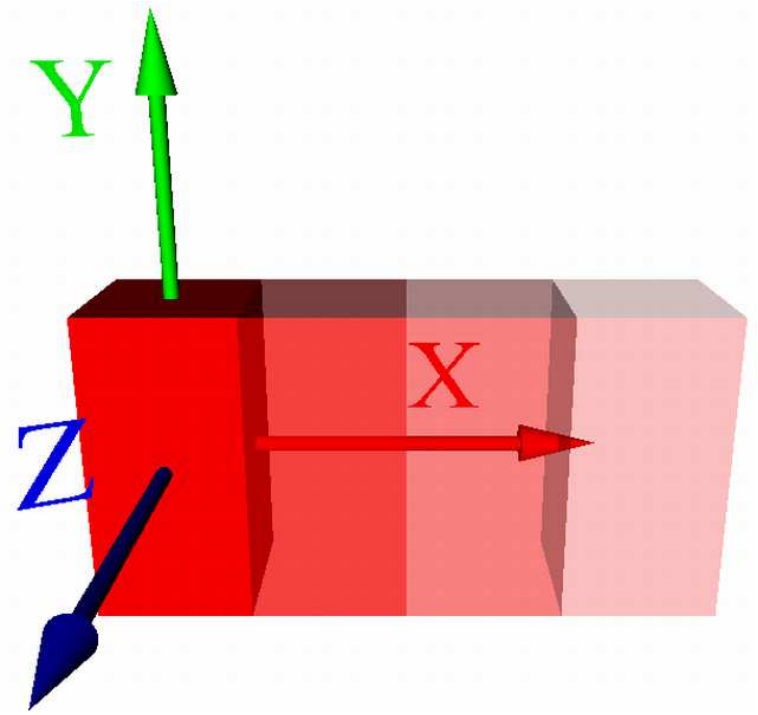
Aby virtuální svět nebyl černobílý, primární barva pozadí je černá a těles bílá, lze vzhled tělesa změnit pomocí uzlu `Appearance`, který je umístěn v parametru `appearance`.

3.9.1 Obarvení těles a textura

Pomocí uzlu `Appearance` můžeme těleso obarvit nebo použít texturu. Barvu vybíráme z klasické palety RGB – red, blue, green, požadovanou barvu namícháme zvolením intenzity jednotlivých složek z intervalu 0 až 1, kde 1 je plná intenzita barvy.

```
Transform {
 children Shape {
       geometry Box {
             size 2 4 2
       }    appearance Appearance {
             material Material {
```

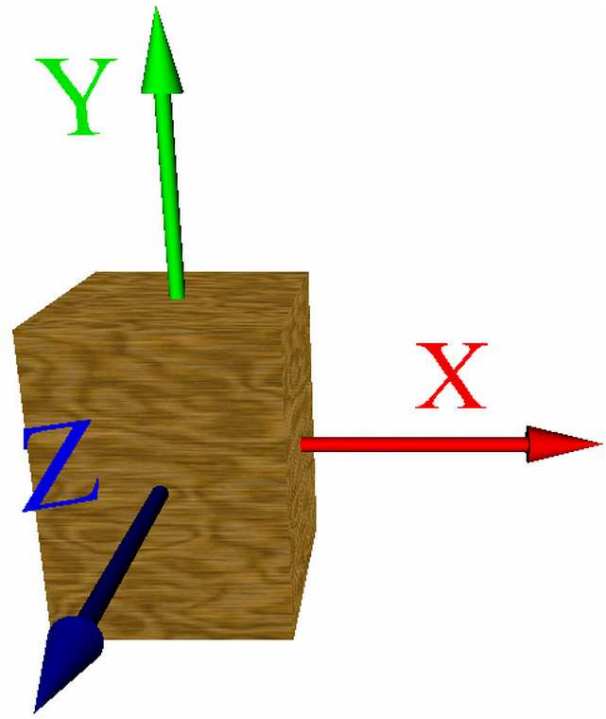
```
                diffuseColor 1 0 0     # Červená barva
                emissiveColor 0 0.5 0 # Barva kterou bude
vyzařovat
                transparency 0.5       # Průhlednoust 1/2
}}}}
```



Obrázek 3.10: Barvy a průhlednost (0, ¼, ½, ¾)

Při použití textury se jedná o obrazový vzorek, který je nanášen na povrch tělesa. Jako texturu lze použít uzel `ImageTexture` (obrázek), `MovieTexture` (video) nebo `PixelTexture` (textura vytvořená pomocí pixelů). Nanášet texturu na obarvené těleso je zbytečné, protože barva není vidět a také je to výpočetně náročnější, vypočítává se barva i textura, také může nastat problém při zobrazení v jiném prohlížeči.

```
Transform {
       children Shape {
             geometry Box {
                   size 3 4 3
             }    appearance Appearance {
                  texture    ImageTexture {
                   url ["drevo.jpg"]
}}}}
```



Obrázek 3.11: Textura – dřevo

3.10 Obecnější tělesa

Abychom mohli vytvořit realističtější vzhled virtuálního světa, budeme potřebovat obecnější tělesa než jsou čtyři základní. K tomu slouží dalších šest uzlů. Skupina prvních tří pracuje s plochami na vytvoření 3D tělesa, zbylé tři mají dvojrozměrný charakter:

1. `IndexedFaceSet` (Množina ploch)
2. `Extrusion` (Opláštění)
3. `ElevationGrid` (Výšková mapa)
4. `IndexedLineSet` (Množina čar)
5. `PointSet` (Množina bodů)
6. `Text` (Nápis)

Popis těchto uzlů a jejich parametrů je rozsáhlý, a proto zde vypíšu jen uzel, který jsem použil.

3.10.1 Text

Uzel `Text` nám umožní umístit nápis do prostoru, lze jej otáčet a přemístit (pomocí rodičovského uzlu `Transform`), nanášet barvu, texturu a definovat průhlednost. Nápisy nepodléhají kolizím s Avatarem, lze ho projít a prohlídnout si nápis ze zadní strany, kde je zrcadlově převrácený, a proto je dobré umístit jej na plochu či

těleso. Bohužel nelze používat diakritická znaménka. Text lze nastavovat mnoha různými parametry:

```
Transform {
 translation    2 2 0
  children Shape {
   geometry Text {
      string ["VRML 97",]      # libovolný řetězec
      length 10                # seznam délek řádků
      maxExtent 0              # maximální délka řádku
      fontStyle FontStyle {
      size 1                   # velikost písma
      spacing 1                # relativní řádkování, skutečné
                               řádkování size × spacing
      family "SERIF"           # seznam rodin písma
      style  "PLAIN"           # styl písma
      horizontal TRUE          # psaní vodorovně
      leftToRight TRUE         # psaní zleva doprava
      topToBottom TRUE         # psaní shora dolů
      justify "MIDDLE"         # zarovnání
}}}}
```

Jiné nastavení některých parametrů:

1. `Family`:
   - `SERIF`: patkové písmo
   - `SANS`: bezpatkové písmo
   - `TYPEWRITER`: písmo s konstantní šířkou znaku
2. `Style`:
   - `PLAIN`: normální písmo
   - `BOLD`: polotučné
   - `ITALIC`: kurzíva
   - `BOLDITALIC`: polotučné kurzíva
3. `Justify` (zarovnání)
   - `BEGIN`: zarovnání doleva vůči vztažnému bodu *x*
   - `FIRST`: stejné jako begin, y souřadnice spodní linka řádku
   - `MIDDLE`: zarovnání na střed
   - `END`: zarovnání doprava

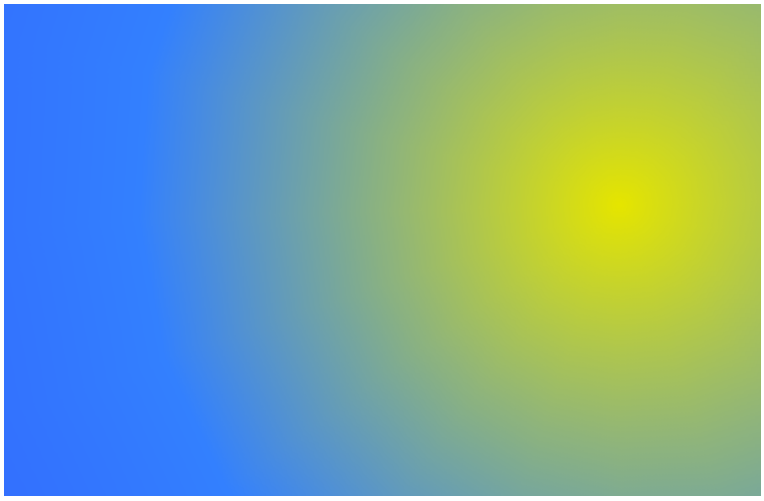
Při používání textu se zvyšují výpočetní nároky, protože text je převeden do velkého množství malých plošek, která se vypočítává každá zvlášť.

3.11 Další uzly a parametry použité ve virtuálním modelu

3.11.1 Background (pozadí)

Vzhled virtuálního světa je vystižen primárně nastavenou černou barvou pozadí. Pomocí uzlu `Background` (pozadí) můžeme toto pozadí změnit a změnit tak celkový vzhled našeho modelu. Pozadí může mít podobu krychle nebo koule, krychle se využívá hlavně při použití textury a koule při nanesení barev. Koule je rozdělena do dvou polokoulí představující nebe (sky) a zemi (ground). Uzel `Background` obsahuje parametry `skyAngel`, `skyColor`, `groundAngel` a `groundColor` určující barvy a úhly. Barev musí být definováno o jednu více než úhlů první barva se použije na úhel od vrcholu k prvnímu zadanému úhlu.

```
Background {
  groundAngle [ 0.1, 1.5 ] # rozdělení dolní polokoule
  groundColor [ 0 0.75 0, 0 0.75 0, 0 0.75 0 ] # barvy
  skyAngle [ 0.7, 1.5 ] # rozdělení hodní polokoule
  skyColor [ 0.9 0.9 0, 0.2 0.5 1, 0.2 0.3 1.0 ]}
# barvy oblohy žlutá, světle modrá a modrá vytvoří iluzi
slunce na nebi
```

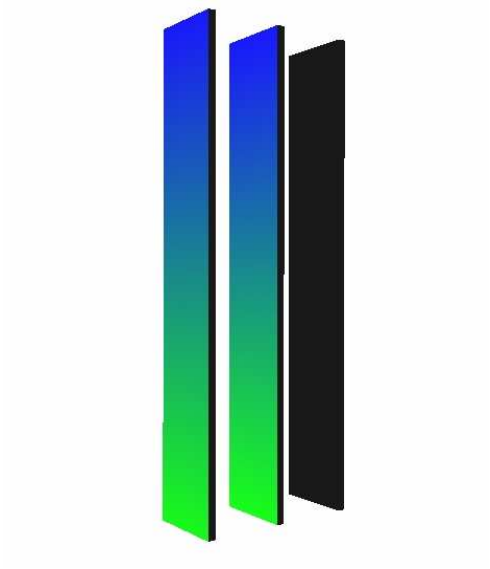


Obrázek 3.12: BackGround (Slunce)

3.11.2 PointLight (světlo)

Zatím náš virtuální svět osvětlovalo pouze světlo z Avatarovy svítilny. Když ji vypneme, jsou vidět pouze objekty s nastaveným vyzařováním světla. Virtuální svět můžeme osvětlit ještě dalšími uzly: `PointLight` (bodové světlo), `DirectionalLight` (zdroj rovnoběžných paprsků), `SpotLight` (reflektor). Nevýhodou světel je, že prochází objekty a osvětlují objekty, které by měly být ve stínu.

Uzel `PointLight` (bodové světlo), osvětluje celý virtuální svět ve svém definovaném okolí. Intenzita světla klesá směrem k okraji kulové hranice až k nule. Útlum světla lze upravit parametrem `attenuation` s koeficienty a_1, a_2, a_3, pak se počítá útlum podle vzorce $\frac{1}{a_0 + a_1 d + a_2 d^2}$, kde d je vzdálenost od zdroje. Na obrázku 3.13 svítí modré a zelené světlo na šedé kvádry a je vidět, že světlo prochází skrz a zadní kvádr je mimo dosvit.



Obrázek 3.13: Osvětlení

3.11.3 Anchor (odkaz)

Uzel `Anchor` nám umožní umístit do virtuálního světa odkaz na nový `ViewPoint`, na jiný svět nebo na www stránku.

```
Transform{
     children Anchor {url "#POHLED"
   # po kliknuti na kvádr se přesune Avatar na ViewPoint POHLED
          children Shape {
          geometry Box {size 5 5 0.1}}}
```

```
Transform {
      children Anchor {
      parameter "target=_blank"     # otevře se v novém okně
      url  "http://www.ite.tul.cz/" # hypertextový odkaz
      children Shape {
            geometry Text {
            string ["WWW.ITE.CZ"] # po kliknutí na text se otevře
nové okno s www stránkou }}}}
```

3.11.4 Inline

Uzel `Inline` využijeme v případě, že chceme použít již vytvořený soubor. Svět vložíme pomocí parametru `url,` do kterého zapíšeme cestu k požadovanému souboru. Vložený svět se zobrazí tak, že jeho nulová souřadnice souhlasí s nulou v aktuálním souboru, proto je dobré uzel `Inline` zapsat do uzlu `Transform` a pomocí něho jej posunovat a otáčet.

```
Transform {
      translation     0 5 0
      children Inline {
            url  "ite.wrl"
}}
```

3.11.5 ProximitySensor

`ProximitySensor` je dynamický uzel, který detekuje přítomnost vatara v určitém prostoru definovaném jako neviditelný kvádr.

```
DEF   SENSOR ProximitySensor {
      center 1.175 1 5.8 # umístění sensoru
      size 1.05 2 4 } # rozměry snímače
ROUTE SENSOR.enterTime TO CASOVAC.startTime # propojení
parametrů, při detekci Avatara se spustí časovač
```

3.11.6 TimeSensor

Chceme-li ve virtuálním světě něčím pohnout jinak než skokově, budeme potřebovat `TimeSensor` pro nastavení času, jak dlouho bude děj probíhat.

```
 TimeSensor {
      cycleInterval 10} # čas deset sekund
```

3.11.7 Interpolátory

Jedná se o dynamické uzly umožňující pohybovat objekty ve virtuálním světě. Důležité jsou uzly `PositionInterpolator` (lineární změna polohy) a OrientationInterpolator (lineární změna otočení). Použití interpolátorů je vidět v kapitole 3.11.8.

3.11.8 Průvodce

Kromě virtuálního světa s různými pohledy na něj, lze vytvořit předem definovanou trajektorii, po které se bude Avatar pohybovat a natáčet.

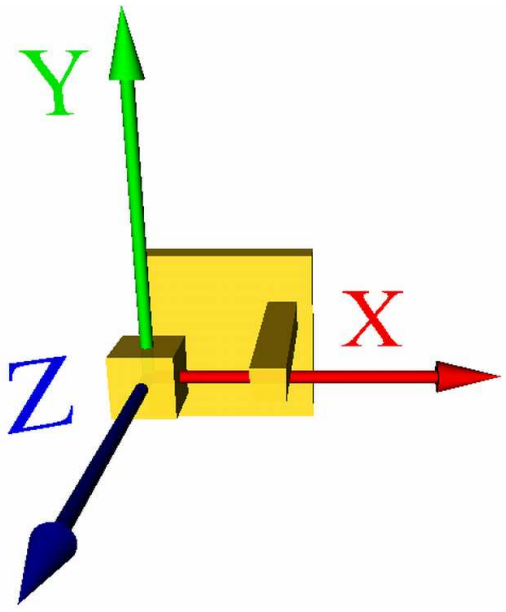
```
DEF   PRUVODCE Viewpoint { # náš průvodce
      description     "pruvodce" position   0 0 5}

DEF   CAS TimeSensor {
      cycleInterval 10} # čas našeho procházení

DEF   TRAJEKTORIE PositionInterpolator {
# interpolátor na změnu polohy
      key   [0.25, 0.5, 0.75, 1] # rozdělení na časové intervaly
      keyValue [0 0 5,0 0 0,2 0 0,2 0 -5  ]} # pozice kterých
Avatar dosáhne za zvolený interval
DEF   OTOCENI OrientationInterpolator {
      key   [ 0.75, 1]
      keyValue [0 1 0 0, 0 1 0 1.57,] # otočení Avatara
}
ROUTE PRUVODCE.bindTime    TO CAS.startTime
# přiřadí start časovače aktivaci pohledu průvodce
ROUTE CAS.fraction_changed TO TRAJEKTORIE.set_fraction
# aktivuje PositionInterpolator
ROUTE CAS.fraction_changed TO OTOCENI.set_fraction
# aktivuje OrientationInterpolator
ROUTE TRAJEKTORIE.value_changed TO PRUVODCE.position
# změní polohu Avatara
ROUTE OTOCENI.value_changed     TO PRUVODCE.orientation
# změní otočení Avatara
```

3.12 DEF, USE

Jedná se o příkazy, které přijdou vhod ve chvíli, kdy budeme potřebovat použít vícekrát barvu, těleso či více těles. Stačí si pojmenovat nějaký uzel příkazem DEF a poté ho můžeme používat příkazem USE a jméno. Tyto příkazy ušetří práci jak nám, tak i počítači při výpočtu.

Při použití DEF a USE je dobré dát si pozor na pár pravidel: definovaný uzel lze použít pouze v tomtéž souboru, když z nějakého důvodu použiji jedno jméno dvakrát, po použití příkazu USE se vykreslí uzel, který je definován blíže. Poslední pravidlo uvádí, že nelze použít definovaný uzel uvnitř sebe sama, to by vedlo k zacyklení prohlížeče, který by se snažil vytvořit rodiče s nekonečným počtem potomků.

```
Transform {
      Translation 0 0 0
      children DEF Kostka Shape {
      geometry Box { size 1 1 1 }}}
Transform {
      translation 1 0 -2
      scale 3 3 0.5
      children USE Kostka }
Transform {
      Translation 2 0.5 0
      scale 0.5 0.5 3
      children USE Kostka }
```



Obrázek 3.14: DEF, USE

4 Vytvoření virtuálního modelu

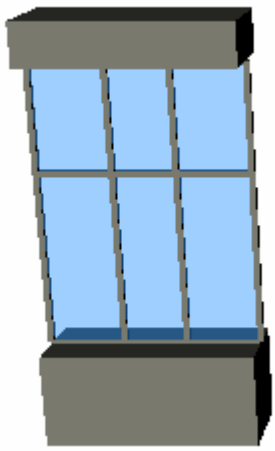
Cílem bakalářské práce je vytvoření virtuálního modelu Ústavu informačních technologií a elektroniky (dále pouze ITE) včetně informací o umístění jejích členů.

4.1 Vytváření modelu

4.1.1 Vytvoření zdí a oken

Prvním krokem pro vytvoření virtuálního modelu ústavu ITE bylo získání nějakého rastru budovy a přibližných rozměrů. Jako plánek budovy posloužila evakuační mapa umístěná v budově. Do zvětšené mapy ústavu jsem si zanesl několik rozměrů, které jsem si přesně přeměřil, jednalo se hlavně o tloušťky zdí, šířky dveří a oken. Abych si ušetřil práci, neměřil jsem všechny rozměry všech místností, ale změřil jsem si pouze několik rozměrů a zbylé rozměry jsem si dopočítával pomocí měřítka změřeného z plánku. Samostatně jsem si přeměřil rozměry oken a dveří. Naštěstí šíře oken i dveří byly pouze dva typy oken a dveří a tak si stačilo do plánku poznamenat, který typ kam patří.

Po vyzkoušení různých editorů jsem se rozhodl pro VrmlPad, jeho funkce byla plně dostačující pro mé požadavky na rychlost a přesnost objektů a jejich umístění. Nejprve jsem si vytvořil model okna, abych ho mohl pomocí uzlu `Inline` vkládat do výsledného modelu. Pro ušetření práce jsem okno vytvořil už v požadované výšce a se zdí nad a pod oknem. Použití uzlu `Inline` (viz kapitola 3.11.4).

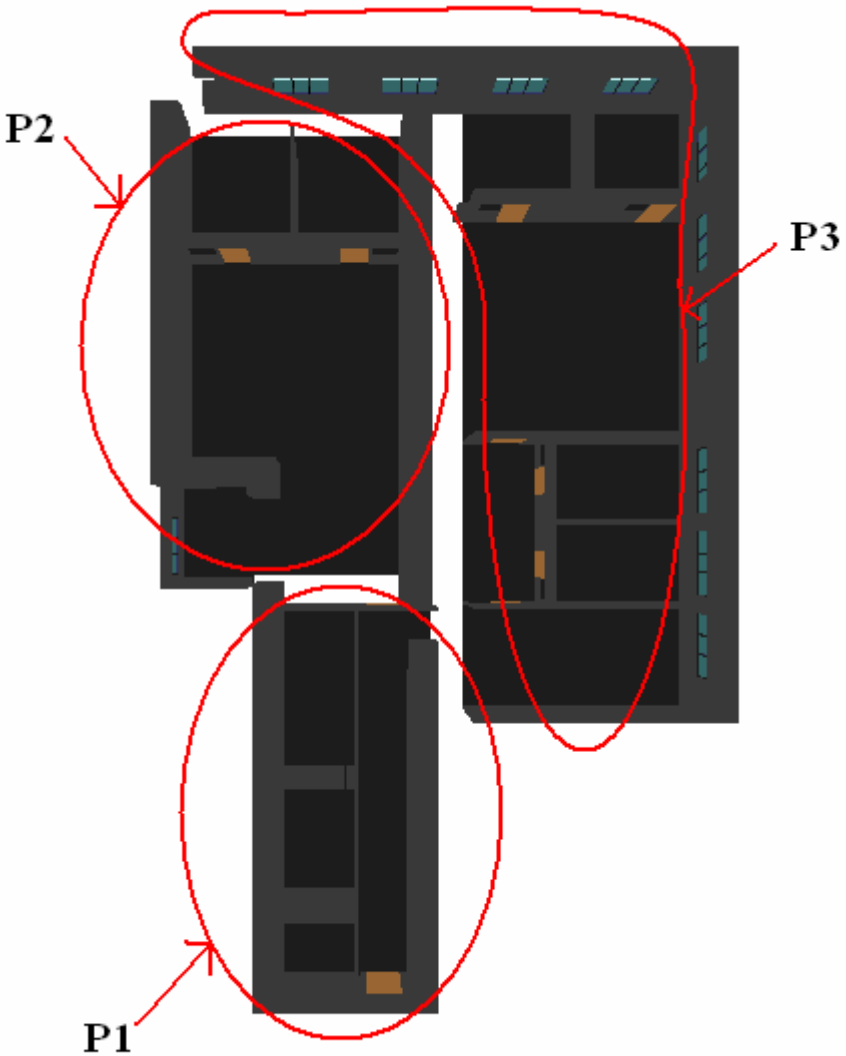


Obrázek 4.1: Předefinované okno

Dveře jsem si zatím udělal jako pouhé kvádry o dvou šířkách 85 cm a 105 cm, abych po vytvoření modelu mohl dveře vylepšit a udělat je otevírací. Kvůli snadnější orientaci jsem se rozhodl vytvářet model po směru hodinových ručiček a dodržovat

komentování každého uzlu. Při prvním pokusu vytvořit stěnu s okny se naskytla příležitost použít příkaz DEF a USE (viz kapitola 3.12), okna jsou stejně daleko a se stejnými rozestupy, a tak jsem si sloup a okno definoval pomocí DEF a poté jej dvakrát použil na vytvoření celé stěny, pouze jsem dodělal krajní stěnu tak, aby souhlasila její celková délka. Všechny stěny jsem definoval jednou šedivou barvou tak, aby bylo možné později barvu jednoduše a rychle změnit, pouze místo pro dveře jsem nechal bílou barvou, abych viděl jsou-li správně umístěné. Pro snadnější orientaci v modelu jsem občas použil jako orientační bod vysoký tenký kvádr, který představoval osu *y*.

Po vytvoření několika stěn, byl zdrojový kód už docela dlouhý, a proto jsem se rozhodl rozdělit celý ústav do tří samostatných souborů P1, P2 a P3, jak je vidět na obrázku 4.2.



Obrázek 4.2: Rozdělení

Aby tyto tři části na sebe přesně navazovaly, udělal jsem společnou prostřední zeď tak, aby část přečnívala do stěny v druhém souboru a při jejich přesném spojení není tento přesah vidět. Při vytváření druhé části jsem si nejdříve celou část nakreslil na papír, vypočítal a zapsal kóty s rozměry. Zadní společnou stěnu jsem vytvořil až v posledním souboru P3, protože bylo jednoduší udělat celou stěnu najedou. Jinou šíři oken, než jsem měl vytvořené okno, vyřešila vhodná změna měřítka, parametr `size`. Příkazy `DEF` a `USE` jsem využil i na vnitřních zdech, které jsou stejné jen posunuté. Po vytvoření všech tří částí jsem je v souboru ITE spojil v jeden celek uzlem `Inline`.

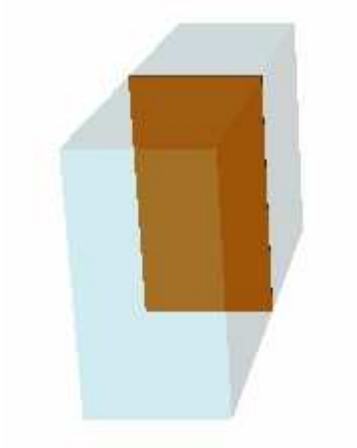
Celý model jsem vytvořil bez stropů, protože jsem chtěl zachovat možnost prohlédnout si rozmístění pokojů shora tak, jak to není možné ve skutečnosti.

4.1.2 Přidání dveří

Po vytvoření zdí a oken přišlo na řadu nahrazení bílých kvádrů za dveře, původně jsem to plánoval tak, že bych si vytvořil dva soubory s dveřmi o šířce 85 cm a 105 cm a pomocí `Inline` je vložil místo vytvořeného bílého kvádru. Takto vytvořené dveře se otevíraly všechny najednou, proto jsem vytvořil dveře každé zvlášť.

Vytvoření dveří:

```
DEF SNIMAC01 ProximitySensor { # snímač přítomnosti Avatara
   center 0 1 0
   size 1.05 2 4 }
DEF DVERE01 Transform {        # dveře
   translation     0 1 0       # posunutí dveří
   children [Transform {
         translation      0.525 0 0
         children Shape {
               geometry Box  {
                     size 1.05 2 0.02 }}} # velikost dveří
DEF CASOVAC TimeSensor {cycleInterval 10 }
         # cyklus otevření a zavření dveří
DEF OTOC OrientationInterpolator {
   key [0 0.2 0.8 1]     # rozdělení 2s, 6s a 2s
   keyValue [0 1 0 0, 0 1 0 1.57, 0 1 0 1.57, 0 1 0 0]}]}
ROUTE SNIMAC01.enterTime TO CASOVAC.startTime
ROUTE CASOVAC.fraction_changed TO OTOC.set_fraction
ROUTE OTOC.value_changed TO DVERE01.rotation
```



Obrázek 4.3: Dveře a senzor

Dveře fungují tak, že při detekci Avatara senzorem, na obrázku 4.3 světle modrá, se budou dveře dvě sekundy otvírat do úhlu -90° zůstanou šest sekund otevřené a nakonec se budou dvě sekundy zavírat. Použití uzlu `ProximitySensor` je uvedeno v kapitole 3.11.5

4.1.3 Umístění modelu do prostoru

Takto vytvořený model vypadá jako plovoucí v černém prostoru, pro lepší reálnost modelu je lepší vytvořit pozadí pomocí uzlu `BackGround`, který vytvoří kolem modelu pozadí v podobě koule viz kapitola 3.11.1 .

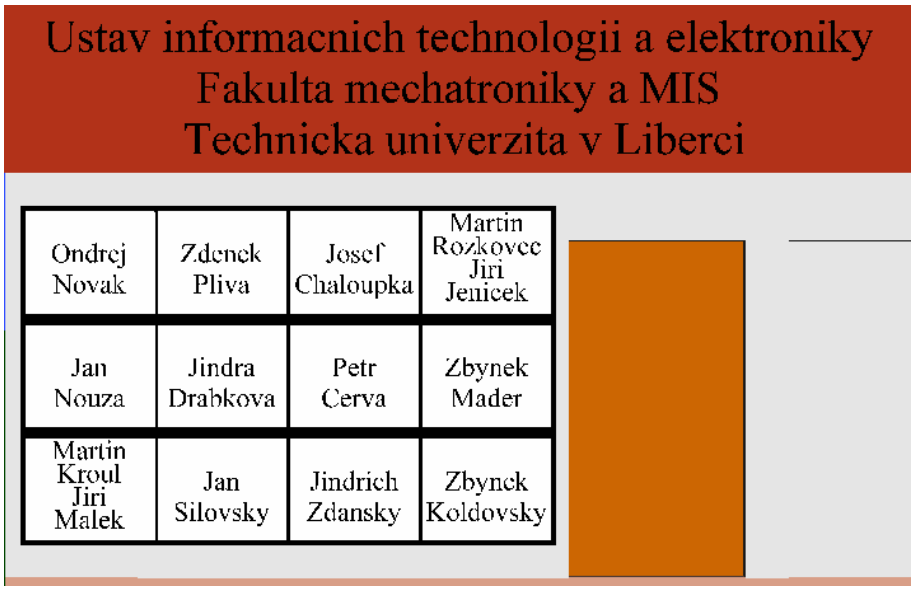
4.1.4 Informace o umístění členů ITE

Dalším krokem bylo doplnění informací o umístění členů ITE. Udělal jsem to tak, že jsem umístil vedle vstupních dveří tabulku, na které je jméno toho, kdo sídlí v místnosti za dveřmi. Tabulka se jménem obsahuje internetový odkaz na stránky katedry s informacemi o konkrétním členu, použil jsem uzel `Anchor` viz kapitola 3.11.2 .

4.1.5 Interaktivní průchody ITE

Posledním bodem zadání bylo navrhnout pohledy na model a interaktivní průchody ústavem. Pohledy na model jsem udělal tři, jeden hlavní před modelem a dva vedlejší, pohled shora a zprava. Na začátku modelu se nachází tabule se jmény členů katedry (obrázek 4.4). Tabulka obsahuje odkazy na průchody podle toho, kterého si vyberete, k tomu Vás prohlížeč dovede. Průchody jsem vytvářel tak, že jsem začal těmi co jsou nejblíže a u vzdálenějších jsem použil už vytvořenou trajektorii jen s úpravou celkového času a poslední pozice. Do vzdálenějších míst, například v oblasti P3, by se

Avatar přemísťoval dlouho, a proto jsou tyto průchody rychlejší tak, aby to bylo uživatelsky příjemnější. Vytvoření průchodu je ukázáno v kapitole 3.11.6.



Obrázek 4.4: Vchod a tabule

4.2 Problémy a jejich řešení

V průběhu vytváření virtuálního modelu jsem narazil na několik různých problémů.

4.2.1 Okno

První problém nastal s použitím okna a zdi nad a pod oknem, po přidání sloupu vedle oken začaly být vidět přechody. Smazal jsem proto z definovaného okna zeď nad a pod oknem a ty potom do modelu doplnil jako dlouhou zeď nad a pod oky.

4.2.2 Dveře

Při pokusu vytvořit si zvlášť v souboru dveře o dvou šířích, které bych poté pomocí uzlu `Inline` vložil do modelu na určené místo do modelu, se stalo, že při přiblížení k jakýmkoliv dveřím došlo k otevření všech dveří, které byly vloženy ze stejného souboru. Musel jsem dveře vytvořit každé zvlášť.

4.2.3 Osvětlení

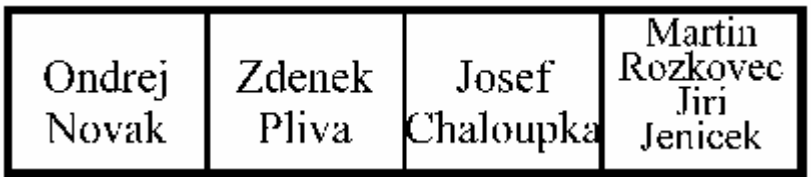
Největší problém nastal s osvětlením celého modelu, nejprve jsem vyzkoušel celý model osvětlit pomocí reflektorů namířených tak, aby osvětlily model ze všech směrů. Po vypnutí Avatarovy svítilny byl model nasvícen ze všech směrů *x*, *y*, *z*, to mělo za následek, vzhledem ke stejné barvě stěn, zmizení rohů, bez viditelnosti

vrcholku stěn nebylo poznat, kde jedna končí a druhá pod úhlem začíná. Smazal jsem tedy reflektory a nahradil je symetricky rozmístěnými bodovými světly, `PointLight` viz kapitola 3.11.2. Při rozmístění zdrojů do tří řad jednu doprostřed a dvě po krajích modelu tak, aby osvítily celý model i trochu ze strany. To, že světlo prochází i objekty, má tu výhodu, že jsem nemusel dávat zdroj světla nad každou místnost. Bohužel umístění tolika zdrojů nad model způsobilo nadměrné osvětlení vodorovných ploch natolik, že byly bílé. Objevil se taky problém na některých zdech, které byly tvořeny více kvádry, osvětlená stěna nepůsobila jako celek, ale kvádry byly nestejně osvícené.

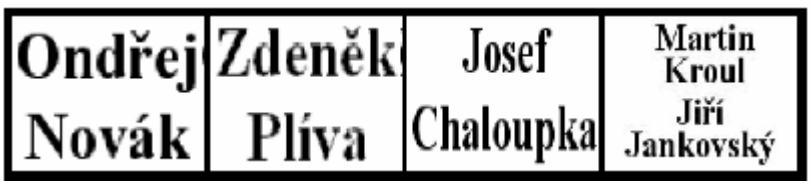
Problém s osvětlením modelu vyřešilo až umístění jednoho bodového světla vysoko nad model s velikým poloměrem dosvitu s nízkou intenzitou osvětlení, ponecháním zapnuté Avatarovy svítilny a upravení vlastností barvy všech použitých objektů tak, aby vyzařovaly nepatrné světlo. Díky tomu zmizelo různorodé osvětlení stěn a osvětlení působí reálně.

4.2.4 Text

Použití uzlu `Text` má pro český jazyk nevýhodu při použití diakritických znamének. Text se při použití těchto znamének nezobrazí. Zkusil jsem proto vytvořit cedule s nápisem jako texturu tak, že jsem vytvořil obrázek se jménem a poté nanesl jako texturu na určitý kvádr. Použití kvádru mělo za následek, že text byl nanesen na všechny stěny kvádru, lepší by bylo použít plochu `IndexFaceSet`, kdy se vykreslí pouze jedna plocha. Bohužel při použití obrázku se mi nepodařilo dodržet stejně veliké obrázky se jmény, a tak písmo vypadalo na každém obrázku jinak. Rozhodl jsem se tedy použít uzel text. Porovnání použití textu a textury je vidět na obrázku 4.5 a 4.6.



Obrázek 4.5: Použití uzlu Text



Obrázek 4.6: Použití textury

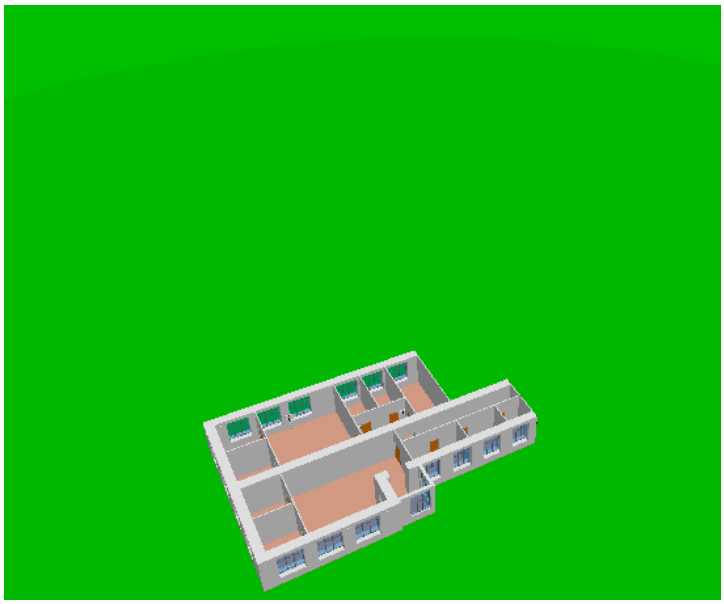
Použití textu jako odkaz má nevýhodu v tom, že kurzor musí být přímo na textu, což je vzhledem k členitosti textu složité. Proto je dobré umístit text a desku pod ním do jednoho uzlu `Anchor` viz kapitola 3.11.3.

4.2.5 Inline

Při použití `Inline` se mi jednou při kopírování souborů stalo, že se nevykreslily připojené soubory. Problém byl v dlouhé adresářové cestě nebo v dlouhých názvech adresářů. Přesunutím výše v adresářové struktuře nebo přejmenováním lze problém vyřešit.

4.2.6 Model v prostoru

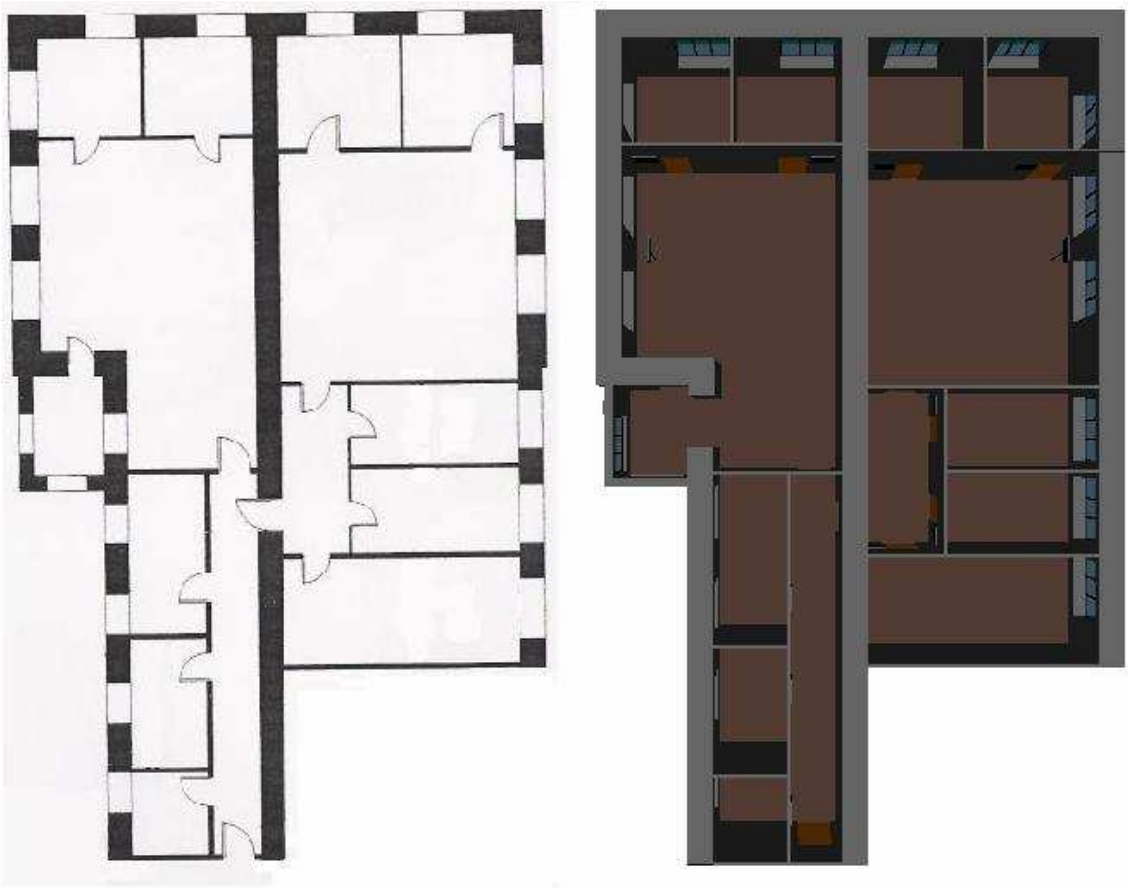
Při prohlížení hotového modelu vypadá model z venku jakoby levitující v prostoru. Použití pozadí tento jev neodstranilo, nakonec jsem pod model vložil tenký model se stejnou barvou jako má pozadí země. Při použití kvádru byly v dálce vidět rohy podstavy, a proto jsem použil tenký válec, aby v dálce nebylo vidět, kde končí podstava a kde je pozadí.



Obrázek 4.7: Podstava a pozadí

4.3 Porovnání s realitou

V porovnání s realitou se nejvíce model podobá hlavně rozměrově, rozmístěním oken a dveří, jak je vidět na porovnání na obrázku 4.8. Dveře se otevírají přesně podle plánku.



Obrázek 4.8: Porovnání plánku a modelu

Realitě se moc nepodobají okna, reálná okna jsou příliš členitá, a proto jsem je vytvořil jednodušší.

4.4 Možnosti zlepšení

Asi největší možností na zlepšení vzhledu blíže k realitě bych viděl v použití textury na zdech a dveře tak, aby vypadaly více jako reálné. Další možností by mohlo být rozmístění nábytku do modelu, například stoly a židle, dalo by se to provést rychle a jednoduše pomocí `Inline`, stačilo by si vytvořit například stůl se židlí. Rozmístění nábytku by ale ztížilo procházení Avatara modelem, průchody by se musely změnit tak, aby stoly a židle obcházel, proto jsem vložení nábytku vynechal. Možností na zlepšení pro návštěvníka by mohlo být umístění další informace nad jméno učitele u dveří a přidat i fotku učitele sídlícího v této místnosti. Dále by se mohlo upravit pozadí

z kulového na krychli a použít fotky skutečného okolí budovy. Nakonec by se dal model dodělat jako model celého patra a umístit jej do celé budovy.

5 Závěr

Cílem bakalářské práce bylo seznámit se s jazykem VRML, s možnostmi jeho editace a prohlížení. Naučit se pracovat se základními prvky VRML a pomocí těchto znalostí vytvořit virtuální model Ústavu informačních technologií a elektroniky, doplnit informace o umístění členů ústavu, navrhnout pohledy a interaktivní průchody modelem.

Seznámil jsem se s prací v prostředí VRML a prohlížeči. Vyzkoušel jsem si pracovat ve dvou odlišných editorech s grafickým a textovým způsobem programování virtuálních světů. Naučil jsem se vytvářet jednoduché objekty virtuálního světa pomocí různých uzlů a jejich parametrů. Pomocí těchto znalostí jsem vytvořil virtuální model Ústavu informačních technologií a elektroniky. Model obsahuje interaktivní průchody modelem tak, že provede návštěvníka modelem od hlavního vchodu do ústavu ITE k místnosti, kde sídlí zvolený člen ústavu. Vedle dveří visí na zdi tabulka se jménem člena s odkazem na www stránky, kde si návštěvník může přečíst důležité informace o členovi. Hlavním úkolem virtuálního modelu ústavu ITE je zjednodušit návštěvníkům orientaci při následné návštěvě skutečného ústavu a poskytnout rychle informace o členech.

V budoucnu se počítá s umístěním virtuálního modelu na internetové stránky ústavu ITE tak, aby usnadnil orientaci návštěvníkům a především studentům v Ústavu informačních technologií a elektroniky. Pro zlepšení využitelnosti modelu by bylo možné rozšířit model na celé patro budovy a poté ho v rámci možností rozšířit na celou budovu.

Seznam použité literatury

[1] GERGELITSOVÁ, Šárka. *VRML v příkladech*. Praha: BEN, 2004. 224 s. ISBN 80-7300-138-1

[2] ŽÁRA, Jiří. *VRML 97 Laskavý průvodce virtuálními světy*. Brno: Computer Press, 1999. 238 s. ISBN 80-7226-143-6.

[3] Accad.*osu.edu* [online]. 1999 [cit. 2008-04-14]. Dostupný z WWW: <http://accad.osu.edu/~pgerstma/class/vnv/resources/info/AnnotatedVrmlRef/ch1.htm>.

[4] TIŠNOVSKÝ, Pavel. *Root.cz* [online]. 2007 [cit. 2008-05-07]. Dostupný z WWW: <http://www.root.cz/clanky/tvorime-trojrozmerne-sceny-v-jazyce-vrml-2/>. ISSN 1212-8309.

[5] *Grafman Productions* [online]. 1996 [cit. 2008-05-07]. Dostupný z WWW: <http://graphcomp.com/grafman/vrml/tips.html>.

[6] NADEAU, David , MORELAND, John, HECK, Michael. *SIGGRAPH 98* [online]. 1998 [cit. 2008-05-06]. Dostupný z WWW: <http://www.sdsc.edu/~moreland/courses/Siggraph98/vrml97/vrml97.htm>.

[7] SERBUSOVÁ, Marie. *Jazyk VRML 2.0*. [s.l.], 2005. 119 s. Bakalářská práce.

[8] FARKAŠOVÁ, Blanka, KRČÁL, Martin. *Projekt Bibliografické citace* [online]. 2004-2008 [cit. 2008-05-07]. Dostupný z WWW: <www.citace.com>.